University of Liège
School of Engineering
Department of Electrical Engineering & Computer Science

INDUCTIVE BIAS IN DEEP PROBABILISTIC MODELLING

a PhD dissertation
by Antoine Wehenkel

Advisor: Prof. Gilles Louppe
September 2022

This dissertation has been submitted and accepted in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Computer Science.

A word from the artist:
This piece was created in collaboration with artificial intelligence, taking an original human-made artwork and reimagining it with VQGAN-CLIP, a text-to-image AI network. The end result mixes 1960s Brutalist architecture and 18th-century botanical illustration to unique effect.

JURY MEMBERS

Pierre Sacré, Professor at the Université de Liège (President);

Gilles Louppe, Professor at the Université de Liège (Advisor);

Pierre Geurts, Professor at the Université de Liège;

Patrick Gallinari, Professor at Sorbonne University;

Jörn-Henrik Jacobsen, Senior research scientist at Apple Inc.;

Alexandros Kalousis, Professor at the HES-SO Genève.

ABSTRACT

One of the most notable distinctions between humans and most other animals is our ability to grow collective intelligence along generations. This development appears exponential; we are witnessing an explosion of knowledge and technical capabilities. Since the invention of computers, artificial intelligence (AI) has enabled machines to push further the boundaries of our collective intelligence. The rapid progress of information technology and the profusion of data has made machine learning (ML), a sub-field of AI, crucial for the collective intelligence growth.

The probabilistic modelling framework unifies the ML-based and human-based knowledge discovery processes, i.e. the creation of mathematical descriptions of real-world phenomena. Thus it is at the root of this disruption of innovation. In this context, this thesis collects five scientific papers that have contributed to developing modern approaches to probabilistic modelling between 2018 to 2022.

This thesis provides a thorough introduction to modern probabilistic modelling. We discuss the why and the how of probabilistic modelling, and we introduce two important classes of models: *probabilistic graphical models* and *deep probabilistic models*. Then, we contrast our work into contributions to *uninformed* and *informed* models. The former models are prefered when data contains enough information to retrieve the targetted model instance. In contrast, *informed* models embed stronger prior knowledge of the phenomenon of interest. Data is only there to complement this knowledge. The quality of *informed* model instances depends on the data and validity of the prior knowledge.

The second part of the thesis focuses on three distinct contributions to *uninformed* probabilistic models. First, we are interested in bringing together distinct model classes; the combination of diffusion models and variational auto-encoders unlocks new modelling features. Second, we draw explicit connections between Bayesian networks and normalizing flows. We exploit this connection to study some representational aspects of normalizing flows. Finally, we present a new neural network architecture that enforces a monotonic response. We demonstrate the effectiveness of this representation in modelling continuous probability distributions.

In the third part of the manuscript, we consider *informed* probabilistic models. In the fourth contribution, we introduce the graphical normalizing flows, a new normalizing flow architecture that embeds independence assumptions. Finally, our last contribution shows that informing deep probabilistic models with a partial physical understanding of the studied phenomenon unlocks generalisation capabilities inaccessible to non-informed models. We conclude this work with a summary and a brief prospect of future developments in deep probabilistic modelling.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to the members of my jury, Alexandros Kalousis, Jörn Jacobsen, Patrick Gallinari, Pierre Sacré, and Pierre Geurts. I greatly appreciated their insightful comments, constructive criticism, and the engaging discussions we had during the public defence.

I feel lucky to have had the opportunity to learn from Pierre during his data structure and algorithms course. His teaching ignited my passion for computer science. I am grateful for the foundation he provided that enabled me to pursue my research interests.

I want to give a special thanks to Jörn and Jens Behrmann for the chance they gave me to work with them as an intern at Apple. Their positivity, expertise, and challenging approaches significantly impacted my personal and professional growth. I feel fortunate to have worked with such exceptional researchers and mentors.

I am incredibly fortunate to have had Gilles Louppe as my advisor throughout my doctoral studies. Gilles, you have been an outstanding advisor, constantly challenging my assumptions and supporting me in my desire to explore new ideas. I feel grateful for your unwavering support, encouragement, and guidance. The many coffee breaks we shared were not only enjoyable but also helped me to grow new perspectives and insights on what it is to be a researcher; I will certainly miss them.

I would like to thank my colleagues at the department for their camaraderie, support, and the great memories we shared. Nicolas, my office mate for over three years, you made my workdays awesome, even though the quality of your humour is debatable. I feel honoured that you trusted me to be the godfather of Elena. Pascal, who always brings good vibes to wherever you go, I was lucky to share four years of PhD with you. Nicolas, and Pascal, we should also be grateful to Van Anh for being so kind to tolerate almost all of our terrible jokes. Damien, Nicolas, Pascal, and Gilles, thanks for the tremendous Orval-intensive times we had after work, even though it was sometimes a bit harsh the day after. I always keep an Orval in the fridge in case you visit Zürich. Laurine, thanks for being there to share gossip at Montefiore. Guillaume Drion, Pierre Sacré, and Laurent Mathys, thanks for our small chats during coffee breaks. Matthia, thanks for bringing some international flavours to the department and showing me some of your Italian cooking skills. I also want to thank all the others with whom I had the chance to spend lunch breaks or have small chats in the corridor: Anthony, Antonio, Anaïs, Antoine, Arthur, Balthazar, Eric, Gaspard, Jean-Michel, Jonathan, Kathleen, Marc, Philippe, Romain, Renaud, Sophie, Thibaud, and Yann. Your friendship and support were greatly appreciated, and I feel lucky to have been part of such a wonderful group.

I am grateful to the "Dream Team + Queens", my friends outside of work, for being what they are. Andrea, Beniboy, Clairou, Laroushka, Mathilefou, Maximus, Pierro, Polo,

Rémy, QBask8, Thibaud, TZ: I am lucky to be part of such a beautiful group of friends. Spending time with you has helped me to maintain a healthy work-life balance and to have fun away from work-related things.

I want to express my heartfelt gratitude to my family. They have always been a source of love, encouragement, and inspiration. Their unwavering support and belief in me have been critical in helping me pursue my dreams. Thomas and mother, thanks for making fun of me and keeping me aware of the reality outside of academia. Marie, for always being my number one supporter. You are one of those who motivated me to pursue this PhD, and I will always be grateful for that. Dad, thank you for transmitting your love for engineering and science to me and for always being present with good advice when I need it.

Finally, I would like to thank you, Alice, for your patience, love, and understanding throughout my PhD journey. Your unwavering support and encouragement have helped me overcome various challenges and motivated me to achieve my goals. I am deeply grateful to have you in my life.

CONTENTS

First, we guess a law. Then we compute the consequences of the guess and then we compare the computation results to nature to see if it works. If it disagrees with experiment, it's wrong. In that simple statement is the key to science. It doesn't make any difference how beautiful your guess is, it doesn't matter how smart you are who made the guess. If it disagrees with experiment, it's wrong. That's all there is to it.

Richard Feynman

1 INTRODUCTION

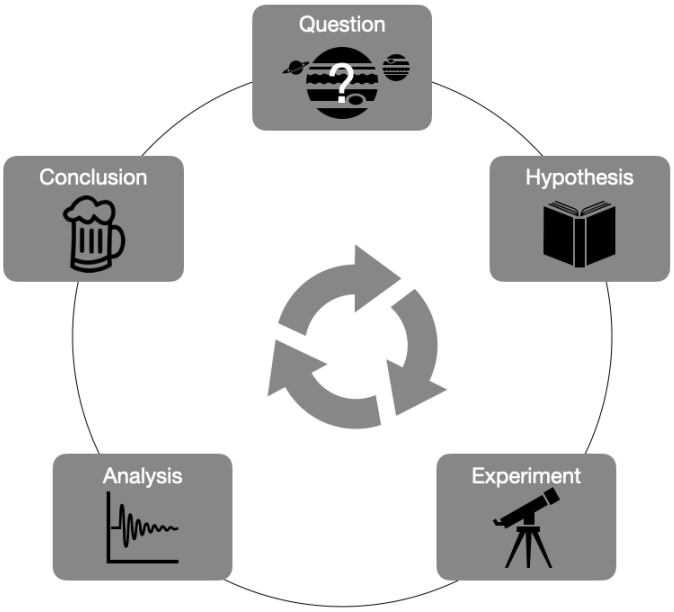


Figure 1.1: Illustration of the scientific inquiry as a simplified 5-step process. The pictograms sketch a cartoon of the TRAPPIST-1 planetary system discovery made by astronomers from Liège University in 2016 [Gillon et al., 2017]. Scientists shall first formulate a *research question* and a set of reasonable *hypotheses*, together they define a class of hypothetical models of the world. In order to answer their question, scientists gather data from *experiments* and analyze them in the context of their modelling assumptions. Checking the consistency of different hypothetical models allows them *to provide an answer* to their initial question up to a certain degree of certainty.

Ever since the beginning of their existence, humans have always been curious to understand the world's complexity. This is not only true at the individual level, as we keep building and improving our knowledge over our lifespan, but also at the level of humankind, where knowledge has never really stopped rising since the beginning of civilisation. As an objectification of our curiosity, science aims to ground the construction of this common knowledge on rationality. In particular, the modern scientific method arguably relies on five pillars, as depicted in Figure 1.1, which ensure discoveries are made out of rigour and based on current scientific knowledge. Although the scientific method can answer questions in the context of a specified model of the world as stated by the

hypotheses, the more fundamental goal of science is to refine these models by criticising their ability to predict the real world. In this context, this thesis aims to explore and contribute to modern techniques for building or refining models of the world.

Classically, scientists or engineers use their domain expertise to build incrementally complex models and improve their faithfulness to reality. The scientific method is then applied to validate or reject the new class of models. This strategy has not only led to today's state of science but also to the uttermost engineering accomplishments of modern times. In engineering, these successes impact our daily life, such as by enabling nuclear electricity – as predicted by special relativity – or allowing one to read this manuscript on a smart tablet – thanks to Maxwell equations' implications on the design of modern computers. More abstract but arguably as impactful on our vision of the universe, the classical model refinement strategy led to all modern scientific breakthroughs. As an example, these models allow astronomers to observe the furthest human-known objects, thanks to general relativity and gravitational lensing and enable the indirect observations of black holes thanks to gravitational waves theory.

Despite these numerous achievements, machine learning has recently challenged the classical modelling approach. Where humans fall short of finding patterns in a large amount of data and build arbitrarily complex models by themselves, machines can automatically perform these tasks day and night. The paradigm shift from hand-crafted to automated modelling happens in a world where the most ambitious scientific experiments generate petabytes (10^{15}) of data per second [CERN]. On the one hand, while the computing capacity continuously increases, the human brain is not better wired to apprehend vast amounts of data than it was when Galileo Galilei set the basis of modern science 400 years ago. On the other hand, deep learning has recently proven its ability to build accurate generative models that outperform the ones designed out of human expertise. This is, for example, true in the context of voice synthesis [Van Den Oord et al., 2016], text-translation [Brown et al., 2020; Devlin et al., 2018], chatbots [Alayrac et al., 2022], or even text-to-image synthesis [Ramesh et al., 2022; Saharia et al., 2022], to cite a few. Among these models, some impersonates human artists. For instance, Figure 1.2 shows images produced by a small version of DALL · E 2, a machine learning model capable of generating images from text descriptions. Under these circumstances, the paradigm shift seems natural, even in a scientific context.

Machine learning has revolutionised the way we build models over the past decades. Yet, modelling *uncertainty* with machine learning models is still a very active research field. Models that account for uncertainty are called *probabilistic models* and describe non-deterministic relationships between the quantities of interest. When it comes to machine learning, these descriptions often take the form of a *generative model* that synthesise realisations of the phenomenon of interest. For example, DALL · E 2 is what we call a *deep generative model*; it parameterises a generative process from textual description to plausible images with deep neural networks.

Figure 1.2: *A dog writing a PhD thesis*, generated by DALL · E mini [Dayma et al., 2021].

Despite some great successes, automating probabilistic model discovery remains an active research area. One of the great features of ML algorithms, their genericity, is also an important flaw. The weak assumptions behind ML models contrast with the classical modelling approach, which is grounded on a deep understanding of the phenomenon of interest. The term *inductive bias* refers to all the weak assumptions made by the machine learning algorithm, such as continuity or invariance. Together with large amounts of data, the inductive bias of current ML algorithms suffices for some modelling tasks such as image or audio synthesis. However, this approach fails in scarce data settings or on data modality for which the inductive bias of existing learning algorithms is not appropriate. In contrast to the classical modelling approach, ML algorithms do not exploit effectively existing prior knowledge. Moreover, existing algorithms exhibit other limitations, such as training instabilities and lack of expressivity, to cite a few.

> *Explicit vs implicit models*
>
> In general, accurate modelling requires accounting for uncertainty. Modelling uncertainty is essential as, by definition, models are simplified representations of reality; they cannot explicitly represent all possible sources of perturbations deterministically. For instance, we determine the precision of any sensor by fitting a model that accounts stochastically for internal and external perturbations that may arise in practical settings such as temperature, humidity or pressure variations. This inherent stochasticity exists both for small models that are simplistic representations and for larger models that are a combination of smaller stochastic models. These models describe the stochastic relationship between observations $\boldsymbol{x}$ provided the models' parameters $\boldsymbol{\theta}$.
> While simple models lead to tractable likelihood functions $p(\boldsymbol{x}|\boldsymbol{\theta})$, larger models are computer programs for which the likelihood is usually intractable because of the multiple sources of randomness. We use the terms *explicit* and *implicit* to distinguish between models that provide direct access to the likelihood function and those that do not. We use the same terms for deep probabilistic models that provide access to the likelihood or solely to the generative process.

1.1 RESEARCH QUESTION

Motivated by this paradigm shift and the remaining challenges in deep probabilistic modelling, this thesis contributes to answering the following research question: **How to automate the discovery of probabilistic models with deep learning algorithms?** The rapid progress of information technology and the profusion of data makes this question timely. Failing to provide answers would miss an extraordinary opportunity for scientific and technological discoveries.

In pursuing this objective, we explore several directions that study and improve upon various aspects of deep probabilistic models. We distinguish between data-driven models and hybrid models. On the one hand, the performance of data-driven models mainly

depends on the learning algorithm's inductive bias and the availability of representative data. These algorithms are well suited to modelling tasks for which we do not have strong domain knowledge but access to many representative data. On the other hand, hybrid models are informed by solid prior knowledge, e.g. independencies or partial understanding of the underlying physics. Hybrid modelling allows the combination of domain knowledge with data and is thus better equipped against small or less-representative datasets. We argue that contributions to both modelling strategies are complementary and will improve the range of application of deep probabilistic models.

1.2 OUTLINE AND STRUCTURE

Before diving into the core contributions of this thesis, a review of probabilistic modelling in Part i naturally follows this introduction. We provide the notions necessary for the appreciation of this thesis by a reader equipped with a technical background. Chapter 2 is an accesible primer on probabilistic modelling. Then, we introduce graphical probabilistic models and their technicalities in Chapter 3. We conclude the background by presenting deep probabilistic models in Chapter 4. Then, Part ii focuses on *uninformed* deep probabilistic models, which only rely on the standard inductive bias of machine learning algorithms. In contrast, with Part iii studies the effect of solid modelling assumptions such as independencies or prescribed physical equations, leading to a class of models that we dub *informed* deep probabilistic models in this thesis.

In Part ii, we aim to understand existing probabilistic models better and improve their expressivity. To this end, we explore three distinct directions, each giving rise to its own chapter. First, Chapter 5 chapter studies the complementarity of variational auto-encoders [Kingma and Welling, 2013] and diffusion models [Sohl-Dickstein et al., 2015; Ho et al., 2020; Song and Ermon, 2019]. We demonstrate that diffusion models are a suitable replacement for the simple Gaussian prior classically used in variational auto-encoders. Then, Chapter 6 aims to understand better normalizing flows [Tabak and Vanden-Eijnden, 2010; Tabak and Turner, 2013; Rezende and Mohamed, 2015, NFs] a popular class of probabilistic models. In particular, Chapter 6 shows that affine normalizing flows, a class of explicit models, have limited modelling capacities. The second part of this manuscript ends with Chapter 7 where we address the limited expressivity of affine normalizing flows by introducing unconstrained monotonic neural networks.

Part iii explores hybrid modelling, a family of algorithms that embrace the opportunity to combine expert knowledge and deep probabilistic models. In particular, Chapter 8 suggests explicitly handling independence assumptions in normalizing flows, and Chapter 9 studies the generalisation capabilities of deep probabilistic models equipped with a partial model of the studied process. As a conclusion and summary, Chapter 10 reflects upon the contributions of this thesis.

1.3 PUBLICATIONS

Setting aside this introduction, an original primer on probabilistic modelling in Part i and the conclusion, the scientific content of this manuscript is exclusively borrowed from original contributions made to deep probabilistic modelling over the last four years. Each selected contribution sets its own chapter complemented with a prologue and an epilogue. The manuscript builds upon the following list of publications, ordered by publication date,

> [Wehenkel and Louppe, 2019] *Unconstrained monotonic neural networks*, **Wehenkel Antoine** and Louppe Gilles.
> Advances in neural information processing systems, 2019.
> → Chapter 7.

> [Wehenkel and Louppe, 2020] *You say Normalizing Flows I see Bayesian Networks*, **Wehenkel Antoine** and Louppe Gilles.
> International Conference on Machine Learning, Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models, 2020.
> → Chapter 6.

> [Wehenkel and Louppe, 2021b] *Graphical Normalizing Flows*, **Wehenkel Antoine** and Louppe Gilles.
> International Conference on Artificial Intelligence and Statistics, 2021.
> → Chapter 8.

> [Wehenkel and Louppe, 2021a] *Diffusion Priors In Variational Autoencoders*, **Wehenkel Antoine** and Louppe Gilles.
> International Conference on Machine Learning, Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models, 2021.
> → Chapter 5.

> [Wehenkel et al., 2022] *Robust Hybrid Learning With Expert Augmentation*, **Wehenkel Antoine**, Behrmann Jens, Hsu Hsiang, Sapiro Guillermo, Louppe Gilles, and Jacobsen Jörn-Henrik.
> In preparation, arXiv preprint arXiv:2202.03881.
> → Chapter 9.

1.4 ADDITIONAL PUBLICATIONS

Along the pursuit of my PhD degree, I had the chance to take part in fruitful collaborations not directly related to the scope of this thesis. The following list of co-authored publications stemmed from these collaborations:

[Pesah et al., 2018] *Recurrent Machines For Likelihood-free Inference*, Pesah Arthur, **Wehenkel Antoine** and Louppe Gilles.
Advances in neural information processing systems, MetaLearn Workshop, 2018.

[Wehenkel et al., 2020] *Parameter Estimation Of Three-phase Untransposed Short Transmission Lines From Synchrophasor Measurements*, **Wehenkel Antoine**, Mukhopadhyay Arpan, Le Boudec Jean-Yves, Paolone Mario.
IEEE Transactions on Instrumentation and Measurement, 2020.

[Vecoven et al., 2020] *Introducing Neuromodulation In Deep Neural Networks To Learn Adaptive Behaviours*, Vecoven Nicolas, Ernst Damien, **Wehenkel Antoine**, Drion Guillaume.
PloS one 15 (1), e0227922.

[Vandegar et al., 2021] *Neural Empirical Bayes: Source Distribution Estimation and its Applications to Simulation-Based Inference*, Vandegar Maxime, Kagan Michael, **Wehenkel Antoine** and Louppe Gilles.
International Conference on Artificial Intelligence and Statistics, 2021.

[Delaunoy et al., 2020] *Lightning-Fast Gravitational Wave Parameter Inference through Neural Amortization*, Delaunoy Arnaud, **Wehenkel Antoine**, Hinderer Tanja, Nissanke Samaya, Weniger Christoph, Williamson Andrew R, and Louppe Gilles.
Advances in neural information processing systems, ML4Science Workshop, 2020.

[Hermans et al., 2021] *Averting A Crisis In Simulation-based Inference*, Hermans Joeri, Delaunoy Arnaud, Rozet François, **Wehenkel Antoine**, and Louppe Gilles.
In preparation, arXiv preprint arXiv:2110.06581.

[Dumas et al., 2021] *A Probabilistic Forecast-driven Strategy For A Risk-aware Participation In The Capacity Firming Market*, Dumas Jonathan, Cointe Colin, **Wehenkel Antoine**, Sutera Antonio, Fettweis Xavier, and Cornélusse Bertrand.
IEEE Transactions on Sustainable Energy, 2021.

[Dumas et al., 2022] *A Deep Generative Model For Probabilistic Energy Forecasting In Power Systems: Normalizing Flows*, Dumas Jonathan, **Wehenkel Antoine**, Lanaspeze Damien, Cornélusse Bertrand ,and Sutera Antonio.
Applied Energy, 2022.

[Delaunoy et al., 2022] *Towards Reliable Simulation-Based Inference with Balanced Neural Ratio Estimation*, Delaunoy Arnaud, Hermans Joeri, Rozet François, **Wehenkel Antoine** and Louppe Gilles.
In preparation, arXiv preprint arXiv:2208.13624.

Part I

BACKGROUND

[We must avoid] false confidence bred from an ignorance of the probabilistic nature of the world, from a desire to see black and white where we should rightly see gray.

Immanuel Kant
as paraphrased by Maria Konnikova

2

INTRODUCTION TO PROBABILISTIC MODELLING

Outline

This chapter provides a broadly accessible introduction to the probabilistic modelling framework. We review the concepts of a model, inference, and learning. In addition, we motivate probabilistic modelling in the context of artificial intelligence and present it as a generalisation of machine learning. Our discussion shall convince the reader that this framework enables abstract reasoning and is a real-world problem solver. It shall motivate the relevance of improving this framework as a research goal.

2.1 INTRODUCTION

The invention of computers has enabled the automatisation of many tasks historically accomplished by humans. One key ingredient of this revolution is making computers *reason* – to make an informed judgment based on logical arguments and observations. The set of hypotheses on which this logic builds is a *model*. Both humans and machines rely on reasoning to perform tasks. For instance, let us consider the problem of ordering a bottle of wine at a restaurant. We first need some hypotheses, e.g., • What kind of wine do people like at the table? Red or white? Strong or delicate? • What is the budget? • What do we have on the menu? • What is the wine list? Etc. Provided with this model, we can make an informed choice: remove the wines that are too expensive or would displease someone at the table and pick one of the remaining that goes well with tonight's dinner. More scientific examples include a climatologist who needs a model of the earth and its atmosphere to explain climate change; or streaming platforms which use a model that represents users' tastes to make video recommendations.

In these contexts and many others, the model plays a central part. In practice, we aim for **faithful models** – models for which reasoning leads to a proper judgment. In the examples mentioned above, we strive for models that help us choose the right wine; that explain the climate over the last centuries; that keep the user longer on the platform. The exact definition of the model's quality depends on the targetted application. However, certain classes of models are usually strictly better than others. A model that embeds notions of uncertainty is more expressive than a deterministic model; the latter models cannot accurately represent phenomena that exhibit randomness.

It is unknown whether the laws that rule our reality are deterministic or stochastic. Nevertheless, modelling requires simplifying assumptions, which cause uncertainty. For instance, a model that acknowledges partial observability must express uncertainty about the system's state. Uncertainty also arises from the finite precision of measurement devices or from simplifying approximations that are not strictly correct, among other reasons.

We thus need a formal language to express this stochasticity, the language of probability [Kolmogorov and Bharucha-Reid, 2018]. *Probability* is the language that enables formal statements about uncertain events. It allows contrast between what is *possible* and what is *plausible*. This distinction is essential as it constructs deterministic decisions by focusing on the most plausible events and discarding unlikely events. For example, wine amateurs know that an old bottle has a higher chance of being corked than a recent one but might also taste better. Probability allows us to express this fact and consider it to select an appropriate bottle of wine.

> *Stochasticity, randomness, or uncertainty?*
>
> In our discussion, we use these terms as follows. *Stochasticity* refers to the property of being well-described by the probabilistic language. It is a statement about a model. In contrast, *randomness* refers to the phenomenon itself, not to a model. *Uncertainty* is the result of randomness or stochasticity. It is our way of acknowledging that we do not know things perfectly. It is equally Nature itself than our modelling assumptions that cause uncertainty.

2.2 PROBABILISTIC MODEL

A probabilistic model is a model that describes a phenomenon of interest in probabilistic terms. Practically, it defines a probability distribution over the set of variables considered valuable to describe the phenomenon (e.g., X, Y, Z). The distribution modelled can be the joint between all variables (e.g., $P(X, Y, Z)$) or a conditional distribution (e.g., $P(X \mid Y, Z)$). Our discussion focuses on the former case for simplicity but with no loss of generality.

One goal of building a probabilistic model, arguably the main, is to perform *inference*. That is to answer questions in the context of a model, to reason. These questions come in different flavours. One could for example wonder: What is the most likely value of Y if we are to observe X? What is the conditional distribution of Y in this case? It is also different whether we want to evaluate the value $P(Y \mid X)$ or just sample from it. For these purposes, the probabilistic models may have to handle different queries: *marginalisation, conditioning, sampling, and probability evaluation.*

Certain representations are appropriate for a subset of queries and not for others. For example, we can represent the discrete distribution between X, Y, Z with a 3D table where each entry stores the corresponding joint probability. The evaluation of the joint

Formalism

In this background, we will abstract the domain (discrete or continuous) of the random variables when possible. The term (conditional) probability distribution refers to the object P that fully describes the stochastic behaviour of a random variable X taking values in $\mathcal{X}$. When the domain $\mathcal{X}$ is discrete, P corresponds to the probability function $P(X = \cdot) : \mathcal{X} \to [0, 1]$ that satisfies the three Kolmogorov's axioms (*non-negativity*, *unitarity*, and σ-*additivity*). In the case of a continuous domain, we will narrow our discussion to real values $\mathcal{X} \triangleq \mathbb{R}^d$, where d is the dimensionality of the random variable. In this context, the probability distribution is described by the density function $P(X = \cdot) : \mathcal{X} \to \mathbb{R}_+$ which defines the probability that a realization $\boldsymbol{x}$ of X lies in a sub-domain $\mathcal{A}$ as $\int_{\boldsymbol{x} \in \mathcal{A}} P(X = \boldsymbol{x}) \mathrm{d}\boldsymbol{x}$. The probability functions implied by the density must also satisfy Kolmogorov's axioms. In this chapter, we clearly mention the domain of the random variable when it matters for the discussion. In the next chapters, we will focus on the continuous case, hence we will use the standard notation p to denote the probability density function.

probability is very efficient with this representation. However, evaluating a conditional distribution requires going over each entry corresponding to the conditioning value and re-normalising the probabilities by their sum. Sampling, marginalisation, and conditioning become very inefficient as the number of entries in the table grows exponentially with the number of dimensions. Fortunately, alternative representations exist, such as the ones we review in the following two chapters.

Frequentist vs Bayesian interpretation

Two interpretations of probabilities co-exist. In the above discussion, we brought probabilities as a language to express our uncertainty about the truth of facts. We took the *Bayesian* interpretation of probabilities. The reference to sir Bayes comes from the application of Bayes' rule to update our prior belief in the presence of new evidence. In this context, the prior is part of the model and affects its quality. The main drawback of the Bayesian interpretation is the potential complexity of defining the prior appropriately. The other view, referred to as *frequentist*, interprets a probability as a frequency of events. With this perspective, probability does not quantify uncertainty; it expresses intrinsic randomness. Frequentists reject the notion of prior belief, which has pros and cons. In general, there is no interpretation better than the other. However, the Bayesian interpretation provides motivations for many popular algorithms in machine learning and is the one we will often implicitly use in our discussions. At the same time, we do not strictly reject the frequentist point of view. For example, we discuss the maximum likelihood principle, a frequentist concept.

2.3 LEARNING

Until now, we have implicitly assumed the model was given to us. However, this is not realistic in many settings. For example, how can we accurately model someone's wine taste? Ranking all world's wines is not a reasonable option. However, we could ask a list of questions and then summarise the taste from the principal characteristics of wine. The task of summarising these answers is *learning* – to build a compressed representation of observations. Afterwards, we can use this representation to perform an informed guess. This representation is a model, a simplified representation, of someone's wine taste. Learning is thus the task of instantiating a model from data.

In practice, we do not perform *learning* without additional assumptions. Instead, we define a set of models among which we believe at least one would be a good representation of the phenomenon of interest. If we are Bayesians, we even prescribe a probability that encode our faith in each model. We will return to the Bayesian prospect later but for now, let us assume we do not have any a priori of the models' quality.

The class of possible models can be finite, e.g. the class contains two models – one for people who like red wine and white wine; the other for people who only like red wine. Compressing someone's taste into one of these models goes with a significant loss of information but might already be helpful in some settings. The class of models can be infinite, e.g. if parameterised by real values. For example, we can summarise wine taste by attributing an affinity score to each of the main features of wine. Let us now review concrete learning strategies.

2.3.1 *Maximum likelihood estimation*

A learning strategy is a set of rules that produces a model from data. When we only consider a finite number of models, a simple strategy exists. We test the predictive performance of each model and select the one that is the most consistent with our data. If the models describe the phenomenon with discrete events, we maximise the probability. If it considers a continuous set of events, we maximise the density. In the case where one of the models is *correct* – it is the one that generates the data – this selection algorithm will eventually select the right model as the number of independently and identically distributed (iid) data points tends to ∞. This selection technique is said *consistent*.

We can use a similar approach when the models are parameterized by a real vector $\boldsymbol{\theta} \in \mathbb{R}^d$. In this case, our goal is to estimate a good value for $\boldsymbol{\theta}$. One approach, denoted maximum likelihood estimation [Fisher, 1922, MLE], is to select the model's parameter $\boldsymbol{\theta}$ that maximizes the joint distribution (density or probability) of a dataset $\mathcal{D} := \{\boldsymbol{x}_i\}_{i=1}^N$ of points $\boldsymbol{x}_i \in \mathcal{X}$. This quantity is called the likelihood function of the parameter $\boldsymbol{\theta}$, denoted $\mathcal{L}(\boldsymbol{\theta}) \triangleq P(\mathcal{D} \mid \boldsymbol{\theta})$. Hence the MLE estimator is formally defined as

$$\boldsymbol{\theta}_{\text{MLE}} = \arg\max_{\boldsymbol{\theta}} P(\mathcal{D} \mid \boldsymbol{\theta}). \tag{2.1}$$

In the presence of iid data, this estimator is consistent – provided a class of models that contains the 'true' generative process, it eventually recovers the 'true' model as the number of points tends to ∞. Formally, the consistency property is a convergence in probability of the estimator to the exact value, and it requires additional assumptions that ensure the model is identifiable and the likelihood function is well behaved.

The consistency of the MLE is an appealing property. However, we must consider the central assumption it relies on very carefully! While ensuring that the model class contains the true generative process is fine in artificial settings, this assumption is a metaphysical question for real data. The law of large numbers saves us if we only look at things on average and are provided with many data, but it does not apply to all modelling tasks. Sometimes, we know this assumption does not hold, but we would still like to learn a good model; the MLE principle does not say much. Even if we know the model class contains the correct model (e.g., if we consider a parametric universal density approximator), the convergence speed with respect to the dataset size is unknown. Moreover, the optimisation of Equation (2.1) is often non-trivial.

2.3.2 *Learning as inference*

The strict delimitation between possible and impossible models is another limitation of the MLE approach. Instead of interpreting learning as model selection, we can reformulate learning as an inference task to alleviate this limitation. In this context, the model is a generic description of the phenomenon parameterised by some explanatory variables. For example, the model can be a parametric function, exactly as when we define a class of models in the MLE approach. The distinction between learning as inference and MLE lies in our interpretation of the parameters. In the former, we consider the parameter $\boldsymbol{\theta}$ as part of the model rather than defining a class of models. The MLE is usually associated with a frequentist interpretation of probability, whereas learning as inference is Bayesian.

Let us say we want to learn a model that predicts the conditional distribution $P(Y = y \mid X = \boldsymbol{x})$ that someone appreciates a wine with features $\boldsymbol{x} \in \mathbb{R}^d$, where $y \in \mathbb{R}$ is a real value that represents someone's affinity with the wine. Learning aims at summarising the information in the data to perform the task of interest. Thus, the objective is to predict the output $P(Y \mid X = \boldsymbol{x}, \mathcal{D})$, the conditional distribution over the output space given features $X = \boldsymbol{x}$ and the set of observations $\mathcal{D}$, which is often a set of input-output pairs $\mathcal{D} := \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$.

For example, the class of models can be a neural network $f_{\boldsymbol{\theta}}(\cdot; \boldsymbol{x}) : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^+$, parameterised by $\boldsymbol{\theta}$, and which defines a parametric density function over $\mathbb{R}$ conditioned on an input $\boldsymbol{x}$. Said otherwise, $f_{\boldsymbol{\theta}}$ represents the conditional distribution $P(Y \mid \boldsymbol{x}, \boldsymbol{\theta})$. We denote with $P(Y \mid X)$ the unknown "true" data distribution, the conditional distribution that has generated the data we observe. We assume the parameters $\boldsymbol{\theta}$ are expressive enough to summarize all information about the "true" distribution contained in the data $\mathcal{D}$. This assumption is equivalent to the conditional independence $Y \perp \mathcal{D} \mid X, \boldsymbol{\theta}$. Then

learning requires i) to condition our predictions on the data $\mathcal{D}$ and; ii) to marginalize with respect to the unknown value of parameters $\boldsymbol{\theta}$. Taking advantage of the conditional independence $Y \perp \mathcal{D} \mid X, \boldsymbol{\theta}$, the model for making new predictions is expressed as

$$P(Y = y \mid X = \boldsymbol{x}, \mathcal{D}) = \int P(Y = y \mid X = \boldsymbol{x}, \boldsymbol{\theta}) P(\boldsymbol{\theta} \mid \mathcal{D}) \mathrm{d}\boldsymbol{\theta} \tag{2.2}$$

$$= \int f_{\boldsymbol{\theta}}(y; \boldsymbol{x}) P(\boldsymbol{\theta} \mid \mathcal{D}) \mathrm{d}\boldsymbol{\theta}. \tag{2.3}$$

The data are only used through the posterior distribution $P(\boldsymbol{\theta} \mid \mathcal{D})$, representing our updated belief about the best models in light of the observed data. Learning thus amounts to compute $P(\boldsymbol{\theta} \mid \mathcal{D})$, which is an inference task.

We have presented Bayesian inference in parametric spaces; however, these concepts generalise to functional spaces [MacKay et al., 1998]. In this context, we need to define a prior distribution over functions (e.g., L2 integrable functions) and be able to evaluate the likelihood defined by each function. This is usually handled by kernel methods such as Gaussian processes and will not be discussed in this thesis.

Keeping track of the complete posterior distribution can be cumbersome in practice. We can avoid this by selecting the maximum a posteriori (MAP) sub-model. In the case of a parametric model this means freezing the parameter $\boldsymbol{\theta}$ to their most plausible value $\boldsymbol{\theta}_{MAP} = \arg\max_{\boldsymbol{\theta}} P(\boldsymbol{\theta} \mid \mathcal{D})$.

Learning as inference strictly generalises the MLE principle to settings where the prior knowledge is more subtle than a binary choice between possible and impossible models. If the prior is non-informative, i.e. the prior does not attribute more credibility to one value of parameters than another, the method is equivalent to the MLE. When we have prior knowledge, this strategy naturally reduces the importance of unlikely model instantiations and prefers models that are the most plausible given the data and our prior beliefs. Moreover, the MAP is also consistent [Schwartz, 1965]; it eventually selects a model that encode the "true" distribution.

The advantage of the Bayesian learning paradigm is to force the explicit formalisation of modelling assumptions and the prior knowledge associated with each learnable component of the model. It acknowledges that learning a model is a subjective task. Occam's razor says we should always favour the simplest of potential explanations. The Bayesian approach may naturally handle this principle by attributing higher plausibility to simpler model instantiations. This is not true for the MLE approach, which requires ad-hoc algorithms or regularisation objectives to favour simpler models.

2.4 MACHINE LEARNING = PROBABILISTIC MODELING

The attentive reader will notice that machine learning (ML) was only mentioned once until now, when discussing the Bayesian and frequentist interpretations of probability. This may sound surprising as this thesis aims to build bridges between ML algorithms

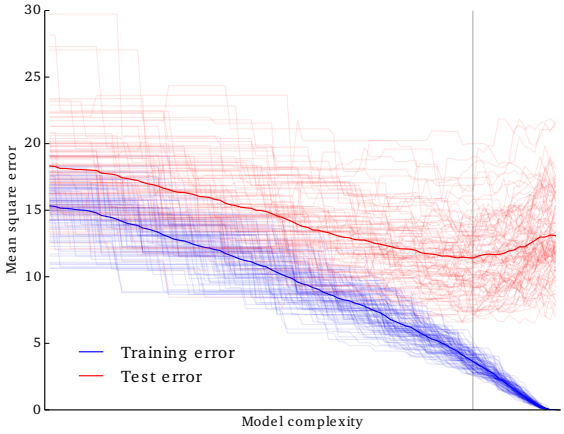


Figure 2.1: The light blue curves show the training error while the light red curves show the estimated test error for 100 pairs of training and test sets drawn at random from a known distribution. The thick blue curve is the average training error while the thick red curve is the average test error. It shows the necessary trade-off between the model's complexity and generalisation performance. As the model gets too complex, it starts overfitting the training data, decreasing test performance. This plot is taken from Louppe [2014].

and the classical modelling approach. We did this on purpose as the distinction between classical modelling (as performed by domain experts, e.g. in science or engineering) and ML is often irrelevant.

We have described probabilistic modelling in generic terms that are valid for both approaches. Whether the class of models is small, made of well-understood pieces, or a deep neural network does not matter when describing the key steps to building and using a model. Even a model designed with domain knowledge usually has degrees of freedom to adapt the model to contexts. At the same time, we must not forget that learning relies on assumptions even when using deep learning with large datasets.

Classical modelling and machine learning aim to find a model that generalises well. On the one hand, classical modelling generally considers simple classes of models which naturally aligns with Occam's razor. It often leads to faithful models when the studied process is well understood. On the other hand, machine learning tackles problems for which classical modelling would fail because the studied phenomenon is less well understood. It does not mean ML's job is to learn a complex model of the phenomenon, quite the opposite. As depicted in Figure 2.1, an ML model achieves its best predictive performance by balancing the model's complexity and goodness of fit. This behaviour is arguably observed with all ML algorithms, although defining the model's complexity is not always straightforward. A standard method to control the complexity of an ML model is to split the data into training, validation and test sets. We use the validation set to find the hyperparameters of the learning algorithm that lead to a trained model that generalises well, that is, a model that has a good balance between faithfulness (good

predictive accuracy for the task of interest) and complexity. We then use these hyperparameters to learn a new model with both train and validation sets and the test set to assess how well the model generalises [Hastie et al., 2009].

Machine learning algorithms are sometimes described in deterministic terms. For example, a classical ML task is to predict a real value y provided a set of features $\boldsymbol{x}$. At first glance, we might have trouble interpreting this in the probabilistic framework, limiting the scope of our previous discussion to a subset of ML algorithms. However, we can always map a deterministic model into the probabilistic framework. Indeed, deterministic learning objectives correspond to the MAP or the MLE of a probabilistic model that assumes the uncertainty a priori. For example, fitting a regression model $f_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Y}$ with mean squared error would lead to the same model as learning a probabilistic model via the MLE and considering a family of models that are Gaussian distributions with a fixed variance and a mean parameterised by $f_{\boldsymbol{\theta}}$. Similarly, mean absolute error corresponds to assuming a Laplace distribution. The duality between the deterministic and the probabilistic interpretations goes further if we observe that regularisation strategies correspond to the prior in the MAP, e.g. ridge regression assumes a Normal prior on the coefficients of the linear model. Khan and Rue [2021] discuss formally the relationships between a wide range of machine-learning algorithms and the Bayesian learning framework.

Another important aspect of modelling is to select the appropriate class of models. This choice varies with the end application which may require performing distinct types of queries on the model. In addition, the learning scenario may also differ and impacts the suitability of different models. As we will see soon, different models may lead to different inference algorithms. Some models represent the distribution of interest as a sampling procedure, while others provide access to the probability distribution. If our goal is to generate samples, we might prefer the former models, although we could also use Markov chain Monte Carlo to draw samples from the others. The following sections shed light on popular classes of probabilistic models that exhibit different advantages and shortcomings.

2.5 CONCLUSION

We have discussed the general principles of probabilistic modelling but have avoided practical technicalities. The following chapters present concrete classes of probabilistic models and the corresponding algorithms for learning and inference. In particular, Chapter 3 discusses **probablistic graphical models** [Koller and Friedman, 2009, PGMs] and Chapter 4 reviews various **deep probabilistic models** [Tomczak, DPMs]. As its name says, the former class aims for a graphical representation of the distribution. It allows understanding the modelling assumptions, such as independence hypotheses, quickly. The latter focuses on models whose internal representations use deep neural networks. Some of these models are especially well suited for sampling and are often called **deep generative models** in this case. We discuss the particularities of each class of models and

provide a thorough description of the main algorithms that perform the different queries aforementioned.

In this manuscript, we argue on multiple occasions that we shall not make a rigid distinction between graphical and deep generative models as they are just different representations of the same mathematical object. It is why we have first started our discussion with a generic introduction to probabilistic modelling that is valid for all classes of models. Some deep probabilistic models directly correspond to a graphical model, enabling abstract reasoning independent of the neural network architectures. However, for clarity, we will first introduce probabilistic graphical models. We then borrow the newly introduced notations and concepts to describe several deep probabilistic models.

Figure 2.2: *A painting by Kandinsky of a mathematician writing formulas and graphs on a blackboard.* As seen by a stable diffusion model [Rombach et al., 2022].

3

PROBABILISTIC GRAPHICAL MODELS

Outline

This chapter introduces the class of probabilistic graphical models. We motivate graphs as effective representations of probabilistic independence assumptions. We distinguish between directed models – a.k.a. Bayesian networks – and undirected models – a.k.a. Markov networks. We discuss each representation's qualities and limitations and present concrete inference and learning algorithms. This chapter provides all notions necessary for the practical and theoretical understanding of probabilistic graphical models in the context of this thesis.

3.1 A GRAPHICAL MODEL IS WORTH A THOUSAND WORDS

As the saying goes, a picture is worth a thousand words. It is why we start our journey in the probabilistic-model lands by revisiting probabilistic graphical models (PGMs). Our trip will pass by Bayesian networks [Pearl, 2011] and make a slight detour by Markov networks [Kindermann, 1980]. As their name hints, PGMs rely on a graphical representation of the probability distributions. We will observe that directed and undirected graphs are appropriate to represent known (in)dependence relations. These representations lead to specialised inference and learning algorithms which we will discuss as well.

The introduction of an undirected representation of the distribution of interacting particles in 1902 by Gibbs [Gibbs, 1902] might be one of the first PGM. We can also attribute one of the first directed PGM to Sewall Wright [Wright, 1921, 1934], who introduced path analysis in genetics in the 1920s. The statistics community only started to acknowledge the graphical framework in the 1960s [Li, 1968; Wright, 1960]. Only later, in the late 80s, PGMs began to creep into the field of artificial intelligence [Russell, 2010, AI] with the seminal works [Pearl, 2022; Kim and Pearl, 1983; Pearl, 1985] of Judea Pearl and his colleagues that provided algorithms to take advantage of Bayesian networks, a class of directed PGMs. Since then, many communities have recognised the graphical representation as a powerful tool. It has achieved great success, such as in the modelling of gene regulatory networks [Werhli and Husmeier, 2007], data compression [McEliece et al., 1998], and many others. Recently, causality [Pearl, 2009; Peters et al., 2017], which under some aspects builds upon Bayesian networks, has arguably become one of the hottest topics in ML and might be part of the next successes in AI.

Many great resources on PGMs exist, and this chapter does not aim to replace them. We aim to provide sufficient materials to get the reader interested in PGMs and understand standard algorithms' main advantages and limitations. This provides a common ground between the reader and the author to motivate the connections with deep probabilistic models and improvements to classical PGMs we have brought into the scope of this thesis. We invite the reader interested in additional details to check the book from Koller and Friedman [2009], the primary reference used to guide this introduction to PGMs. We now motivate the requirement for careful design of model classes.

3.2 THE CURSES OF DIMENSIONALITY

Learning is hard. Let us consider a set of d unfair coins $X = [X_1, \dots, X_d]^T$. Given a dataset of simultaneous throws $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^N$, we want to learn a probabilistic model $P(X)$. A natural approach is to represent the joint probability as a d-dimensional array with an entry for each possible realisation. In this context, learning corresponds to filling the 2^d values in the table. We can reduce this number by factorising the distribution as

$$P(X) = P(X_1)\Pi_{i=2}^d P(X_i \mid X_1, \dots, X_{i-1}),$$

and by acknowledging that the (conditional) probabilities of a tail and a head sum up to 1. Unfortunately, we do not gain much as the number of entries in the table still grows exponentially ($\sum_{i=1}^d 2^{i-1} = 2^{d-1} - 1$). Learning becomes intractable as the number of dimensions grows. This phenomenon is broadly referred to as a *curse of dimensionality* and also hits continuous variables. However, this is just a recall to reality: we need assumptions to create models. The good news is that we can fight the curse of dimensionality with modelling assumptions. For example, it is reasonable to assume the coin throws are independent events. The joint distribution then factorises into d terms: $P(X) = \Pi_{i=1}^d P(X_i)$. For continuous variables, smoothness and constraints on the possible interactions between variables may allow us to achieve modelling results that challenge the curse of dimensionality.

Sampling is hard. We want to sample realisations provided the joint distribution $P(X)$. To this end, we may use approximate sampling schemes, e.g., Markov Chain Monte Carlo [Gilks et al., 1995; Geyer, 1992, MCMC] or importance sampling [Tokdar and Kass, 2010]. These algorithms rely on a proposal distribution (e.g., a normal distribution) and an acceptance/rejection strategy. As the number of dimensions increases, the gap between the proposal distribution expands, and the acceptance rate collapses. Hence, developing efficient sampling strategies that rely on the modelling assumptions is necessary. We will see later how rewarding is the joint development of the model class and the sampling strategy. For instance, sampling algorithms for directed graphical models exploit the network structure.

Interpreting is hard. The complexity of a model naturally grows with the number of variables we consider. Clearly, humans are not able to apprehend correctly more than a few dimensions. Thus, we shall rely on specific modelling frameworks that enable understanding how different assumptions impact the model. Probabilistic graphical models offer a nice balance between expressivity and interpretability.

3.3 DIRECTED GRAPHICAL MODELS – BAYESIAN NETWORKS

The curses of dimensionality prevent probabilistic modelling without appropriate assumptions on the modelled distribution. We now introduce Bayesian networks (BNs), which fight this intractability with independence assumptions. As we have seen, representing d simultaneous coins tosses requires the specification of at least $2^{d-1} - 1$ numbers. This number drops to d if we consider all the variables independent, which is reasonable as the realisation of one coin toss does not impact the outcome for another coin. BNs explicit these (conditional) independencies with a graphical representation and help parameterise distributions more compactly. The term Bayesian takes source from the Bayes' rule, which factorises the joint distribution into compact factors that encode the dependence between variables represented by the graph.

3.3.1 *Bayesian networks*

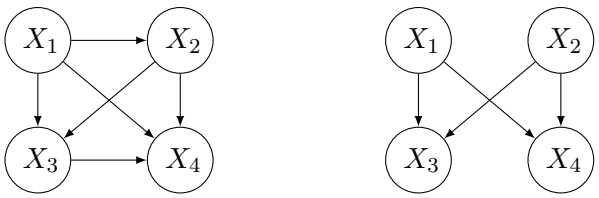


Figure 3.1: Two Bayesian networks of a 4D variable. (**a**) No independence. (**b**) A couple of independencies, hence a reduced number of edges and of parameters.

A Bayesian network is a directed acyclic graph (DAG) that represents independence assumptions between the components of a random vector. Formally, let $X = [X_1, \dots, X_d]^T$ be a random vector taking values $\boldsymbol{x} \in \mathcal{X}_1 \times \cdots \times \mathcal{X}_d$ distributed under $P(X)$. A BN for X has one vertex for each random variable X_i of X. In this DAG, the absence of edges models conditional independence between groups of components through the concept of d-separation [Geiger et al., 1990]. A BN is a valid representation of a random vector X iff its probability distribution (continuous or discrete) factorizes as

$$P(X) = \prod_{i=1}^{d} P(X_i \mid \mathcal{P}_i), \tag{3.1}$$

where $\mathcal{P}_i = \{X_j : A_{j,i} = 1\}$ denotes the set of parents of the vertex i and $A \in \{0,1\}^{d\times d}$ is the adjacency matrix of the BN. A BN, together with the related conditional probability distributions, is a PGM. For simplicity, we will use the term BN to refer to this couple and explicitly mention the terms topology or structure when talking about the BN's structure.

Figure 3.1a is a valid BN for any distribution over X as it does not state any independence and leads to the chain rule factorisation. However, in practice, we seek a sparse BN faithful to existing independencies in X. These sparse networks lead to an efficient factorisation of the modelled probability distributions. It is worth noting that making hypotheses on the graph structure is equivalent to assuming certain conditional independencies between some of the vector's components.

Understanding the independence assumptions underlying a DAG is key to appreciating BNs. For this purpose, d-separation [Geiger et al., 1990] describes rules to check whether certain conditional independence holds in all distributions that factorise over a DAG. Algorithm 3.1 describes this algorithm which allows checking whether the topology is well suited to model a phenomenon of interest. In addition, it also enables characterising all (conditional) independencies that follow from a set of distinct independence assumptions. D-separation is *sound*, it only detects existing independencies. However, it is not *complete* as it misses independence assumptions hidden in the parameterisation of conditional distributions. For example, the BN in Figure 3.1a can model any joint distribution for X; applying d-separation to this graph would reject all independence, even though the distribution modelled could satisfy some independence relations.

Algorithm 3.1 D-separation.

function IsIndependent(X, Y, Z, G)
▷ X, Y and Z are three sets of nodes from G.
▷ Return True iff $X \perp Y \mid Z$ is modelled by the Bayesian network with topology G.
  $A \leftarrow$ AllAncestorsOf(Z, G)
  for $P_i \neq P_j \in A$ **do** ▷ Getting rid of immoralities.
    if HasACommonChild(P_i, P_j, G) **then**
      $G \leftarrow$ AddUndirectedEdge(P_i, P_j, G) ▷ Marry parents.
    end if
  end for
  for $N \in Z$ **do**
    $G \leftarrow$ RemoveNode(N, G)
  end for
  return IsNotReachable(X, Y, G)
end function

3.3.2 *Parameterisation*

BNs reduce the description of a joint distribution into 1) a DAG and 2) conditional distributions. As mentioned earlier, learning aims at selecting (or weighting) within a class of models. A natural way to define a model class is to use parameters to describe the free parts of the model. For BNs, domain knowledge often prescribes the topology, while learning the conditional distributions from data is more common. Moreover, it is simpler to parameterise and learn (conditional) distribution than graph structures that lead to combinatorial problems.

For discrete variables, we use categorical distributions, and each conditioning factor corresponds to distinct parameter values in the worst case. The parameterisation of continuous variables distribution offers many alternatives. For instance, we can use an exponential family with linear functions that compute the natural parameters given the conditioning factors. Gaussians with linear interactions are arguably one of the most popular parameterisations. These models, dubbed Gaussian Bayesian networks, were historically the only ones with an efficient training algorithm as they correspond to multivariate Gaussian distributions [Wermuth, 1980] for which closed-form MLE exists. In Chapter 8, we introduce normalising flows as a more expressive parameterisation.

3.3.3 *Inference*

Almost any task we may want to perform on a model can be cast as inference. With no loss of generality, we focus on the generic the conditional probability query $P(Y \mid E = \boldsymbol{e})$, where Y denotes a subset of the model's variable and $\boldsymbol{e}$ is the value of another subset E of the variables, the remaining variables $M = \{X_i : X_i \notin Y \cup E\}$ are marginalised out. For example, we might be interested in evaluating $P(X_1 \mid X_3 = x_3, X_4 = x_4)$ in Figure 3.1.

3.3.3.1 *Exact inference*

As mentioned earlier, inference gets more complicated when the number of variables increases. However, a BN adds structure to this problem by making some of the independencies explicit. For BNs, inference stays NP-hard in the worst case. However, there exist algorithms able to exploit the structure for most inference tasks efficiently. For example, Pearl's message-passing algorithm [Pearl, 1987] is an efficient exact inference algorithm for polytrees, a subclass of BNs that have acyclic skeleton. Variable elimination (VE) is another popular algorithm that works on all discrete BNs and reduces the complexity of inference to the width (maximum distance between two nodes) of the graph. Sanner and Abbasnejad [2012] adapts VE to continuous variables with a symbolic version of VE. However, VE uses marginalisation and distribution products, which are not straightforward operations in the continuous setting.

3.3.3.2 *Inference as sampling*

The case of continuous variables motivates us to formulate inference differently. We look for formulations that do not explicitly mention marginalisation or the product of densities. A solution is to replace exact inference with conditional sampling from $P(Y \mid E = \boldsymbol{e})$. In BNs, *ancestral sampling* (AS) draws samples from the joint distribution by traversing the graph from parents to children and attributing a value to each node by sampling conditionally on the parents' values. AS can thus perform inference if the conditioning variables are ancestors of all variables in Y.

Ancestral sampling draws samples from each factor $P(X_i \mid \mathcal{P}_i)$ which is usually straightforward. For example, provided the corresponding cumulative distribution (CDF) function $F(\cdot; \mathcal{P}_i) : \mathcal{X}_i \to [0, 1]$ and a uniform random variable U in $[0, 1]$, $x_i = F^{-1}(u; \mathcal{P}_i)$ follows $P(X_i \mid \mathcal{P}_i)$. This method is known as *inversion sampling* (IS).

Rejection sampling (RS) is an alternative to inversion sampling, for example, when we cannot evaluate the inverse CDF directly. The idea is to sample $z \in \mathcal{X}_i$ from a proposal distribution P_z and u from a uniform between 0 and 1, then accept the sample if $u < \frac{P_{x_i}(z)}{KP_z(z)}$ where K is chosen such that $KP_z(x_i) \geq P_{x_i}(x_i) \quad \forall x_i \in \mathcal{X}_i$. RS works for multidimensional variables even when the joint distribution is only known up to a normalising factor. In that sense, it has more broadly applicable than IS. However, it becomes inefficient as the number of dimensions grows. Another limitation appears if we want to condition on a subset of the sampled variables, in which case the rejection rate blows up with the number of possible values for the conditioning variables (e.g., an infinite number in the continuous setting).

Another possibility called *likelihood weighting* (LW) consists in freezing the conditioning factors E to their value $\boldsymbol{e}$ and to sample values $(\boldsymbol{y}_1, \boldsymbol{m}_1), \ldots, (\boldsymbol{y}_k, \boldsymbol{m}_k)$ via AS. These samples does not follow $P(Y, M \mid E = \boldsymbol{e})$ however we may approximate the expected value of a function g with respect to $P(Y \mid E = \boldsymbol{e})$ by keeping track of the corresponding weights $P(Y, M, E = \boldsymbol{e})$ given by Equation (3.1). The approximation of the expected value takes the following form

$$\mathbb{E}_{P(Y|E=\boldsymbol{e})}\left[g(y)\right] \approx \frac{\sum_{i=1}^{k} g(\boldsymbol{y}_i) P(Y = \boldsymbol{y}_i, M = \boldsymbol{m}_i, E = \boldsymbol{e})}{\sum_{i=1}^{k} P(Y = \boldsymbol{y}_i, M = \boldsymbol{m}_i, E = \boldsymbol{e})}.$$

For discrete variables, LW provides an approximate alternative to VE as

$$P(Y = \boldsymbol{y} \mid E = \boldsymbol{e}) \approx \frac{\sum_{i=1}^{k} P(Y = \mathbf{1}\{y_i = y\}, M = \boldsymbol{m}_i, E = \boldsymbol{e})}{\sum_{i=1}^{k} P(Y = y_i, M = \boldsymbol{m}_i, E = \boldsymbol{e})}.$$

Likelihood weighting is a special case of *importance sampling* when the samples of the proposal distribution comes from ancestral sampling. Likelihood weighting is more efficient if the conditioning factors are close to the root of the BN.

Importance sampling does not provide iid samples from the posterior distribution, but MCMC (with independent resampling) does. MCMC is a family of algorithms that

samples from distributions known up to a normalising factors. Hence, MCMC algorithms are particularly interesting to sample from conditional distributions. These algorithms run a Markov chain whose stationary distribution follows the given distribution. Each MCMC algorithm defines its own transition probability $P(Y^k \mid Y^{k-1})$. Many instances exist, such as Metropolis-Hastings MCMC [Hastings, 1970], Hamiltonian MCMC [Neal et al., 2011], slice sampling [Neal, 2003], etc. Gibbs sampling [Sorensen et al., 1995] is an instance that exploits the BN structure in the transition probability. Let us suppose that M is empty, then at each transition step, Gibbs sampling only updates one component Y_i of the vector Y^k. It samples Y_i from the posterior $P(Y_i \mid Y_{-i}^{k-1})$ where Y_{-i}^{k-1} contains the values of the evidence E and the previous state Y^{k-1} except the i^{th} component.

3.3.3.3 *Inference as optimization*

We finally introduce variational inference [Blei et al., 2017, VI] as a third option for inference. VI formulates inference as an optimisation problem in which we look for a distribution $Q(Y)$ that is as close as possible to the posterior of interest $P(Y \mid E = \boldsymbol{e})$. For this purpose, we consider a parametric family of distributions $Q_{\boldsymbol{\theta}}(Y)$, e.g., a BN with a prescribed structure but parametric distributions. Our goal is to optimise for the set of parameters $\boldsymbol{\theta}^\star$ that minimise the discrepancy between the targetted and learnt distributions,

$$\boldsymbol{\theta}^\star = \arg\min_{\boldsymbol{\theta}} \mathbb{D}\left[Q_{\boldsymbol{\theta}}(Y) \| P(Y \mid E = \boldsymbol{e})\right], \tag{3.2}$$

where $\mathbb{D}$ is an appropriate divergence between distributions. For the rest of this discussion, let us fix $\mathbb{D}$ to the Kullback-Leibler ($\mathbb{KL}$) divergence which is an appropriate choice as it is only minimized when $Q = P$. As of now, we also restrain our discussion to the case of continuous variables. Formally, the $\mathbb{KL}$ compares two distributions $P(X)$ and $Q(X)$ as

$$\mathbb{KL}\left[P \| Q\right] \triangleq \int_{\boldsymbol{x} \in \mathcal{X}} P(\boldsymbol{x}) \log \frac{P(\boldsymbol{x})}{Q(\boldsymbol{x})} \mathrm{d}\boldsymbol{x}.$$

The integral can be replaced by a sum for discrete random variables. The $\mathbb{KL}$ is not a proper distance as it is not symmetric. Depending on the context, we will switch the two arguments P and Q, which is fine as in both cases Equation (3.2) is minimized iff the two arguments are equal.

Solving *Equation* (3.2) is possible if we have a parametric family of distributions for which density evaluation and sampling are differentiable functions of the parameters. This family could be another BN but also a normalizing flow, a deep probabilistic model we describe in the next chapter. Let us denote by $Q_{\boldsymbol{\theta}}$ the probability density function and the procedure to generate samples as

$$y := f_{\boldsymbol{\theta}}(z) \quad \text{where} \quad z \sim \mathcal{N}(0, I),$$

where $\mathcal{N}(0, I)$ denotes a normal distribution. In this ideal case, the optimization problem becomes:

$$\boldsymbol{\theta}^\star = \arg\min_{\boldsymbol{\theta}} \mathbb{KL}\left[Q_{\boldsymbol{\theta}}(Y) \| P(Y \mid E = \boldsymbol{e})\right] \tag{3.3}$$

$$= \arg\min_{\boldsymbol{\theta}} \mathbb{KL}\left[Q_{\boldsymbol{\theta}}(Y) \| P(Y, E = \boldsymbol{e})\right] + \mathbb{E}_{\boldsymbol{y}}\left[\log P(E = \boldsymbol{e})\right] \tag{3.4}$$

$$= \arg\min_{\boldsymbol{\theta}} \mathbb{E}_z\left[\log \frac{Q_{\boldsymbol{\theta}}(f_{\boldsymbol{\theta}}(z))}{P(f_{\boldsymbol{\theta}}(z), E = \boldsymbol{e})}\right]. \tag{3.5}$$

We can optimise this equation with stochastic gradient descent (SGD) by approximating the objective function with Monte Carlo samples at each gradient step.

Evaluating $P(f_{\boldsymbol{\theta}}(z), E = \boldsymbol{e})$ in the last equation is not always possible (e.g.; if the variables in M are not leaves). In this context, we may create an approximation $Q_{\boldsymbol{\theta}^\star}(Y, M \mid E = \boldsymbol{e})$ of $P(Y, M \mid E = \boldsymbol{e})$ instead. We can then generate samples from the joint and thus from the marginal $P(Y \mid E = \boldsymbol{e})$. If we need to evaluate the approximate

$$Q_{\boldsymbol{\theta}^\star}(Y \mid E = \boldsymbol{e}) := \int_{\boldsymbol{m} \in \mathcal{M}} Q_{\boldsymbol{\theta}^\star}(Y, M = \boldsymbol{m} \mid E = \boldsymbol{e}) \mathrm{d}\boldsymbol{m},$$

we can find a surrogate model $\tilde{Q}_\psi$ by solving the following optimization problem:

$$\psi^\star = \arg\min_{\psi} \mathbb{KL}\left[Q_{\boldsymbol{\theta}^\star}(Y, M \mid E = \boldsymbol{e}) \| \tilde{Q}_\psi(Y)\right]. \tag{3.6}$$

We can also optimize this equation with SGD and Monte Carlo samples as it only requires to sample from and evaluate $Q_{\boldsymbol{\theta}^\star}(Y, M \mid E = \boldsymbol{e})$, and to evaluate $Q_\psi(Y)$. We can convince ourselves that the solution to Equation (3.6) corresponds to $Q_{\boldsymbol{\theta}^\star}(Y \mid E = \boldsymbol{e})$,

$$\arg\min_{Q} \mathbb{KL}\left[P(Y, M) \| Q(Y)\right] \tag{3.7}$$

$$= \arg\min_{Q} \mathbb{E}_{P(Y)P(M|Y)}\left[\log P(Y) + \log P(M \mid Y) - \log Q(Y)\right] \tag{3.8}$$

$$= \arg\min_{Q} \mathbb{E}_{P(Y)P(M|Y)}\left[\log P(Y) - \log Q(Y)\right] \tag{3.9}$$

$$= \arg\min_{Q} \mathbb{E}_{P(Y)}\left[\log P(Y) - \log Q(Y)\right] \tag{3.10}$$

$$= \arg\min_{Q} \mathbb{KL}\left[P(Y) \| Q(Y)\right]. \tag{3.11}$$

We observe that VI can only achieve good results if the family of distributions contain instances close to the true posterior. In addition, solving the optimisation can be complicated even if provided with a parametric universal density approximator. However, VI is often preferable to sampling strategies for multiple reasons. It eventually becomes faster than sampling because Equation (3.2) returns a model from which we can generate as many samples as we want. This approach also automatically provides a surrogate of the

posterior density. Finally, we can apply this approach to learning a parametric posterior distribution that approximates $P(Y \mid E = \boldsymbol{e})$ for any value of e.

Another interesting application of VI is for estimating a bound on the log-marginal $\log P(E)$ of the larger model $P(Y, E)$. We can directly derive from Equation (3.4):

$$\mathbb{KL}\left[Q_{\boldsymbol{\theta}}(Y) \| P(Y \mid E = \boldsymbol{e})\right] \geq 0 \tag{3.12}$$

$$\mathbb{KL}\left[Q_{\boldsymbol{\theta}}(Y) \| P(Y, E = \boldsymbol{e})\right] + \mathbb{E}_y\left[\log P(E = \boldsymbol{e})\right] \geq 0 \tag{3.13}$$

$$\mathbb{E}_{\boldsymbol{y} \sim Q_{\boldsymbol{\theta}}(Y=\boldsymbol{y})}\left[\log \frac{P(E = \boldsymbol{e} \mid Y = \boldsymbol{y})P(Y = \boldsymbol{y})}{Q_{\boldsymbol{\theta}}(Y = \boldsymbol{y})}\right] \leq \log P(E = \boldsymbol{e}) \tag{3.14}$$

$$\underbrace{\mathbb{E}_{\boldsymbol{y} \sim Q_{\boldsymbol{\theta}}(Y=\boldsymbol{y})}\left[\log P(E = \boldsymbol{e} \mid Y = \boldsymbol{y})\right] - \mathbb{KL}\left[Q_{\boldsymbol{\theta}}(Y) \| P(Y)\right]}_{\text{ELBO}} \leq \log P(E = \boldsymbol{e}) \tag{3.15}$$

This estimated lower bound on the evidence (ELBO) is useful to approximate the log-likelihood of a parametric model $P_{\boldsymbol{\psi}}(E)$ defined as the marginal of a larger model $P_{\boldsymbol{\psi}}(Y, E)$ parameterised by $P_{\boldsymbol{\psi}}(Y \mid E)$ and $P_{\boldsymbol{\psi}}(E)$. VI allows learning deep probabilistic models that formulates the generative process as a stochastic map from latent variables to observations, as we will see in the next chapter.

3.3.4 *Learning*

With VI, we have seen for the second time that inference and learning are two faces of the same coin. We now describe learning strategies to fit the parameters of a BN given data. A BN combines a graph $G \in \mathcal{G}$, where $\mathcal{G}$ denotes the set of DAGs, and $\mathcal{F} = \{P_{\boldsymbol{\theta}_i}(X_i \mid \mathcal{P}_i)\}_{i=1}^d$ is the corresponding set of parametric conditional distributions that together model a joint distribution. Ideally, we would like to adapt both G and $f_{\boldsymbol{\theta}} \in \mathcal{F}$ to the learning dataset $\mathcal{D} = \{X^i\}_{i=1}^N$.

Assuming iid data, we can express the learning task as a generic optimisation problem:

$$B^\star = \arg\max_{B \in \mathcal{B}} \sum_{i=1}^{N} \log P_B(X^i) - R(B), \tag{3.16}$$

where $\mathcal{B} = \mathcal{G} \times \mathcal{F}$ denotes the set of possible BNs and $R(\cdot) : \mathcal{G} \times \mathcal{F} \to \mathbb{R}$ is a regularisation term which can be interpreted as the prior over plausible BNs and is constant if we do MLE. We insist on the importance of $R(B)$ for obtaining a good model. Although BNs with complete DAGs are strictly more expressive than sparse structures, we prefer the latter as they explicit independence assumptions and lead to simpler conditional distributions.

The formulation in Equation (3.16) does not say much about concrete strategies to fit the parameters of a BN. In practice, we often separate structure learning from distribution fitting. Structure learning is a combinatorial problem, whereas fitting the parameters of conditional distributions is a continuous optimisation problem; solving both issues jointly remains difficult. In Chapter 8, we will see one strategy to formulate the combinatorial

topology learning problem into a continuous one; these are advanced concepts beyond this background's scope. Now, we focus on the two sub-problems independently.

3.3.4.1 *Distribution learning*

We suppose a prescribed topology and only focus on learning the best factors in Equation (3.1). For this purpose we can parameterize the factors with a vector $\boldsymbol{\theta} := \{\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_d\}$ and update Equation (3.16) into

$$\boldsymbol{\theta}^\star = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{N} \log \prod_{j=1}^{d} P_{\boldsymbol{\theta}_j}(X_j^i \mid \mathcal{P}_j) + \log \pi(\boldsymbol{\theta}). \tag{3.17}$$

In practice, we approach $\boldsymbol{\theta}^\star$ via stochastic gradient ascent on the objective of Equation (3.17). The choice of the parameterisation and the prior π are crucial to finding a good model when N is finite.

3.3.4.2 *Structure learning*

It is sometimes difficult to define a relevant structure a priori, and it may be interesting to learn it from data instead. Structure learning aims to find a sparse topology that does not imply (conditional) independence in contradiction with the data. We can formulate this search as an optimisation problem under constraints in the space of DAGs $\mathcal{G}$:

$$G^\star = \arg\min_{G \in \mathcal{G}} \sum_{i=1}^{d} \sum_{j=1}^{d} A_{i,j}(G) \quad \text{such that} \quad I(G) \subseteq I(\mathcal{D}), \tag{3.18}$$

where $A(G)$ is the adjacency matrix of the graph G, $I(G)$ is the set of (conditional) independence encoded by G, and $I(\mathcal{D})$ is the set of independence observed in the data. The constraints ensure that only independencies observed in the data are encoded in the structure, and the objective penalizes the number of edges.

Fixing the topological ordering in advance reduces drastically the size of the search space and allows greedy algorithms. In this case, the solution's quality depends on the chosen order. In addition to being intractable, the formulation in Equation (3.18) does not take advantage the prior knowledge. A partial solution is to reduce the search space $\mathcal{G}$ to the structures we consider plausible. Expressing more subtle priors over graphs and handling them efficiently in structure learning is still an active research question.

The evaluation of $I(\mathcal{D})$ is essential for structure learning. In theory, testing for independence in the data is difficult as it requires accessing the (conditional) distribution between the two variables tested, which is what we are trying to learn. In practice, we create statistical independence tests by making assumptions on the class of interactions and marginal distributions (e.g. linear Gaussian, discretisation of variables, etc.). These tests are reasonably accurate for unconditional independence but not for conditional independence. Interestingly, while pure independence between continuous variables is hard

to prove in practice, it is often beneficial to consider weakly dependent variables as if they were independent. This usually simplifies the model and improves its generalisation to unseen data.

In parallel to this, there exist specialized algorithms for learning BNs' structures. Some implement greedy solutions which provide a useful approximation of the true solution [Tsamardinos et al., 2006]. Others focus on sub-class of BNs for which efficient solutions exist [Cooper and Herskovits, 1992; Chow and Liu, 1968]. In general, topology learning is hard. It has recently regained great attention from the machine learning community in the context of causal discovery [Khemakhem et al., 2020; Balgi et al., 2022a; Vowels et al., 2021; Brouillard et al., 2020].

3.3.5 *Duality between directed graphs and distributions*

If a distribution P factorises over a Bayesian network with graph G, we can check whether some conditional independence holds with d-separation. In this case, we say that the Bayesian network is an *I-map* of the independence in P – any independence expressed by the BN is present in P. However, we have also observed that any distribution factorises over the complete graph. Complete Bayesian networks are unsatisfactory as they do not reflect any independence assumptions necessary for manipulating the distribution efficiently. For example, Figure 3.1a depicts a BN that is an I-map for any distribution over the four variables X_1, X_2, X_3, X_4. In contrast, Figure 3.1b is an I-map over a distribution over these variables only if it factorises under the ancestral ordering of the network, which implies some independencies such as $X_1 \mid X_2$.

Our goal is thus to find a sparse graph that only represents independence present in the distribution we aim to model. In particular, we say that a graph G is a *minimal I-map* for a set of independencies if it is an I-map and if the removal of even a single edge breaks the I-map property. We are generally only interested in these minimal I-map. For example, if the independencies modelled by Figure 3.1b holds in the distribution, we will prefer it over Figure 3.1a. Nevertheless, *minimal I-map* are not perfect because they can miss some independencies in P. When possible, we aim to find a graph G that is a *P-map* for a set of independencies $\mathcal{I}$ present in the distribution we aim to model. It is the case if the independence modelled by the graphs $\mathcal{I}_G$ are equal to $\mathcal{I}$ – that is, any independence modelled by G is in $\mathcal{I}$, and all independencies in $\mathcal{I}$ are modelled by G.

Clearly, any P-map is a minimal I-map, and the best BNs are the ones that are P-maps of the distribution of interest. Unfortunately, Bayesian networks are not universal P-maps. Some sets of independencies $\mathcal{I}$ are impossible to express faithfully with directed graphs. In such a case, either the graph misses some independence or expresses false independence. We will see that undirected graphs can faithfully represent such independencies but have other limitations and are not universal P-maps either.

3.3.6 *Causality*

It is natural to interpret the arrows in a BN as causal relationships between variables. This interpretation is dangerous as it is not necessarily correct. For example, two complete BNs with opposed arrows may express the same distribution but opposite causal relationships. However, the reverse interpretation is ok – causal graphs are BNs. In these graphs, a directed edge from X to Y represents the causation of Y by X. The BN corresponding to a causal graph is an I-map of the distributions between the variables present in the causal graph.

We can recycle algorithms and interpretation from BNs to causal graphs but not vice-versa. We can also try to understand some algorithms by assuming the BN is a valid causal graph. For example, causality implies that we shall first sample parents to generate the children and provides a natural interpretation of the ancestral sampling algorithm. As we will see in the next section, some (in)dependence assumptions are inherently non-causal, which hints at why BNs are not universal P-maps.

Causality has become a significant sub-field of artificial intelligence and strongly impacts machine learning. Do-calculus [Pearl, 1994] is a tool that allows inference with causal rather than probabilistic facts. Such reasoning patterns are necessary to answer counterfactual questions scientifically [Morgan and Winship, 2015]. In addition, causal assumptions prove the generalisation and robustness of some machine learning algorithms. The recent interest in causal modelling provides additional motivations for using directed graphs, hence BNs, to model distributions.

3.4 UNDIRECTED GRAPHICAL MODELS – MARKOV NETWORKS

The duality between Bayesian and causal networks provides a simple procedure to constrain the structure of the networks when we have a causal understanding of the studied process. However, the duality also implies that BNs are not universal P-maps, as shown in the following example. Let X_1, X_2, X_3, X_4 be four random variables related by the following independence statements: $\{X_1 \perp X_3 \mid (X_2, X_4), X_2 \perp X_4 \mid (X_1, X_3)\}$. A Bayesian network cannot faithfully represent such independence statements. Indeed the assumptions impose that X_2 and X_4 d-separate X_1 and X_3 and vice versa. This imposes that the structure of the BN resembles Figure 3.2b. However, it is impossible to add directions to the edges without adding a cycle or a V-structure that would contradict at least one of the independence hypotheses.

This example is even more straightforward if we take a concrete example where such assumptions would be reasonable and observe that it contradicts any causal interpretation of the BN. Let us consider that the four random variables correspond to the opinion of four wine amateurs about a *Côtes de Provence rosé*. Each amateur tastes the wine twice, on two different days, with two distinct amateurs. As they discuss together, the opinion of each amateur is influenced by the view of the others and vice versa. Because

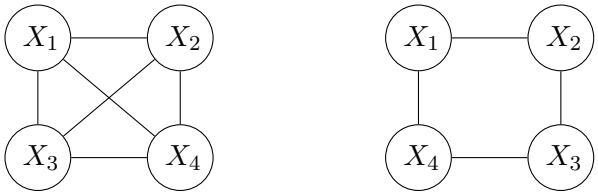


Figure 3.2: Two Markov networks of a 4D variable. (**a**) No independence. (**b**) Cycle dependencies. A Bayesian network cannot represent all implied independencies but a Markov network can.

we expect each pair to reach some consensus, all opinions should influence each other, sometimes indirectly. Each amateur will have a final opinion about the wine eventually. This contradicts a causal interpretation in which at least one of the variables, a root, should not be influenced by another.

It sounds more natural to express similar configurations with an undirected graph. Undirected graphical models can indeed represent such conditional independence faithfully. However, undirected graphs are not universal P-maps either, as they cannot represent independence assumptions rooted in causal interpretation. For example, it is impossible to express that two independent variables can become dependent when conditioned on a third variable (V-structure in BNs) which naturally arises when the two independent variables cause the third. Nevertheless, undirected representations sometimes handle non-causal assumptions more naturally than Bayesian networks. We briefly introduce these models, united under the term Markov networks. Our discussion is intentionally more superficial than for directed representations, as Markov networks are not directly related to any of the contributions in this thesis.

3.4.1 *Markov networks*

A Markov network (MN), also called Markov Random Field, is an undirected graph that describes *Markov properties* between random variables. Formally, let $X = [X_1, \dots, X_d]^T$ be a vector collecting d random variables. The global Markov property states that any two subsets of variables are conditionally independent given a separating subset. In mathematical terms: $X_A \perp X_B \mid X_S$, where the subset of variables X_S separates the subsets X_A and X_B; X_S blocks all path from X_A to X_B in the graph. For example, the Markov network in Figure 3.2a does not impose any independence whereas the one in Figure 3.2b implies the following independence: $\{X_1 \perp X_3 \mid (X_2, X_4), X_2 \perp X_4 \mid (X_1, X_3)\}$.

X_1 \ X_2	0	1
0	10	30
1	50	10

(a)

X_1 \ X_2	0	1
0	0.1	0.3
1	0.5	0.1

(b)

Table 3.1: The numerical values associated with a two nodes (X_1 and X_2) Markov network. (**a**) The unnormalised factor. (**b**) The factor normalised corresponds to the joint distribution.

3.4.2 *Parameterisation.*

The unidirectionality of Markov networks imposes some symmetry in their parameterisation. In contrast to BNs, the numerical values cannot represent conditional distributions which would break the symmetry of the undirected relationship between two nodes ($P(A \mid B) \neq P(B \mid A)$). An alternative parameterisation could be to encode all the 2D joint distributions. Unfortunately, this would be impractical. Let us suppose we parameterise the wine-amateurs Markov network in Figure 3.2b with the corresponding 2D joint distributions ($P(X_1, X_2), P(X_2, X_3), P(X_3, X_4), P(X_1, X_4)$). Satisfying the Kolmogorov's axioms with such a parameterisation is a real challenge; e.g., it would imply that $P(X_1) = \sum_{x_2} P(X_1, X_2 = x_2)$ must be equal to $P(X_1) = \sum_{x_4} P(X_1, X_4 = x_4)$. It is unclear how we could practically ensure this equality with a simple parameterisation.

It is why instead Markov networks rely on **unormalised** functions called *factors* for their parameterisation. A factor is a real-valued function $\phi(\cdot) : \mathcal{X} \to \mathbb{R}$ that describes the interactions between a set of random variables X. We can combine multiple factors to create a joint distribution. This parameterisation is called a Gibbs distribution. It parameterises the joint distribution of a random vector $X = [X_1, \ldots, X_d]^T$ with a set of factors $\boldsymbol{\phi} = \{\phi_1(D_1), \ldots, \phi_K(D_K)\}$. Each D_i is a, potentially overlapping, different subset of variables. The Gibbs distribution is defined as follows:

$$P_{\boldsymbol{\phi}}(X) = \frac{1}{Z}\tilde{P}_{\boldsymbol{\phi}}(X),$$

where

$$\tilde{P}_{\boldsymbol{\phi}}(X) = \phi_1(D_1) \times \cdots \times \phi_K(D_K).$$

The normalising factor is $Z = \sum_{\boldsymbol{x} \in \mathcal{X}} \tilde{P}_{\boldsymbol{\phi}}(X = \boldsymbol{x})$ if the random vector is discrete or $Z = \int_{x \in \mathcal{X}} \tilde{P}_{\boldsymbol{\phi}}(X = \boldsymbol{x}) \mathrm{d}\boldsymbol{x}$ if it is continuous.

We sometimes write the unnormalized probability distribution as the negative exponential of an energy function $E(\cdot) : \mathcal{X} \to \mathbb{R}$, $\tilde{P}_{\boldsymbol{\phi}}(X) = e^{-E_{\boldsymbol{\phi}}(X)}$. The denomination energy comes from physics which describes the probability of observing a system in a given state as a Gibbs distribution parameterised by the state's energy (and temperature) [Gibbs, 1902].

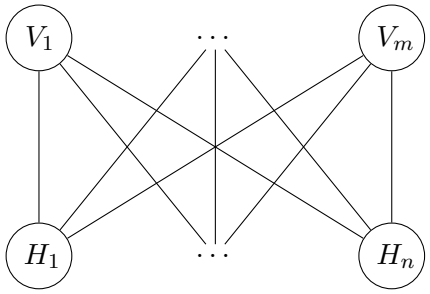


Figure 3.3: A Restricted Boltzmann machines compose of n hidden units H_i and m visible units V_j. The Markov network is a bipartite graph separating hidden and visible units into two groups fully connected to each other.

In BNs, the ancestral factorisation ensures that the parameterisation respects the independencies modelled by the graph structure. Similarly, a Gibbs distribution respects the independence of a Markov network if the factors only take a subset of variables that are complete network subgraphs. For example, we can parameterise the wine-amateur joint distribution with four 2D factors $\{\phi_k(\cdot,\cdot) : \{0,1\} \times \{0,1\} \to \mathbb{R}\}_{k=1}^4$ that encode the interactions between the four amateurs, e.g. as the one in Table 3.1a. In addition, we could also use 1D factors. However, the parameterisation with maximal complete subgraphs, also called maximal clique potentials, is sufficiently expressive to encode the marginals over subsets. Thus we can stick to the 2D factors in this example and, in general, parameterise the maximal clique potentials only.

Parameterising Markov networks with factors over maximal clique potentials may obscure the structure in the lower-dimensional original set of factors. As the number of nodes grows, the graph's complexity and the maximal clique's size grow. For discrete variables, the number of entries in the table representing the factor grows exponentially with the dimensionality of the input vector. Hence we usually try to minimise the number of edges in the graph, or we only parameterise 2D factors and ignore higher-order interactions.

3.4.3 *Toward neural networks*

Provided the graph structure and factors, we can use sampling algorithms, such as MCMC algorithms, to generate realisations. However, without additional constraints on the structure manipulating these models is difficult. Restricted Boltzmann machines [Hinton, 2002, RBMs] are a popular class of Markov networks where binary variables are split into visible units, denoted $V \in \{0,1\}^m$, and hidden units, $H \in \{0,1\}^n$, with a bipartite graph as depicted in Figure 3.3. This restriction allows efficient learning algorithms inspired by the Hebbian plasticity of the brain. The parameterisation of RBMs is a matrix of weights $W \in \mathbb{R}^{m \times n}$ that describes the interactions between visible and hidden units and two vec-

tors of offset $\boldsymbol{b}_H$ and $\boldsymbol{b}_V$ respectively for hidden and visible units. The joint distribution between visible and hidden units is computed as

$$P(V = \boldsymbol{v}, H = \boldsymbol{h}) = \frac{1}{Z}\phi(V = \boldsymbol{v}, H = \boldsymbol{h}),$$

where the factor is $\phi(V, H) = e^{-E(V,H)}$ and the energy function takes the particular form

$$E(V = \boldsymbol{v}, H = \boldsymbol{h}) = -\boldsymbol{b}_H^T\boldsymbol{h} - \boldsymbol{b}_V^T\boldsymbol{v} - \boldsymbol{v}^T W\boldsymbol{h}.$$

We can train these networks on a dataset of visible variables with an algorithm that combines Gibbs sampling (on the hidden units) and gradient descent to update the weights. RBMs are among the first success of neural networks. They have been used for unsupervised learning problems such as dimensionality reduction, collaborative filtering, and others. RBMs naturally transition between PGMs and deep probabilistic models we will discuss in the next chapter.

3.5 CONCLUSION

In this chapter, we have discussed motivations for a graphical representation of probabilistic models. We have introduced Bayesian and Markov networks as the main classes of probabilistic graphical models. In addition, we have presented various inference algorithms. Some algorithms apply to all probabilistic models, while others exploit the graphical structure. Undoubtedly, this chapter has outlooked the parameterisation of continuous distributions, even when they are unidimensional. Our discussion ended with a preamble on neural networks. The next chapter goes further and details various strategies to parameterise probability distributions with neural networks.

The basic laws of the universe are simple, but because our senses are limited, we can't grasp them. There is a pattern in creation.

Albert Einstein

4

DEEP PROBABILISTIC MODELS

Outline

This chapter comprehensively reviews various strategies to parameterise probabilistic models with neural networks. We present the main algorithms used in practice, and we discuss the pros and cons of each class of models. The distinction between two models is sometimes artificial. It is why, we highlight relevant connections between these strategies when possible. Our aim is to convince the reader of the effectiveness and flexibility of deep probabilistic models. Finally, we provide the primary motivations for the contributions presented in this thesis.

4.1 INTRODUCTION

Restricted Botzmann Models [Hinton, 2002] are arguably one of the first generative models built upon neural networks. This chapter discusses other successful approaches for parameterising probabilistic models with neural networks. While graphical models describe probabilistic models with graphs and are appealing for understanding and prescribing modelling assumptions, they do not rigorously describe efficient parameterisations of the conditional probability distributions in the continuous setting. Deep probabilistic models encompass methods for parameterising these probability distributions with neural networks. We call some of these models *deep generative models* when they are well suited for sample generation. We will highlight the correspondence of some of these models within the probabilistic graphical models when possible.

4.2 WHY NEURAL NETWORKS?

We abstract neural networks as parametric functions that map an input space $\mathcal{X}$ to an output space $\mathcal{Y}$ and are differentiable functions $f_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Y}$ of their parameters $\boldsymbol{\theta}$. Moreover, backpropagation allows computating of the first-order derivative $\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}$ of neural networks automatically. Although neural networks are universal function approximators, the parameterisation of a probability distribution is challenging and requires the right architectural choices. We often present neural networks as deterministic models; however, many strategies exist to parameterise probability distributions, as we will see in the next sections.

For example, normalizing flows are neural networks that directly parameterise the probability distribution but require strong architectural constraints, as we will see later. This is particularly inconvenient when working with structured data (e.g. images, sound, graphs, etc.) because specialised architectures (e.g. convolutional networks [LeCun et al., 1995]) do not directly satisfy the architectural constraints. Another possibility is to use a neural network to parameterise a known distribution such as a Gaussian. For example, we can model $P(Y|X)$ as a multivariate Gaussian $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ where a free-form neural network computes the mean vector and covariance matrix as a parametric function of the input X. This strategy makes a strong assumption on the form of $P(Y \mid X)$, which may be inaccurate, e.g. if X is a piece of text describing the image Y. An alternative is to see the neural network as a generator of samples similar to the training set, mapping a noise vector to realisations [Goodfellow et al., 2020]. These deep generative models define the probability distribution implicitly and allow us to use complex architectures well suited to the modelling task. However, they also require specialised training algorithms.

Neural networks are better suited to parameterise distributions than other machine learning models. In particular, their differentiability and training procedure based on stochastic gradient descent allows us to optimise objectives corresponding to the MLE or MAP directly. In parallel, considerable efforts have been made over the last year to create architectures with inductive biases adapted to different data types, e.g. graph neural networks [Xu et al., 2018; Satorras et al., 2021] and recurrent neural networks [Hochreiter and Schmidhuber, 1997; Cho et al., 2014]. The availability of large pre-trained models is another advantage of working with neural networks [Bommasani et al., 2021].

In the following we ignore the conditioning variable X. Indeed, the main challenges in modelling distributions with neural networks is usually orthogonal to handling conditioning variables. Instead we focus on the modelling of the joint distributions of a d-dimensional random vector $Y = [Y_1, \dots, Y_d]$ that takes value in $\mathcal{Y} = \mathcal{Y}_1 \times \cdots \times \mathcal{Y}_d$. For simplicity, we consider that $\mathcal{Y}_1 = \cdots = \mathcal{Y}_d$ and the domains are either discrete classes (e.g. a pixel value in $\{0, \dots, 255\}^3$) or real numbers $\mathbb{R}$.

4.3 AUTOREGRESSIVE MODELS

Autoregressive models [Graves, 2013; Germain et al., 2015; Van den Oord et al., 2016; Wong and Li, 2000] reduce the modelling of d-dimensional distributions into d distributions via the product rule ($P(A, B) = P(A)P(B \mid A)$) as shown in Figure 4.1. The autoregressive structure is there to account for the piling-up conditioning factors. These models can represent all kinds of dependencies between the components of the random vector modelled. Formally, autoregressive models decompose the joint distribution as

$$P(Y) = P(Y_1)\Pi_{i=2}^{d} P(Y_i|Y_{<i}), \quad \text{where} \quad Y_{<i} \triangleq [Y_1, \dots, Y_{i-1}]\,. \tag{4.1}$$

When the domain is discrete, we can parameterise each 1D factor with a classifier. If the domain is continuous, the common practice is to parameterise a 1D Gaussian distribution

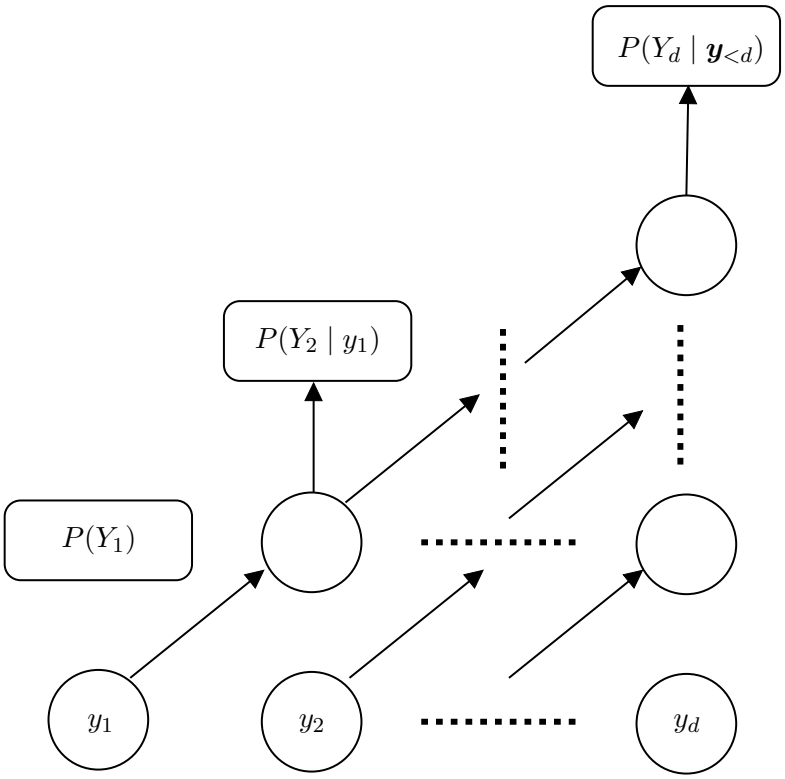


Figure 4.1: The forward computation of an autoregressive model. Each 1D distribution is conditioned autoregressively on the input vector.

with a neural network that predicts the mean and log-variance given the conditioning vector. Mixture density networks [Bishop, 1994, MDNs] are an effective alternative that easily accounts for multimodality. MDNs parameterise a mixture of m $1D$ distributions (e.g., Gaussian distributions) with a neural network that assigns a normalised weight to each mixture component, in addition to their parameters (e.g. the mean and log-variance). Autoregressive models involve the parameterization of d functions with different input dimensionalities which might be very inefficient if d is large (e.g. 2^{16} for 256×256 images).

Sharing parameterisation between factors is an important aspect of autoregressive models. One strategy is to use recurrent neural networks where the hidden vector carries and updates the information of the successive conditioning variables[Van Oord et al., 2016]. However, this strategy is slow because it processes the input vector sequentially. Moreover, the inductive bias of recurrent neural networks is not ideal for signals with long-range or spatial dependencies such as audio or images. A better solution is to use causal convolutional neural networks [Oord et al., 2016a, Causal CNNs], which process the complete vector simultaneously and enforce the autoregressive structure in the outputs. Causal CNNs have achieved great success in modelling audio [Van Den Oord et al., 2016, 2018]. The PixelCNN autoregressive model [Oord et al., 2016b] generalises this architecture to 2D signals and has shown great modelling capabilities.

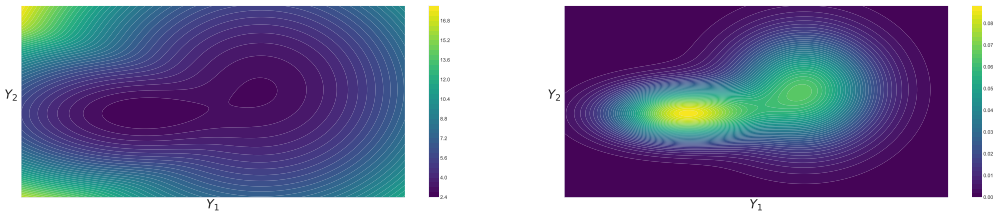
We can train autoregressive models efficiently by maximising their likelihood given data. Evaluating the likelihood only requires passing the data once in the network and computing the product of the d conditional distribution evaluated at the input vectors. Unfortunately, sampling is a sequential process as each variable is sampled according to the preceding. Sampling from of autoregressive models is slow for images or long time series and is one of their main drawbacks. The lack of independence assumptions and poor inductive bias are other limitations of these models. This is especially true with images for which the enforced causality between successive pixels and sampling algorithm is ineffective. We rarely use autoregressive models alone but in combination with other models, such as variational auto-encoders [Kingma and Welling, 2013].

4.4 ENERGY BASED MODELS

We have already encountered the term *energy* when discussing the parameterisation of Markov networks. There, and in the context of energy-based models [Teh et al., 2003, EBMs], the energy relates to the definition of a Gibbs distribution, taking the form

$$P_{\boldsymbol{\theta}}(Y = \boldsymbol{y}) = \frac{1}{Z(\boldsymbol{\theta})} e^{-E_{\boldsymbol{\theta}}(\boldsymbol{y})}, \tag{4.2}$$

where $E_{\boldsymbol{\theta}}(\cdot) : \mathcal{Y} \to \mathbb{R}$ is the energy function and $Z(\boldsymbol{\theta})$ is called the partition function. Figure 4.2a shows the energy function of a 2D distribution with two modes, the integral over the support is not equal to 1, but the energy is proportional to the logarithm of the probability density function.



(a) (b)

Figure 4.2: The energy landscape of a 2D bi-modal distribution (**a**) and the corresponding probability density function (**b**).

The interest of EBMs is to parameterise probability distributions with free-form real functions. For instance, any neural network can potentially parameterise an EBM. This property is very appealing as it allows combining the flexibility and efficient inductive bias of neural networks, which was fundamental reasons for the greatest successes of deep learning. Unfortunately, this flexible parameterisation prevents direct access to the likelihood function, which requires the intractable computation of the partition function. This prevents us from directly using the MLE learning strategy. Inspired by the recent review paper on EBMs by Song and Kingma [2021], we now review the main approaches for learning EBMs for continuous variables.

4.4.1 *Markov chain Monte Carlo*

In order to find the best set of parameters $\boldsymbol{\theta}$, we need to evaluate $P_{\boldsymbol{\theta}}$ and potentially its derivative, not only the energy function. A natural strategy would be to compute the integral $Z(\boldsymbol{\theta}) = \int_{y \in \mathcal{Y}} e^{-E_{\boldsymbol{\theta}}(\boldsymbol{y})} \mathrm{d}y$. However, as soon as the energy function is non-trivial, this integral becomes intractable. Thus, we need to find a better strategy. We see below how sampling from the EBM can save us the pain of explicitly computing $Z(\boldsymbol{\theta})$.

Let us formalise the learning problem for a given dataset $\mathcal{D} := \{\boldsymbol{y}_i\}_{i=1}^N$ of N iid samples. A reasonable strategy is to find the MLE,

$$\theta^\star = \arg\max_\theta P_{\boldsymbol{\theta}}(\mathcal{D}) \tag{4.3}$$

$$= \arg\max_\theta \log P_{\boldsymbol{\theta}}(\mathcal{D}) \tag{4.4}$$

$$= \arg\max_\theta \frac{1}{N} \sum_{i=1}^N \log P_{\boldsymbol{\theta}}(Y = \boldsymbol{y}_i). \tag{4.5}$$

In practice we cannot solve this last equation directly and we would like to use standard optimization techniques such as (stochastic) gradient ascent [Amari, 1993; Bottou, 2012].

In this case, learning the EBMs eventually reduces to evaluating $\nabla_{\boldsymbol{\theta}} \log P_{\boldsymbol{\theta}}(Y = \boldsymbol{y}_i)$ for any $\boldsymbol{y}_i \in \mathcal{D}$. Unfortunately, this gradient explicitly depends on the parition function,

$$\nabla_{\boldsymbol{\theta}} \log P_{\boldsymbol{\theta}}(\boldsymbol{y}) = \nabla_{\boldsymbol{\theta}} \log Z(\boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\boldsymbol{y}). \tag{4.6}$$

We can obtain the first term with automatic differentiation. In contrast, the second term requires computing the gradient of an intractable integral and requires additional tricks to be evaluated.

However, we can rewrite the problematic term as an expectation over samples from the EBMs:

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} \log Z(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} \log \int_{\boldsymbol{y} \in \mathcal{Y}} e^{-E_{\boldsymbol{\theta}}(\boldsymbol{y})} \mathrm{d}\boldsymbol{y} && (4.7)\\
&= \left(\int_{\boldsymbol{y} \in \mathcal{Y}} e^{-E_{\boldsymbol{\theta}}(\boldsymbol{y})} \mathrm{d}\boldsymbol{y} \right)^{-1} \int_{\boldsymbol{y} \in \mathcal{Y}} \nabla_{\boldsymbol{\theta}} \left(e^{-E_{\boldsymbol{\theta}}(\boldsymbol{y})} \right) \mathrm{d}\boldsymbol{y} && (4.8)\\
&= \int_{\boldsymbol{y} \in \mathcal{Y}} \underbrace{\left(\int_{\boldsymbol{y} \in \mathcal{Y}} e^{-E_{\boldsymbol{\theta}}(\boldsymbol{y})} \mathrm{d}\boldsymbol{y} \right)^{-1} e^{-E_{\boldsymbol{\theta}}(\boldsymbol{y})}}_{= P_{\boldsymbol{\theta}}(\boldsymbol{y})} \nabla_{\boldsymbol{\theta}} \left(-E_{\boldsymbol{\theta}}(\boldsymbol{y}) \right) \mathrm{d}\boldsymbol{y} && (4.9)\\
&= -\mathbb{E}_{P_{\boldsymbol{\theta}}(\boldsymbol{y})} \left[\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\boldsymbol{y}) \right]. && (4.10)
\end{aligned}$$

Then, we can use MCMC to sample from the EBM and estimate the gradient of the log-likelihood function. There exist MCMC algorithms, such as Langevin MCMC [Parisi, 1981; Grenander and Miller, 1994] or Hamiltonian Monte Carlo [Duane et al., 1987; Neal et al., 2011], that are specifically designed to exploit the fact that the score function $\nabla_{\boldsymbol{y}} \log P_{\boldsymbol{\theta}}(\boldsymbol{y}) = -\nabla_{\boldsymbol{y}} E_{\boldsymbol{\theta}}(\boldsymbol{y})$ is directly accessible. These algorithms are usually more efficient than Metropolis-Hastings MCMC [Hastings, 1970].

Running MCMC until convergence is expensive. Hence, the requirement to do it at each gradient step leads to inefficient learning algorithms. Many improvements to the naive learning strategy, which starts each MCMC run from scratch, have been proposed. For example, contrastive divergence [Hinton, 2002] starts each chain from a data point rather than a random initial state. A similar strategy named persistent contrastive divergence [Tieleman, 2008] use persistent states between successive gradient estimations and update steps.

4.4.2 *Contrastive learning*

Sampling is inefficient in highly-dimensional spaces or even for low-dimensional that are multimodal. Although strategies to improve the efficiency of MCMC-based training exists, it often leads to biased gradient estimates [Fischer and Igel, 2011]. This observation has motivated the development of alternative learning strategies. For instance, Gutmann and Hyvärinen [2012] proposed noise contrastive learning, which jointly estimates the partition and energy functions. Gutmann and Hyvärinen [2012] notice that if the energy

function models correctly the training distribution, it enables discriminating between noise and samples from the distribution.

Let us consider a balanced binary classification problem where samples from class $C = 0$ follow a known noise distribution P_n. The positive class $C = 1$ is the training set and it follows the unknown distribution $P(Y)$. As long as the noise and the training distributions' supports match, we can learn a Bayes optimal classifier as a function of the ratio between the two densities [Devroye et al., 2013; Faragó and Lugosi, 1993]. This classifier attributes the posterior probability to C given a sample $\boldsymbol{y}$ as

$$P(C = 0 \mid \boldsymbol{y}) = \frac{P_n(\boldsymbol{y})}{P_n(\boldsymbol{y}) + P(\boldsymbol{y})}. \tag{4.11}$$

We can replace $P(\boldsymbol{y})$ by the EBM $P_{\boldsymbol{\theta}}(\boldsymbol{y})$ and solve the corresponding classification problem as a function of the EBM's parameters $\boldsymbol{\theta}$ and of the normalizing constant $\alpha \triangleq Z(\boldsymbol{\theta})$.

Formally we aim to recover the Bayes optimal classifier, that is, for all $\boldsymbol{y}$ to be able to return the posterior distribution $P(C \mid \boldsymbol{y})$. This objective gives the following optimisation problem:

$$\begin{aligned}
\boldsymbol{\theta}^\star &= \arg\min_{\boldsymbol{\theta}} \mathbb{KL}\left[P(C \mid \boldsymbol{y}) \| P_{\boldsymbol{\theta}}(C \mid \boldsymbol{y})\right] \quad \forall \boldsymbol{y} \in \mathcal{Y} && (4.12)\\
&= \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{P(\boldsymbol{y})} \mathbb{KL}\left[P(C \mid \boldsymbol{y}) \| P_{\boldsymbol{\theta}}(C \mid \boldsymbol{y})\right] && (4.13)\\
&= \arg\min_{\boldsymbol{\theta}} -\mathbb{E}_{P(\boldsymbol{y})} \mathbb{E}_{P(C|\boldsymbol{y})} \left[\log P_{\boldsymbol{\theta}}(C \mid \boldsymbol{y})\right] + C && (4.14)\\
&= \arg\min_{\boldsymbol{\theta}} -\mathbb{E}_{P(C)} \mathbb{E}_{P(\boldsymbol{y}|C)} \left[\log P_{\boldsymbol{\theta}}(C \mid \boldsymbol{y})\right] && (4.15)\\
&= \arg\min_{\boldsymbol{\theta}} -\mathbb{E}_{P_n(\boldsymbol{y})} \left[\log \left(\frac{P_n(\boldsymbol{y})}{P_n(\boldsymbol{y}) + P_{\boldsymbol{\theta}}(\boldsymbol{y})}\right)\right] - \mathbb{E}_{P(\boldsymbol{y})} \left[\log \left(\frac{P_{\boldsymbol{\theta}}(\boldsymbol{y})}{P_n(\boldsymbol{y}) + P_{\boldsymbol{\theta}}(\boldsymbol{y})}\right)\right]. && (4.16)
\end{aligned}$$

The last term is the standard cross-entropy loss applied to a classifier that resembles Equation (4.11) parameterised by an EBM.

Then learning corresponds to minimising a cross-entropy loss of a classifier expressed as a function of the EBM. It will eventually lead to the correct EBM if the classifier gets optimal. In practice, this strategy is sensitive to the distribution of noise. Indeed, the cross-entropy loss is more sensitive to errors on $P_{\boldsymbol{\theta}}(\boldsymbol{y})$ in regions where the data and the noise distributions overlap. Intuitively, in the low-density regions of the noise, the classification is simple because $P_n(\boldsymbol{y})$ is very small. Hence the sensitivity of Equation (4.11) to $P_{\boldsymbol{\theta}}(\boldsymbol{y})$ is small. In contrast, in high-density regions of both the noise and data distributions, the discriminator is very sensitive to errors in $P_{\boldsymbol{\theta}}(\boldsymbol{y})$.

The sensitivity of contrastive learning to the noise distribution restricts its applicability. For instance, high-dimensional data often lives in manifolds corresponding to low-density regions of tractable noise distributions. Thus, many methods have been proposed to address this issue by tuning the noise distribution automatically, e.g., [Bose et al., 2018; Ceylan and Gutmann, 2018; Gao et al., 2020].

4.4.3 *Score matching*

Score matching [Hyvärinen and Dayan, 2005] fixes the innacuracy of contrastive learning in low density regions of the noise distribution. This training strategy notices that the data score function, the gradient of the log-likelihood with respect to the data, is independent from the partition function. Thus learning a good energy function can be achieved by minimizing the distance between the model score function $-\nabla_Y E_{\boldsymbol{\theta}}(Y = \cdot) : \mathbb{R}^d \to \mathbb{R}^d$, and the data score function $\boldsymbol{s}(\cdot) \triangleq \nabla_Y \log P(Y = \cdot) : \mathbb{R}^d \to \mathbb{R}^d$. Formally this leads to the following optimization problem

$$\boldsymbol{\theta}^\star = \arg\min_{\boldsymbol{\theta}} \int_{\boldsymbol{y}\in\mathbb{R}^d} P(Y = \boldsymbol{y}) \|\boldsymbol{s}(\boldsymbol{y}) + \nabla_Y E_{\boldsymbol{\theta}}(\boldsymbol{y})\|^2 \mathrm{d}\boldsymbol{y}. \tag{4.17}$$

This formulation is still difficult to optimise as it would require accessing the data's score function, which is what we are implicitly trying to estimate by learning the energy function. However, under weak regularity conditions, we can express the objective function in Equation (4.17) as

$$J(\boldsymbol{\theta}) = \int_{\boldsymbol{y}\in\mathbb{R}^d} P(Y = \boldsymbol{y}) \sum_{i=1}^{d} \left[\frac{-\partial \nabla_Y E_{\boldsymbol{\theta}}[i]}{\partial y_i} + \frac{1}{2} (\nabla_Y E_{\boldsymbol{\theta}[i]})^2 \right] \mathrm{d}\boldsymbol{y} + C, \tag{4.18}$$

where $\nabla_Y E_{\boldsymbol{\theta}}[i] = \frac{\partial E_{\boldsymbol{\theta}}}{\partial y_i}$ is the partial derivative of the energy function with respect to the i^{th} component of the input vector $\boldsymbol{y}$. The constant C is independent from the parameters $\boldsymbol{\theta}$. For neural network-based energy functions, the objective directly translates into a loss function that depends on the diagonal of the Hessian and can be optimised via stochastic gradient descent. We note that score matching does not perform maximum likelihood estimation of the parameters. MLE corresponds to searching for the best models with the KL divergence and score matching the Fisher divergence [Lyu, 2012].

Back in 2010, Vincent [2011] suggested to use score matching to learn denoising models. Denoising models represents the conditional distribution $P(Y \mid \tilde{Y})$ of a clean random variable Y given a noisy version $\tilde{Y} = Y + n$ perturbed by some noise n (e.g. a white Gaussian noise). The idea of stacking many denoising models lead to diffusion models that have recently become state-of-the-art deep generative models for image synthesis and are presented below.

4.5 DIFFUSION MODELS

Diffusion models encompass deep generative models that are all connected by the idea of corrupting structured data into noise and learning a model that reverses the corruption process. We distinguish two sub-classes of diffusion models: i) Continuous-time models [Song and Ermon, 2019; Song et al., 2020b] that formalise the diffusion and generative process as stochastic differential equations and learn the model with denoising

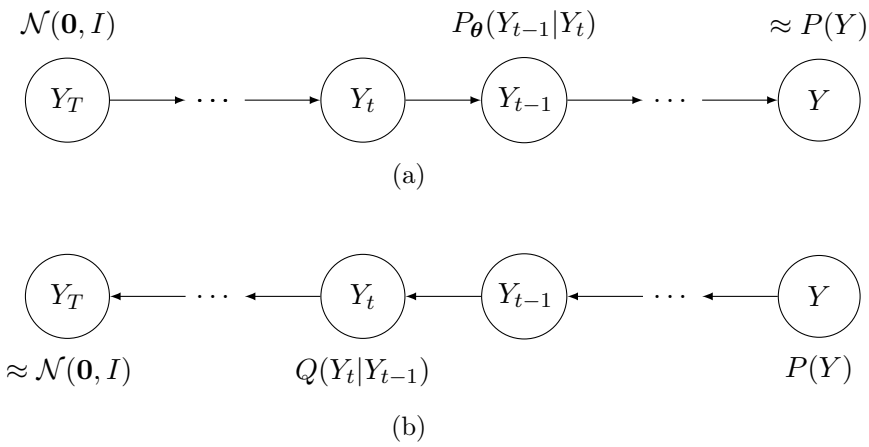


Figure 4.3: The description of a discrete-time diffusion with Bayesian networks, more precisely Markov chains. **(a)** The reverse (generative) process samples an initial state from a normal distribution and generates observations Y by transiting between states with the learned conditional distributions $P_{\boldsymbol{\theta}}(Y_{t-1}|Y_t)$. **(b)** The diffusion process progressively corrupts observations from the dataset $Y \sim P(Y)$ with prescribed corruption kernels $Q(Y_t|Y_{t-1})$ that eventually converge to noise.

score-matching; ii) Discrete-time models dubbed *denoising diffusion probabilistic models* [DDPM Sohl-Dickstein et al., 2015; Ho et al., 2020] that fix the number of corruption steps as a hyperparameter of the model and use variational inference to derive a bound on the model's likelihood.

We first provide a thorough description of discrete-time models and then discuss intuitively continuous-time. We encourage the reader interested in continuous-time models to look at the following papers for more information: [Song and Ermon, 2019; Song et al., 2020b, 2021; Dockhorn et al., 2021].

4.5.1 *Discrete-time diffusion*

Inspired by non-equilibrium statistical physics, Sohl-Dickstein et al. [2015] originally introduced DDPMs. Ho et al. [2020] demonstrated only more recently how to train these models for image synthesis and achieved results close to the state-of-the-art on this task. DDPMs formulate generative modelling as the reverse operation of diffusion, a physical process which progressively destroys information. These processes take the form of Markov chains as depicted in Figure 4.3.

Formally, the reverse process is a latent variable model of the form

$$P_{\boldsymbol{\theta}}(Y = \boldsymbol{y}) := \int P_{\boldsymbol{\theta}}(Y_{0:T} = \boldsymbol{y}_{0:T}) d\boldsymbol{y}_{1:T},$$

where $Y_0 := Y$ takes value in $\mathcal{Y} \triangleq \mathbb{R}^d$ and denotes the observations. The latent variables $Y_1, \ldots, Y_T$ have the same dimensionality as Y. The joint distribution $P_{\boldsymbol{\theta}}(Y_{0:T})$ is modelled as a first order Markov chain with Gaussian transitions, that is

$$P_{\boldsymbol{\theta}}(Y_{0:T}) := P_{\boldsymbol{\theta}}(Y_T) \prod_{t=1}^{T} P_{\boldsymbol{\theta}}(Y_{t-1}|Y_t), \tag{4.19}$$

$$\text{where } P_{\boldsymbol{\theta}}(Y_T) := \mathcal{N}(\mathbf{0}, \mathrm{I}) \quad \text{and} \quad P_{\boldsymbol{\theta}}(Y_{t-1}|Y_t) := \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\psi}}(Y_t, t), \sigma_t^2 \mathrm{I}). \tag{4.20}$$

We want to estimate the parameters of the generative (reverse) process with maximum likelihood estimation. However, we do not have access to the likelihood function $P_{\boldsymbol{\theta}}(Y)$ explicitly but only to the joint distribution between the observation and the latent variables $P_{\boldsymbol{\theta}}(Y_{0:T})$. This is exactly the setting that motivated us to discuss and derive the ELBO in Equation (3.15). Here, the approximate posterior $Q(Y_{1:T}|Y_0)$ is a diffusion process that is also a first order Markov chain with Gaussian transitions,

$$Q(Y_{1:T}|Y_0) := \prod_{t=1}^{T} Q(Y_t|Y_{t-1}), \tag{4.21}$$

$$Q(Y_t|Y_{t-1}) := \mathcal{N}(\sqrt{1-\beta_t} Y_{t-1}, \beta_t \mathrm{I}), \tag{4.22}$$

where $\beta_1, \ldots, \beta_T$ are the variance schedule that is either fixed as training hyper-parameters or learned. The ELBO is then given by

$$\text{ELBO} := \mathbb{E}_Q \left[\log \frac{P_{\boldsymbol{\theta}}(Y_{0:T})}{Q(Y_{1:T}|Y_0)} \right] \leq \log P_{\boldsymbol{\theta}}(Y_0). \tag{4.23}$$

Provided that the variance schedule β_t is small and that the number of timesteps T is large enough, the Gaussian assumptions on the generative process $P_{\boldsymbol{\theta}}$ are reasonable. Ho et al. [2020] take advantage of the Gaussian transitions by expressing the ELBO as

$$\mathbb{E}_Q \left[\mathbb{KL}\left[Q(Y_T|Y_0) \| P_{\boldsymbol{\theta}}(Y_T)\right] - \log P_{\boldsymbol{\theta}}(Y_0|Y_1) + \sum_{t=2}^{T} \mathbb{KL}\left[Q(Y_{t-1}|Y_t, Y_0) \| P_{\boldsymbol{\theta}}(Y_{t-1}|Y_t)\right] \right]. \tag{4.24}$$

The inner sum in Equation (4.24) is made of comparisons between the Gaussian generative transitions $P_{\boldsymbol{\theta}}(Y_{t-1}|Y_t)$ and the conditional forward posterior $Q(Y_{t-1}|Y_t, Y_0)$ which can also be expressed in closed form as Gaussians $\mathcal{N}(\tilde{\mu}_t(Y_0, Y_t), \tilde{\beta}_t \mathrm{I})$, where $\tilde{\mu}_t$ and $\tilde{\beta}_t$ depends on the variance schedule. The KL can thus be calculated with closed form expressions which reduces the variance of the final expression. In addition, Ho et al. [2020] empirically demonstrate that it is sufficient to take optimisation steps on uniformly sampled terms of the sum instead of computing it completely. The final objective resembles denoising score matching over multiple noise levels [Vincent, 2011]. These observations, combined with additional simplifications, lead to a simplified objective

$$L_{\text{DDPM}}(Y_0; \boldsymbol{\theta}) := \mathbb{E}_{t, \boldsymbol{y}_t | Y_0 = \boldsymbol{y}_0} \left[\frac{1}{2\sigma_t^2} \| \boldsymbol{\mu}_{\boldsymbol{\theta}}(Y_t = \boldsymbol{y}_t, t) - \tilde{\mu}_t(Y_0 = \boldsymbol{y}_0, Y_t = \boldsymbol{y}_t) \|^2 \right], \tag{4.25}$$

where $\tilde{\mu}_t(\boldsymbol{y}_0, \boldsymbol{y}_t)$ is the mean of $Q(Y_{t-1} = \boldsymbol{y}_{t-1} | Y_0 = \boldsymbol{y}_0, Y_t = \boldsymbol{y}_t)$, the forward diffusion posterior conditioned on the observation $\boldsymbol{y}_0$. We refer the reader to Ho et al. [2020] for the detailed derivation of the simplified loss function.

4.5.2 *Continuous-time diffusion*

Denoising score matching and DDPM rest on perturbing data at multiple noise scales. Continuous-time diffusion generalises this idea to an infinite number of noise scales corresponding to formulating the forward and reverse processes as stochastic differential equations (SDE). Song et al. [2020b] formally describe the diffusion process as the solution to an Itô SDE:

$$\mathrm{d}\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{y}, t)\mathrm{d}t + g(t)\mathrm{d}\boldsymbol{w}, \tag{4.26}$$

where $\boldsymbol{w}$ is the standard Wiener process (a.k.a., Brownian motion) [Wiener and Masani, 1957; Freedman, 2012], $\boldsymbol{f}(\cdot, t) : \mathbb{R}^d \to \mathbb{R}^d$ is a vector-valued function called the drift coefficient of $\boldsymbol{y}(t)$, and $g(\cdot) : \mathbb{R} \to \mathbb{R}$ is a scalar function known as the diffusion coefficient of $\boldsymbol{y}(t)$.

In this formulation, $\boldsymbol{y}(\cdot) : \mathbb{R} \to \mathbb{R}^d$ becomes a function of time where initial states $\boldsymbol{y}(0)$ are provided by the training samples. We design the drift and diffusion coefficients to enforce a known noise distribution P_T (e.g., $P_T = \mathcal{N}(\mathbf{0}, I)$) after running the SDE from $t = 0$ to $t = T$. We can then generate artificial samples from $P_0 = P(Y)$ by sampling an initial state $\boldsymbol{y}_T \sim P_T$ and running the reverse SDE defined by Anderson [1982] as

$$\mathrm{d}\boldsymbol{y} = \left[\boldsymbol{f}(\boldsymbol{y}, t)\mathrm{d}t - g(t)^2 \nabla_Y \log P_t(Y = \boldsymbol{y})\right] \mathrm{d}t + g(t)\mathrm{d}\boldsymbol{w}, \tag{4.27}$$

where the Wiener process and time flow from $t = T$ to $t = 0$. The reverse process closely resembles stochastic Langevin dynamics [Welling and Teh, 2011] with infinitesimal steps.

The score functions $\nabla_Y \log P_t(Y = \boldsymbol{y})$ are indexed by time t and parameterized by a neural network $\boldsymbol{s}_{\boldsymbol{\theta}}(\cdot, t) : \mathbb{R}^d \to \mathbb{R}^d$ that takes both the state $\boldsymbol{y}(t)$ and the time t. We train the neural networks with score matching at all noise scales. Formally, the learning problem is

$$\boldsymbol{\theta}^\star = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_t \left[\mathbb{E}_{\boldsymbol{y}(0)} \mathbb{E}_{\boldsymbol{y}(t)|\boldsymbol{y}(0)} \left[\| \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{y}(t), t) - \nabla_{Y_t} \log P_{0:t}\Big(\boldsymbol{y}(t) \mid \boldsymbol{y}(0)\Big) \|^2 \right] \right]. \tag{4.28}$$

For adequate drift and diffusion coefficients, the diffusion kernel $P_{0:t}\Big(\boldsymbol{y}(t) \mid \boldsymbol{y}(0)\Big)$ is easy to sample from and can be evaluated in closed-form. In this case, we can train a neural network with stochastic gradient descent and Monte Carlo estimations of the objective function in Equation (4.28). When the diffusion kernel cannot be evaluated directly, we can resort to sliced score matching [Song et al., 2020a] to optimize $\boldsymbol{s}_{\boldsymbol{\theta}}$ from samples of the forward process.

There exists a duality between SDEs and ordinary differential equations (ODE) that allows us to transform one into the other while maintaining the same marginal distribution over the states. The ODE corresponding to Equation (4.27) is

$$\mathrm{d}\boldsymbol{y} = \left[\boldsymbol{f}(\boldsymbol{y}, t) - \frac{1}{2} g(t)^2 \nabla_Y \log P_t(Y = \boldsymbol{y})\right] \mathrm{d}t. \tag{4.29}$$

Provided the approximation $\boldsymbol{s_\theta}$ and an initial random state $\boldsymbol{y}_T \sim P_T$ we can then use an ODE solver to generate samples.

This duality is very important as it provides a direct way of evaluating the model's likelihood via the instantaneous change of variables [Chen et al., 2018]. This theorem states that the change in log probability of a continuous random variables $\boldsymbol{y}(t) \sim P(\boldsymbol{y}(t))$ transformed by an ODE $\frac{\mathrm{d}\boldsymbol{y}}{\mathrm{d}t} = \boldsymbol{h}(\boldsymbol{y}(t), t)$, where $\boldsymbol{h}$ is uniformly Lisphitz, follows a differential equation

$$\frac{\partial \log P(\boldsymbol{y}(t))}{\partial t} = -\operatorname{tr}\left(\frac{d\boldsymbol{h}}{\mathrm{d}\boldsymbol{y}(t)}\right), \tag{4.30}$$

where $\operatorname{tr}\left(\frac{d\boldsymbol{h}}{\mathrm{d}\boldsymbol{y}(t)}\right)$ is the trace of the Jacobian of $\boldsymbol{h}$. This draws a direct connection between diffusion models and normalizing flows that constitute the next class of deep probabilistic models we are going to discuss.

4.6 NORMALIZING FLOWS

A Normalizing Flow [NF, Rezende and Mohamed, 2015] is defined as a sequence of invertible transformations $\boldsymbol{u}_i : \mathbb{R}^d \to \mathbb{R}^d$ $(i = 1, ..., k)$ composed together to create an expressive invertible mapping $\boldsymbol{u} = \boldsymbol{u}_1 \circ \cdots \circ \boldsymbol{u}_k : \mathbb{R}^d \to \mathbb{R}^d$. This mapping can be used to perform density estimation, using $\boldsymbol{u}(\cdot; \boldsymbol{\theta}) : \mathbb{R}^d \to \mathbb{R}^d$ to map a sample $\boldsymbol{y} \in \mathbb{R}^d$ onto a latent vector $\boldsymbol{z} \in \mathbb{R}^d$ equipped with a prescribed density $P_{\boldsymbol{z}}(\boldsymbol{z})$ such as an isotropic Normal. The transformation $\boldsymbol{u}$ implicitly defines a density $p(\boldsymbol{x}; \boldsymbol{\theta})$ as given by the change of variables formula,

$$P(\boldsymbol{y}; \boldsymbol{\theta}) = P_Z(\boldsymbol{u}(\boldsymbol{y}; \boldsymbol{\theta})) \left|\det J_{\boldsymbol{u}(\boldsymbol{y};\boldsymbol{\theta})}\right|, \tag{4.31}$$

where $J_{\boldsymbol{u}(\boldsymbol{y};\boldsymbol{\theta})}$ is the Jacobian of $\boldsymbol{u}(\boldsymbol{y}; \boldsymbol{\theta})$ with respect to $\boldsymbol{x}$. The resulting model is trained by maximising the likelihood of the data $\mathcal{D} := \{\boldsymbol{y}_i\}_{i=1}^N$.

4.6.1 *Discrete normalizing flows*

It is common for normalizing flows to stack the same parametric function $\boldsymbol{u}_i$ (with different parameters values) and to reverse variables ordering after each transformation. For this reason, we will focus on presenting a popular strategy to build one of these repeated transformations, which we further refer to as $\boldsymbol{g} : \mathbb{R}^d \to \mathbb{R}^d$.

In general the step $\boldsymbol{g}$ take any form as long as it defines a bijective map. Many neural architectures of normalizing flows can be mathematically described as

$$\mathbf{g}(\boldsymbol{y}) = \begin{bmatrix} g^1(y_1; \boldsymbol{c}^1(\boldsymbol{y})) & \dots & g^d(y_d; \boldsymbol{c}^d(\boldsymbol{y})) \end{bmatrix}^T, \tag{4.32}$$

where the $\boldsymbol{c}^i$ are the **conditioners** which role is to constrain the structure of the Jacobian of $\boldsymbol{g}$ into triangularizable matrices. The functions g^i, called **normalizers**, partially parameterized by their conditioner, must be invertible with respect to their input variable y_i. For such architectures, the determinant of the Jacobian reduces to the product of the partial derivatives $\frac{\partial g^i}{\partial y_i}$, which is sign-constant. This implies that the determinant of the Jacobian never cancels out and convinces us that $\boldsymbol{g}$ is bijective.

The conditioners impose an autoregressive structure or, more generally, any structure representing a directed acyclic graph. In Chapter 6 and Chapter 8, we show why this is true and draw a clear relationship between normalizing flows and Bayesian networks. The normalizer g^i can be any function as long as it is a monotonic function of its main input y_i. In terms of neural networks, an affine normalizer $g : \mathbb{R} \times \mathbb{R}^2 \to \mathbb{R}$ can be expressed as $g(x; m, s) = x \exp(s) + m$, where $m \in \mathbb{R}$ and $s \in \mathbb{R}$ are computed by the conditioner. There also exist methods to parameterize monotonic normalizers [Huang et al., 2018; De Cao et al., 2020; Durkan et al., 2019; Jaini et al., 2019] with neural networks and one contribution of this thesis is to introduce one of them called Unconstrained Monotonic Neural Networks [UMNNs, Wehenkel and Louppe, 2019] in Chapter 7.

4.6.2 *Continuous normalizing flows.*

We have described normalizing flows for which k is discrete. There also exist continuous normalizing flows that correspond to infinitesimal transformations defined by an ODE $\frac{\mathrm{d}\boldsymbol{y}}{\mathrm{d}t} = \boldsymbol{h}(\boldsymbol{y}(t), t)$ as mentioned at the end of the discussion on diffusion models. Continuous NFs were first introduced in the seminal work on neural ordinary equations by Chen et al. [NODE, 2018]. Soon after, Grathwohl et al. [2018] proposed to use the Hutchinson trace estimator in Equation (4.30) to reduce the computation cost of continuous NFs. As previously mentioned, continuous NFs can be parameterised by any Lipshitz continuous function and are thus easy to parameterise with neural networks. For a thorough review of normalizing flow archirecture we refer the reader to Papamakarios et al. [2019a]; Kobyzev et al. [2020-08] .

4.6.3 *Discussion*

Remarkably, NFs are among the rare deep probabilistic models that explicitly provide access to the likelihood function, hence to the learned density. In contrast to other deep probabilistic models that do not directly lead to a tractable optimisation objective, the learning algorithm of NFs is straightforward. It is just the gradient descent of the negative

Figure 4.4: The description of a variational auto-encoder with Bayesian networks. **(a)** The decoding process samples the latent variables from $P(Z)$ and generates observations Y by sampling conditionaly from $P_{\boldsymbol{\theta}}(Y \mid Z)$. **(b)** The encoding process takes an observation from the dataset $Y \sim P(Y)$ and computes the approximate posterior $Q_{\boldsymbol{\psi}}(Z \mid Y)$ corresponding to the model in **(a)**.

log-likelihood. Explicit models are also particularly interesting for parameterising the approximate posterior in VI [Rezende and Mohamed, 2015] and have played an essential role in many simulation-based inference algorithms as well [Papamakarios et al., 2019b; Greenberg et al., 2019].

The tractability of the likelihood function has a price. Discrete NFs impose strong constraints on the neural architecture used to parameterise the bijective transformations. This often leads to poor inductive bias and reduces these models' efficiency for some data modalities such as images. For continuous models, the principal cost is the potential complexity of solving the associated neural ODE and the difficulty of optimising NODE models. In general, the most fundamental issue of NFs is to enforce a latent space that has the same dimensionality as the data whereas it is often more reasonable to assume these data lie on a lower-dimensional manifold. People have worked at solving this issue, e.g., Brehmer and Cranmer [2020] introduced $\mathcal{M}$-flow that learns a manifold and a density on it jointly; however these methods either require a prescribed manifold, or they resort to adversarial optimization which complicates the simple training loop of classical NFs.

A simpler solution is to formulate the generative process as a stochastic mapping between low dimensional latent variables to observations instead and use NFs to model conditional distributions. These models are called variational auto-encoders and are covered below.

4.7 VARIATIONAL AUTO-ENCODERS

We have already discussed some generative models based on latent variables with diffusion models. The main difference here is to not assume a prescribed mapping from the observations Y to the latent Z. Instead a variational auto-encoder [VAE, Kingma and Welling, 2013] trains jointly an encoder network that models the posterior distribution $Q_{\boldsymbol{\psi}}(Z \mid Y) \approx P(Z \mid Y)$ and a decoder network that parameterizes the stochastic mapping $P_{\boldsymbol{\theta}}(Y \mid Z)$ from latent variables to observations. This allows us to embed good inductive bias in the decoder that generates observations from latent variables. A good example is the NVAE [Vahdat and Kautz, 2020] that formulates this mapping hierarchically for

images, using different latent variables to describe the high-level and low-level structure of the image.

Formally, we want to learn a generative model of an unknown distribution $P(Y)$ given a dataset $\mathcal{D} \triangleq \{\boldsymbol{y}^i\}_{i=1}^N$ of N i.i.d observations $\boldsymbol{y}^i \sim P(Y)$ sampled from this unknown distribution. The original VAE postulates a two-step generative process in which some unobserved variables $\boldsymbol{z} \in \mathbb{R}^h$ are first sampled from a prior distribution $P(Z)$ and then observations $\boldsymbol{y}$ are generated from a conditional distribution $P_{\boldsymbol{\theta}}(Y \mid Z = \boldsymbol{z})$. The generative process can be expressed mathematically as

$$\mathbf{z} \sim P(Z) \quad \text{and} \quad \boldsymbol{y} \sim P_{\boldsymbol{\theta}}(Y \mid Z = \boldsymbol{z}). \tag{4.33}$$

The prior $P(Z)$ is often an isotropic Gaussian while the likelihood $P_{\boldsymbol{\theta}}(Y \mid Z = \boldsymbol{z})$ is parameterised with a neural network. The likelihood model decodes latent variables into observations and is thus usually referred to as the decoder in the literature. In its original formulation, the likelihood is parameterized with a multivariate Gaussian $\mathcal{N}\Big(\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{z}), \operatorname{diag}\left(\sigma^2_{\boldsymbol{\theta}}(\boldsymbol{z})\right)\Big)$ when the domain $\mathcal{Y}$ is continuous, and a categorical distribution when it is discrete.

Training aims to find the parameters $\boldsymbol{\theta}$ of the decoder that maximize the sum of the marginal likelihoods of individual points, the MLE which optimises

$$\log P_{\boldsymbol{\theta}}(\mathcal{D}) = \sum_{\boldsymbol{y} \in \mathcal{D}} \log \int P_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{z}) P(\boldsymbol{z}) \mathrm{d}\boldsymbol{z}.$$

These integrals are intractable but we rely on variational inference again to approximate the MLE objective. The introduction of an encoder network that approximates the posterior distribution $Q_{\boldsymbol{\psi}}(Z \mid Y)$ allows maximizing the associated evidence lower bound

$$\begin{aligned} \operatorname{ELBO}(\boldsymbol{y}; \boldsymbol{\theta}, \boldsymbol{\psi}) &:= \mathbb{E}_Q \left[\log \frac{P_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{z}) P(\boldsymbol{z})}{Q_\psi(\boldsymbol{z} \mid \boldsymbol{y})} \right] && (4.34) \\ &= \log P_{\boldsymbol{\theta}}(\boldsymbol{y}) - \mathbb{KL}\left[Q_{\boldsymbol{\psi}}(Z \mid \boldsymbol{y}) \| P_{\boldsymbol{\theta}}(Z \mid \boldsymbol{y}) \right] && (4.35) \\ &\leq \log P_{\boldsymbol{\theta}}(\boldsymbol{x}). && (4.36) \end{aligned}$$

The ELBO becomes tighter as the approximate posterior $Q_{\boldsymbol{\psi}}(\boldsymbol{z}|\boldsymbol{y})$ gets closer to the true posterior. Learning the generative model is finally performed by jointly optimising the parameters $\boldsymbol{\theta}$ of the decoder and $\boldsymbol{\psi}$ of the approximate posterior via stochastic gradient ascent. In the original VAE, the encoder models the approximate posterior as a conditional multivariate Gaussian distribution $\mathcal{N}(\mu_{\boldsymbol{\psi}}(\boldsymbol{y}), \operatorname{diag}(\sigma^2_{\boldsymbol{\psi}}(\boldsymbol{y})))$.

In practice, the good optimisation of VAEs depends on the ability of the encoder $Q_{\boldsymbol{\psi}}(Z|Y)$ to approximate the posterior well at all possible observations Y. This also implies that marginalizing out Y from the encoder, i.e., $Q(Z) \triangleq \frac{1}{N} \sum_{\boldsymbol{y}_i \in \mathcal{D}} Q_{\boldsymbol{\psi}}(Z|Y)$, should closely match the prior distribution $P(Z)$ which is often difficult. A strategy is to make both the posterior and prior distribution generic by parameterising them with normalizing flows [Berg et al., 2018].

4.8 DISCUSSION

As we have seen, the separation between two classes of models is sometimes blurry. Normalizing flows and autoregressive models have more in common than differences. Similarly, the most notable difference between a VAE and a diffusion model is in the parameterisation of the approximate posterior distribution. Moreover, diffusion models achieve the same goal as NFs; they map a noise distribution to the data distribution with an invertible transformation. Energy-based models are slightly more distinct in contrast. Still, most algorithms designed for unnormalised models are also relevant for other models. For instance, continuous diffusion models define implicitly the distribution via the gradient of its logarithm – it is an energy-based model.

In this background, we have set aside some aspects of deep generative models because we do not believe these are necessary to apprehend the rest of this thesis. In particular, we have avoided providing details about the neural architectures used in practice, although this is decisive for achieving good performance with deep probabilistic models. These choices are usually motivated by experience and depend on the data type. New architectures leading to better performance in discriminative tasks often lead to the corresponding advances in unsupervised modelling tasks. In addition, there also exist architectures designed for sub-classes of deep probabilistic models such as hierarchical VAEs [Vahdat and Kautz, 2020], causal convolutions [Van Den Oord et al., 2016], etc. Furthermore, years of research in deep probabilistic models has lead to many tricks that can improve the performance of these models in different contexts.

4.9 CHALLENGES AND OPPORTUNITIES

We have presented the main deep probabilistic models (DPMs). Each model offers a different balance between simplicity, tractability and expressivity. Some models, such as diffusion and energy-based models, define the probabilistic model implicitly and even involve sampling algorithms to express the modelled distribution. In contrast, NFs and autoregressive models parameterise the distribution explicitly – we have direct access to the model's likelihood. Sampling these models is not always computationally efficient. Nevertheless, it is a deterministic procedure that does not depend on hyperparameters in contrast to sampling diffusion or energy-based models. Finally, VAEs offer a nice balance between expressivity and tractability. Although the model's likelihood is intractable, sampling from these models is straightforward.

Chapter 3 has introduced graphical models. This approach to probabilistic modelling is closer to classical modelling than DPMs. Indeed, the topology, often prescribed, defines a set of strong assumptions on the modelled phenomenon. The learning task mostly amounts to fitting conditional distributions. Similarly, classical modelling combines existing models that are assumed faithful to sub-components of the phenomenon. In oppo-

sition, deep models mainly rely on the soft constraints induced by the neural networks' architecture and training algorithm.

In this thesis, we reject the classical separation of somewhat distinct classes of models. We argue that there exist meaningful connections between all probabilistic models, sometimes yet to be disovered. Ignoring these connections leads to an unnecessary profusion of equivalent algorithms and terminologies. This profusion reduces the accessibility of probabilistic modelling, yet one of the most fundamental tools of engineers and scientists. In addition, we foresee at least two other motivations for drawing connections between different types of models. First, it provides a new prospect on the concerned models, which has countless positive outcomes: it unlocks the sharing of algorithms, interpretations or pre-trained models, to cite a few advantages. The second motivation is to simplify the compositionality of different classes of models together, which is critical for creating models aligned with prescribed knowledge. Such informed models may generalise outside the training distributions and require fewer data as they rely on a more substantial inductive bias.

Aligned with this objective, we discuss and improve the expressivity of VAEs and NFs in Part ii. In Chapter 5. We show that modelling the prior of VAEs with diffusion models is beneficial. Then, in Chapter 6, we draw connections between NFs and Bayesian networks and prove the limitations of existing affine NFs. In Chapter 7, we introduce a new neural architecture for modelling monotonic functions. This architecture overcomes the expressivity issues of affine NFs. In Part iii, we explore informed probabilistic modelling. We demonstrate that Bayesian networks and NFs can serve each other's purpose in Chapter 8. In Chapter 9, we show that combining deterministic models and VAEs leads to hybrid models that generalise outside the training distributions under appropriate assumptions. Finally, Chapter 10 concludes with a summary of our contribution and presents our perspectives on the future of probabilistic modelling.

Part II

UNINFORMED PROBABILISTIC MODELLING

The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point.

Claude Shannon

5

COMBINING MODELS

Outline

We demonstrate the benefits of mixing distinct classes of probabilistic models. In particular, we study the complementary between variational autoencoders (VAEs) and denoising diffusion probabilistic models (DDPMs). While VAEs offer scalable amortised posterior inference and fast sampling, they are often outperformed by competing models such as normalizing flows (NFs) or deep-energy models. We improve VAEs by modelling the prior distribution of the latent variables with a diffusion process. The diffusion prior model improves upon Gaussian priors of classical VAEs and is competitive with NF-based priors. This contribution shows that connecting different classes of models can unlock modelling capacities and properties that are unreachable by each model class independently.

5.1 PROLOGUE

This chapter expresses the relevance of unifying the frameworks associated with distinct (deep) probabilistic models. Indeed, building a broad and deep understanding of different probabilistic modelling frameworks unlocks potential model combinations. These combinations may help reduce the weaknesses of each class of models separately while maintaining their assets. We notice the relevance of this unification for deep probabilistic models. Indeed, gradient-descent-based training framework allows jointly training all components of a 'super' model if the objective function is a differentiable function of the model's components. We rely on this observation to combine denoising diffusion probabilistic models and variational autoencoders.

As will see in the paper, combining DDPMs and VAEs is pretty straightforward and leads to better modelling for image synthesis. We believe that further interplay between different probabilistic models is essential in developing tomorrow's probabilistic modelling toolbox. In an ideal world, combining two classes of deep probabilistic models should be as simple as replacing a Normal distribution with a Laplace distribution in a probabilistic program. Building this ideal world should eventually help practitioners to define models with all the key properties required for their final application. The paper shall highlight the relevance of this idea.

5.2 THE PAPER: DIFFUSION PRIORS IN VARIATIONAL AUTOENCODERS

5.2.1 *Author contributions*

Gilles Louppe and I co-authored the paper. As the leading author, I developed the connections between diffusion models and variational autoencoders, did experiments, and wrote the article. I derived the ELBO associated with the denoising diffusion priors in VAES. Gilles supervised me throughout this project, offered suggestions, and helped write the paper.

5.2.2 *Reading tips*

The reader may skip Section 2, which presents VAEs and DDPMs already introduced in the background chapter. The reader interested in deeply understanding the implementation of DDPMs should look at Ho et al. [2020]. The rest of the paper should flow smoothly.

5.2.3 *Minor corrections*

There is a missing negative sign in Equation (20) which becomes

$$\mathcal{L}(\boldsymbol{x}; \phi, \theta, \psi) := -\mathbb{E}_{q_\psi}\left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})}{q_\psi(\boldsymbol{z}|\boldsymbol{x})}\right] + \mathbb{E}_{q_\psi}\left[L_{\text{DDPM}}(\boldsymbol{z}_0; \phi)\right].$$

Diffusion Priors In Variational Autoencoders

Antoine Wehenkel [1] **Gilles Louppe** [1]

Abstract

Among likelihood-based approaches for deep generative modelling, variational autoencoders (VAEs) offer scalable amortized posterior inference and fast sampling. However, VAEs are also more and more outperformed by competing models such as normalizing flows (NFs), deep-energy models, or the new denoising diffusion probabilistic models (DDPMs). In this preliminary work, we improve VAEs by demonstrating how DDPMs can be used for modelling the prior distribution of the latent variables. The diffusion prior model improves upon Gaussian priors of classical VAEs and is competitive with NF-based priors. Finally, we hypothesize that hierarchical VAEs could similarly benefit from the enhanced capacity of diffusion priors.

1. Introduction

Over the last few years, the interest of the deep learning community for generative modelling has increased steadily. Among the likelihood-based approaches for deep generative modelling, variational autoencoders (Kingma & Welling, 2013, VAEs) stand as one of the most popular, although competing approaches now demonstrate better performance. In particular, Ho et al. (2020); Nichol & Dhariwal (2021); Dhariwal & Nichol (2021) recently showed that denoising diffusion probabilistic models (DDPMs) are competitive deep generative models, obtaining samples quality similar to those of the best implicit deep generative models such as ProgressiveGAN (Karras et al., 2017) and StyleGAN (Karras et al., 2019). Similarly to VAEs, DDPMs train on a variational bound and may be interpreted under the encoding-decoding framework.

In the original formulation of VAEs, the prior and the posterior distributions over the latent variables are assumed to be both Gaussian. However, these assumptions are often incompatible and are thus limiting the performance of VAEs for complex modelling tasks (Tomczak & Welling, 2018; Chen et al., 2018). A natural solution to this problem is to parameterize the prior, sometimes also the posterior, with more expressive distributions. In this preliminary work, we improve VAEs by demonstrating how DDPMs can be used for modelling the prior distribution of the latent variables. In addition to boosting DDPM with the compression properties of VAEs, combining the two models should eventually lead to greater generative performance by enabling complex generative modelling even with simple decoder architecture. Finally, working in the latent space should eventually reduce the computational burden associated with diffusion generative models.

[*]Equal contribution [1]University of Liège, Liège, Belgium. Correspondence to: Antoine Wehenkel <antoine.wehenkel@uliege.be>.

Third workshop on *Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models* (ICML 2021).

2. Latent generative models

2.1. Variational autoencoder

We want to learn a generative model of an unknown distribution $p(\mathbf{x})$ given a dataset $X \in \mathbb{R}^{N\times d}$ of N i.i.d observations $\mathbf{x}$ sampled from this unknown distribution. The original VAE postulates a two-step generative process in which some unobserved variables $\mathbf{z} \in \mathbb{R}^h$ are first sampled from a prior distribution $p(\mathbf{z})$ and then observations $\mathbf{x}$ are generated from a conditional distribution $p_\theta(\mathbf{x}|\mathbf{z})$. The generative process can be expressed mathematically as

$$\mathbf{z} \sim p(\mathbf{z}) \quad \text{and} \quad \mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z}). \tag{1}$$

The prior $p(\mathbf{z})$ is chosen Gaussian while the likelihood $p_\theta(\mathbf{x}|\mathbf{z})$ is modeled with a neural network. The likelihood model decodes latent variables into observations and is thus usually refereed as the decoder in the literature. In its original formulation, the likelihood is parameterized with a multivariate Gaussian $\mathcal{N}(\mu_\theta(\mathbf{z}), \text{diag}(\sigma^2_\theta(\mathbf{z})))$ when the observations are continuous, and a categorical distribution when they are discrete.

Training the generative model is achieved by finding the parameters θ of the decoder that maximize the sum of the marginal likelihoods of individual points

$$p_\theta(X) = \sum_{\mathbf{x}\in X} \log \int p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})\text{d}\mathbf{z}.$$

These integrals are intractable but the introduction of an

encoder network that approximates the posterior distribution $q_\phi(\mathbf{z}|\mathbf{x})$ allows maximizing the associated evidence lower bound

$$\begin{aligned}\text{ELBO} &:= \mathbb{E}_q\left[\log \frac{p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{q_\psi(\mathbf{z}|\mathbf{x})}\right] && (2)\\ &= \log p_\theta(\mathbf{x}) - \mathbb{KL}\left[q_\psi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})\right] && (3)\\ &\leq \log p_\theta(\mathbf{x}). && (4)\end{aligned}$$

The ELBO becomes tighter as the approximate posterior $q_\psi(\mathbf{z}|\mathbf{x})$ gets closer to the true posterior. Learning the generative model is finally performed by jointly optimizing the parameters θ of the decoder and ϕ of the approximate posterior via stochastic gradient ascent. In the original VAE, the encoder models the approximate posterior as a conditional multivariate Gaussian distribution $\mathcal{N}(\mu_\phi(\mathbf{x}), \text{diag}(\sigma^2_\phi(\mathbf{x})))$.

The ELBO loss presents two antagonistic goals to the encoder. It should be able to both encodes the data accurately while being as close as possible to the prior distribution. Consequently, the Gaussian assumptions made on both the prior and the posterior distributions are often incompatible and limit the generative performance. A possible solution consists in learning a prior distribution that is compatible with the learned posteriors. For example, Habibian et al. (2019) and Chen et al. (2017) respectively showed that autoregressive models and normalizing flows (Rezende & Mohamed, 2015, NFs) greatly improve the performance of VAEs when used as prior distributions. In the following we present how denoising diffusion probabilistic models can be used to improve the performance of classical VAEs.

2.2. Denoising diffusion probabilistic models

Inspired by non-equilibrium statistical physics, Sohl-Dickstein et al. (2015) originally introduced DDPMs while Ho et al. (2020) demonstrated only more recently how to train these models for image synthesis, achieving results close to the state-of-the-art on this task. DDPMs formulate generative modelling as the reverse operation of diffusion, a physical process which progressively destroys information. Formally, the reverse process is a latent variable model of the form

$$p_\phi(\mathbf{x}_0) := \int p_\phi(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T},$$

where $\mathbf{x}_0 := \mathbf{x}$ denotes the observations and $\mathbf{x}_1, \dots, \mathbf{x}_T$ denote latent variables of the same dimensionality as $\mathbf{x}_0$. The joint distribution $p_\phi(\mathbf{x}_{0:T})$ is modelled as a first order Markov chain with Gaussian transitions, that is

$$\begin{aligned}p_\phi(\mathbf{x}_{0:T}) &:= p_\phi(\mathbf{x}_T)\prod_{t=1}^{T} p_\phi(\mathbf{x}_{t-1}|\mathbf{x}_t), && (5)\\ p_\phi(\mathbf{x}_T) &:= \mathcal{N}(\mathbf{0}, \mathbf{I}), && (6)\\ p_\phi(\mathbf{x}_{t-1}|\mathbf{x}_t) &:= \mathcal{N}(\mu_\phi(\mathbf{x}_t, t), \sigma_t^2\mathbf{I}). && (7)\end{aligned}$$

Similar to VAEs, the reverse Markov chain is trained on an ELBO. However, the approximate posterior $q(\mathbf{x}_{1:T}|\mathbf{x}_0)$ is fixed to a diffusion process that is also a first order Markov chain with Gaussian transitions,

$$\begin{aligned}q(\mathbf{x}_{1:T}|\mathbf{x}_0) &:= \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}), && (8)\\ q(\mathbf{x}_t|\mathbf{x}_{t-1}) &:= \mathcal{N}(\sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}), && (9)\end{aligned}$$

where $\beta_1, \dots, \beta_T$ are the variance schedule that is either fixed as training hyper-parameters or learned. The ELBO is then given by

$$\text{ELBO} := \mathbb{E}_q\left[\log \frac{p_\phi(\mathbf{x_{0:T}})}{q(\mathbf{x_{1:T}}|\mathbf{x_0})}\right] \leq \log p_\phi(\mathbf{x}_0). \quad (10)$$

Provided that the variance schedule β_t is small and that the number of timesteps T is large enough, the Gaussian assumptions on the generative process p_ϕ are reasonable. Ho et al. (2020) take advantage of the Gaussian transitions to express the ELBO as

$$\begin{aligned}\mathbb{E}_q\Big[&\mathbb{KL}\left[q(\mathbf{x}_T|\mathbf{x}_0)||p(\mathbf{x}_T)\right] - \log p_\phi(\mathbf{x}_0|\mathbf{x}_1)\\ &+ \sum_{t=2}^{T}\mathbb{KL}\left[q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\phi(\mathbf{x}_{t-1}|\mathbf{x}_t)\right]\Big]. \quad (11)\end{aligned}$$

The inner sum in Equation (11) is made of comparisons between the Gaussian generative transitions $p_\phi(\mathbf{x}_{t-1}|\mathbf{x}_t)$ and the conditional forward posterior $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ which can also be expressed in closed form as Gaussians $\mathcal{N}(\tilde{\mu}_t(\mathbf{x}_0, \mathbf{x}_t), \tilde{\beta}_t\mathbf{I})$, where $\tilde{\beta}_t$ are functions of the variance schedule. The KL can thus be calculated with closed form expressions which reduces the variance of the final expression. In addition, Ho et al. (2020) empirically demonstrate that it is sufficient to take optimization steps on uniformly sampled terms of the sum instead of computing it completely. The final objective closely resembles denoising score matching over multiple noise levels (Song & Ermon, 2019). These observations combined with additional simplifications leads to a simplified loss

$$L_{\text{DDPM}}(\mathbf{x}_0; \phi) := \mathbb{E}_{t, \mathbf{x}_0, \mathbf{x}_t}\left[\frac{1}{2\sigma_t^2}||\mu_\phi(\mathbf{x}_t, t) - \tilde{\mu}_t(\mathbf{x}_0, \mathbf{x}_t)||^2\right], \quad (12)$$

where $\tilde{\mu}_t(\mathbf{x}_0, \mathbf{x}_t)$ is the mean of $q(\mathbf{x}_{t-1}|\mathbf{x}_0, \mathbf{x}_t)$, the forward diffusion posterior conditioned on the observation $\mathbf{x}_0$.

3. Prior modelling with denoising diffusion

We now introduce our contribution which consists in using a DDPM for modelling the prior distribution in VAEs. We

formulate the generative model as

$$\mathbf{z}_T \sim \mathcal{N}(\mathbf{0}, \mathrm{I}) \tag{13}$$
$$\mathbf{z}_{t-1|t} \sim p_\phi(\mathbf{z}_{t-1}|\mathbf{z}_t) \quad \forall t \in [T, \dots, 1] \tag{14}$$
$$\mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z}_0), \tag{15}$$

where ϕ denotes the parameters of the reverse diffusion model encoding the prior distribution. Equations (13) and (14) implicitly define a prior distribution over the usual latent variables $\mathbf{z}_0$ which is modelled with a reverse diffusion process.

Unfortunately, we cannot train a VAE with a diffusion prior directly on the ELBO as expressed in Equation (2) as $p_\phi(\mathbf{z}_0)$ cannot be evaluated. However, Equation (2) can be further developed as

$$\mathbb{E}_{q_\psi}\left[\log p_\theta(\mathbf{x}|\mathbf{z}_0)\right] - \mathbb{E}_{q_\psi}\left[\log q(\mathbf{z}_0|\mathbf{x})\right] + \mathbb{E}_{q_\psi}\left[\log p_\phi(\mathbf{z}_0)\right] \tag{16}$$

in which a lower bound on the last term can be expressed by Equation (10). This finally leads to the following expression

$$\mathbb{E}_{q_\psi}\left[\log p_\theta(\mathbf{x}|\mathbf{z}_0) - \log q(\mathbf{z}_0|\mathbf{x}) + \mathbb{E}_q\left[\log \frac{p_\phi(\mathbf{z_{0:T}})}{q(\mathbf{z_{1:T}}|\mathbf{z_0})}\right]\right] \tag{17}$$
$$\leq \mathbb{E}_{q_\psi}\left[\log p_\theta(\mathbf{x}|\mathbf{z}_0) - \log q(\mathbf{z}_0|\mathbf{x}) + \log p_\phi(\mathbf{z}_0)\right] \tag{18}$$
$$\leq \log p_\theta(\mathbf{x}), \tag{19}$$

which is a valid ELBO. Finally, the diffusion prior p_ϕ is trained jointly with the approximate posterior q_ψ and the likelihood models p_θ which are optimized as in a classical VAE. This leads to the following loss function:

$$\mathcal{L}(\mathbf{x}; \phi, \theta, \psi) := \mathbb{E}_{q_\psi}\left[\log \frac{p_\theta(\mathbf{x}|\mathbf{z})}{q_\psi(\mathbf{z}|\mathbf{x})}\right] + \mathbb{E}_{q_\psi}\left[L_{\text{DDPM}}(\mathbf{z}_0; \phi)\right]. \tag{20}$$

4. Related work

Various approaches have been proposed to improve the modelling capacity and the training of VAEs. As a first example, some state-of-the-art deep generative models based on VAEs model the posterior with normalizing flows or autoregressive models (Kingma et al., 2016; Vahdat & Kautz, 2020). Autoregressive models are also often used as a replacement of the original likelihood parameterization, which assumes conditional independencies that are often unrealistic (Oord et al., 2016). Another popular improvement made to the original VAE is the embedding of structure in the latent variables. In particular, hierarchical VAEs (Sønderby et al., 2016; Kingma et al., 2016) combined with careful training demonstrate impressive results on generative modelling for images (Vahdat & Kautz, 2020).

Vahdat et al. (2021) concurrently proposed to use diffusion for modelling the prior distributions of VAEs. They obtain state-of-the-art results on image synthesis by combining continuous diffusion models and VAEs. Not as close to our work but related, Chen et al. (2017) proposed to learn the prior as a solution to the mismatch between the approximate and the true posteriors. They model the prior with an autoregressive flow, which also closely relates to modelling the posterior distribution with an inverse autoregressive flow (Kingma et al., 2016). Tomczak & Welling (2018) takes inspiration from the aggregated posterior $\frac{1}{N}\sum_{i=1}^{N} q_\psi(z|x)$ (Hoffman & Johnson, 2016; Makhzani et al., 2015) to introduce the VampPrior defined as a mixture of learned pseudo-inputs. An orthogonal line of work suggests that the mismatch between the approximate posterior and the exact posterior can be reduced by over-weighting the terms related to the prior and to the approximate posterior in the ELBO (Higgins et al., 2016; Chen et al., 2018).

5. Experiments

We now compare VAEs for different choices of priors, including the original Gaussian prior, an NF prior, and the proposed diffusion prior. All models share a same backbone encoder-decoder architecture inspired from DCGAN (Radford et al., 2015). Optimization is performed with Adam for 250 epochs with a learning rate set to 0.0005. After each epoch, the models are evaluated on a validation set used to select the best one for each training setting. We compare the models on the CIFAR10 and CelebA datasets for 3 different latent variables dimensionality (40, 100, 200). The NF used in our experiments is a 3-step autoregressive affine flow with simple MLP backbones similar to the one used to model the transition function of DDPM.

Table 1 presents the FID scores for the different models. We first notice the large scores reached by all models on the CIFAR10 dataset. This can be explained by the simplicity of the models trained in our experiments. We believe these scores could be greatly improved by using a more sophisticated likelihood model such as a PixelCNN (Oord et al., 2016). Although FID scores suggest that the Gaussian prior outperforms the diffusion prior in terms of generative performance, the visual inspection of Figure 1 shows that the diffusion prior results in samples slightly more realistic than those of the classical VAE. The best FID score is achieved by the NF prior, although its samples do not seem to reflect this superiority. In this case, we believe the FID scores are not entirely informative about the quality of the images synthesized by the models and should be interpreted with a grain of salt. Although learned priors seem to improve generative performance on CIFAR10, additional work is needed to reach results that would justify using a diffusion prior for this dataset.

On CelebA however, we observe in Table 1 that diffusion priors outperform the Gaussian prior. This is in line with

Table 1. FID scores for different models for prior modelling in VAEs and for different latent size. *Diffusion priors outperform classical VAE on CelebA but are slightly worse than NFs. FID scores do not reveal the superiority of any method on CIFAR10.*

Dataset	**CelebA**			**CIFAR10**		
Latent Size	40	100	200	40	100	200
Gaussian	154.3	149.4	139.1	176.0	126.2	123.9
NF	72.9	59.49	54.7	167.6	129.1	129.6
Diffusion	114.8	67.95	88.3	177.9	160.5	153.1

the visual inspection of Figure 2a and Figure 2c. As for CIFAR10, the NF prior outperforms the Gaussian and diffusion priors in terms of FID scores, although the visual inspection of the corresponding samples in Figure 2b does not reveal a much better quality of images when compared to those resulting from the diffusion prior. We conclude from these observations that diffusion priors offer an interesting alternative to NFs for modelling the prior in a VAE.

6. Conclusion and future work

This preliminary work presents how denoising diffusion probabilistic models can be used as a new class of learnable priors for VAEs. As a notable contribution, we empirically demonstrate that optimizing implicitly a prior on an ELBO can be performed jointly to training the encoder and the decoder of the VAE. In addition, our results suggest DDPM performs on par with NFs for modelling prior distribution.

A large spectrum of future research directions could benefit from the basic idea expressed in this preliminary work. As an example, recent advances in diffusion models such as the continuous formulation (Song et al., 2020) or improvement to the training procedure of DDPM (Nichol & Dhariwal, 2021) could be implemented in the prior model. Similarly, many improvements could be made to the architectures used for the VAE and to the training procedure. In particular, image synthesis with hierarchical VAEs which organizes the latent variables into multiple scales images could reveal the full potential of diffusion priors. This would indeed combine the structural knowledge embed by such type of VAEs with the impressive performance of DDPM for modelling distributions over images. Finally, diffusion does not constrain the neural networks architectures and so enables the embedding of a larger choice of inductive biases in the prior distribution compared to autoregressive models and NFs.

References

Chen, R. T., Li, X., Grosse, R., and Duvenaud, D. Isolating sources of disentanglement in variational autoencoders. *arXiv preprint arXiv:1802.04942*, 2018.

Chen, X., Kingma, D. P., Salimans, T., Duan, Y., Dhariwal, P., Schulman, J., Sutskever, I., and Abbeel, P. Variational lossy autoencoder. *ICLR*, 2017.

Dhariwal, P. and Nichol, A. Diffusion models beat gans on image synthesis. *arXiv preprint arXiv:2105.05233*, 2021.

Habibian, A., Rozendaal, T. v., Tomczak, J. M., and Cohen, T. S. Video compression with rate-distortion autoencoders. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7033–7042, 2019.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *arXiv preprint arXiv:2006.11239*, 2020.

Hoffman, M. D. and Johnson, M. J. Elbo surgery: yet another way to carve up the variational evidence lower bound. In *Workshop in Advances in Approximate Bayesian Inference, NIPS*, volume 1, pp. 2, 2016.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

Karras, T., Laine, S., and Aila, T. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4401–4410, 2019.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

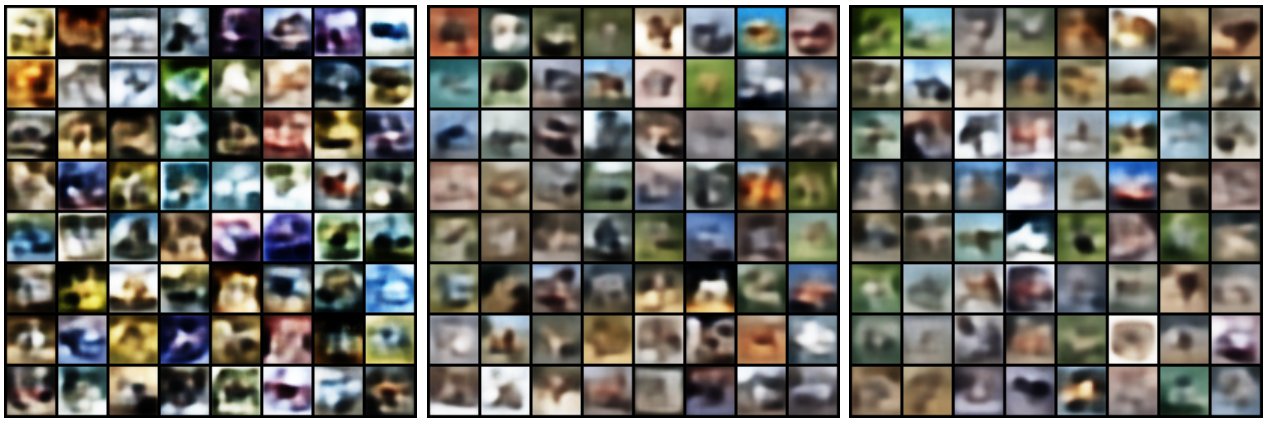
Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. Improving variational inference with inverse autoregressive flow. *arXiv preprint arXiv:1606.04934*, 2016.

Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., and Frey, B. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

Nichol, A. and Dhariwal, P. Improved denoising diffusion probabilistic models. *arXiv preprint arXiv:2102.09672*, 2021.

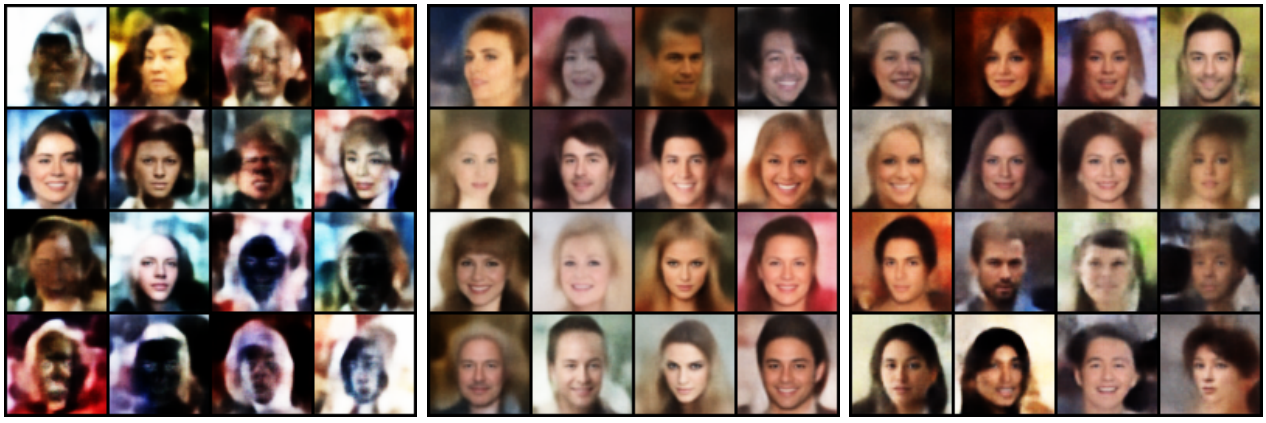
Oord, A. v. d., Kalchbrenner, N., Vinyals, O., Espeholt, L., Graves, A., and Kavukcuoglu, K. Conditional image generation with pixelcnn decoders. *arXiv preprint arXiv:1606.05328*, 2016.

Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.



(a) Gaussian prior (b) NF prior (c) Diffused prior

Figure 1. Samples generated by a VAE trained on CIFAR10 for three different prior models. *The diffusion prior leads to slightly better sampling quality than the Gaussian distribution and similar to the NF prior.*



(a) Gaussian prior (b) NF prior (c) Diffused prior

Figure 2. Samples generated by a VAE trained on CelebA for three different prior models. *The diffusion prior leads to better sampling quality than the Gaussian distribution and similar to the NF prior.*

Rezende, D. and Mohamed, S. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pp. 1530–1538. PMLR, 2015.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pp. 2256–2265. PMLR, 2015.

Sønderby, C. K., Raiko, T., Maaløe, L., Sønderby, S. K., and Winther, O. Ladder variational autoencoders. *arXiv preprint arXiv:1602.02282*, 2016.

Song, Y. and Ermon, S. Generative modeling by estimating gradients of the data distribution. In *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems*, 2019.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Tomczak, J. and Welling, M. Vae with a vampprior. In *International Conference on Artificial Intelligence and Statistics*, pp. 1214–1223. PMLR, 2018.

Vahdat, A. and Kautz, J. Nvae: A deep hierarchical variational autoencoder. *arXiv preprint arXiv:2007.03898*, 2020.

Vahdat, A., Kreis, K., and Kautz, J. Score-based generative modeling in latent space. *arXiv preprint arXiv:2106.05931*, 2021.

5.3 EPILOGUE

5.3.1 *Diffusion in the latent space*

One of the main limitations of diffusion models is their computational inefficiency both at training and synthesis time. Indeed, diffusion models use a one-to-one function and thus work directly in the data space. Training the largest models costs millions of euros without accounting for the cost of architecture search and hyperparameter optimisation [alexjc]. Synthesis is not better, as it requires the successive application of the same U-Net hundreds of times and can take up to a few minutes on modern hardware. Leaving the data space for a lower-dimensional latent space alleviates this limitation but necessitates the development of new algorithms. In addition to the paper presented in this chapter, we now discuss three alternative strategies respectively dubbed *Score-based generative modeling in latent space* [Vahdat et al., 2021], *Diffusion-Decoding Models* [Sinha et al., 2021], and *Stable diffusion* [Rombach et al., 2022].

Concurrently to our work, Vahdat et al. [2021] proposed to train a score-based model, i.e., a continuous-time diffusion model, for modelling the latent space of variational auto-encoder. Similar to us, Vahdat et al. [2021] decompose the ELBO of the VAE into three components,

$$\text{ELBO}(\boldsymbol{x};\phi,\psi) = \underbrace{\mathbb{E}_{q_\psi}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}_0)\right]}_{\text{reconstruction term}} - \underbrace{\mathbb{E}_{q_\psi}\left[\log q(\boldsymbol{z}_0|\boldsymbol{x})\right]}_{\text{encoder entropy}} + \underbrace{\mathbb{E}_{q_\psi}\left[\log p_{\boldsymbol{\phi}}(\boldsymbol{z}_0)\right]}_{\text{cross entropy}}. \tag{5.1}$$

In our case, we lower-bounded the cross entropy (CE) term to get a proper ELBO. We use this ELBO to train the encoder, the decoder, and the diffusion model jointly. However, this approach does not directly work for continuous-time models trained via denoising score matching at multiple noise levels. We refer the reader to Section 4.5.2 for more details about the notations and continous diffusion models. Instead, Vahdat et al. [2021] show that the cross entropy term is equal to

$$\mathbb{E}_{q_\psi}\left[\log p_{\boldsymbol{\phi}}(\boldsymbol{z}_0)\right] = \mathbb{E}_{t\sim\mathcal{U}[0,1]}\left[\frac{g(t)^2}{2}\mathbb{E}_{q(\boldsymbol{z}_t,\boldsymbol{z_0}|\boldsymbol{x})}\left[|\nabla_{\boldsymbol{z}_t}\log q(\boldsymbol{z}_t \mid \boldsymbol{z}_0) - \log p(\boldsymbol{z}_t)|_2^2\right]\right] + C, \tag{5.2}$$

under mild conditions. The symbol t denotes the time; it starts from 0 with structured data and ends in 1 with noise. We can use stochastic gradient descent to optimise this term, the encoder entropy, and reconstruction terms jointly. In order to evaluate Equation (5.2), we first encode the sample ($q(\boldsymbol{z}_0 \mid \boldsymbol{x})$), draw $t \sim \mathcal{U}[0,1]$ and the corresponding $\boldsymbol{z_t} \sim q(\boldsymbol{z}_t \mid \boldsymbol{z}_0)$ that follows a known Normal transition kernel $\mathcal{N}\left(\boldsymbol{\mu}_t(\boldsymbol{z}_0), \sigma_t^2 I\right)$. We notice that Equation (5.2) resembles the ELBO objective used to train discrete-time models, which is what we use to replace the CE. However, there remains a surprising difference between the equation (12) in the paper and Equation (5.2). In our case, the inverse variance $\frac{1}{\sigma_t^2}$ weights each term while Equation (5.2) weights with $g(t)^2$ which is supposed to

have a similar meaning to σ_t^2 in the discrete case. Vahdat et al. [2021] suggest moving away from this weighting to obtain better results in practice. We wonder if there is a connection between this practical trick and the observed discrepancy between the losses of the continuous and discrete latent variables models.

Diffusion-Decoder models [Sinha et al., 2021, D2C] is another architecture that combines VAEs with discrete-time diffusion models. They combine a loss similar to the ELBO we derive in our paper with a self-supervised loss that encourages meaningful latent representations via data augmentation. They also conclude that replacing the CE with a diffusion loss leads to a proper ELBO. In addition, they discuss the ability of latent diffusion models to close "prior holes" resulting from the mismatch between the aggregated posteriors and the prior of VAEs. Finally, they also show that the representations learned by D2C enables effective few-shot conditional generation.

Stable diffusion [Rombach et al., 2022] is arguably one of the most notable contributions to openly accessible machine learning models over the last decades. Rombach et al. [2022] combines an auto-encoder architecture with a discrete-time diffusion model to reduce the computing cost associated with pure diffusion models. Large pre-trained stable diffusion models have been publicly released and have had a significant impact. In a few weeks only, many tools that exploit the computational efficiency and impressive synthesis results of Stable diffusion have appeared. For example, Dream studio is a website that allows synthesising images from text interactively.

In contrast to our work, Stable diffusion does not train the auto-encoder and the diffusion model synchronously. Instead, they first train a vanilla auto-encoder with the combination of a perceptual loss [Zhang et al., 2018] and a patch-based adversarial objective [Dosovitskiy and Brox, 2016; Esser et al., 2021; Yu et al., 2021]. Only then do they train a diffusion model on the latent representations. The diffusion model should eventually learn the distribution over the latent space and allows synthesis. However, this would never happen in practice without additional care. First, Rombach et al. [2022] also regularises the latent space's entropy to avoid high-variances latent spaces that would be hard to learn. They rely on a strong architectural inductive bias that conserves the spatial structure of images but allows perceptual compression. In contrast, the diffusion model completes the semantic compression.

5.3.2 *Behind the scenes*

Our initial motivation for combining diffusion models and VAEs was not to improve VAEs, nor to speed up diffusion models, although these two outcomes are highly relevant in practice. Instead, our broader objective was to enable new types of noise for training diffusion models. As with any other component of the DDPM learning algorithm, the noise model is part of the inductive bias. Adapting the noise to the data modality might thus help to learn diffusion models for new modalities. We thought heat equations would make an excellent inductive bias for image synthesis. Heat equation blurs the image

through time, which amounts to first discarding perceptual information and later the semantic content.

Our idea was to use a VAE formulation to bypass the closed-form gaussian kernel required for obtaining a tractable training objective. We thought using distinct diffusion speeds inside the latent variables of the VAE would eventually encourage disentanglement and allow extracting high and low-level semantic information from images automatically.

Although we did not manage to achieve sufficiently good results to write a publication on it, Rissanen et al. [2022] had a similar idea and achieved state-of-the-art image synthesis with a diffusion model based on the heat equation. This work hints that combining heat diffusion with VAEs to compress data into human-interpretable latent variables representing high and low-level features might still be a valuable research direction.

5.3.3 *Scientific impact*

According to Google Scholar, our article has received five citations between its publication in June 2021 and August 2022. The impact of our work is arguably marginal compared to the publications discussed above, particularly Stable diffusion [Rombach et al., 2022]. However, we believe testing our joint learning of the VAE and diffusion latent prior might still be beneficial. Our idea to place diffusion models for the latent variable distribution was original at the time of its publication. We are glad to notice a scientific and practical interest in this idea.

5.3.4 *Conclusion and opportunities*

Since the publication of this article, diffusion models have become very popular, owing part of their success to the astonishing results achieved by large text-to-images models created by OpenAI [DALL · E 2 Ramesh et al., 2022] and Google [Imagen Saharia et al., 2022]. Diffusion models have also percolated in audio modelling [Kong et al., 2020]. Close to our work, Yu et al. [2022] recently proposed to use energy-based models trained with denoising score matching to model the prior distribution of VAEs for interpretable text modelling.

Shortly, the attractivity of diffusion models for modelling distributions over images, and other data modalities, should stay high. However, diffusion models also have some limitations, and many interesting questions remain. *Why do diffusion models represent high dimensional data such as images so well? Can we fasten the sampling process of diffusion models?* An answer to the former question relies upon the inductive bias of diffusion models; however, which part exactly is an open question. Related to the latter question, pushing further the connections between diffusion models and other probabilistic models, such as normalizing flows and VAEs, might help reduce the sample synthesis's complexity.

Combining auto-encoders and other probabilistic models is not new to diffusion models. Similarly to Stable diffusion, VQ-VAE [Razavi et al., 2019a,b] achieved state-of-the-art image synthesis by combining a classical auto-encoder with an autoregressive model to encode the distribution of the latent variables. Another popular strategy discussed in our paper is to use flexible density approximators such as normalizing flows. Indeed, both flows and autoregressive models provide access to the corresponding density and do not require re-formulating the ELBO. However, the inductive bias of normalizing flows limits their effectiveness for spatially structured data, which is part of the success of Stable diffusion. Encoding spatial structure with normalizing flows is possible [Kingma and Dhariwal, 2018]. However, it is not straightforward because of the strict invertibility requirement. In the next chapter, we show that some normalizing flows have a limited capacity to model distributions.

Retrospectively, the complementarity of auto-encoders and (quasi-)invertible probabilistic models is not surprising. On the one hand, auto-encoders naturally embed the inductive bias that data lie in a low-dimensional manifold which maps to a low-dimensional Euclidean space. However, finding a continuous mapping from the data space to this manifold is difficult. The classical Normal prior used in VAEs enforces an orthogonal latent space which is often hard to identify robustly. On the other hand, (quasi-)invertible models build a dimensionality-preserving invertible mapping from an unstructured noise distribution to the data space. While this is a bad inductive bias when the data lie in a lower-dimensional space, this constraint is beneficial when the data lives in its own space.

Technological progress has merely provided us with more efficient means for going backwards.

Aldous Huxley

6

UNDERSTANDING MODELS

Outline

We show that drawing connections between different classes of models can help us better understand existing classes of models. In particular, we use connections between normalizing flows and Bayesian networks to prove three essential properties of affine normalizing flows. First, we show that stacking multiple transformations in a normalizing flow relaxes independence assumptions and entangles the model distribution. Second, we show that a fundamental leap of capacity emerges when the depth of affine flows exceeds three transformation layers. Third, we prove the non-universality of the affine normalizing flow, regardless of its depth.

6.1 PROLOGUE

This chapter further highlights the relevance of connecting distinct classes of probabilistic models. These connections provide new viewpoints that can help us focus on some models' components and abstract others less relevant to what we want to understand. We derive properties of affine normalizing flows relevant to practitioners. We show that stacking more than three steps of normalizing flows is poorly motivated. Adding more steps unnecessarily slows down the inversion of the flow for no gain in expressivity. We also reveal one failure mode of affine normalizing flows, which implies they are not universal density approximators.

This chapter shall convince the reader that gaining a theoretical understanding of probabilistic models is key to using these models appropriately. For this purpose, taking a new perspective by drawing connections between different types of models, here graphical and deep probabilistic models, is a worthy effort. We can then reason intuitively on the class of models with simple analogies.

6.2 THE PAPER: YOU SAY NORMALIZING FLOWS I SEE BAYESIAN NETWORKS

6.2.1 *Author contributions*

The paper is co-authored by Gilles Louppe and I. As the leading author, I developed the connections between normalizing flows and Bayesian networks, conducted experiments, and wrote the paper. Gilles helped me throughout this project, offered suggestions and actively participated in writing the paper.

6.2.2 *Reading tips*

The reader may skip section 2 which presents normalizing flows and Bayesian networks already introduced in Part i. We encourage the reader to first focus on understanding the relationship between Figure 1 and coupling layers presented in section 3.2. The reader can then focus on understanding section 3.3 which provides the ground for the rest of the paper.

You say Normalizing Flows I see Bayesian Networks

Antoine Wehenkel [1] **Gilles Louppe** [1]

Abstract

Normalizing flows have emerged as an important family of deep neural networks for modelling complex probability distributions. In this note, we revisit their coupling and autoregressive transformation layers as probabilistic graphical models and show that they reduce to Bayesian networks with a pre-defined topology and a learnable density at each node. From this new perspective, we provide three results. First, we show that stacking multiple transformations in a normalizing flow relaxes independence assumptions and entangles the model distribution. Second, we show that a fundamental leap of capacity emerges when the depth of affine flows exceeds 3 transformation layers. Third, we prove the non-universality of the affine normalizing flow, regardless of its depth.

1. Introduction

Normalizing flows [NF, 5] have gained popularity in the recent years because of their unique ability to model complex data distributions while allowing both for sampling and exact density computation. This family of deep neural networks combines a base distribution with a series of invertible transformations while keeping track of the change of density that is caused by each transformation.

Probabilistic graphical models (PGMs) are well-established mathematical tools that combine graph and probability theory to ease the manipulation of joint distributions. They are commonly used to visualize and reason about the set of independencies in probabilistic models. Among PGMs, Bayesian networks [BN, 4] offer a nice balance between readability and modeling capacity. Reading independencies stated by a BN is simple and can be performed graphically with the d-separation algorithm [2].

In this note, we revisit NFs as Bayesian networks. We first briefly review the mathematical grounds of these two worlds. Then, for the first time in the literature, we show that the modeling assumptions behind coupling and autoregressive transformations can be perfectly expressed by distinct classes of BNs. From this insight, we show that stacking multiple transformation layers relaxes independencies and entangles the model distribution. Then, we show that a fundamental change of regime emerges when the NF architecture includes 3 transformation steps or more. Finally, we prove the non-universality of affine normalizing flows.

[1]University of Liège. Correspondence to: Antoine Wehenkel <antoine.wehenkel@uliege.be>.

2. Background

2.1. Normalizing flows

A normalizing flow is defined as a sequence of invertible transformation steps $\mathbf{g}_k : \mathbb{R}^d \to \mathbb{R}^d$ $(k = 1, ..., K)$ that are composed together to create an expressive invertible mapping $\mathbf{g} = \mathbf{g}_1 \circ \cdots \circ \mathbf{g}_K : \mathbb{R}^d \to \mathbb{R}^d$. This mapping can be used to perform density estimation, using $\mathbf{g}(\cdot; \theta) : \mathbb{R}^d \to \mathbb{R}^d$ to map a sample $\mathbf{x} \in \mathbb{R}^d$ to a latent vector $\mathbf{z} \in \mathbb{R}^d$ equipped with a density $p_{\mathbf{z}}(\mathbf{z})$. The transformation $\mathbf{g}$ implicitly defines a density $p(\mathbf{x}; \theta)$ as given by the change of variables formula,

$$p(\mathbf{x}; \theta) = p_{\mathbf{z}}(\mathbf{g}(\mathbf{x}; \theta)) \left| \det J_{\mathbf{g}(\mathbf{x};\theta)} \right| ,$$

where $J_{\mathbf{g}(\mathbf{x};\theta)}$ is the Jacobian of $\mathbf{g}(\mathbf{x}; \theta)$ with respect to $\mathbf{x}$. The resulting model is trained by maximizing the likelihood of the data $\{\mathbf{x}^1, ..., \mathbf{x}^N\}$. NFs can also be used for data generation tasks while keeping track of the density of the generated samples such as to improve the latent distribution in variational auto-encoders [5]. In the rest of this paper, we will not distinguish between $\mathbf{g}$ and $\mathbf{g}_k$ when the discussion will be focused on only one of these steps $\mathbf{g}_k$.

In general, steps $\mathbf{g}$ can take any form as long as they define a bijective map. Here, we focus on a sub-class of normalizing flows for which these steps can be mathematically described as

$$\mathbf{g}(\mathbf{x}) = \left[g^1(x_1; \mathbf{c}^1(\mathbf{x})) \quad \dots \quad g^d(x_d; \mathbf{c}^d(\mathbf{x})) \right] ,$$

where the $\mathbf{c}^i$ are denoted as the **conditioners** and constrain the structure of the Jacobian of $\mathbf{g}$. The functions g^i, partially parameterized by their conditioner, must be invertible with respect to their input variable x_i. These are usually defined as affine or strictly monotonic functions, with the latter being the most general class of invertible scalar continuous functions. In this note, we mainly discuss affine normalizers

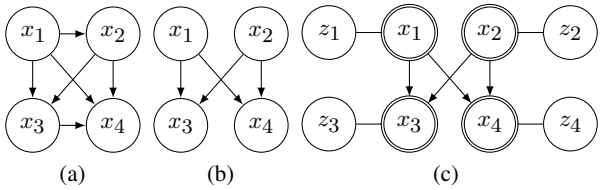


Figure 1. Bayesian networks for single-step normalizing flows on a vector $\mathbf{x} = [x_1, x_2, x_3, x_4]^T$. (**a**) BN for an autoregressive conditioner. (**b**) BN for a coupling conditioner. (**c**) Pseudo BN for a coupling conditioner, with the latent variables shown explicitly. Double circles stand for deterministic functions of the parents and non-directed edges stand for bijective relationships.

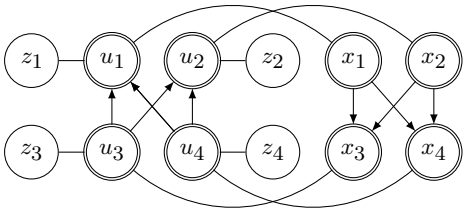


Figure 2. A Bayesian network equivalent to a 2-step normalizing flow based on coupling layers. Independence statements are relaxed by the second step.

that can be expressed as $g(x; m, s) = x \exp(s) + m$ where $m \in \mathbb{R}$ and $s \in \mathbb{R}$ are computed by the conditioner.

2.2. Bayesian networks

Bayesian networks allow for a compact and natural representation of probability distributions by exploiting conditional independence. More precisely, a BN is a directed acyclic graph (DAG) which structure encodes for the conditional independencies through the concept of d-separation [2]. Equivalently, its skeleton supports an efficient factorization of the joint distribution.

A BN is able to model a distribution p if and only if it is an I-map with respect to p. That is, iff the set of independencies stated by the BN structure is a subset of the independencies that holds for p. Equivalently, a BN is a valid representation of a random vector $\mathbf{x}$ iff its density $p_{\mathbf{x}}(\mathbf{x})$ can be factorized by the BN structure as

$$p_{\mathbf{x}}(\mathbf{x}) = \prod_{i=1}^{d} p(x_i|\mathcal{P}_i), \tag{1}$$

where $\mathcal{P}_i = \{j : A_{i,j} = 1\}$ denotes the set of parents of the vertex i and $A \in \{0, 1\}^{d \times d}$ is the adjacency matrix of the BN. As an example, Fig. 1a is a valid BN for any distribution over $\mathbf{x}$ because it does not state any independence, leading to a factorization that results in the chain rule.

3. Normalizing flows as Bayesian networks

3.1. Autoregressive conditioners

Autoregressive conditioners can be expressed as

$$\mathbf{c}^i(\mathbf{x}) = \mathbf{h}^i\left(\begin{bmatrix} x_1 & \dots & x_{i-1} \end{bmatrix}^T\right),$$

where $\mathbf{h}^i : \mathbb{R}^{i-1} \to \mathbb{R}^l$ are functions of the first $i-1$ components of $\mathbf{x}$ and whose output size depends on architectural choices. These conditioners constrain the Jacobian of $\mathbf{g}$ to be lower triangular, making the computation of its determinant $\mathcal{O}(d)$. The multivariate density $p(\mathbf{x}; \theta)$ induced by $\mathbf{g}(\mathbf{x}; \theta)$ and $p_{\mathbf{z}}(\mathbf{z})$ can be expressed as a product of d univariate conditional densities,

$$p(\mathbf{x}; \theta) = p(x_1; \theta) \prod_{i=2}^{d} p(x_i|\mathbf{x}_{1:i-1}; \theta). \tag{2}$$

When $p_{\mathbf{z}}(\mathbf{z})$ is a factored distribution $p_{\mathbf{z}}(\mathbf{z}) = \prod_{i=1}^{d} p(z_i)$, we identify that each component z_i coupled with the corresponding function g^i encodes for the conditional $p(x_i|\mathbf{x}_{1:i-1}; \theta)$. An explicit connection between BNs and autoregressive conditioners can be made if we define $\mathcal{P}_i = \{x_1, \dots, x_{i-1}\}$ and compare (2) with (1). Therefore, and as illustrated in Fig. 1a, autoregressive conditioners can be seen as a way to model the conditional factors of a BN that does not state any independence.

3.2. Coupling conditioners

Coupling conditioners [1] are another popular type of conditioners used in normalizing flows. The conditioners $\mathbf{c}^i$ made from coupling layers are defined as

$$\mathbf{c}^i(\mathbf{x}) = \begin{cases} \underline{\mathbf{h}}^i & \text{if} \quad i < k \\ \mathbf{h}^i(\mathbf{x}_{<k}) & \text{if} \quad i \geq k \end{cases}$$

where the $\underline{\mathbf{h}}^i$ symbol define constant values. As for autoregressive conditioners, the Jacobian of $\mathbf{g}$ made of coupling layers is lower triangular. Assuming a factored latent distribution, the density associated with these conditioners can be written as follows:

$$p(\mathbf{x}; \theta) = \prod_{i=1}^{k-1} p(x_i) \prod_{i=k}^{d} p(x_i|\mathbf{x}_{<k}),$$

$$\text{where} \quad p(x_i) = p(g^i(x_i; \underline{\mathbf{h}}^i)) \frac{\partial g^i(x_i; \underline{\mathbf{h}}^i)}{\partial x_i}$$

$$\text{and} \quad p(x_i|\mathbf{x}_{<k}) = p(g^i(x_i; \mathbf{h}^i(\mathbf{x}_{<k}))) \frac{\partial g^i(x_i; \mathbf{h}^i(\mathbf{x}_{<k}))}{\partial x_i}.$$

The factors define valid 1D conditional probability distributions because they can be seen as 1D changes of variables between z_i and x_i. This factorization can be graphically

expressed by a BN as shown in Fig. 1b. In addition, we can see Fig. 1b as the marginal BN of Fig. 1c which fully describes the stochastic process modeled by a NF that is made of a single transformation step and a coupling conditioner. In contrast to autoregressive conditioners, coupling layers are not by themselves universal density approximators, even when associated with very expressive normalizers g^i. Indeed, d-separation reveals independencies stated by this class of BN, such as the conditional independence between each pair in $\mathbf{x}_{\geq k}$ knowing $\mathbf{x}_{<k}$. These independence statements do not hold in general.

3.3. Stacking transformation steps

In practice, the invertible transformations discussed above are often stacked together in order to increase the representation capacity of the flow, with the popular good practice of permuting the vector components between two transformation steps. The structural benefits of this stacking strategy can be explained from the perspective of the underlying BN.

First, a BN that explicitly includes latent variables is faithful as long as the sub-graph made only of those latent nodes is an I-map with respect to their distribution. Normalizing flows composed of multiple transformation layers can therefore be viewed as single transformation flows whose latent distribution is itself recursively modeled by a normalizing flow. As an example, Fig. 2 illustrates a NF made of two transformation steps with coupling conditioners. It can be observed that the latent vector $\mathbf{u}$ is itself a normalizing flow whose distribution can be factored out by a class of BN.

Second, from the BN associated to a NF, we observe that additional layers relax the independence assumptions defined by its conditioners. The distribution modeled by the flow gets more entangled at each additional layer. For example, Fig. 2 shows that for coupling layers, the additional steps relax the strong conditional independencies between x_1 and x_2 of the single transformation NF of Fig. 1c. Indeed, we can observe from the figure that x_1 and x_2 have common ancestors (z_3 and z_4) whereas they are clearly assumed independent in Fig. 1b.

In general, we note that edges between two nodes in a BN do not model dependence, only the absence of edges does model independence. However, because some of the relationship between nodes are bijective, this implies that these nodes are strictly dependent on each other. We represent these relationships with undirected edges in the BN, as it can be seen in Fig. 2.

4. Affine normalizing flows unlock their capacity with 3 transformation steps

We now show how some of the limitations of affine normalizers can be relaxed by stacking multiple transformation steps. We also discuss why some limitations cannot be relaxed even with a large number of transformation steps. We intentionally put aside monotonic normalizers because they have already been proven to lead to universal density approximators when the conditioner is autoregressive [3]. We focus our discussion on a multivariate normal with an identity covariance matrix as base distribution $p_{\mathbf{z}}(\mathbf{z})$.

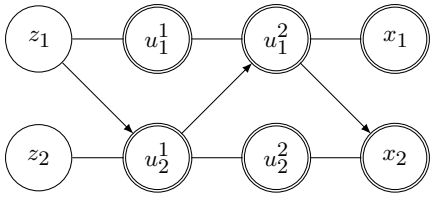


Figure 3. The Bayesian network of a three-steps normalizing flow on vector $\mathbf{x} = [x_1, x_2]^T \in \mathbb{R}^4$. It can be observed that the distribution of the intermediate latent variables, and at the end of the vector $\mathbf{x}$, becomes more entangled at each additional transformation step. Considering the undirected edges as affine relationships, we see that while u_1^1 and u_2^1 are affine transformations of the latent $\mathbf{z}$, the vector $\mathbf{x}$ cannot be expressed as a linear function of the latent $\mathbf{z}$.

We first observe from Fig. 1 that in a NF with a single transformation step at least one component of $\mathbf{x}$ is a function of only one latent variable. If the normalizer is affine and the base distribution is normal, then this necessarily implies that the marginal distribution of this component is normal as well, which will very likely not lead to a good fit. We easily see that adding steps relaxes this constraint. A more interesting question to ask is what exactly the modeling capacity gain for each additional step of affine normalizer is. Shall we add steps to increase capacity or shall we increase the capacity of each step instead? We first discuss a simple 2-dimensional case, which has the advantage of unifying the discussion for autoregressive and coupling conditioners, and then extend it to a more general setting.

Affine NFs made of a single transformation step induce strong constraints on the form of the density. In particular, these models implicitly assume that the data distribution can be factorized as a product of conditional normal distributions. These assumptions are rexaled when accumulating steps in the NF. As an example, Fig. 3 shows the equivalent BN of a 2D NF composed of 3 steps. This flow is mathematically described with the following set of equations:

$$
\begin{aligned}
u_1^1 &:= z_1 & u_2^1 &:= \exp(s_2^1(z_1))z_2 + m_2^1(z_1) \\
u_2^2 &:= u_2^1 & u_1^2 &:= \exp(s_1^2\left(u_2^1\right))z_1 + m_1^2\left(u_2^1\right) \\
x_1 &:= u_1^2 & x_2 &:= \exp(s_2^3\left(u_1^2\right))u_2^2 + m_2^3\left(u_1^2\right)
\end{aligned}
$$

From these equations, we see that after one step the latent variables u_1^1 and u_2^1 are respectively normal and conditionally normal. This is relaxed with the second step, where the latent variable u_1^2 is a non-linear function of two random variables distributed normally (by assumption on the dis-

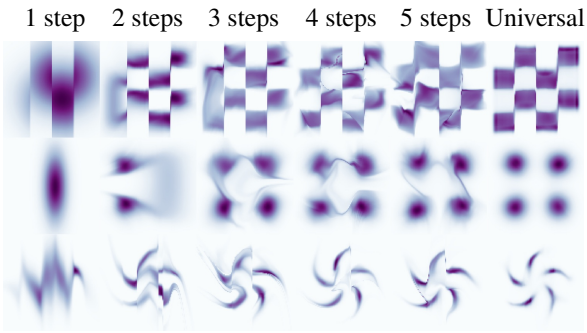


Figure 4. Evolution of an affine normalizing flow's capacity as the number of steps increases. For comparison, the density learned by a universal density approximator is shown on the last column.

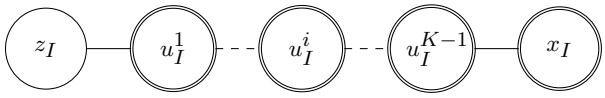


Figure 5. The equivalent BN of a component with a unique latent variable as ancestor.

tribution of z_1 and z_2). However, u_2^2 is a stochastic affine transformation of a normal random variable. In addition, we observe that the expression of u_1^2 is strictly more expressive than the expression of u_2^2. Finally, x_1 and x_2 are non-linear functions of both latent variables z_1 and z_2. Assuming that the functions s_j^i and m_j^i are universal approximators, we argue that the stochastic process that generates x_1 and the one that generates x_2 are as expressive as each other. Indeed, by making the functions arbitrarily complex the transformation for x_1 could be made arbitrarily close to the transformations for x_2 and vice versa. This is true because both transformations can be seen as an affine transformation of a normal random variables whose scaling and offset factors are non-linear arbitrarily expressive transformations of all the latent variables. Because of this equilibrium between the two expressions, additional steps do not improve the modeling capacity of the flow. The same observations can be made empirically as illustrated in Fig. 3 for 2-dimensional toy problems. A clear leap of capacity occurs from 2-step to 3-step NFs, while having 4 steps or more does not result in any noticeable improvement when s_j^i and m_j^i already have enough capacity.

For $d > 2$, autoregressive and coupling conditioners do not correspond to the same set of equations or BN. However, if the order of the vector is reversed between two transformation steps, the discussion generalizes to any value of d for both conditioners. Indeed, in both cases each component of the intermediate latent vectors can be seen as having a set of conditioning variables and a set of independent variables. At each successive step the indices of the non-conditioning variables are exchanged with the conditioning ones and thus any output vector's component can be expressed either as a component of the vector form of x_1 or of x_2.

5. Affine normalizing flows are not universal density approximators

We argue that affine normalizers do not lead to universal density approximators in general, even for an infinite number of steps. In the following, we assume again that the latent variables are distributed according to a normal distribution with a unit covariance matrix.

To prove the non-universality of affine normalizing flows, one only needs to provide a counter-example. Let us consider the simple setup in which one component x_I of the random vector $\mathbf{x}$ is independent from all the other components. Let us also assume that x_I is distributed under a non-normal distribution. We can then consider two cases. First, x_I has only one component of the latent vector $\mathbf{z}$ as an ancestor. This implies that the equivalent BN would be as in Fig. 5, hence that x_I is a linear function of this ancestor and is therefore normally distributed. Else, x_I has n components of the latent vector as ancestors. However, this second case would imply that at least one undirected edge is removed from the original BN considered in Section 3.3. This cannot happen since it would deadly hurt the bijectivity of the flow.

Besides proving the non-universality of affine NFs, this discussion provides the important insight that when affine normalizers must transform non-linearly some latent variables they introduce dependence in the model of the distribution. In some sense, this means that the additional disorder required to model this non-normal component is performed at the cost of some loss in entropy caused by mutual information between the random vector components.

6. Summary

In this preliminary work, we have revisited normalizing flows from the perspective of Bayesian networks. We have shown that stacking multiple transformations in a normalizing flow relaxes independence assumptions and entangles the model distribution. Then, we have shown that affine normalizing flows benefit from having at least 3 transformation layers. Finally, we demonstrated that they remain non-universal density approximators regardless of their depths.

We hope these results will give practitioners more intuition in the design of normalizing flows. We also believe that this work may lead to further research. First, unifying Bayesian networks and normalizing flows could be pushed one step further with conditioners that are specifically designed to model Bayesian networks. Second, the study could be extended for other type of normalizing flows such as non-autoregressive monotonic flows. Finally, we believe this study may spark research at the intersection of structural equation modeling, causal networks and normalizing flows.

Acknowledgments

The authors would like to thank Matthia Sabatelli, Johann Brehmer and Louis Wehenkel for proofreading the manuscript. Antoine Wehenkel is a research fellow of the F.R.S.-FNRS (Belgium) and acknowledges its financial support. Gilles Louppe is recipient of the ULiège - NRB Chair on Big data and is thankful for the support of NRB.

References

[1] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real nvp. In *International Conference in Learning Representations*, 2017.

[2] D. Geiger, T. Verma, and J. Pearl. d-separation: From theorems to algorithms. In *Machine Intelligence and Pattern Recognition*, volume 10, pages 139–148. Elsevier, 1990.

[3] C.-W. Huang, D. Krueger, A. Lacoste, and A. Courville. Neural autoregressive flows. In *International Conference on Machine Learning*, pages 2083–2092, 2018.

[4] J. Pearl. Bayesian networks. 2011.

[5] D. Rezende and S. Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538, 2015.

6.3 EPILOGUE

6.3.1 *Scientific impact*

After the publication of this short article, the idea of combining NFs and BNs has led to Graphical Normalizing Flows presented in Chapter 8. At the same workshop, Khemakhem et al. [2020] presented connections between autoregressive NFs and causal networks, a sub-class of BNs featured with causal interpretation. Subsequent work has also focused on combining probabilistic graphical models with normalizing flows such as Mouton and Kroon [2022a,b].

We regret that our work has not gained more attention from practitioners who use affine normalizing flows. As of August 2022, this article has received 8 citations according to Google Scholar since its publication in July 2019. It is unfortunately still widespread to stack tens of NF steps (e.g., [Dax et al.]). This inconsistent usage still happens, although other work reached the same conclusion regarding the number of steps in affine NFs. In particular, Koehler et al. [2021] shows that three steps of affine coupling flows are sufficient to express any distribution on $\mathbb{R}^d$ when d is even.

On the one hand, Koehler et al. [2021]'s result aligns with ours; it confirms that stacking more than three steps does not increase the expressivity of an NF. On the other hand, it also contradicts our statement about the non-universality of affine flows. Similarly, Huang et al. [2020] showed that normalizing flows padded with zeros are universal density approximators. However, these networks are not trainable anymore via direct MLE and may issue numerical instabilities.

We explain the mismatch between our result and Koehler et al. [2021] by observing that, similarly to Huang et al. [2020], Koehler et al. [2021] allows degenerate flows. These flows exhibit exploding or vanishing Jacobians, which corresponds to non-invertibility and was implicitly discarded in our discussion. In addition, our definition of universality is different from theirs. In our case, universality occurs when the model class contains all possible continuous distributions. In contrast, Koehler et al. [2021] defines universality as the ability to approach any distribution as close as wanted in Wasserstein distance. Our study highlights the numerical instabilities of modelling independent multi-modal distributions with coupling layers. This issue is related to the problem pointed out by Behrmann et al. [2021], which shows that exploding Jacobians cause numerical instabilities with the training and sampling of NFs and can reduce their effectiveness.

6.3.2 *Conclusion and opportunities*

It is now clear that stacking more than three steps does not increase the expressivity of the class of models. However, we must acknowledge that our study hides the positive impacts additional steps might have on training the flow in practice. Indeed the log-likelihood of a flow directly uses the Jacobian of each step; this may act as some

skip connection in the gradient flow and potentially overcome numerical instabilities at training time. Understanding this aspect of normalizing flows should help practitioners efficiently parameterise these probabilistic models.

At a higher level, this chapter has shown in what sense drawing connections between distinct model classes may provide insights for a better understanding of models. Building such understanding is relevant for the real-world application of deep probabilistic modelling because it helps practitioners correctly use the model's key features. This chapter also highlights the limitations of affine transformations. This limitation motivates the next chapter, which introduces more expressive transformations.

All generalizations, with the possible exception of this one, are false.

Kurt Gödel

7

IMPROVING MODELS

Outline

In this chapter, we improve the expressivity of deep probabilistic models; we introduce unconstrained monotonic neural networks, a new neural parameterisation of monotonic functions. Architectures that ensure monotonicity typically enforce constraints on weights and activation functions, limiting the expressiveness of the resulting transformations. In contrast, unconstrained monotonic neural networks lean on the insight that monotonic functions have sign-constant derivatives. Hence, any free-form neural network with a positive output satisfies this simple condition. We define a new class of density approximators by combining these networks within autoregressive flows. This new class is a universal approximator of continuous distributions. We demonstrate the effectiveness of this new transformation on density estimation experiments.

7.1 PROLOGUE

Finding an appropriate parameterisation of deep probabilistic models is essential in practice. As for any machine learning model, we aim to find a flexible parameterisation and satisfy prescribed constraints. These constraints may take many forms. For instance, hierarchical layers structure the transformation into discrete processing steps. Convolutional neural networks respect time or space equivariance, which are appropriate constraints for structured signals. Autoregressive layers causally process the input and lead to a tractable likelihood. Thus, an essential part of research in DPMs is devoted to finding new differentiable layers that match the requirements of certain DPMs.

One particular class of models that forces us to invent specific neural parameterisations are normalizing flows. These models require bijective transformations, which are not guaranteed with free-form neural networks. A common solution is to combine scalar invertible transformations with a constrained structure of the Jacobian. For instance, autoregressive or coupling layers enforce a triangular Jacobian. In addition, Rezende and Mohamed [2015], and shortly after Kingma et al. [2016]; Dinh et al. [2017], use sign-constant affine layers to parameterise invertible scalar transformations. The previous chapter has highlighted the limitations of these transformations. Their inability to split the density of a unimodal base distribution into multiple modes. This limitation lies in

the lack of expressivity of affine transform that only plays with the first two modes of the base distribution.

This chapter introduces a new parameterisation of monotonic transformations with neural networks. The term monotonic is a synonym for continuously bijective scalar functions. Thus, monotonic functions are appealing parameterisations for normalizing flows. We observe that having a constant-sign first-order derivative enforces a monotonic behaviour in any continuous function. We propose to parameterise this first-order derivative rather than the monotonic function itself. This parameterisation allows us to use any neural network as long as its output is signed-constant, which we induce with an appropriate output activation function. We discuss the consequences of our new architecture in detail and compare it to alternative parameterisations in the paper and Section 7.3.

7.2 THE PAPER: UNCONSTRAINED MONOTONIC NEURAL NETWORKS

7.2.1 *Author contributions*

Gilles Louppe and I co-authored the paper. The idea of enforcing monotonicity via sign-constant first-order derivative came out during a discussion with Gilles. It is attributable to him. I had the original ideas of implicit differentiation via the Leibniz rule and the use of binary search for inverting the transformation. As the leading author, I wrote the code for the Clenshaw-Curtis quadrature and its implicit differentiation, together with the code for the autoregressive normalizing flows and corresponding experiments. Gilles gave substantial help writing the paper.

7.2.2 *Reading tips*

The reader can skip section 3.1 and 3.2, which describe autoregressive normalizing flows and are very similar to the corresponding section in the background. The rest of the paper flows by itself.

Unconstrained Monotonic Neural Networks

Antoine Wehenkel
University of Liège

Gilles Louppe
University of Liège

Abstract

Monotonic neural networks have recently been proposed as a way to define invertible transformations. These transformations can be combined into powerful autoregressive flows that have been shown to be universal approximators of continuous probability distributions. Architectures that ensure monotonicity typically enforce constraints on weights and activation functions, which enables invertibility but leads to a cap on the expressiveness of the resulting transformations. In this work, we propose the Unconstrained Monotonic Neural Network (UMNN) architecture based on the insight that a function is monotonic as long as its derivative is strictly positive. In particular, this latter condition can be enforced with a free-form neural network whose only constraint is the positiveness of its output. We evaluate our new invertible building block within a new autoregressive flow (UMNN-MAF) and demonstrate its effectiveness on density estimation experiments. We also illustrate the ability of UMNNs to improve variational inference.

1 Introduction

Monotonic neural networks have been known as powerful tools to build monotone models of a response variable with respect to individual explanatory variables [Archer and Wang, 1993, Sill, 1998, Daniels and Velikova, 2010, Gupta et al., 2016, You et al., 2017]. Recently, strictly monotonic neural networks have also been proposed as a way to define invertible transformations. These transformations can be combined into effective autoregressive flows that can be shown to be universal approximators of continuous probability distributions. Examples include Neural Autoregressive Flows [NAF, Huang et al., 2018] and Block Neural Autoregressive Flows [B-NAF, De Cao et al., 2019]. Architectures that ensure monotonicity typically enforce constraints on weight and activation functions, which enables invertibility but leads to a cap on the expressiveness of the resulting transformations. For neural autoregressive flows, this does not impede universal approximation but typically requires either complex conditioners or a composition of multiple flows.

Nevertheless, autoregressive flows defined as stacks of reversible transformations have proven to be quite efficient for density estimation of empirical distributions [Papamakarios et al., 2019, 2017, Huang et al., 2018], as well as to improve posterior modeling in Variational Auto-Encoders (VAE) [Germain et al., 2015, Kingma et al., 2016, Huang et al., 2018]. Practical successes of these models include speech synthesis [van den Oord et al., 2016, Oord et al., 2018], likelihood-free inference [Papamakarios et al., 2019], probabilistic programming [Tran et al., 2017] and image generation [Kingma and Dhariwal, 2018]. While stacking multiple reversible transformations improves the capacity of the full transformation to represent complex probability distributions, it remains unclear which class of reversible transformations should be used.

In this work, we propose a class of reversible transformations based on a new Unconstrained Monotonic Neural Network (UMNN) architecture. We base our contribution on the insight that a function is monotonic as long as its derivative is strictly positive. This latter condition can be enforced with a free-form neural network whose only constraint is for its output to remain strictly positive.

We summarize our contributions as follows:

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

- We introduce the Unconstrained Monotonic Neural Network (UMNN) architecture, a new reversible scalar transformation defined via a free-form neural network.
- We combine UMNN transformations into an autoregressive flow (UMNN-MAF) and we demonstrate competitive or state-of-the-art results on benchmarks for normalizing flows.
- We empirically illustrate the scalability of our approach by applying UMNN on high dimensional density estimation problems.

2 Unconstrained monotonic neural networks

Our primary contribution consists in a neural network architecture that enables learning arbitrary monotonic functions. More specifically, we want to learn a strictly monotonic scalar function $F(x;\psi) : \mathbb{R} \to \mathbb{R}$ without imposing strong constraints on the expressiveness of the hypothesis class. In UMNNs, we achieve this by only imposing the derivative $f(x;\psi) = \frac{\partial F(x;\psi)}{\partial x}$ to remain of constant sign or, without loss of generality, to be strictly positive. As a result, we can parameterize the bijective mapping $F(x;\psi)$ via its strictly positive derivative $f(x;\psi)$ as

$$F(x;\psi) = \int_0^x f(t;\psi)\ \mathrm{d}t + \underbrace{F(0;\psi)}_{\beta}, \tag{1}$$

where $f(t;\psi) : \mathbb{R} \to \mathbb{R}_+$ is a strictly positive parametric function and $\beta \in \mathbb{R}$ is a scalar. We make f arbitrarily complex using an unconstrained neural network whose output is forced to be strictly positive through an ELU activation unit increased by 1. ψ denotes the parameters of this neural network.

Forward integration The forward evaluation of $F(x;\psi)$ requires solving the integral in Equation (1). While this might appear daunting, such integrals can often be efficiently approximated numerically using Clenshaw-Curtis quadrature. The better known trapezoidal rule, which corresponds to the two-point Newton-Cotes quadrature rule, has an exponential convergence when the integrand is periodic and the range of integration corresponds to its period. Clenshaw-Curtis quadrature takes advantage of this property by using a change of variables followed by a cosine transform. This extends the exponential convergence of the trapezoidal rule for periodic functions to any Lipschitz continuous function. As a result, the number of evaluation points required to reach convergence grows with the Lipschitz constant of the function.

Backward integration Training the integrand neural network f requires evaluating the gradient of F with respect to its parameters. While this gradient could be obtained by backpropagating directly through the integral solver, this would also result in a memory footprint that grows linearly with the number of integration steps. Instead, the derivative of an integral with respect to a parameter ω can be expressed with the Leibniz integral rule:

$$\frac{\mathrm{d}}{\mathrm{d}\omega}\left(\int_{a(\omega)}^{b(\omega)} f(t;\omega)\ \mathrm{d}t\right) = f(b(\omega);\omega)\frac{\mathrm{d}}{\mathrm{d}\omega}b(\omega) - f(a(\omega);\omega)\frac{\mathrm{d}}{\mathrm{d}\omega}a(\omega) + \int_{a(\omega)}^{b(\omega)} \frac{\partial}{\partial\omega} f(t;\omega)\ \mathrm{d}t\,. \tag{2}$$

Applying Equation (2) to evaluate the derivative of Equation (1) with respect to the parameters ψ, we find

$$\begin{aligned}\nabla_\psi F(x;\psi) &= f(x;\psi)\nabla_\psi(x) - f(0;\psi)\nabla_\psi(0) + \int_0^x \nabla_\psi f(t;\psi)\ \mathrm{d}t + \nabla_\psi\beta \\ &= \int_0^x \nabla_\psi f(t;\psi)\ \mathrm{d}t + \nabla_\psi\beta.\end{aligned} \tag{3}$$

When using a UMNN block in a neural architecture, it is also important to be able to compute its derivative with respect to its input x. In this case, applying Equation (2) leads to

$$\frac{\mathrm{d}}{\mathrm{d}x}F(x;\psi) = f(x;\psi). \tag{4}$$

Equations (3) and (4) make the memory footprint for the backward pass independent from the number of integration steps, and therefore also from the desired accuracy. Indeed, instead of computing the gradient of the integral (which requires keeping track of all the integration steps), we integrate the gradient (which is memory efficient, as this corresponds to summing gradients at different evaluation points). We provide the pseudo-code of the forward and backward passes using Clenshaw-Curtis quadrature in Appendix B.

Numerical inversion In UMMNs, the modeled monotonic function F is arbitrary. As a result, computing its inverse cannot be done analytically. However, since F is strictly monotonic, it admits a unique inverse x for any point $y = F(x; \psi)$ in its image, therefore inversion can be computed efficiently with common root-finding algorithms. In our experiments, search algorithms such as the bisection method proved to be fast enough.

3 UMNN autoregressive models

3.1 Normalizing flows

A Normalizing Flow [NF, Rezende and Mohamed, 2015] is defined as a sequence of invertible transformations $\mathbf{u}_i : \mathbb{R}^d \to \mathbb{R}^d$ $(i = 1, ..., k)$ composed together to create an expressive invertible mapping $\mathbf{u} = \mathbf{u}_1 \circ \cdots \circ \mathbf{u}_k : \mathbb{R}^d \to \mathbb{R}^d$. It is common for normalizing flows to stack the same parametric function $\mathbf{u}_i$ (with different parameters values) and to reverse variables ordering after each transformation. For this reason, we will focus on how to build one of these repeated transformations, which we further refer to as $\mathbf{g} : \mathbb{R}^d \to \mathbb{R}^d$.

Density estimation NFs are most commonly used for density estimation, that map empirical samples to unstructured noise. Using normalizing flows, we define a bijective mapping $\mathbf{u}(\cdot; \theta) : \mathbb{R}^d \to \mathbb{R}^d$ from a sample $\mathbf{x} \in \mathbb{R}^d$ to a latent vector $\mathbf{z} \in \mathbb{R}^d$ equipped with a density $p_Z(\mathbf{z})$. The transformation $\mathbf{u}$ implicitly defines a density $p(\mathbf{x}; \theta)$ as given by the change of variables formula,

$$p(\mathbf{x}; \theta) = p_Z(\mathbf{u}(\mathbf{x}; \theta)) \left| \det J_{\mathbf{u}(\mathbf{x};\theta)} \right|, \tag{5}$$

where $J_{\mathbf{u}(\mathbf{x};\theta)}$ is the Jacobian of $\mathbf{u}(\mathbf{x}; \theta)$ with respect to $\mathbf{x}$. The resulting model is trained by maximizing the likelihood of the data $\{\mathbf{x}^1, ..., \mathbf{x}^N\}$.

Variational auto-encoders NFs are also used in VAE to improve posterior modeling. In this case, a normalizing flow transforms a distribution p_Z into a complex distribution q which can better model the variational posterior. The change of variables formula yields

$$q(\mathbf{u}(\mathbf{z}; \theta)) = p_Z(\mathbf{z}) \left| \det J_{\mathbf{u}(\mathbf{z};\theta)} \right|^{-1}. \tag{6}$$

3.2 Autoregressive transformations

To be of practical use, NFs must be composed of transformations for which the determinant of the Jacobian can be computed efficiently, otherwise its evaluation would be running in $\mathcal{O}(d^3)$. A common solution consists in making the transformation $\mathbf{g}$ autoregressive, i.e., such that $\mathbf{g}(\mathbf{x}; \theta)$ can be rewritten as a vector of d scalar functions,

$$\mathbf{g}(\mathbf{x}; \theta) = \begin{bmatrix} g^1(x_1; \theta) & \dots & g^i(\mathbf{x}_{1:i}; \theta) & \dots & g^d(\mathbf{x}_{1:d}; \theta) \end{bmatrix},$$

where $\mathbf{x}_{1:i} = \begin{bmatrix} x_1 & \dots & x_i \end{bmatrix}^T$ is the vector including the i first elements of the full vector $\mathbf{x}$. The Jacobian of this function is lower triangular, which makes the computation of its determinant $\mathcal{O}(d)$. Enforcing the bijectivity of each component g^i is then sufficient to make $\mathbf{g}$ bijective as well.

For the multivariate density $p(\mathbf{x}; \theta)$ induced by $\mathbf{g}(\mathbf{x}; \theta)$ and $p_Z(\mathbf{z})$, we can use the chain rule to express the joint probability of $\mathbf{x}$ as a product of d univariate conditional densities,

$$p(\mathbf{x}; \theta) = p(x_1; \theta) \prod_{i=1}^{d-1} p(x_{i+1} | \mathbf{x}_{1:i}; \theta). \tag{7}$$

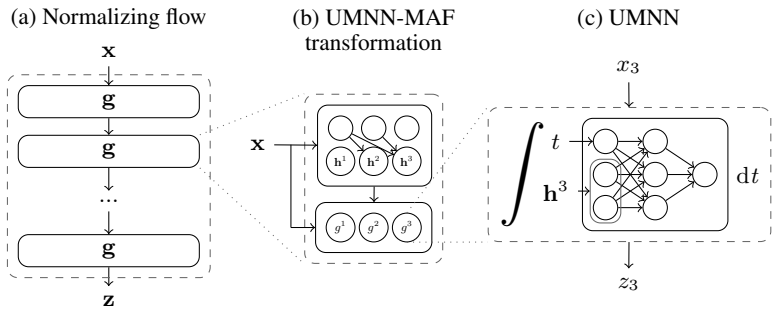


Figure 1: **(a)** A normalizing flow made of repeated UMNN-MAF transformations $\mathbf{g}$ with identical architectures. **(b)** A UMNN-MAF which transforms a vector $\mathbf{x} \in \mathbb{R}^3$. **(c)** The UMNN network used to map x_3 to z_3 conditioned on the embedding $\mathbf{h}^3(\mathbf{x}_{1:2})$.

When $p_Z(\mathbf{z})$ is a factored distribution $p_Z(\mathbf{z}) = \prod_{i=1}^d p(z_i)$, we identify that each component z_i coupled with the corresponding function g^i encodes for the conditional $p(x_i|\mathbf{x}_{1:i-1}; \theta)$. Autoregressive transformations strongly rely on the expressiveness of the scalar functions g^i. In this work, we propose to use UMNNs to create powerful bijective scalar transformations.

3.3 UMNN autoregressive transformations (UMNN-MAF)

We now combine UMNNs with an embedding of the conditioning variables to build invertible autoregressive functions g^i. Specifically, we define

$$\begin{aligned} g^i(\mathbf{x}_{1:i}; \theta) &= F^i(x_i, \mathbf{h}^i(\mathbf{x}_{1:i-1}; \phi^i); \psi^i) \\ &= \int_0^{x_i} f^i(t, \mathbf{h}^i(\mathbf{x}_{1:i-1}; \phi^i); \psi^i)\ \mathrm{d}t + \beta^i(\mathbf{h}^i(\mathbf{x}_{1:i-1}; \phi^i)), \end{aligned} \tag{8}$$

where $\mathbf{h}^i(\cdot; \phi^i) : \mathbb{R}^{i-1} \to \mathbb{R}^q$ is a q-dimensional neural embedding of the conditioning variables $\mathbf{x}_{1:i-1}$ and $\beta(\cdot)^i : \mathbb{R}^{i-1} \to \mathbb{R}$. Both degenerate into constants for $g^1(\mathbf{x}_1)$. The parameters θ of the whole transformation $\mathbf{g}(\cdot; \theta)$ is the union of all parameters ϕ^i and ψ^i. For simplicity we remove the parameters of the networks by rewriting $f^i(\cdot; \psi^i)$ as $f^i(\cdot)$ and $\mathbf{h}^i(\cdot; \phi^i)$ as $\mathbf{h}^i(\cdot)$.

In our implementation, we use a Masked Autoregressive Network [Germain et al., 2015, Kingma et al., 2016, Papamakarios et al., 2017] to simultaneously parameterize the d embeddings. In what follows we refer to the resulting UMNN autoregressive transformation as UMNN-MAF. Figure 1 summarizes the complete architecture.

Log-density The change of variables formula applied to the UMMN autoregressive transformation results in the log-density

$$\begin{aligned} \log p(\mathbf{x}; \theta) &= \log p_Z(\mathbf{g}(\mathbf{x}; \theta)) \left|\det J_{\mathbf{g}(\mathbf{x};\theta)}\right| \\ &= \log p_Z(\mathbf{g}(\mathbf{x}; \theta)) + \log \left| \prod_{i=1}^d \frac{\partial F^i(x_i, \mathbf{h}^i(\mathbf{x}_{1:i-1}))}{\partial x_i} \right| \\ &= \log p_Z(\mathbf{g}(\mathbf{x}; \theta)) + \sum_{i=1}^d \log f^i(x_i, \mathbf{h}^i(\mathbf{x}_{1:i-1})). \end{aligned} \tag{9}$$

Therefore, the transformation leads to a simple expression of (the determinant of) its Jacobian, which can be computed efficiently with a single forward pass. This is different from FFJORD [Grathwohl et al., 2018] which relies on numerical methods to compute both the Jacobian and the transformation between the data and the latent space. Therefore our proposed method makes the computation of the Jacobian exact and efficient at the same time.

Sampling Generating samples require evaluating the inverse transformation $\mathbf{g}^{-1}(\mathbf{z}; \theta)$. The components of the inverse vector $\mathbf{x}^{\text{inv}} = \mathbf{g}^{-1}(\mathbf{z}; \theta)$ can be computed recursively by inverting each com-

ponent of $\mathbf{g}(\mathbf{x};\theta)$:

$$x_1^{\text{inv}} = \left(g^1\right)^{-1}\left(z_1;\mathbf{h}^1\right) \qquad \text{if} \quad i=1 \tag{10}$$

$$x_i^{\text{inv}} = \left(g^i\right)^{-1}\left(z_i;\mathbf{h}^i\left(\mathbf{x}_{1:i-1}^{\text{inv}}\right)\right) \qquad \text{if} \quad i>1 \tag{11}$$

where $(g^i)^{-1}$ is the inverse of g^i. Another approach to invert an autoregressive model would be to approximate its inverse with another autoregressive network [Oord et al., 2018]. In this case, the evaluation of the approximated inverse model is as fast as the forward model.

Universality Since the proof is straightforward, we only sketch that UMNN-MAF is a universal density approximator of continuous random variables. We rely on the inverse sampling theorem to prove that UMNNs are universal approximators of continuously derivable ($\mathbb{C}^1$) monotonic functions. Indeed, if UMNNs can represent any $\mathbb{C}^1$ monotonic function, then they can also represent the (inverse) cumulative distribution function of any continuous random variable. Any continuously derivable function $f : \mathcal{D} \rightarrow \mathcal{I}$ can be expressed as the following integral: $f(x) = \int_a^x \frac{df}{dx}dx + f(a), \quad \forall x, a \in \mathcal{D}$. The derivative $\frac{df}{dx}$ is a continuous positive function and the universal approximation theorem of NNs ensures it can be successfully approximated with a NN of sufficient capacity (such as those used in UMNNs).

4 Related work

The most similar work to UMNN-MAF are certainly Neural Autoregressive Flow [NAF, Huang et al., 2018] and Block Neural Autoregressive Flow [B-NAF, De Cao et al., 2019], which both rely on strictly monotonic transformations for building bijective mappings. In NAF, transformations are defined as neural networks which activation functions are all constrained to be strictly monotonic and which weights are the output of a strictly positive and autoregressive HyperNetwork [Ha et al., 2017]. Huang et al. [2018] shows that NAFs are universal density approximators. In B-NAF, the authors improve on the scalability of the NAF architecture by making use of masking operations instead of HyperNetworks. They also present a proof of the universality of B-NAF, which extends to UMNN-MAF. Our work differs from both NAF and B-NAF in the sense that the UMNN monotonic transformation is based on free-form neural networks for which no constraint, beyond positiveness of the output, is enforced on the hypothesis class. This leads to multiple advantages: it enables the use of any state-of-the-art neural architecture, simplifies weight initialization, and leads to a more lightweight evaluation of the Jacobian.

More generally, UMNN-MAF relates to works on normalizing flows built upon autoregressive networks and affine transformations. Germain et al. [2015] first introduced masking as an efficient way to build autoregressive networks, and proposed autoregressive networks for density estimation of high dimensional binary data. Masked Autoregressive Flows [Papamakarios et al., 2017] and Inverse Autoregressive Flows [Kingma et al., 2016] have generalized this approach to real data, respectively for density estimation and for latent posterior representation in variational auto-encoders. More recently, Oliva et al. [2018] proposed to stack various autoregressive architectures to create powerful reversible transformations. Meanwhile, Jaini et al. [2019] proposed a new Sum-of-Squares flow that is defined as the integral of a second order polynomial parametrized by an autoregressive NN.

With NICE, Dinh et al. [2015] introduced coupling layers, which correspond to bijective transformations splitting the input vector into two parts. They are defined as

$$\mathbf{z}_{1:k} = \mathbf{x}_{1:k} \quad \text{and} \quad \mathbf{z}_{k+1:d} = e^{\sigma(\mathbf{x}_{1:k})} \odot \mathbf{x}_{k+1:d} + \mu(\mathbf{x}_{1:k}), \tag{12}$$

where σ and μ are two unconstrained functions $\mathbb{R}^{d-k} \rightarrow \mathbb{R}^{d-k}$. The same authors introduced RealNVP [Dinh et al., 2017], which combines coupling layers with normalizing flows and multi-scale architectures for image generation. Glow [Kingma and Dhariwal, 2018] extends RealNVP by introducing invertible 1x1 convolutions between each step of the flow. In this work we have used UMNNs in the context of autoregressive architectures, however UMNNs could also be applied to replace the linear transformation in coupling layers.

Finally, our architecture also shares a connection with Neural Ordinary Differential Equations [NODE, Chen et al., 2018]. The core idea of this architecture is to learn an ordinary differential equation which dynamic is parameterized by a neural network. Training can be carried

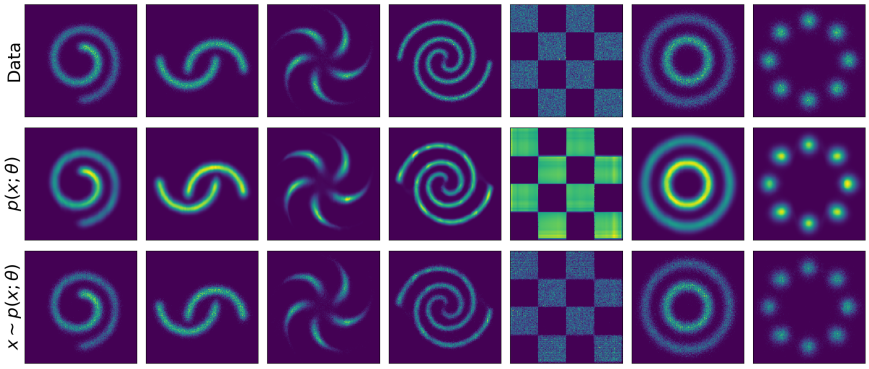


Figure 2: Density estimation and sampling with a UMNN-MAF network on 2D toy problems. **Top**: Samples from the empirical distribution $p(\mathbf{x})$. **Middle**: Learned density $p(\mathbf{x};\theta)$. **Bottom**: Samples drawn by numerical inversion. UMNN-MAF manages to precisely capture multi-modal and/or discontinuous distributions. Sampling is possible even if the model is not invertible analytically.

out by backpropagating efficiently through the ODE solver, with constant memory requirements. Among other applications, NODE can be used to model a continuous normalizing flow with a free-form Jacobian as in FFJORD [Grathwohl et al., 2018]. Similarly, a UMNN transformation can be seen as a structured neural ordinary differential equation in which the dynamic of the vector field is separable and can be solved efficiently by direct integration.

5 Experiments

In this section, we evaluate the expressiveness of UMNN-MAF on a variety of density estimation benchmarks, as well as for approximate inference in variational auto-encoders. The source code is accessible at `https://github.com/AWehenkel/UMNN`.

Experiments were carried out using the same integrand neural network in the UMNN component – i.e., in Equation 8, $f^i = f$ with shared weights $\psi^i = \psi$ for $i \in \{1, \dots, d\}$. The functions β^i are taken to be equal to one of the outputs of the embedding network. We observed in our experiments that sharing the same integrand function does not impact performance. Therefore, the neural embedding function $\mathbf{h}^i$ must produce a fixed size output for $i \in \{1, \dots, d\}$.

5.1 2D toy problems

We first train a UMNN-MAF on 2-dimensional toy distributions, as defined by Grathwohl et al. [2018]. To train the model, we minimize the negative log-likelihood of observed data

$$L(\theta) = -\sum_{n=1}^{N}\left[\log p_Z(\mathbf{g}(\mathbf{x}^n;\theta)) + \sum_{i=1}^{d}\log f(x_i^n, \mathbf{h}^i(\mathbf{x}_{1:i-1}^n))\right]. \tag{13}$$

The flow used to solve these tasks is the same for all distributions and is composed of a single transformation. More details can be found in Appendix A.1.

Figure 2 demonstrates that our model is able to learn a change of variables that warps a simple isotropic Gaussian into multimodal and/or discontinuous distributions. We observe from the figure that our model precisely captures the density of the data. We also observe that numerical inversion for generating samples yields good results.

5.2 Density estimation

We further validate UMNN-MAF by comparing it to state-of-the-art normalizing flows. We carry out experiments on tabular datasets (POWER, GAS, HEPMASS, MINIBOONE, BSDS300) as well as on MNIST. We follow the experimental protocol of Papamakarios et al. [2017]. All training hyper-parameters and architectural details are given in Appendix A.1. For each dataset, we report

Table 1: Average negative log-likelihood on test data over 3 runs, error bars are equal to the standard deviation. Results are reported in nats for tabular data and bits/dim for MNIST; lower is better. The best performing architecture for each dataset is written in bold and the best performing architecture per category is underlined. (a) Non-autoregressive models, (b) Autoregressive models, (c) Monotonic and autoregressive models. UMNN outperforms other monotonic transformations on 4 tasks over 6 and is the overall best performing model on 2 tasks over 6.

	Dataset	**POWER**	**GAS**	**HEPMASS**	**MINIBOONE**	**BSDS300**	**MNIST**
(a)	RealNVP - Dinh et al. [2017]	$-0.17_{\pm.01}$	$-8.33_{\pm.14}$	$18.71_{\pm.02}$	$13.55_{\pm.49}$	$-153.28_{\pm1.78}$	-
	Glow - Kingma and Dhariwal [2018]	$-0.17_{\pm.01}$	$-8.15_{\pm.40}$	$19.92_{\pm.08}$	$11.35_{\pm.07}$	$-155.07_{\pm.03}$	-
	FFJORD - Grathwohl et al. [2018]	$\underline{-0.46}_{\pm.01}$	$\underline{-8.59}_{\pm.12}$	$\underline{14.92}_{\pm.08}$	$\underline{10.43}_{\pm.04}$	$\underline{-157.40}_{\pm.19}$	-
(b)	MADE - Germain et al. [2015]	$3.08_{\pm.03}$	$-3.56_{\pm.04}$	$20.98_{\pm.02}$	$15.59_{\pm.50}$	$-148.85_{\pm.28}$	$2.04_{\pm.01}$
	MAF - Papamakarios et al. [2017]	$-0.24_{\pm.01}$	$-10.08_{\pm.02}$	$17.70_{\pm.02}$	$11.75_{\pm.44}$	$-155.69_{\pm.28}$	$1.89_{\pm.01}$
	TAN - Oliva et al. [2018]	$\underline{-0.60}_{\pm.01}$	$\underline{\mathbf{-12.06}}_{\pm.02}$	$\underline{\mathbf{13.78}}_{\pm.02}$	$\underline{11.01}_{\pm.48}$	$\underline{\mathbf{-159.80}}_{\pm.07}$	$\underline{1.19}$
(c)	NAF - Huang et al. [2018]	$-0.62_{\pm.01}$	$-11.96_{\pm.33}$	$15.09_{\pm.40}$	$\underline{\mathbf{8.86}}_{\pm.15}$	$-157.73_{\pm.30}$	-
	B-NAF - De Cao et al. [2019]	$-0.61_{\pm.01}$	$\underline{-12.06}_{\pm.09}$	$14.71_{\pm.38}$	$8.95_{\pm.07}$	$-157.36_{\pm.03}$	-
	SOS - Jaini et al. [2019]	$-0.60_{\pm.01}$	$-11.99_{\pm.41}$	$15.15_{\pm.1}$	$8.90_{\pm.11}$	$-157.48_{\pm.41}$	1.81
	UMNN-MAF (ours)	$\underline{\mathbf{-0.63}}_{\pm.01}$	$-10.89_{\pm.7}$	$\underline{13.99}_{\pm.21}$	$9.67_{\pm.13}$	$\underline{-157.98}_{\pm.01}$	$\underline{\mathbf{1.13}}_{\pm.02}$

results on test data for our best performing model (selected on the validation data). At testing time we use a large number of integration steps (100) to compute the integral, this ensures its correctness and avoids misestimating the performance of UMNN-MAF.

Table 1 summarizes our results, where we can see that on tabular datasets, our method is competitive with other normalizing flows. For POWER, our architecture slightly outperforms all others. It is also better than other monotonic networks (category (c)) on 3 tabular datasets over 5. From these results, we could conclude that Transformation Autoregressive Networks [TAN, Oliva et al., 2018] is overall the best method for density estimation. It is however important to note that TAN is a flow composed of many heterogeneous transformations (both autoregressive and non-autoregressive). For this reason, it should not be directly compared to the other models which respective results are specific to a single architecture. However, TAN provides the interesting insight that combining heterogeneous components into a flow leads to better results than an homogeneous flow.

Notably, we do not make use of a multi-scale architecture to train our model on MNIST. On this task, UMNN-MAF slightly outperforms all other models by a reasonable margin. Samples generated by a conditional model are shown on Figure 3, for which it is worth noting that UMNN-MAF is the first monotonic architecture that has been inverted to generate samples. Indeed, MNIST can be considered as a high dimensional dataset ($d = 784$) for standard feed forward neural networks which autoregressive networks are part of. NAF and B-NAF do not report any result for this benchmark, presumably because of memory explosion. In comparison, BSDS300, which data dimension is one order of magnitude smaller than MNIST ($63 \ll 784$), are the largest data they have tested on. Table 2 shows the number of parameters used by UMNN-MAF in comparison to B-NAF and NAF. For bigger datasets, UMNN-MAF requires less parameters than NAF to reach similar or better performance. This could explain why NAF has never been used for density estimation on MNIST.

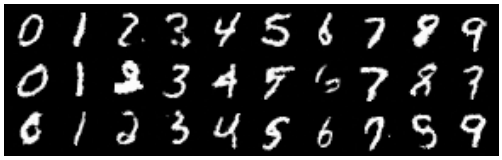


Figure 3: Samples generated by numerical inversion of a conditional UMNN-MAF trained on MNIST. Samples $\mathbf{z}$ are drawn from an isotropic Gaussian with $\sigma = .75$. See Appendix C for more details.

Table 2: Comparison of the number of parameters between NAF, B-NAF and UMNN-MAF. In high dimensional datasets, UMNN-MAF requires fewer parameters than NAF and a similar number to B-NAF.

Dataset	NAF	B-NAF	UMNN-MAF
POWER ($d = 6$)	4.14e5	3.07e5	5.09e5
GAS ($d = 8$)	4.02e5	5.44e5	8.15e5
HEPMASS ($d = 21$)	9.27e6	3.72e6	3.62e6
MINIBOONE ($d = 43$)	7.49e6	4.09e6	3.46e6
BSDS300 ($d = 63$)	3.68e7	8.76e6	1.56e7

Table 3: Average negative evidence lower bound of VAEs over 3 runs, error bars are equal to the standard deviation. Results are reported in bits per dim for Freyfaces and in nats for the other datasets; lower is better. UMNN-NAF is performing slightly better than IAF but is outperformed by B-NAF. We believe that the gap in performance between B-NAF and UMNN is due to the way the NF is conditioned by the encoder's output.

	Dataset	**MNIST**	**Freyfaces**	**Omniglot**	**Caltech 101**
(a)	VAE - Kingma and Welling [2013]	$86.65_{\pm.06}$	$4.53_{\pm.02}$	$104.28_{\pm.39}$	$110.80_{\pm.46}$
	Planar - Rezende and Mohamed [2015]	$86.06_{\pm.32}$	$4.40_{\pm.06}$	$102.65_{\pm.42}$	$109.66_{\pm.42}$
	IAF - Kingma et al. [2016]	$84.20_{\pm.17}$	$4.47_{\pm.05}$	$102.41_{\pm.04}$	$111.58_{\pm.38}$
	Sylvester - Berg et al. [2018]	$83.32_{\pm.06}$	$4.45_{\pm.04}$	$99.00_{\pm.04}$	$104.62_{\pm.29}$
	FFJORD - Grathwohl et al. [2018]	$82.82_{\pm.01}$	$4.39_{\pm.01}$	$98.33_{\pm.09}$	$104.03_{\pm.43}$
(b)	B-NAF - De Cao et al. [2019]	$83.59_{\pm.15}$	$4.42_{\pm.05}$	$100.08_{\pm.07}$	$105.42_{\pm.49}$
	UMNN-MAF (ours)	$84.11_{\pm.05}$	$4.51_{\pm.01}$	$100.98_{\pm.13}$	$110.45_{\pm.69}$

5.3 Variational auto-encoders

To assess the performance of our model, we follow the experimental setting of Berg et al. [2018] for VAE. The encoder and the decoder architectures can be found in the appendix of their paper. In VAE it is usual to let the encoder output the parameters of the flow. For UMNN-MAF, this would cause the encoder output's dimension to be too large. Instead, the encoder output is passed as additional entries of the UMNN-MAF. Like other architectures, the UMNN-MAF also takes as input a vector of noise drawn from an isotropic Gaussian of dimension 64.

Table 3 presents our results. It shows that on MNIST and Omniglot, UMNN-MAF slightly outperforms the classical VAE as well as planar flows. Moreover, on these datasets and Freyfaces, IAF, B-NAF and UMNN-MAF achieve similar results. FFJORD is the best among all, however it is worth noting that the roles of encoder outputs in FFJORD, B-NAF, IAF and Sylvester are all different. We believe that the heterogeneity of the results could be, at least in part, due to the different amortizations.

6 Discussion and summary

Static integral quadrature can be inaccurate. Computing the integral with static Clenshaw-Curtis quadrature only requires the evaluation of the integrand at predefined points. As such, batches of points can be processed all at once, which makes static Clenshaw-Curtis quadrature well suited for neural networks. However, static quadratures do not account for the error made during the integration. As a consequence, the quadrature is inaccurate when the integrand is not smooth enough and the number of integration steps is too small. In this work, we have reduced the integration error by applying the normalization described by Gouk et al. [2018] in order to control the Lipschitz constant of the integrand and appropriately set the number of integration steps. We observed that as long as the Lipschitz constant of the network does not increase dramatically (< 1000), a reasonable number of integration steps (< 100) is sufficient to ensure the convergence of the quadrature. An alternative solution would be to use dynamic quadrature such as dynamic Clenshaw-Curtis.

Efficiency of numerical inversion. Architectures relying on linear transformations [Papamakarios et al., 2017, Kingma et al., 2016, Dinh et al., 2017, Kingma and Dhariwal, 2018] are trivially exactly and efficiently invertible. In contrast, the UMNN transformation has no analytic inverse. Nevertheless, it can be inverted numerically using root-finding algorithms. Since most such algorithms rely on multiple nested evaluations of the function to be inverted, applying them naively to a numerical integral would quickly become very inefficient. However, the Clenshaw-Curtis quadrature is part of the nested quadrature family, meaning that the evaluation of the integral at multiple nested points can take advantage of previous evaluations and thus be implemented efficiently. As an alternative, Oord et al. [2018] have shown that an invertible model can always be distilled to learn its inverse, and thus make the inversion efficient whatever the cost of inversion of the original model.

Scalability and complexity analysis. UMNN-MAF is particularly well suited for density estimation because the computation of the Jacobian only requires a single forward evaluation of a NN.

Together with the Leibniz integral rule, they make the evaluation of the log-likelihood derivative as memory efficient as usual supervised learning, which is equivalent to a single backward pass on the computation graph. By contrast, density estimation with previous monotonic transformations typically requires a backward evaluation of the computation graph of the transformer NN to obtain the Jacobian. Then, this pass must be evaluated backward again in order to obtain the log-likelihood derivative. Both NAF and B-NAF provide a method to make this computation numerically stable, however both fail at not increasing the size of the computation graph of the log-likelihood derivative, hence leading to a memory overhead. The memory saved by the Leibniz rule may serve to speed up the quadrature computation. In the case of static Clenshaw-Curtis, the function values at each evaluation point can be computed in parallel using batch of points. In consequence, when the GPU memory is large enough to store "meta-batches" of size $d \times N \times B$ (with d the dimension of the data, N the number of integration steps and B the batch size) the computation is approximately as fast as a forward evaluation of the integrand network.

Summary We have introduced Unconstrained Monotonic Neural Networks, a new invertible transformation built upon free-form neural networks allowing the use of any state-of-the-art architecture. Monotonicity is guaranteed without imposing constraints on the expressiveness of the hypothesis class, contrary to classical approaches. We have shown that the resulting integrated neural network can be evaluated efficiently using standard quadrature rule while its inverse can be computed using numerical algorithms. We have shown that our transformation can be composed into an autoregressive flow, with competitive or state-of-the-art results on density estimation and variational inference benchmarks. Moreover, UMNN is the first monotonic transformation that has been successfully applied for density estimation on high dimensional data distributions (MNIST), showing better results than the classical approaches.

We identify several avenues for improvement and further research. First, we believe that numerical integration could be fasten up during training, by leveraging the fact that controlled numerical errors can actually help generalization. Moreover, the UMNN transformation would certainly profit from using a dynamic integration scheme, both in terms of accuracy and efficiency. Second, it would be worth comparing the newly introduced monotonic transformation with common approaches for modelling monotonic functions in machine learning. On a similar track, these common approaches could be combined into an autoregressive flow as shown in Section 3.3. Finally, our monotonic transformation could be used within other neural architectures than generative autoregressive networks, such as multi-scale architectures [Dinh et al., 2017] and learnable 1D convolutions [Kingma and Dhariwal, 2018].

Acknowledgments

The authors would like to acknowledge Matthia Sabatelli, Nicolas Vecoven, Antonio Sutera and Louis Wehenkel for useful feedback on the manuscript. They would also like to thank the anonymous reviewers for many relevant remarks. Antoine Wehenkel is a research fellow of the F.R.S.-FNRS (Belgium) and acknowledges its financial support.

References

N. P. Archer and S. Wang. Application of the back propagation neural network algorithm with monotonicity constraints for two-group classification problems. *Decision Sciences*, 24(1):60–75, 1993.

R. v. d. Berg, L. Hasenclever, J. M. Tomczak, and M. Welling. Sylvester normalizing flows for variational inference. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2018.

T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, pages 6571–6583, 2018.

H. Daniels and M. Velikova. Monotone and partially monotone neural networks. *IEEE Transactions on Neural Networks*, 21(6):906–917, 2010.

N. De Cao, I. Titov, and W. Aziz. Block neural autoregressive flow. *arXiv preprint arXiv:1904.04676*, 2019.

L. Dinh, D. Krueger, and Y. Bengio. Nice: Non-linear independent components estimation. In *International Conference in Learning Representations workshop track*, 2015.

L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real nvp. In *International Conference in Learning Representations*, 2017.

M. Germain, K. Gregor, I. Murray, and H. Larochelle. Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pages 881–889, 2015.

H. Gouk, E. Frank, B. Pfahringer, and M. Cree. Regularisation of neural networks by enforcing lipschitz continuity. *arXiv preprint arXiv:1804.04368*, 2018.

W. Grathwohl, R. T. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. In *International Conference on Machine Learning*, 2018.

M. Gupta, A. Cotter, J. Pfeifer, K. Voevodski, K. Canini, A. Mangylov, W. Moczydlowski, and A. Van Esbroeck. Monotonic calibrated interpolated look-up tables. *The Journal of Machine Learning Research*, 17(1):3790–3836, 2016.

D. Ha, A. M. Dai, and Q. V. Le. Hypernetworks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

C.-W. Huang, D. Krueger, A. Lacoste, and A. Courville. Neural autoregressive flows. In *International Conference on Machine Learning*, pages 2083–2092, 2018.

P. Jaini, K. A. Selby, and Y. Yu. Sum-of-squares polynomial flow. *arXiv preprint arXiv:1905.02325*, 2019.

D. P. Kingma and P. Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pages 10236–10245, 2018.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations (ICLR)*, 2013.

D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751, 2016.

J. Oliva, A. Dubey, M. Zaheer, B. Poczos, R. Salakhutdinov, E. Xing, and J. Schneider. Transformation autoregressive networks. In *International Conference on Machine Learning*, pages 3895–3904, 2018.

A. Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. Driessche, E. Lockhart, L. Cobo, F. Stimberg, et al. Parallel wavenet: Fast high-fidelity speech synthesis. In *International Conference on Machine Learning*, pages 3915–3923, 2018.

G. Papamakarios, T. Pavlakou, and I. Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pages 2338–2347, 2017.

G. Papamakarios, D. C. Sterratt, and I. Murray. Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. In *22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019.

D. Rezende and S. Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538, 2015.

J. Sill. Monotonic networks. In *Advances in neural information processing systems*, pages 661–667, 1998.

D. Tran, M. D. Hoffman, R. A. Saurous, E. Brevdo, K. Murphy, and D. M. Blei. Deep probabilistic programming. In *5th International Conference on Learning Representations (ICLR)*, 2017.

A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. In *9th ISCA Speech Synthesis Workshop*, pages 125–125, 2016.

S. You, D. Ding, K. Canini, J. Pfeifer, and M. Gupta. Deep lattice networks and partial monotonic functions. In *Advances in Neural Information Processing Systems*, pages 2981–2989, 2017.

A Experimental setup

A.1 Density estimation and toy problems hyperparameters

Table 4 reports the training configurations for the 2D toy problems and the 5 tabular datasets. For tabular data the best performing architecture has been found after some preliminary experiments, while this was not needed for the 2D toy problems. During our preliminary experiments we tested different integrand network architectures, we tested on the number of hidden layers $L \in \{3, 4\}$ and on their dimension $D \in \{50, 100, 150, 200\}$. The architecture of the embedding networks is the best performing MADE network used in NAF [Huang et al., 2018]. We used the Adam optimizer and tried different learning rate $\lambda \in \{10^{-3}, 5 \times 10^{-4}, 10^{-4}\}$. When the learning rate chosen was greater than 10^{-4} we schedule once the learning rate to 10^{-4} after the first plateau. We also tested for different weights decay values $W \in \{10^{-5}, 10^{-2}\}$. The batch size was chosen to be as big as possible while not harming the learning procedure. We observed during our preliminary experiments that choosing the number of integration steps at random (uniformly from 20 to 100) for each batch regularizes the complexity of the integral. For MNIST, we observed that 25 integration steps was enough if the Lipschitz constant of the network is constraint (with the normalization proposed by Gouk et al. [2018]) to be smaller than 1.5.

Dataset	**POWER**	**GAS**	**HEPMASS**	**MINIBOONE**	**BSDS300**	**MNIST**	**2D Toys**
Lipschitz	-	-	-	-	2.5	1.5	-
N°integ. steps	rand	rand	rand	rand	rand	25	50
Embedding net	2×100	2×100	2×512	1×512	2×1024	1×1024	4×50
Integrand net ($L \times D$)	4×150	3×200	4×200	3×50	4×150	3×150	4×50
Learning rate (λ)	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-4}	10^{-3}	10^{-3}
N°flows	5	10	5	3	5	5	1
Embedding Size	30	30	30	30	30	30	10
Weight decay (W)	10^{-5}	10^{-2}	10^{-4}	10^{-2}	10^{-2}	10^{-2}	10^{-5}
Batch size	10000	10000	100	500	100	100	100

Table 4: Training configurations for density estimation and toy problems.

A.2 Variational auto-encoders

Table 5 presents the architectural settings of the normalizing flows used inside the variational auto-encoders. The number of values outputted by the encoder is always taken to be equal to 320. These values as well as the 64-dimensional noise vector are the inputs of the embedding network which is constantly made of one hidden layer of 1280 neurons. We have performed a small grid search on the integrand network architecture, we took a look at 2 different number $L \in \{3, 4\}$ of hidden layers of dimensions $D \in \{100, 150\}$.

Dataset	**MNIST**	**Freyfaces**	**Omniglot**	**Caltech 101**
Lipschitz	-	-	-	-
N°integ. steps	rand	rand	rand	rand
Encoder Output	320	320	320	320
Embedding net	1×1280	1×1280	1×1280	1×1280
Integrand net	4×100	3×100	4×100	4×100
N°flows	16	8	16	16
Embedding Size	30	30	30	30

Table 5: Training configurations of variational auto-encoder.

B Clenshaw-Curtis module

Algorithm 1 Clenshaw-Curtis quadrature

Input: x: A tensor of scalar values that represent the superior integration bounds.
$\mathbf{h}$: A tensor of vectors that representing embeddings.
Output: F: A tensor of scalar values that represent the integral of $\int_0^x f(t;\mathbf{h})\ \mathrm{d}t$.
Hyper-parameters: f: A derivable function $\mathbb{R} \to \mathbb{R}$.
N: The number of integration steps.

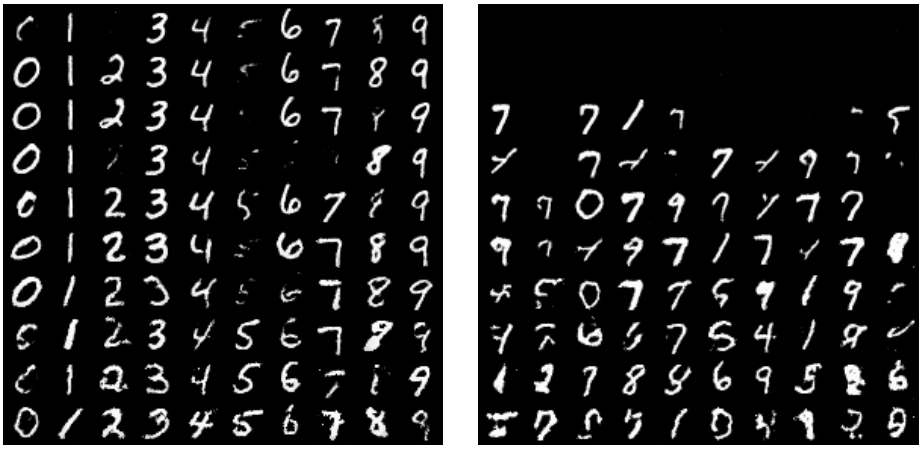
procedure FORWARD(x, $\mathbf{h}$; f, N)
  ▷ Compute weights and evaluation steps for Clenshaw-Curtis quadrature
  $\mathbf{w}$, δ_x = COMPUTE_CC_WEIGHTS(N)
  $F = 0$
  for $i \in [1, N]$ **do**
    $x_i = x_0 + \frac{1}{2}(x - x_0)(\delta_x[i] + 1)$ ▷ Compute the next point to evaluate
    $\delta_F = f(x_i; \mathbf{h})$
    $F = F + \mathbf{w}[i]\delta_F$
  end for
  $F = \frac{F}{2}(x - x_0)$
  return F
end procedure

Inputs: x: A tensor of scalar values that represent the superior integration bounds.
$\mathbf{h}$: A tensor of vectors that representing embeddings.
∇_{out} : The derivatives of the loss function with respect to $\int_0^x f(t;\mathbf{h})\ \mathrm{d}t$ for all x.
Outputs: ∇_x: The gradient of $\int_0^x f(t;\mathbf{h})\ \mathrm{d}t$ with respect to x.
∇_θ: The gradient of $\int_0^x f(t;\mathbf{h})\ \mathrm{d}t$ with respect to f parameters.
$\nabla_\mathbf{h}$: The gradient of $\int_0^x f(t;\mathbf{h})\ \mathrm{d}t$ with respect to h.
Hyper-parameters: f: A derivable function $\mathbb{R} \to \mathbb{R}$.
N: The number of integration steps.

procedure BACKWARD(x, h, ∇_{out}; f, N)
  ▷ Compute weights and evaluation steps for Clenshaw-Curtis quadrature
  $\mathbf{w}$, δ_x = COMPUTE_CC_WEIGHTS(N)
  $F, \nabla_\theta, \nabla_\mathbf{h} = 0, 0, 0$
  for $i \in [1, N]$ **do**
    $x_i = x_0 + \frac{1}{2}(x - x_0)(\delta_x[i] + 1)$ ▷ Compute the next point to evaluate
    $\delta_F = f(x_i; \mathbf{h})$
    ▷ Sum up for all samples of the batch the gradients with respect to inputs $\mathbf{h}$
    $\delta_{\nabla_\mathbf{h}} = \sum_{j=1}^{B} \nabla_{\mathbf{h}^j} \left(\delta_F^j\right) \nabla_{out}^j (x^j - x_0^j)$
    ▷ Sum up for all samples of the batch the gradients with respect to parameters θ
    $\delta_{\nabla_\theta} = \sum_{j=1}^{B} \nabla_\theta \left(\delta_F^j\right) \nabla_{out}^j (x^j - x_0^j)$
    $\nabla_\mathbf{h} = \nabla_\mathbf{h} + \mathbf{w}[i]\delta_{\nabla_\mathbf{h}}$
    $\nabla_\theta = \nabla_\theta + \mathbf{w}[i]\delta_{\nabla_\theta}$
  end for
  ▷ Gradients with respect to superior integration bound.
  $\nabla_x = f(x, \mathbf{h})\nabla_{out}$
  return ∇_x, ∇_θ, $\nabla_\mathbf{h}$
end procedure

C Generated images from MNIST

Figure 4 presents samples generated from two UMNN-MAF trained on MNIST, respectively with (sub-figure a) and without (sub-figure b) labels. The samples are generated with different levels of noise, which are the product of the inversion of the network with random values drawn from $\mathcal{N}(0, T)$, with T being the sampling temperature. The sampling temperature increases linearly from 0.1 (top rows) to 1.0 (bottom rows). We can observe that the unconditional model fails to incorporate digit structure when the level of noise is too small. However, when the level is sufficient it is able to generate random digits with a high level of heterogeneity.



(a) (b)

Figure 4: (a): Class-conditional generated images from MNIST. The temperature of sampling increases from 0.1 (top row) to 1.0 (bottom row). Columns correspond to different classes. (b): Unconditional generated images from MNIST. The temperature of sampling goes from 0.1 at top row to 1.0 at bottom row. Columns are different random noise values.

7.3 EPILOGUE

7.3.1 *Discussion*

Alternative monotonic normalizing flows. In 2018, Huang et al. [2018] proposed replacing the affine transformations of masked autoregressive flows [Papamakarios et al., 2017] with neural autoregressive flows (NAFs). NAFs enforce monotonicity with monotonic activation functions and positive weights. This parameterisation does not only cause monotonicity for the desired monotonic variables but for all input variables. This aspect of NAF is undesirable when we only want monotonicity for one variable at a time, such as in the context of normalizing flows. Huang et al. [2018] overcome this issue by predicting the weights of the monotonic neural network with a hyper network that takes as input the conditioning variables. Computing the Jacobian of such flows requires backpropagation which may be both computationally and memory demanding.

De Cao et al. [2020] replaced the hyper networks of NAFs by block monotonic neural networks. They showed that such parameterisation achieved better results with fewer parameters. Concurrently to UMNNs, Durkan et al. [2019] proposed neural spline flows (NSFs) as a parameterisation of monotonic transformations in NFs. Compared to (block) neural autoregressive flows, their parameterization provides direct access to the Jacobian determinant. We may generally favour NSFs over UMNNs because they do not require solving integrals. However, the spline parameterisation can create discontinuities at the boundary points. These discontinuities sometimes lead to numerical issues in practice. More importantly, Köhler et al. [2021] showed that these discontinuities preclude smoothness which is sometimes an expected feature.

Is universality the goal? We can easily fool ourselves into the non-realistic objective of learning probabilistic models from data only. We have already discussed in Part i why this objective is vain in high dimensionalities. The universality of a class of models does not say anything about the corresponding learning algorithm's generalisation capabilities. Nevertheless, neural networks are universal approximators of continuous functions and frequently generalise well to unseen data. The continuity of the multi-layer perceptron is one possible explanation. In addition, stochastic gradient descent is an implicit regularisation [Smith et al., 2021; Barrett and Dherin, 2020] and other strategies, such as dropout [Srivastava et al., 2014] or weight decay [Krogh and Hertz, 1991], are popular explicit regularisation. Complex architectures, such as CNNs or GNNs, enforce substantial constraints. These constraints, such as equivariance or invariance properties, induce generalisation. Without a similar inductive bias, processing structured signals such as images or audio with machine learning models would be ineffective.

In contrast, UMNN-MAF embeds only weak inductive bias, which may preclude us from learning a meaningful representation of the data. For example, learning a good representation of images, even as small as digits from MNIST (28×28 grey pixels), is very

difficult and hopeless for higher resolutions. The autoregressive structure of the model naturally induces usually irrelevant dependence between all dimensionalities. In the next part, we focus on embedding more substantial inductive bias in deep probabilistic models such as NFs and VAEs. We show that unconstrained monotonic neural networks with (conditional) independence assumptions have strong generalisation capabilities.

Multidimensional unconstrained monotonic neural networks. One limitation of UMNNs is that they are limited to unidimensional monotonicity. However, we can generalise UMNNs to any dimensionality. Indeed, we observe that the sum of k univariate monotonic functions is monotonic with respect to the union of the k monotonic entries. The Kolmogorov-Arnold representation theorem [Kolmogorov, 1956; Arnold, 2009] says that any multivariate continuous function can be written as a double sum over univariate functions, as

$$f(x_1, \dots, x_k) = \sum_{q=0}^{2n} g_q \left(\sum_{p=0}^{n} h_{q,p}(x_p) \right).$$

This decomposition hints that we can effectively parameterise multivariate-monotonic functions with multiple levels of univariate-monotonic transformations. If needed, we can also add non-monotonic inputs to these functions. We believe such generalised UMNNs might also be relevant for machine learning problems where a subset of variables have a monotonic relationship with the output.

7.3.2 *Scientific impact*

The paper introduces an implicit parameterisation of monotonic functions via their first-order derivatives. We show that the Clenshaw-Curtis quadrature efficiently integrates a neural network with respect to its input. We also overcome potential memory issues of direct backpropagation through the numerical integration steps with the Leibniz rule, which describe the derivative of an integral and can be computed on-the-fly. In contrast to alternative monotonic neural networks, UMNNs are continuously differentiable and do not necessitate automatic differentiation to evaluate their first-order derivative with respect to the input variable. Combined with autoregressive transformations, UMNNs lead to an efficient parameterisation of normalizing flows. We must also acknowledge that solving integrals necessitates additional computations compared to feedforward neural networks. However, the integration part is parallelisable on GPUs if the memory is large enough; if this is the case, the forward and backward evaluations take two times as much time for a UMNN than the corresponding feedforward network, not more.

Monotonic transformations are also relevant outside of normalizing flows. Among them, model calibration is arguably an essential issue. Uncalibrated models provide biased confidence scores; calibration corrects this bias. Recently, Gruber and Buettner [2022]; Dey et al.; Rahimi et al. [2020] relied on UMNNs to parameterise the calibration layers

in diverse settings. Another application of UMNNs is to induce a monotonic relationship between a subset of the input variables and the model's output. For example, Yurk and Abu-Mostafa [2021] study the effect of business closures on the speed of propagation of COVID-19 with ML models. A UMNN enforces a monotonic relationship between the tightness of public policies and the observed reproduction rate. UMNNs have also proven helpful in distributional reinforcement learning [Dabney et al., 2018]. Théate et al. [2021] parameterise the 1D distribution of the cumulative reward with UMNNs and study the effect of different divergence or distance functions on the learned distribution.

It is unclear whether parameterising monotonic functions via their first-order derivative is always preferable given the burden of the integration. Nevertheless, UMNNs are effective neural network architectures and have already had a good impact. According to Google Scholar, the paper has received 102 citations as of August 2022. As mentioned, it has been used in diverse settings ranging from model calibration to density estimation. Graphical normalizing flows, introduced in Chapter 8, strongly rely on the universality of UMNNs to define a simplified and unifying framework for normalizing flows.

7.3.3 *Conclusion and opportunities*

This chapter has introduced a new parameterisation of monotonic functions with neural networks. In contrast to other approaches, UMNNs work with free-form neural networks and benefit from all research on activation functions, initialisation strategies, and regularisation techniques. Since UMNNs directly provide their first-order derivative with respect to their input variable, they are particularly well suited to parameterise the normaliser functions of NFs. The corresponding NF is a universal density approximator of continuous distributions and achieves state-of-the-art results in density estimation. Since its publications, UMNNs have also been used successfully in diverse settings to induce monotonic responses in ML models.

Efficient parameterisations of monotonic functions should remain valuable to ML practitioners in the long run. In particular, calibration is an essential issue for applications where the uncertainty of the predictions matters [Minderer et al., 2021; Guo et al., 2017; Cranmer et al., 2015]. Another example is Multi-agent reinforcement learning. These algorithms summarise the cumulative discounted rewards of all agents with a unique value that is monotonic with respect to these rewards [Rashid et al., 2018; Leroy et al., 2020].

We speculate on the broader value that implicit parameterisations, similar to UMNNs, via the first-order derivative, might have in the future. For instance, we can enforce Lipshitzness or convexity with simple constraints on the first-order derivative of the function. Thus applications that require similar properties might benefit from similar implicit parameterisations. Similarly to Neural ODE [Chen et al., 2018] that parameterises dynamical systems via an ordinary differential equation or deep equilibrium models [Bai et al., 2019], UMNNs demonstrate that implicit layers provide a relevant parameterisation strategy for modern machine learning.

Part III

INFORMED PROBABILISTIC MODELLING

I prefer dangerous freedom over peaceful slavery.

Thomas Jefferson

8

STRUCTURED DENSITY ESTIMATION

Outline

n this chapter, we aim to go beyond uninformed probabilistic modelling. We now focus on *informed* probabilistic model. For this purpose, we introduce the graphical normalizing flow. This new normalizing flow embeds explicit inductive bias in the form of prescribed independencies. This transformation unifies Bayesian networks, coupling and autoregressive layers altogether. Graphical normalizing flows embed domain knowledge in the form of independencies while preserving the interpretability of Bayesian networks and the representation capacity of normalizing flows. In addition to the straightforward embedding of prescribed independencies, graphical conditioners can also discover relevant independence from data only. We analyse the effect of l1-penalization on the recovered probabilistic model and show that it improves the generalisation of normalizing flows.

8.1 PROLOGUE

In Chapter 7, we have introduced a universal approximator of continuous density functions called UMNN-MAF. Universality is a desirable property as it implies convergence of the MLE toward the correct model as the number of training points grows. However, in many practical settings, the number of points does not grow sufficiently fast to discriminate between all possible models. This is the curse of dimensionality, and universality becomes an issue rather than an solution.

One potential solution is to take a Bayesian approach and bias the learning toward more plausible models. However, we need to express a prior distribution over the considered class of models to do this. Unfortunately, distributions over complex functions, such as normalizing flows, are challenging to represent. As an alternative, we can exclude models that are irrelevant for describing the phenomenon of interest. For instance, we often make (conditional) independence assumptions when building complex models. These independencies lead to a simplified factorisation of the modelled distribution.

While the first solution is generic, it is also challenging to implement. In contrast, humans are reasonably good at drawing independence assumptions between small pieces of a larger system. We can use Bayesian networks to encode these independencies and understand the big picture produced by these low-level assumptions. The paper featured

in this chapter introduces normalizing flows as a parameterisation of the conditional distributions of Bayesian networks for continuous variables. Similarly to the autoregressive or coupling layers that lift invertible transformations from scalar to vector, the structure of any Bayesian network is a valid transformer for normalizing flows. This unified framework reveals the potential of combining prescribed independencies and expressive bijective transformations together.

Unifying Bayesian networks and normalizing flows allows years of research from each domain to benefit the other. For example, recent advances in topology discovery [Zheng et al., 2018] allows us to introduce a new class of probabilistic models where both the conditional densities and the distribution factorisation are trainable components. This class of models is a universal density approximator provided the appropriate parameterisation. However, in contrast to UMNN-MAF, graphical normalizing flows with learnable structure can be elegantly regularised by penalising the absence of independence. Moreover, the recovered structure provides interesting insights on independence properties observed in the data and hypothesised by the learnt model.

This contribution is again about the interplay between seemingly distinct classes of probabilistic models. We show that unifying frameworks can create new models with unique properties. In this case, we gain interpretability, a new inductive bias for normalizing flows, and a unified vision of coupling and autoregressive layers that were historically seen as separate classes of models. This contribution is also well aligned with the notion that building effective models requires the correct assumptions. Indeed, in contrast to classical normalizing flows, our model naturally digests prescribed knowledge we may have about the distribution we aim to learn.

8.2 THE PAPER: GRAPHICAL NORMALIZING FLOWS

8.2.1 *Author contributions*

Gilles Louppe and I co-authored the paper. Gilles helped me throughout the project to shape the research idea. He also provided substantial help in writing the paper. I developed the connections between Bayesian networks and normalizing flows and the theory to combine graphical normalizing flows with the NOTEARS algorithm for topology discovery. I also wrote the code for all experiments.

8.2.2 *Reading tips*

The reader should be able to skip section 3 as it describes the basics of normalizing flows and Bayesian network which were already described in the background.

Graphical Normalizing Flows

Antoine Wehenkel
ULiège

Gilles Louppe
ULiège

Abstract

Normalizing flows model complex probability distributions by combining a base distribution with a series of bijective neural networks. State-of-the-art architectures rely on coupling and autoregressive transformations to lift up invertible functions from scalars to vectors. In this work, we revisit these transformations as probabilistic graphical models, showing they reduce to Bayesian networks with a pre-defined topology and a learnable density at each node. From this new perspective, we propose the graphical normalizing flow, a new invertible transformation with either a prescribed or a learnable graphical structure. This model provides a promising way to inject domain knowledge into normalizing flows while preserving both the interpretability of Bayesian networks and the representation capacity of normalizing flows. We show that graphical conditioners discover relevant graph structure when we cannot hypothesize it. In addition, we analyze the effect of ℓ_1-penalization on the recovered structure and on the quality of the resulting density estimation. Finally, we show that graphical conditioners lead to competitive white box density estimators. Our implementation is available at `https://github.com/AWehenkel/DAG-NF`.

1 Introduction

Normalizing flows [NFs, Rezende and Mohamed, 2015, Tabak et al., 2010, Tabak and Turner, 2013, Rippel and Adams, 2013] have proven to be an effective way to model complex data distributions with neural networks. These models map data points to latent variables through an invertible function while keeping track of the change of density caused by the transformation. In contrast to variational auto-encoders (VAEs) and generative adversarial networks (GANs), NFs provide access to the exact likelihood of the model's parameters, hence offering a sound and direct way to optimize the network parameters. Normalizing flows have proven to be of practical interest as demonstrated by Van Den Oord et al. [2018], Kim et al. [2018] and Prenger et al. [2019] for speech synthesis, by Rezende and Mohamed [2015], Kingma et al. [2016] and Van Den Berg et al. [2018] for variational inference or by Papamakarios et al. [2019b] and Greenberg et al. [2019] for simulation-based inference. Yet, their usage as a base component of the machine learning toolbox is still limited in comparison to GANs or VAEs. Recent efforts have been made by Papamakarios et al. [2019a] and Kobyzev et al. [2020] to define the fundamental principles of flow design and by Durkan et al. [2019] to provide coding tools for modular implementations. We argue that normalizing flows would gain in popularity by offering stronger inductive bias as well as more interpretability.

Sometimes forgotten in favor of more data oriented methods, probabilistic graphical models (PGMs) have been popular for modeling complex data distributions while being relatively simple to build and read [Koller and Friedman, 2009, Johnson et al., 2016]. Among PGMs, Bayesian networks [BNs, Pearl and Russell, 2011] offer an appealing balance between modeling capacity and simplicity. Most notably, these models have been at the basis of expert systems before the big data era (e.g. [Díez et al., 1997, Kahn et al., 1997, Seixas et al., 2014]) and were commonly used to merge qualitative expert knowledge and quantitative information together. On the one hand, experts stated independence assumptions that should be encoded by the structure of the network. On the other hand, data were used to estimate the parameters of the conditional probabilities/densities encoding the quantitative aspects of the data distribution. These models have progressively received less attention from the machine learning community in favor of other methods

Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, California, USA. PMLR: Volume 130.

that scale better with the dimensionality of the data.

Driven by the objective of integrating intuition into normalizing flows and the proven relevance of BNs for combining qualitative and quantitative reasoning, we summarize our contributions as follows: **(i)** From the insight that coupling and autoregressive transformations can be reduced to Bayesian networks with a fixed topology, we introduce the more general graphical conditioner for normalizing flows, featuring either a prescribed or a learnable BN topology; **(ii)** We show that using a correct prescribed topology leads to improvements in the modeled density compared to autoregressive methods. When the topology is not known we observe that, with the right amount of ℓ_1-penalization, graphical conditioners discover relevant relationships; **(iii)** In addition, we show that graphical normalizing flows perform well in a large variety of density estimation tasks compared to classical black-box flow architectures.

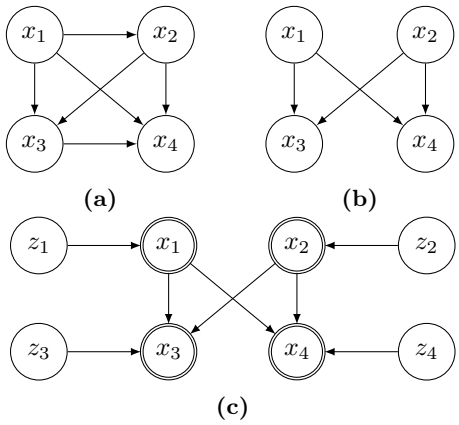


Figure 1: Bayesian networks equivalent to normalizing flows made of a single transformation step. (**a**) Autoregressive conditioner. (**b**) Coupling conditioner. (**c**) Coupling conditioner, with latent variables shown explicitly. Double circles stand for deterministic functions of the parents.

2 Background

Bayesian networks A Bayesian network is a directed acyclic graph (DAG) that represents independence assumptions between the components of a random vector. Formally, let $\mathbf{x} = [x_1, \dots, x_d]^T \in \mathbb{R}^d$ be a random vector distributed under $p_{\mathbf{x}}$. A BN associated to $\mathbf{x}$ is a directed acyclic graph made of d vertices representing the components x_i of $\mathbf{x}$. In this kind of network, the absence of edges models conditional independence between groups of components through the concept of d-separation [Geiger et al., 1990]. A BN is a valid representation of a random vector $\mathbf{x}$ iff its density can be factorized as

$$p_{\mathbf{x}}(\mathbf{x}) = \prod_{i=1}^{d} p(x_i|\mathcal{P}_i), \tag{1}$$

where $\mathcal{P}_i = \{j : A_{i,j} = 1\}$ denotes the set of parents of the vertex i and $A \in \{0,1\}^{d\times d}$ is the adjacency matrix of the BN. As an example, Fig. 1a is a valid BN for any distribution over $\mathbf{x}$ because it does not state any independence and leads to a factorization that corresponds to the chain rule. However, in practice we seek for a sparse and valid BN which models most of the independence between the components of $\mathbf{x}$, leading to an efficient factorization of the modeled probability distribution. It is worth noting that making hypotheses on the graph structure is equivalent to assuming certain conditional independence between some of the vector's components.

Normalizing flows A normalizing flow is defined as a sequence of invertible transformation steps $\mathbf{g}_k : \mathbb{R}^d \to \mathbb{R}^d$ $(k = 1, ..., K)$ composed together to create an expressive invertible mapping $\mathbf{g} := \mathbf{g}_1 \circ \cdots \circ \mathbf{g}_K : \mathbb{R}^d \to \mathbb{R}^d$. This mapping can be used to perform density estimation, using $\mathbf{g}(\cdot;\theta) : \mathbb{R}^d \to \mathbb{R}^d$ to map a sample $\mathbf{x} \in \mathbb{R}^d$ onto a latent vector $\mathbf{z} \in \mathbb{R}^d$ equipped with a prescribed density $p_{\mathbf{z}}(\mathbf{z})$ such as an isotropic Normal. The transformation $\mathbf{g}$ implicitly defines a density $p(\mathbf{x};\theta)$ as given by the change of variables formula,

$$p(\mathbf{x};\theta) = p_{\mathbf{z}}(\mathbf{g}(\mathbf{x};\theta)) \left|\det J_{\mathbf{g}(\mathbf{x};\theta)}\right|, \tag{2}$$

where $J_{\mathbf{g}(\mathbf{x};\theta)}$ is the Jacobian of $\mathbf{g}(\mathbf{x};\theta)$ with respect to $\mathbf{x}$. The resulting model is trained by maximizing the likelihood of the model's parameters θ given the training dataset $\mathbf{X} = \{\mathbf{x}^1, ..., \mathbf{x}^N\}$. Unless needed, we will not distinguish between $\mathbf{g}$ and $\mathbf{g}_k$ for the rest of our discussion.

In general the steps $\mathbf{g}$ can take any form as long as they define a bijective map. Here, we focus on a subclass of normalizing flows for which the steps can be mathematically described as

$$\mathbf{g}(\mathbf{x}) = \begin{bmatrix} g^1(x_1; \mathbf{c}^1(\mathbf{x})) & \dots & g^d(x_d; \mathbf{c}^d(\mathbf{x})) \end{bmatrix}^T, \tag{3}$$

where the $\mathbf{c}^i$ are the **conditioners** which role is to constrain the structure of the Jacobian of $\mathbf{g}$. The functions g^i, partially parameterized by their conditioner, must be invertible with respect to their input variable x_i. They are often referred to as transformers, however in this work we will use the term **normalizers** to avoid any confusion with attention-based transformer architectures.

The conditioners examined in this work can be combined with any normalizer. In particular, we consider affine and monotonic normalizers. An affine normalizer $g : \mathbb{R} \times \mathbb{R}^2 \to \mathbb{R}$ can be expressed as $g(x; m, s) = x \exp(s) + m$, where $m \in \mathbb{R}$ and $s \in \mathbb{R}$

are computed by the conditioner. There exist multiple methods to parameterize monotonic normalizers [Huang et al., 2018, De Cao et al., 2020, Durkan et al., 2019, Jaini et al., 2019], but in this work we rely on Unconstrained Monotonic Neural Networks [UMNNs, Wehenkel and Louppe, 2019] which can be expressed as $g(x;\mathbf{c}) = \int_0^x f(t,\mathbf{c})dt + \beta(\mathbf{c})$, where $\mathbf{c} \in \mathbb{R}^{|\mathbf{c}|}$ is an embedding made by the conditioner and $f : \mathbb{R}^{|\mathbf{c}|+1} \to \mathbb{R}^+$ and $\beta : \mathbb{R}^d \to \mathbb{R}$ are two neural networks respectively with a strictly positive scalar output and a real scalar output. Huang et al. [2018] proved NFs built with autoregressive conditioners and monotonic normalizers are universal density approximators of continuous random variables.

3 Normalizing flows as Bayesian networks

Autoregressive conditioners Due to computing speed considerations, NFs are usually composed of transformations for which the determinant of the Jacobian can be computed efficiently, as otherwise its evaluation would scale cubically with the input dimension. A common solution is to use autoregressive conditioners, i.e., such that

$$\mathbf{c}^i(\mathbf{x}) = \mathbf{h}^i\left(\begin{bmatrix} x_1 & \dots & x_{i-1}\end{bmatrix}^T\right)$$

are functions $\mathbf{h}^i$ of the first $i-1$ components of $\mathbf{x}$. This particular form constrains the Jacobian of $\mathbf{g}$ to be lower triangular, which makes the computation of its determinant $\mathcal{O}(d)$.

For the multivariate density $p(\mathbf{x};\theta)$ induced by $\mathbf{g}(\mathbf{x};\theta)$ and $p_{\mathbf{z}}(\mathbf{z})$, we can use the chain rule to express the joint probability of $\mathbf{x}$ as a product of d univariate conditional densities,

$$p(\mathbf{x};\theta) = p(x_1;\theta)\prod_{i=2}^{d} p(x_i|\mathbf{x}_{1:i-1};\theta). \tag{4}$$

When $p_{\mathbf{z}}(\mathbf{z})$ is a factored distribution $p_{\mathbf{z}}(\mathbf{z}) = \prod_{i=1}^d p(z_i)$, we identify that each component z_i coupled with the corresponding functions g^i and embedding vectors $\mathbf{c}^i$ encode for the conditional $p(x_i|\mathbf{x}_{1:i-1};\theta)$. Therefore, and as illustrated in Fig. 1a, autoregressive transformations can be seen as a way to model the conditional factors of a BN that does not state any independence but relies on a predefined node ordering. This becomes clear if we define $\mathcal{P}_i = \{x_1,\dots,x_{i-1}\}$ and compare (4) with (1).

The complexity of the conditional factors strongly depends on the ordering of the vector components. While not hurting the universal representation capacity of normalizing flows, the arbitrary ordering used in autoregressive transformations leads to poor inductive bias and to factors that are most of the time difficult to learn. In practice, one often alleviates the arbitrariness of the ordering by stacking multiple autoregressive transformations combined with random permutations on top of each other.

Coupling conditioners Coupling layers [Dinh et al., 2015] are another popular type of conditioners that lead to a bipartite structure. The conditioners $\mathbf{c}^i$ made from coupling layers are defined as

$$\mathbf{c}^i(\mathbf{x}) = \begin{cases} \underline{\mathbf{h}}^i & \text{if} \quad i \leq k \\ \mathbf{h}^i\left(\begin{bmatrix} x_1 & \dots & x_k\end{bmatrix}^T\right) & \text{if} \quad i > k \end{cases}$$

where the underlined $\underline{\mathbf{h}}^i \in \mathbb{R}^{|\mathbf{c}|}$ denote constant values and $k \in \{1,\dots,d\}$ is a hyper-parameter usually set to $\lfloor \frac{d}{2} \rfloor$. As for autoregressive conditioners, the Jacobian of $\mathbf{g}$ made of coupling layers is lower triangular. Again, and as shown in Fig. 1b and 1c, these transformations can be seen as a specific class of BN where $\mathcal{P}_i = \{\}$ for $i \leq k$ and $\mathcal{P}_i = \{1,...,k\}$ for $i > k$. D-separation can be used to read off the independencies stated by this class of BNs such as the conditional independence between each pair in $\mathbf{x}_{k+1:d}$ knowing $\mathbf{x}_{1:k}$. For this reason, and in contrast to autoregressive transformations, coupling layers are not by themselves universal density approximators even when associated with very expressive normalizers g^i [Wehenkel and Louppe, 2020]. In practice, these bipartite structural independencies can be relaxed by stacking multiple layers, and may even recover an autoregressive structure. They also lead to useful inductive bias, such as in the multi-scale architecture with checkerboard masking [Dinh et al., 2017, Kingma and Dhariwal, 2018].

Algorithm 1 Sampling

1: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, I)$
2: $\mathbf{x} \leftarrow \mathbf{z}$
3: **repeat**
4: $\quad \mathbf{c}^i \leftarrow \mathbf{h}^i(\mathbf{x} \odot A_{i,:}) \quad \forall i \in \{1,\dots,d\}$
5: $\quad x_i \leftarrow (g^i)^{-1}(z_i;\mathbf{c}^i,\theta) \quad \forall i \in \{1,\dots,d\}$
6: **until** $\mathbf{x}$ converged

4 Graphical normalizing flow

4.1 Graphical conditioners

Following up on the previous discussion, we introduce the graphical conditioner architecture. We motivate our approach by observing that the topological ordering (a.k.a. ancestral ordering) of any BN leads to a

lower triangular adjacency matrix whose determinant is equal to the product of its diagonal terms (proof in Appendix B). Therefore, conditioning factors $\mathbf{c}^i(\mathbf{x})$ selected by following a BN adjacency matrix necessarily lead to a transformation $\mathbf{g}$ whose Jacobian determinant remains efficient to compute.

Formally, given a BN with adjacency matrix $A \in \{0,1\}^{d\times d}$, we define the **graphical conditioner** as being

$$\mathbf{c}^i(\mathbf{x}) = \mathbf{h}^i(\mathbf{x} \odot A_{i,:}), \tag{5}$$

where $\mathbf{x} \odot A_{i,:}$ is the element-wise product between the vector $\mathbf{x}$ and the i^{th} row of A – i.e., the binary vector $A_{i,:}$ is used to mask on $\mathbf{x}$. NFs built with this new conditioner architecture can be inverted by sequentially inverting each component in the topological ordering. In our implementation the neural networks modeling the h^i functions are shared to save memory and they take an additional input that one-hot encodes the value i. An alternative approach would be to use a masking scheme similar to what is done by Germain et al. [2015] in MADE as suggested by Lachapelle et al. [2019].

The graphical conditioner architecture can be used to learn the conditional factors in a continuous BN while elegantly setting structural independencies prescribed from domain knowledge. In addition, the inverse of NFs built with graphical conditioners is a simple as it is for autoregressive and coupling conditioners, Algorithm 1 describes an inversion procedure. We also now note how these two conditioners are just special cases in which the adjacency matrix reflects the classes of BNs discussed in Section 3.

4.2 Learning the topology

In many cases, defining the whole structure of a BN is not possible due to a lack of knowledge about the problem at hand. Fortunately, not only is the density at each node learnable, but also the DAG structure itself: defining an arbitrary topology and ordering, as it is implicitly the case for autoregressive and coupling conditioners, is not necessary.

Building upon *Non-combinatorial Optimization via Trace Exponential and Augmented lagRangian for Structure learning* [NO TEARS, Zheng et al., 2018], we convert the combinatorial optimization of score-based learning of a DAG into a continuous optimization by relaxing the domain of A to real numbers instead of binary values. That is,

$$\begin{aligned} \max_{A\in\mathbb{R}^{d\times d}} \quad & F(A) \\ \text{s.t.} \quad & \mathcal{G}(A) \in \mathsf{DAGs} \end{aligned} \iff \begin{aligned} \max_{A\in\mathbb{R}^{d\times d}} \quad & F(A) \\ \text{s.t.} \quad & w(A) = 0, \end{aligned} \tag{6}$$

where $\mathcal{G}(A)$ is the graph induced by the weighted adjacency matrix A and $F : \mathbb{R}^{d\times d} \to \mathbb{R}$ is the log-likelihood of the graphical NF $\mathbf{g}$ plus a regularization term, i.e.,

$$F(A) = \sum_{j=1}^{N} \log\left(p(\mathbf{x}^j;\theta)\right) + \lambda_{\ell_1}||A||_1, \tag{7}$$

where λ_{ℓ_1} is an ℓ_1-regularization coefficient and N is the number of training samples $\mathbf{x}^i$. The likelihood is computed as

$$p(\mathbf{x};\theta) = p_{\mathbf{z}}(\mathbf{g}(\mathbf{x};\theta)) \prod_{i=1}^{d} \left| \frac{\partial g^i(x_i; \mathbf{h}^i(\mathbf{x} \odot A_{i,:}), \theta)}{\partial x_i} \right|.$$

The function $w(A)$ that enforces the acyclicity is expressed as suggested by Yu et al. [2019] as

$$w(A) := \operatorname{tr}\left((I + \alpha A)^d\right) - d \propto \operatorname{tr}\left(\sum_{k=1}^{d} \alpha^k A^k\right),$$

where $\alpha \in \mathbb{R}_+$ is a hyper-parameter that avoids exploding values for $w(A)$. In the case of positively valued A, an element (i,j) of $A^k = \underbrace{AA...A}_{\text{k terms}}$ is non-null if and only if there exists a path going from node j to node i that is made of exactly k edges. Intuitively $w(A)$ expresses to which extent the graph is cyclic. Indeed, the diagonal elements (i,i) of A^k will be as large as there are many paths made of edges that correspond to large values in A from a node to itself in k steps.

In comparison to our work, Zheng et al. [2018] use a quadratic loss on the corresponding linear structural equation model (SEM) as the score function $F(A)$. By attaching normalizing flows to topology learning, our method has a continuously adjustable level of complexity and does not make any strong assumptions on the form of the conditional factors.

4.3 Stochastic adjacency matrix

In order to learn the BN topology from the data, the adjacency matrix must be relaxed to contain reals instead of booleans. It also implies that the graph induced by A does not formally define a DAG during training. Work using NO TEARS to perform topology learning directly plug the real matrix A in (6) [Zheng et al., 2018, Yu et al., 2019, Lachapelle et al., 2019, Zheng et al., 2020] however this is inadequate because the quantity of information going from node j to node i does not continuously relate to the value of $A_{i,j}$. Either the information is null if $A_{i,j} = 0$ or it passes completely if not. Instead, we propose to build the stochastic pseudo binary valued matrix A' from A,

defined as

$$A'_{i,j} = \frac{e^{\frac{\log(\sigma(A^2_{i,j}))+\gamma_1}{T}}}{e^{\frac{\log(\sigma(A^2_{i,j}))+\gamma_1}{T}} + e^{\frac{\log(1-\sigma(A^2_{i,j}))+\gamma_2}{T}}},$$

where $\gamma_1, \gamma_2 \sim \text{Gumbel}(0,1)$ and $\sigma(a) = 2(\text{sigmoid}(2a^2) - \frac{1}{2})$ normalizes the values of A between 0 and 1, being close to 1 for large values and close to zero for values close to 0. The hyper-parameter T controls the sampling temperature and is fixed to 0.5 in all our experiments. In contrast to directly using the matrix A, this stochastic transformation referred to as the Gumbel-Softmax trick in the literature [Maddison et al., 2016, Jang et al., 2016] allows to create a direct and continuously differentiable relationship between the weights of the edges and the quantity of information that can transit between two nodes. Indeed, the probability mass of the random variables $A'_{i,j}$ is mainly located around 0 and 1, and its expected value converges to 1 when $A_{i,j}$ increases.

4.4 Optimization

We rely on the augmented Lagrangian approach to solve the constrained optimization problem (6) as initially proposed by Zheng et al. [2018]. This optimization procedure requires solving iteratively the following sub-problems:

$$\max_A \mathbb{E}_{\gamma_1,\gamma_2}[F(A)] - \lambda_t w(A) - \frac{\mu_t}{2} w(A)^2, \qquad (8)$$

where λ_t and μ_t respectively denote the Lagrangian multiplier and penalty coefficients of the sub-problem t.

We solve these optimization problems with mini-batch stochastic gradient ascent. We update the values of γ_t and μ_t as suggested by Yu et al. [2019] when the validation loss does not improve for 10 consecutive epochs. Once $w(A)$ equals 0, the adjacency matrix is acyclic up to numerical errors. We recover an exact DAG by thresholding the elements of A while checking for acyclicity with a path finding algorithm. We provide additional details about the optimization procedure used in our experiments in Appendix A.

5 Experiments

In this section, we demonstrate some applications of graphical NFs in addition to unifying NFs and BN under a common framework. We first demonstrate how pre-loading a known or hypothesized DAG structure can help finding an accurate distribution of the data. Then, we show that learning the graph topology leads to relevant BNs that support generalization well when combined with ℓ_1-penalization. Finally, we demonstrate that mono-step normalizing flows made of graphical conditioners are competitive density estimators.

Table 1: Datasets description. d=Dimension of the data. V=Number of edges in the ground truth Bayesian Network.

Dataset	d	V	Train	Test
Arithmetic Circuit	8	8	10,000	5,000
8 Pairs	16	8	10,000	5,000
Tree	7	8	10,000	5,000
Protein	11	20	6,000	1,466
POWER	6	≤ 15	1,659,917	204,928
GAS	8	≤ 28	852,174	105,206
HEPMASS	21	≤ 210	315,123	174,987
MINIBOONE	43	≤ 903	29,556	3,648
BSDS300	63	$\leq 1,953$	1,000,000	250,000

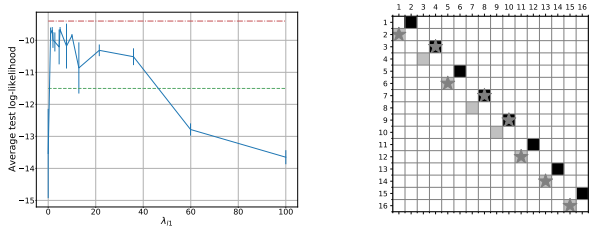
Table 2: Graphical vs autoregressive conditioners combined with monotonic normalizers. Average log-likelihood on test data over 5 runs, under-scripted error bars are equal to the standard deviation. Results are reported in nats; higher is better. The best performing architecture for each dataset is written in bold. *Graphical conditioners clearly lead to improved density estimation when given a relevant prescribed topology in 3 out of the 4 datasets.*

Conditioner	Graphical	Autoreg.
Arithmetic Circuit	$\mathbf{3.99}_{\pm.16}$	$3.06_{\pm.38}$
8 Pairs	$\mathbf{-9.40}_{\pm.06}$	$-11.50_{\pm.27}$
Tree	$\mathbf{-6.85}_{\pm.02}$	$-6.96_{\pm.05}$
Protein	$6.46_{\pm.08}$	$\mathbf{7.52}_{\pm.10}$

5.1 On the importance of graph topology

The following experiments are performed on four distinct datasets, three of which are synthetic, such that we can define a ground-truth minimal Bayesian network, and the fourth is a causal protein-signaling network derived from single-cell data [Sachs et al., 2005]. Additional information about the datasets are provided in Table 1 and in Appendix C. For each experimental run we first randomly permute the features before training the model in order to compare autoregressive and graphical conditioners fairly.

Prescribed topology Rarely do real data come with their associated Bayesian network however oftentimes experts want to hypothesize a network topology and to rely on it for the downstream tasks. Sometimes the topology is known a priori, as an example the sequence of instructions in stochastic simulators can usually be translated into a graph topology (e.g. in probabilistic programming [van de Meent et al., 2018, Weilbach et al., 2020]). In both cases, graphical conditioners allow to explicitly take advantage of this to



(a) (b)

Figure 2: **(a)**: Test log-likelihood as a function of ℓ_1-penalization on **8 pairs** dataset. The upper bound is the average result when given a prescribed topology, the lower bound is the result with an autoregressive conditioner. *Learning the right topology leads to better results than autoregressive conditioners.* **(b)**: The blacked cells corresponds to one correct topology of the 8 pairs dataset and the grey cells to the transposed adjacency matrix. The stars denote the edges discovered by the graphical conditioner when trained with $\lambda_{\ell_1} = 4$. *The optimization discovers a relevant BN (equivalent to the ground truth).*

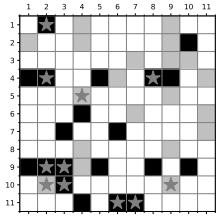


Figure 3: The adjacency matrix of the protein interaction network. The blacked cells are the directed connections proposed by domain experts and the grey is the transposed adjacency matrix. The stars denote the edges discovered by the graphical conditioner when trained with $\lambda_{\ell_1} = 12$. *In a realistic setting, the optimization leads to a graph that shares a lot with the one designed by experts.*

build density estimators while keeping the assumptions made about the network topology valid.

Table 2 presents the test likelihood of autoregressive and graphical normalizing flows on the four datasets. The flows are made of a single step and use monotonic normalizers. The neural network architectures are all identical. Further details on the experimental settings as well as additional results for affine normalizers are respectively provided in Appendix C.2 and C.3. *We observe how using correct BN structures lead to good test performance in Table 2.* Surprisingly, the performance on the protein dataset are not improved when using the ground truth graph. We observed during our experiments (see Appendix C.3) that learning the topology from this dataset sometimes led to improved density estimation performance with respect to the ground truth graph. The limited dataset size does not allow us to answer if this comes from to the limited capacity of the flow and/or from the erroneous assumptions in the ground truth graph. However, we stress out that the graphical flow respects the assumed causal structure in opposition to the autoregressive flow.

Learning the topology The λ_{ℓ_1} coefficient introduced in (7) controls the sparsity of the optimized BN, allowing to avoid some spurious connections. We now analyze the effect of sparsity on the density estimation performance. Fig. 2a shows the test log-likelihood as a function of the ℓ_1-penalization on the 8 pairs dataset. We observe that the worst results are obtained when there is no penalization. Indeed, in this case the algorithm finds multiple spurious connections between independent vector's components and then overfits on these incorrect relationships. Another extreme case shows up when the effect of penalization is too strong, in this case, the learning procedure underfits the data because it ignores too many relevant connections. It can also be concluded from the plot that the optimal ℓ_1-penalization performs on par with the ground truth topology, and certainly better than the autoregressive conditioner. Additional results on the other datasets provided in Appendix C lead to similar conclusions.

Protein network The adjacency matrix discovered by optimizing (8) (with $\lambda_{\ell_1} = 12$) on the protein network dataset is shown in Fig. 3. All the connections discovered by the method correspond to ground truth connections. Unsurprisingly, their orientation do not always match with what is expected by the experts. Overall we can see that the optimization procedure is able to find relevant connections between variables and avoids spurious ones when the ℓ_1-penalization is optimized. Previous work on topology learning such as Zheng et al. [2018], Yu et al. [2019], Lachapelle et al. [2019] compare their method to others by looking at the topology discovered on the same dataset. Here we do not claim that learning the topology with a graphical conditioner improves over previous methods. Indeed, we believe that the difference between the methods mainly relies on the hypothesis and/or inductive biased made on the conditional densities. Which method would perform better is dependent on the application. Moreover, the Structural Hamiltonian Distance (SHD) and the Structural Inference Distance (SID), often used to compare BNs, are not well motivated in general. On the one hand the SHD does not take into account the independence relationships modeled with Bayesian networks, as an example the SHD between the graph found on 8 pairs dataset of Fig. 2b and one possible ground truth is non zero whereas the two BNs are equivalent. On the other hand the SID aims to compare causal graphs whereas the methods

Table 3: Average log-likelihood on test data over 3 runs, under-scripted error bars are equal to the standard deviation. Results are reported in nats; higher is better. The best performing architecture per category for each dataset is written in bold. (a) 1-step affine normalizers (b) 1-step monotonic normalizers. *Graphical normalizing flows outperform coupling and autoregressive architectures on most of the datasets.*

	Dataset	POWER	GAS	HEPMASS	MINIBOONE	BSDS300
(a)	Coup.	$-5.60_{\pm.00}$	$-3.05_{\pm.01}$	$-25.74_{\pm.01}$	$-38.34_{\pm.02}$	$57.33_{\pm.00}$
	Auto.	$-3.55_{\pm.00}$	$-0.34_{\pm.01}$	$-21.66_{\pm.01}$	$\mathbf{-16.70_{\pm.05}}$	$\mathbf{63.74_{\pm.00}}$
	Graph.	$\mathbf{-2.80_{\pm.01}}$	$\mathbf{1.99_{\pm.02}}$	$\mathbf{-21.18_{\pm.07}}$	$-19.67_{\pm.06}$	$62.85_{\pm.07}$
(b)	Coup.	$0.25_{\pm.00}$	$5.12_{\pm.03}$	$-20.55_{\pm.04}$	$-32.04_{\pm.12}$	$107.17_{\pm.46}$
	Auto.	$0.58_{\pm.00}$	$9.79_{\pm.04}$	$-14.52_{\pm.16}$	$\mathbf{-11.66_{\pm.02}}$	$151.29_{\pm.31}$
	Graph.	$\mathbf{0.62_{\pm.04}}$	$\mathbf{10.15_{\pm.15}}$	$\mathbf{-14.17_{\pm.13}}$	$-16.23_{\pm.52}$	$\mathbf{155.22_{\pm.11}}$

Table 4: Average log-likelihood on test data over 3 runs, under-scripted error bars are equal to the standard deviation. Results are reported in nats, higher is better. The results followed by a star are copied from the literature and the number of steps in the flow is indicated in parenthesis for each architecture. *Graphical normalizing flows reach density estimation performance on par with the most popular flow architectures whereas it is only made of 1 transformation step.*

Dataset	POWER	GAS	HEPMASS	MINIBOONE	BSDS300
Graph.-UMNN (1)	$0.62_{\pm.04}$	$10.15_{\pm.15}$	$-14.17_{\pm.13}$	$-16.23_{\pm.52}$	$155.22_{\pm.11}$
MAF (5)	$0.14_{\pm.01}$	$9.07_{\pm.01}$	$-17.70_{\pm.01}$	$-11.75_{\pm.22}$	$155.69_{\pm.14}$
Glow* (10)	$0.42_{\pm.01}$	$12.24_{\pm.03}$	$-16.99_{\pm.02}$	$-10.55_{\pm.45}$	$156.95_{\pm.28}$
UMNN-MAF* (5)	$0.63_{\pm.01}$	$10.89_{\pm.70}$	$-13.99_{\pm.21}$	$-9.67_{\pm.13}$	$157.98_{\pm.01}$
Q-NSF* (10)	$0.66_{\pm.01}$	$12.91_{\pm.01}$	$-14.67_{\pm.02}$	$-9.72_{\pm.24}$	$157.42_{\pm.14}$
FFJORD* (5-5-10-1-2)	$0.46_{\pm.01}$	$8.59_{\pm.12}$	$-14.92_{\pm.08}$	$-10.43_{\pm.04}$	$157.40_{\pm.19}$

discussed here do not learn causal relationships. In general BN topology identification is an ill posed problem and thus we believe using these metrics to compare different methods without additional downstream context is irrelevant. However, we conclude from Fig. 2b and Fig. 3 that the scores computed with graphical NFs can be used to learn relevant BN structures.

5.2 Density estimation benchmark

In these experiments, we compare autoregressive, coupling and graphical conditioners with no ℓ_1-penalization on benchmark tabular datasets as introduced by Papamakarios et al. [2017] for density estimation. See Table 1 for a description. We evaluate each conditioner in combination with monotonic and affine normalizers. We only compare NFs with a single transformation step because our focus is on the conditioner capacity. We observed during preliminary experiments that stacking multiple conditioners improves the performance slightly, however the gain is marginal compared to the loss of interpretability. To provide a fair comparison we have fixed in advance the neural architectures used to parameterize the normalizers and conditioners as well as the training parameters by taking inspiration from those used by Wehenkel and Louppe [2019] and Papamakarios et al. [2017]. The variable ordering of each dataset is randomly permuted at each run. All hyper-parameters are provided in Appendix D and a public implementation will be released on Github.

First, Table 3 presents the test log-likelihood obtained by each architecture. These results indicate that graphical conditioners offer the best performance in general. Unsurprisingly, coupling layers show the worst performance, due to the arbitrarily assumed independencies. Autoregressive and graphical conditioners show very similar performance for monotonic normalizers, the latter being slightly better on 4 out of the 5 datasets. Table 4 contextualizes the performance of graphical normalizing flows with respect to the most popular normalizing flow architectures. Comparing the results together, we see that while additional steps lead to noticeable improvements for affine normalizers (MAF), benefits are questionable for monotonic transformations. Overall, graphical normalizing flows made of a single transformation step are competitive with the best flow architectures with the added value that they can directly be translated into their equivalent BNs. *From these results we stress out that single step graphical NFs are able to model complex densities on par with SOTA while they offer new ways of introducing domain knowledge.*

Second, Table 5 presents the number of edges in the BN associated with each flow. For POWER and GAS, the number of edges found by the graphical conditioners is close or equal to the maximum number of edges. Interestingly, graphical conditioners outperform autoregressive conditioners on these two tasks, demonstrating the value of finding an appropriate ordering particularly when using affine normalizers. Moreover, graphical conditioners correspond to BNs whose sparsity is largely greater than for autoregressive conditioners while providing equivalent if not better performance. The depth [Bezek, 2016] of the equivalent BN directly limits the number of steps required to inverse the flow. Thus sparser graphs that are inevitably shallower correspond to NFs for which sampling, i.e. computing their inverse, is faster.

6 Discussion

Cost of learning the graph structure The attentive reader will notice that learning the topology does not come for free. Indeed, the Lagrangian for-

Table 5: Rounded average number of edges (over 3 runs) in the equivalent Bayesian network. *The graphical conditioners lead to sparser BNs compared to autoregressive conditioners.*

Dataset	P	G	H	M	B
Graph.-Aff.	15	26	152	277	471
Graph.-Mon.	15	27	159	265	1594
Coupling	9	16	110	462	992
Autoreg.	15	28	210	903	1953

mulation requires solving a sequence of optimization problems which increases the number of epochs before convergence. In our experiments we observed different overheads depending on the problems, however in general the training time is at least doubled. This does not impede the practical interest of using graphical normalizing flows. The computation overhead is more striking for non-affine normalizers (e.g. UMNNs) that are computationally heavy. However, we observed that most of the time the topology recovered by affine graphical NFs is relevant. It can thus be used as a prescribed topology for normalizers that are heavier to run, hence alleviating the computation overhead. Moreover, one can always hypothesize on the graph topology but more importantly the graph learned is usually sparser than an autoregressive one while achieving similar if not better results. The sparsity is interesting for two reasons: it can be exploited for speeding up the forward density evaluation; but more importantly it usually corresponds to shallower BNs that can be inverted faster than autoregressive structures.

Bayesian network topology learning Formal BN topology learning has extensively been studied for more than 30 years now and many strong theoretical results on the computational complexity have been obtained. Most of these results however focus on discrete random variables, and how they generalize in the continuous case is yet to be explained. The topic of BN topology learning for discrete variables has been proven to be NP-hard by Chickering et al. [2004]. However, while some greedy algorithms exist, they do not lead in general to a minimal I-map although allowing for an efficient factorization of random discrete vectors distributions in most of the cases. These algorithms are usually separated between the constrained-based family such as the *PC algorithm* [Spirtes et al., 2001] or the *incremental association Markov blanket* [Koller and Friedman, 2009] and the score-based family as used in the present work. Finding the best BN topology for continuous variables has not been proven to be NP-hard however the results for discrete variables suggest that without strong assumptions on the function class the problem is hard.

The recent progress made in the continuous setting relies on the heuristic used in score-based methods. In particular, Zheng et al. [2018] showed that the acyclicity constraint required in BNs can be expressed with NO TEARS, as a continuous function of the adjacency matrix, allowing the Lagrangian formulation to be used. Yu et al. [2019] proposed DAG-GNN, a follow up work of Zheng et al. [2018] which relies on variational inference and auto-encoders to generalize the method to non-linear structural equation models. Further investigation of continuous DAG learning in the context of causal models was carried out by Lachapelle et al. [2019]. They use the adjacency matrix of the causal network as a mask over neural networks to design a score which is the log-likelihood of a parameterized normal distribution. The requirement to pre-define a parametric distribution before learning restricts the factors to simple conditional distributions. In contrast, our method combines the constraints given by the BN topology with NFs which are free-form universal density estimators. Remarkably, their method leads to an efficient one-pass computation of the joint density. This neural masking scheme can also be implemented for NF architectures such as already demonstrated by Papamakarios et al. [2017] and De Cao et al. [2019] for autoregressive conditioners.

Shuffling between transformation steps As already mentioned, consecutive transformation steps are often combined with randomly fixed permutations in order to mitigate the ordering problem. Linear flow steps [Oliva et al., 2018] and 1x1 invertible convolutions [Kingma and Dhariwal, 2018] generalize these fixed permutations. They are parameterized by a matrix $W = PLU$ where P is the fixed permutation matrix, and L and U are respectively a lower and an upper triangular matrix. Although linear flow improves the simple permutation scheme, they do still rely on an arbitrary permutation. To the best of our knowledge, graphical conditioners are the first attempt to get completely rid of any fixed permutation in NFs.

Inductive bias Graphical conditioners eventually lead to binary masks that model the conditioning components of a factored joint distribution. In this way, the conditioners process their input as they would process the full vector.We show experimentally in Appendix E that this effectively leads to good inductive bias for processing images with NFs. In addition, we have shown that normalizing flows built from graphical conditioners combined with monotonic transformations are expressive density estimators. In effect, this means that enforcing some a priori known independencies can be performed thanks to graphical normal-

izing flows without hurting their modeling capacity. We believe such models could be of high practical interest because they cope well with large datasets and complex distributions while preserving some readability through their equivalent BN.

Close to our work, Weilbach et al. [2020] improve amortized inference by prescribing a BN structure between the latent and observed variables into a FFJORD NF, once again showing the interest of using the potential BN knowledge. Similar to our work, Khemakhem et al. [2020] see causal autoregressive flows as structural equation modelling. They show bivariate autoregressive affine flows can be used to identify the causal direction under mild conditions. Under similar mild conditions, discovering causal relationships with graphical normalizing flows could well be an exciting research direction.

Conclusion We have revisited coupling and autoregressive conditioners for normalizing flows as Bayesian networks. From this new perspective, we proposed the more general graphical conditioner architecture for normalizing flows. We have illustrated the importance of assuming or learning a relevant Bayesian network topology for density estimation. In addition, we have shown that this new architecture compares favorably with autoregressive and coupling conditioners and on par to the most common flow architectures on standard density estimation tasks even without any hypothesized topology. One interesting and straightforward extension of our work would be to combine it with normalizing flows for discrete variables. We also believe that graphical conditioners could be used when the equivalent Bayesian network is required for downstream tasks such as in causal reasoning.

Acknowledgments

We thank Vân Anh Huynh-Thu, Johann Brehmer, and Louis Wehenkel for proofreading this manuscript or an earlier version of it. We are also thankful to the reviewers for helpful comments. Antoine Wehenkel is a research fellow of the F.R.S.-FNRS (Belgium) and acknowledges its financial support. Gilles Louppe is recipient of the ULiège - NRB Chair on Big data and is thankful for the support of NRB.

References

Matúš Bezek. Characterizing dag-depth of directed graphs. *arXiv preprint arXiv:1612.04980*, 2016.

David Maxwell Chickering, David Heckerman, and Christopher Meek. Large-sample learning of Bayesian networks is NP-hard. *Journal of Machine Learning Research*, 5(Oct):1287–1330, 2004.

Nicola De Cao, Ivan Titov, and Wilker Aziz. Block neural autoregressive flow. *arXiv preprint arXiv:1904.04676*, 2019.

Nicola De Cao, Wilker Aziz, and Ivan Titov. Block neural autoregressive flow. In *Uncertainty in Artificial Intelligence*, pages 1263–1273. PMLR, 2020.

Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. In *International Conference in Learning Representations workshop track*, 2015.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. In *International Conference in Learning Representations*, 2017.

Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. In *Advances in Neural Information Processing Systems*, pages 7509–7520, 2019.

F.J. Díez, J. Mira, E. Iturralde, and S. Zubillaga. DIAVAL, a Bayesian expert system for echocardiography. *Artif. Intell. Med.*, 10(1):59–73, May 1997. URL https://doi.org/10.1016/s0933-3657(97)00384-9.

Dan Geiger, Thomas Verma, and Judea Pearl. d-separation: From theorems to algorithms. In *Uncertainty in Artificial Intelligence*, volume 10, pages 139–148. Elsevier, 1990. URL https://doi.org/10.1016/b978-0-444-88738-2.50018-x.

Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pages 881–889, 2015.

Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. FFJORD: Free-form continuous dynamics for scalable reversible generative models. In *International Conference on Machine Learning*, 2018.

David S Greenberg, Marcel Nonnenmacher, and Jakob H Macke. Automatic posterior transformation for likelihood-free inference. *arXiv preprint arXiv:1905.07488*, 2019.

Chin-Wei Huang, David Krueger, Alexandre Lacoste, and Aaron Courville. Neural autoregressive flows. *arXiv preprint arXiv:1804.00779*, 2018.

Priyank Jaini, Kira A Selby, and Yaoliang Yu. Sum-of-squares polynomial flow. *arXiv preprint arXiv:1905.02325*, 2019.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Matthew J Johnson, David K Duvenaud, Alex Wiltschko, Ryan P Adams, and Sandeep R Datta.

Composing graphical models with neural networks for structured representations and fast inference. In *Advances in neural information processing systems*, pages 2946–2954, 2016.

Charles E. Kahn, Linda M. Roberts, Katherine A. Shaffer, and Peter Haddawy. Construction of a Bayesian network for mammographic diagnosis of breast cancer. *Comput. Biol. Med.*, 27(1):19–29, January 1997. URL https://doi.org/10.1016/s0010-4825(96)00039-x.

Ilyes Khemakhem, Ricardo Pio Monti, Robert Leech, and Aapo Hyvärinen. Causal autoregressive flows. *arXiv preprint arXiv:2011.02268*, 2020.

Sungwon Kim, Sang-gil Lee, Jongyoon Song, Jaehyeon Kim, and Sungroh Yoon. Flowavenet: A generative flow for raw audio. *arXiv preprint arXiv:1811.02155*, 2018.

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pages 10236–10245, 2018.

Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751, 2016.

Ivan Kobyzev, Simon Prince, and Marcus Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

Daphne Koller and Nir Friedman. *Probabilistic graphical models: Principles and techniques*. MIT press, 2009.

Sébastien Lachapelle, Philippe Brouillard, Tristan Deleu, and Simon Lacoste-Julien. Gradient-based neural dag learning. *arXiv preprint arXiv:1906.02226*, 2019.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

Junier Oliva, Avinava Dubey, Manzil Zaheer, Barnabas Poczos, Ruslan Salakhutdinov, Eric Xing, and Jeff Schneider. Transformation autoregressive networks. In *International Conference on Machine Learning*, pages 3895–3904, 2018.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pages 2338–2347, 2017.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *arXiv preprint arXiv:1912.02762*, 2019a.

George Papamakarios, David C Sterratt, and Iain Murray. Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. In *22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019b.

Judea Pearl and Stuart Russell. *Bayesian networks*. California Digital Library, 2011.

Ryan Prenger, Rafael Valle, and Bryan Catanzaro. Waveglow: A flow-based generative network for speech synthesis. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3617–3621. IEEE, 2019.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538, 2015.

Oren Rippel and Ryan Prescott Adams. High-dimensional probability estimation with deep density models. *arXiv preprint arXiv:1302.5125*, 2013.

K. Sachs, O. Perez, D. Pe'er, D. A. Lauffenburger, and G. P. Nolan. Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 308:523–529, 2005.

Robert Sedgewick and Kevin Wayne. *Algorithms*. Addison-wesley professional, 2011.

Flávio Luiz Seixas, Bianca Zadrozny, Jerson Laks, Aura Conci, and Débora Christina Muchaluat Saade. A Bayesian network decision model for supporting the diagnosis of dementia, alzheimer's disease and mild cognitive impairment. *Comput. Biol. Med.*, 51:140–158, August 2014. URL https://doi.org/10.1016/j.compbiomed.2014.04.010.

Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction, and Search*. The MIT Press, 2001. URL https://doi.org/10.7551/mitpress/1754.001.0001.

Esteban G Tabak and Cristina V Turner. A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, 66 (2):145–164, 2013.

Esteban G Tabak, Eric Vanden-Eijnden, et al. Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences*, 8(1): 217–233, 2010.

Jan-Willem van de Meent, Brooks Paige, Hongseok Yang, and Frank Wood. An introduction

to probabilistic programming. *arXiv preprint arXiv:1809.10756*, 2018.

Rianne Van Den Berg, Leonard Hasenclever, Jakub M Tomczak, and Max Welling. Sylvester normalizing flows for variational inference. *arXiv preprint arXiv:1803.05649*, 2018.

Aäron Van Den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George Driessche, Edward Lockhart, Luis Cobo, Florian Stimberg, et al. Parallel WaveNet: Fast high-fidelity speech synthesis. In *International Conference on Machine Learning*, pages 3915–3923, 2018.

Antoine Wehenkel and Gilles Louppe. Unconstrained monotonic neural networks. In *Advances in Neural Information Processing Systems*, pages 1543–1553, 2019.

Antoine Wehenkel and Gilles Louppe. You say normalizing flows i see bayesian networks. *arXiv preprint arXiv:2006.00866*, 2020.

Christian Weilbach, Boyan Beronov, Frank Wood, and William Harvey. Structured conditional continuous normalizing flows for efficient amortized inference in graphical models. In *International Conference on Artificial Intelligence and Statistics*, pages 4441–4451. PMLR, 2020.

Yue Yu, Jie Chen, Tian Gao, and Mo Yu. DAG-GNN: Dag structure learning with graph neural networks. In *International Conference on Machine Learning*, pages 7154–7163, 2019.

Xun Zheng, Bryon Aragam, Pradeep K Ravikumar, and Eric P Xing. DAGs with NO TEARS: Continuous optimization for structure learning. In *Advances in Neural Information Processing Systems*, pages 9472–9483, 2018.

Xun Zheng, Chen Dan, Bryon Aragam, Pradeep Ravikumar, and Eric Xing. Learning sparse nonparametric dags. In *International Conference on Artificial Intelligence and Statistics*, pages 3414–3425. PMLR, 2020.

A Optimization procedure

Algorithm 2 Main Loop

epoch $\leftarrow 0$
while !Stopping criterion **do**
  foreach batch $\mathbf{X} \in \mathbf{X_{train}}$ **do**
    loss $\leftarrow$ computeLoss(flow, X)
    optimize(flow, loss)
  $\text{loss}_{valid} \leftarrow$ computeLoss(flow, $\mathbf{X_{test}}$)
  epoch $\leftarrow$ epoch $+ 1$
  updateCoefficients(flow, epoch, loss_{valid})
  if isDagConstraintNull(flow) **then**
    PostProcess(flow)

The method computeLoss(flow, X) is computed as described by equation (8). The optimize(flow, loss) method performs a backward pass and an optimization step with the chosen optimizer (Adam in our experiments). The post-processing is peformed by PostProcess(flow) and consists in thresholding the values in A such that the values below a certain threshold are set to 0 and the other values to 1, after post-processing the stochastic door is deactivated. The threshold is the smallest real value that makes the equivalent graph acyclic. The method updateCoefficients() updates the Lagrangian coefficients as described in section 4.4.

B Jacobian of graphical conditioners

Proposition B.1. *The absolute value of the determinant of the Jacobian of a normalizing flow step based on graphical conditioners is equal to the product of its diagonal terms.*

Proof. **Proposition B.1** A Bayesian Network is a directed acyclic graph. Sedgewick and Wayne [2011] showed that every directed acyclic graph has a topological ordering, it is to say an ordering of the vertices such that the starting endpoint of every edge occurs earlier in the ordering than the ending endpoint of the edge. Let us suppose that an oracle gives us the permutation matrix P that orders the components of $\mathbf{g}$ in the topological defined by A. Let us introduce the following new transformation $\mathbf{g}_P(\mathbf{x}_P) = P\mathbf{g}(P^{-1}(P\mathbf{x}))$ on the permuted vector $\mathbf{x}_P = P\mathbf{x}$. The Jacobian of the transformation $\mathbf{g}_P$ (with respect to $\mathbf{x}_P$) is lower triangular with diagonal terms given by the derivative of the normalizers with respect to their input component. The determinant of such Jacobian is equal to the product of the diagonal terms. Finally, we have

$$\begin{aligned} |\det(J_{\mathbf{g}_P(\mathbf{x}_P)})| &= |\det(P)||\det(J_{\mathbf{g}(\mathbf{x})})|\frac{|\det(P)|}{|\det(P)|} \\ &= |\det(J_{\mathbf{g}(\mathbf{x})})|, \end{aligned}$$

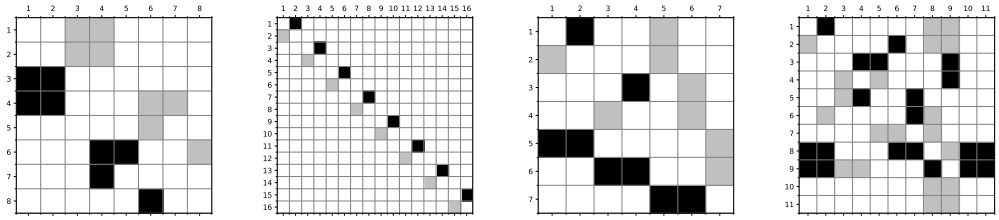
because of (1) the chain rule; (2) The determinant of the product is equal to the product of the determinants; (3) The determinant of a permutation matrix is equal to 1 or -1. The absolute value of the determinant of the Jacobian of $\mathbf{g}$ is equal to the absolute value of the determinant of $\mathbf{g}_P$, the latter given by the product of its diagonal terms that are the same as the diagonal terms of $\mathbf{g}$. Thus the absolute value of the determinant of the Jacobian of a normalizing flow step based on graphical conditioners is equal to the product of its diagonal terms. □

C Experiments on topology learning

C.1 Neural networks architecture

We use the same neural network architectures for all the experiments on the topology. The conditioner functions h_i are modeled by shared neural networks made of 3 layers of 100 neurons. When using UMNNs for the normalizer we use an embedding size equal to 30 and a 3 layers of 50 neurons MLP for the integrand network.

Figure 4: Ground truth adjacency matrices. Black squares denote direct connections and in light grey is their transposed.



(a) Arithmetic Circuit **(b)** 8 Pairs **(c)** Tree **(d)** Human Proteins

C.2 Dataset description

Arithmetic Circuit The arithmetic circuit reproduced the generative model described by Weilbach et al. [2020]. It is composed of heavy tailed and conditional normal distributions, the dependencies are non-linear. We found that some of the relationships are rarely found by during topology learning, we guess that this is due to the non-linearity of the relationships which can quickly saturates and thus almost appears as constant.

8 pairs This is an artificial dataset made by us which is a concatenation of 8 2D toy problems borrowed from Grathwohl et al. [2018] implementation. These 2D variables are multi-modal and/or discontinuous. We found that learning the independence between the pairs of variables is most of the time successful even when using affine normalizers.

Tree This problem is also made on top of 2D toy problems proposed by Grathwohl et al. [2018], in particular a sample $X = [X_1, \dots, X_7]^T$ is generated as follows:

1. The pairs variables (X_1, X_2) and (X_3, X_4) are respectively drawn from *Circles* and *8-Gaussians*;
2. $X_5 \sim \mathcal{N}(\max(X_1, X_2), 1)$;
3. $X_6 \sim \mathcal{N}(\min(X_3, X_4), 1)$;
4. $X_7 \sim 0.5\mathcal{N}(\sin(X_5 + X_6), 1) + 0.5\mathcal{N}(\cos(X_5 + X_6), 1)$.

Human Proteins A causal protein-signaling networks derived from single-cell data. Experts have annoted 20 ground truth edges between the 11 nodes. The dataset is made of 7466 entries which we kept $5,000$ for training and $1,466$ for testing.

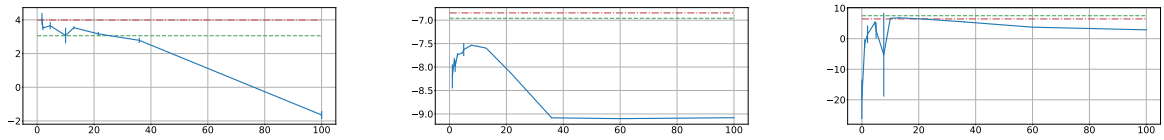
C.3 Additional experiments

Fig. 5 and Fig. 6 present the test log likelihood as a function of the ℓ_1-penalization on the four datasets for monotonic and affine normalizers respectively. It can be observed that graphical conditioners perform better than autoregressive ones for certain values of regularization and when given a prescribed topology in many cases. It is interesting to observe that autoregressive architectures perform better than a prescribed topology when an affine normalizer is used. We believe this is due to the non-universality of mono-step affine normalizers which leads to different modeling trade-offs. In opposition, learning the topology improves the results in comparison to autoregressive architectures.

D Tabular density estimation - Training parameters

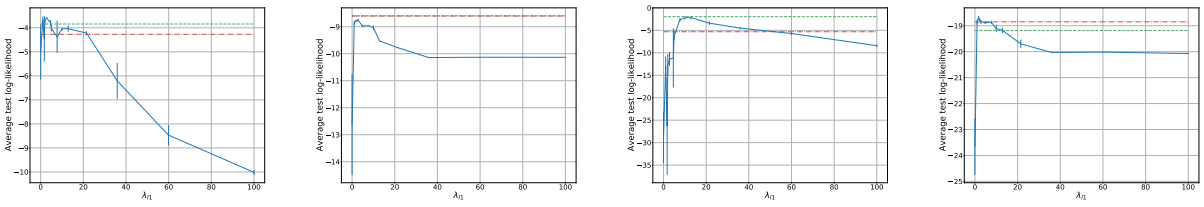
Table 6 provides the hyper-parameters used to train the normalizing flows for the tabular density estimation tasks. In our experiments we parameterize the functions $\mathbf{h}^i$ with a unique neural network that takes a one hot encoded version of i in addition to its expected input $\mathbf{x} \odot A_{i,:}$. The embedding net architecture corresponds to the network that computes an embedding of the conditioning variables for the coupling and DAG conditioners,

Figure 5: Test log-likelihood as a function of ℓ_1-penalization for monotonic normalizers. The red horizontal line is the average result when given a prescribed topology, the green horizontal line is the result with an autoregressive conditioner.



(a) Arithmetic Circuit **(b)** Tree **(c)** Human Proteins

Figure 6: Test log-likelihood as a function of ℓ_1-penalization for affine normalizers. The red horizontal line is the average result when given a prescribed topology, the green horizontal line is the result with an autoregressive conditioner.



(a) Arithmetic Circuit **(b)** Tree **(c)** Human Proteins **(d)** 8 Pairs

for the autoregressive conditioner it corresponds to the architecture of the masked autoregressive network. The output of this network is equal to 2 ($2 \times d$ for the autoregressive conditioner) when combined with an affine normalizer and to an hyper-parameter named *embedding size* when combined with a UMNN. The number of dual steps corresponds to the number of epochs between two updates of the DAGness constraint (performed as in Yu et al. [2019]).

Dataset	POWER	GAS	HEPMASS	MINIBOONE	BSDS300
Batch size	2500	10000	100	100	100
Integ. Net	3×100	3×200	3×200	3×40	3×150
Embedd. Net	3×60	3×80	3×210	3×430	3×630
Embed. Size	30	30	30	30	30
Learning Rate	0.001	0.001	0.001	0.001	0.001
Weight Decay	10^{-5}	10^{-3}	10^{-4}	10^{-2}	10^{-4}
λ_{ℓ_1}	0	0	0	0	0

Table 6: Training configurations for density estimation tasks.

In addition, in all our experiments (tabular and MNIST) the integrand networks used to model the monotonic transformations have their parameters shared and receive an additional input that one hot encodes the index of the transformed variable. The models are trained until no improvement of the average log-likelihood on the validation set is observed for 10 consecutive epochs.

E Density estimation of images

We now demonstrate how graphical conditioners can be used to fold in domain knowledge into NFs by performing density estimation on MNIST images. The design of the graphical conditioner is adapted to images by parameterizing the functions $\mathbf{h}^i$ with convolutional neural networks (CNNs) whose parameters are shared for all $i \in \{1, ..., d\}$ as illustrated in Fig. 7. Inputs to the network $\mathbf{h}^i$ are masked images specified by both the adjacency matrix A and the entire input image $\mathbf{x}$. Using a CNN together with the graphical conditioner allows for an inductive bias suitably designed for processing images. We consider single step normalizing flows whose conditioners are either coupling, autoregressive or graphical-CNN as described above, each combined with

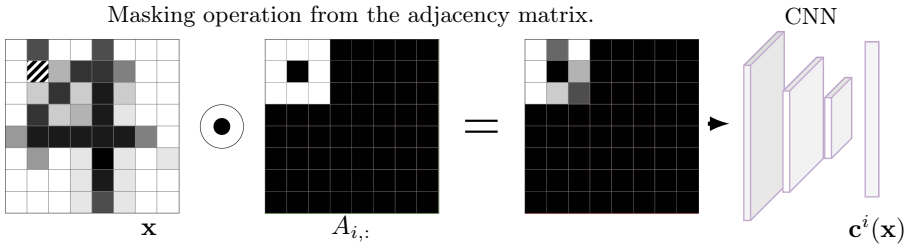


Figure 7: Illustration of how a graphical conditioner's output $\mathbf{c}^i(\mathbf{x})$ is computed for images. The sample $\mathbf{x}$, on the left, is an image of a 4. The stripes denote the pixel x_i. The parents of x_i in the learned DAG are shown as white pixels on the mask $A_{i,:}$, the other pixels are in black. The element-wise product between the image $\mathbf{x}$ and the mask $A_{i,:}$ is processed by a convolutional neural network that produces the embedding vector $\mathbf{c}^i(\mathbf{x})$ conditioning the pixel x_i.

	Model	Neg. LL.	Parameters	Edges	Depth
(a)	G-Affine (1)	$1.81_{\pm.01}$	1×10^6	5016	103
	G-Monotonic (1)	$1.17_{\pm.03}$	1×10^6	2928	125
(b)	A-Affine (1)	$2.12_{\pm.02}$	3×10^6	306936	783
	A-Monotonic (1)	$1.37_{\pm.04}$	3.1×10^6	306936	783
	C-Affine (1)	$2.39_{\pm.03}$	3×10^6	153664	1
	C-Monotonic (1)	$1.67_{\pm.08}$	3.1×10^6	153664	1
(c)	A-Affine (5)	$1.89_{\pm.01}$	6×10^6	5×306936	5×783
	A-Monotonic (5)	$1.13_{\pm.02}$	6.6×10^6	5×306936	5×783

Table 7: Results on MNIST. The negative log-likelihood is reported in bits per pixel on the test set over 3 runs on MNIST, error bars are equal to the standard deviation. The number of edges and the depth of the equivalent Bayesian network is reported. Results are divided into 3 categories: (a) The architectures introduced in this work. (b) Classical single-step architectures. (c) The best performing architectures based on multi-steps autoregressive flows.

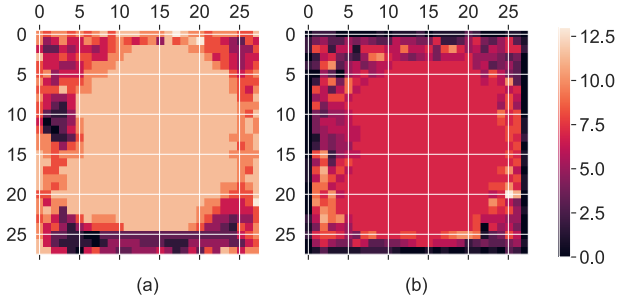


Figure 8: The in (a) and out (b) degrees of the nodes in the equivalent BN learned in the MNIST experiments.

either affine or monotonic normalizers. The graphical conditioners that we use include an additional inductive bias that enforces a sparsity constraint on A and which prevents a pixel's parents to be too distant from their descendants in the images. Formally, given a pixel located at (i, j), only the pixels $(i \pm l_1, j \pm l_2), l_1, l_2 \in \{1, ..., L\}$ are allowed to be its parents. In early experiments we also tried not constraining the parents and observed slower but successful training leading to a relevant structure.

Results reported in Table 7 show that graphical conditioners lead to the best performing affine NFs even if they are made of a single step. This performance gain can probably be attributed to the combination of both learning a masking scheme and processing the result with a convolutional network. These results also show that when the capacity of the normalizers is limited, finding a meaningful factorization is very effective to improve performance. The number of edges in the equivalent BN is about two orders of magnitude smaller than for coupling and autoregressive conditioners. This sparsity is beneficial for the inversion since the evaluation of the inverse of the flow requires a number of steps equal to the depth [Bezek, 2016] of the equivalent BN. Indeed, we find that while obtaining density models that are as expressive, the computation complexity to generate samples is approximately divided by $\frac{5\times784}{100} \approx 40$ in comparison to the autoregressive flows made of 5 steps and comprising many more parameters.

These experiments show that, in addition to being a favorable tool for introducing inductive bias into NFs, graphical conditioners open the possibility to build BNs for large datasets, unlocking the BN machinery for modern datasets and computing infrastructures.

F MNIST density estimation - Training parameters

For all experiments the batch size was 100, the learning rate 10^{-3}, the weight decay 10^{-5}. For the graphical conditioners the number of epochs between two coefficient updates was chosen to 10, the greater this number the better were the performance however the longer is the optimization. The CNN is made of 2 layers of 16 convolutions with 3×3 kernels followed by an MLP with two hidden layers of size 2304 and 128. The neural network used for the Coupling and the autoregressive conditioner are neural networks with 3×1024 hidden layers. For all experiments with a monotonic normalizer the size of the embedding was chosen to 30 and the integral net was made of 3 hidden layers of size 50. The models are trained until no improvements of the average log-likelihood on the validation set is observed for 10 consecutive epochs.

8.3 EPILOGUE

8.3.1 *Inductive bias in normalizing flows.*

The Lipschitz constant can be a good summary of the continuous machine learning models' complexity [Virmaux and Scaman, 2018; Weng et al., 2018; Bartlett et al., 2017]. Hence we may prevent overfitting by choosing the model with the smallest Lipschitz constant between many fitting equally well the train set [von Luxburg and Bousquet, 2004]. We usually control the Lipschitz constant of the model only implicitly, e.g. by penalising the norm of the weights [Krogh and Hertz, 1991] or adding stochasticity in the learning algorithm [Smith et al., 2021; Bottou, 2012]. However, such strategies turn inefficient when the loss function favours models with large Lipschitz constants.

As shown in Figure 8.1, the Lipschitz constant of normalizing flows and overfitting are correlated. Indeed, we optimise the likelihood of the NF, which is directly related to the Lipschitz constant of the modelled density. The density is a function of its first-order derivative with respect to the input – the Lipshitz constant bounds the likelihood and the highest density peak. The Lipschitz constant can quickly explode for small training sets. Indeed, expressive models may discover spurious relationships between variables. These spurious relationships are complicated and lead to large Lipschitz constants. Thus by enforcing independencies, prescribed or discovered in the data, graphical normalizing flows may avoid these traps and overfit less. Meanwhile, the relationships discover by graphical normalizing flows are simpler and generalise better.

There may also exist a genuine deterministic relationship between variables which implies that the data lie on a manifold. For instance, let us consider predicting the distribution of birds on the world atlas. We must choose between a distribution over 3D real numbers or 2D longitudinal and latitudinal coordinates that encode the birds' position. On the one hand, the 3D parameterisation contains a non-spurious deterministic relationship which is the equation of the surface of the atlas. On the other hand, the latter formalisation does not operate with real numbers, nor in an euclidean space. In this context, graphical normalizing flows cannot help much. However, there is a rich litterature [Köhler et al., 2021; Mathieu and Nickel, 2020; Gemici et al., 2016; Kalatzis et al., 2021; Rezende et al., 2020] about normalizing flows on manifolds, which can help prevent numerical instabilities and also provide a powerful inductive bias to ease the learning task.

Finally, another way to improve the inductive bias of normalizing flows is to design invertible layers that mimic effective non-invertible architectures. This translation is not always straightforward, but it can be worth the time spent. Glow [Kingma and Dhariwal, 2018] is a perfect example; they borrow ideas from convolutions, pooling, and hierarchical VAEs to achieve state-of-the-art image synthesis at the time.

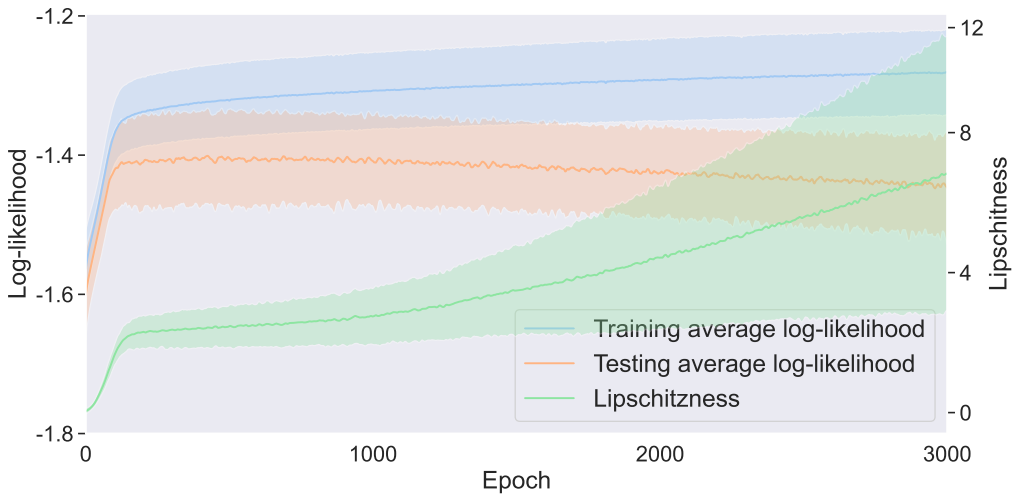


Figure 8.1: Evolution of the Lipschitz constant of a normalizing flow's log-likelihood along training with the corresponding learning and testing log-likelihood. The training and test sets are composed of 100 iid samples from a mixture of two Gaussian distributions. We reproduce the training 20 times and plot the corresponding mean signals. The shaded area represents one standard deviation. As training continues overfitting, the gap between training and testing performance increases with the Lipschitz constant of the flow's log-likelihood.

8.3.2 *Scientific impact*

The graphical conditioner unifies autoregressive and coupling layers under a unique architecture with a non-singular binary matrix that controls the topology. This unification can ease the search for the best architecture and simplify the code behind different types of normalizing flows. As discussed, the main feature of graphical normalizing flows is to provide an explicit treatment of independencies. Not only does this offers more interpretability, but it is also a new inductive bias that can help avoid overfitting. One-step graphical normalizing flows achieve performance on par with multiple steps flows but have a reduced complexity for generating samples. Graphical normalizing flows also emphasise that powerful models rely on strong assumptions and are not often purely data-driven.

According to Google Scholar, our paper has received 17 citations between its publication at AISTATS in April 2021 and August 2022. Among these, we notice the work from Mouton and Kroon [2022a] that combine residual networks with graphical normalizing flows to improve the stability and efficiency of computing the inverse transformation. In Mouton and Kroon [2022b], the same authors exploit the additional structure of graphical normalizing to improve VAEs. Another line of work proposed by Balgi et al. [2022b] is using graphical normalizing flows to discover causal structures. They show that graphical NFs may find relevant relationships and are well suited for counterfactual inference.

8.3.3 *Conclusion and opportunities*

We have introduced a new type of normalizing flow that combines Bayesian networks with neural networks. This architecture emphasises the importance of making assumptions when learning probabilistic models and eases the handling of independencies within the Normalizing flow framework. As a nice byproduct, graphical normalizing flows unify autoregressive and coupling transformers under a unique layer. In addition to being an expressive graphical and deep probabilistic model, subsequent work has also demonstrated that graphical normalizing flows can be equipped with a causal flag in specific contexts.

Graphical normalizing flows also have substantial limitations. One of them is that they lose the Bayesian network interpretation as soon as the number of NF steps exceeds one. This restriction is inconsequential as one-step graphical flows are universal density approximators; there is no substantial motivation for stacking multiple steps. When independence assumptions live, we think that graphical normalizing flows should be tested and might better represent the phenomenon of interest than other types of flows.

A critical limitation of normalizing flows exists when there is no prescribed independence, and we aim to discover them from data. Although graphical NFs may learn relevant Bayesian network topologies and generalise better than alternative flow architectures, the corresponding optimisation problem is hard to solve. An exciting line of future work would be to work out the numerical stability of the Lagrangian optimisation, as proposed by Ng et al. [2022]. Another issue of our method happens during optimisation when the

topology does not correspond to a directed acyclic graph. Hence, the corresponding normalizing flow is not invertible, which impedes the inverse theorem from providing an exact likelihood. In the future, it would be worth exploring whether sampling sub-graphs that are acyclic improves the overall learning algorithm.

A complete Bayesian treatment of normalizing flows is out-of-reach. They rely on neural networks, and Bayesian neural networks [MacKay, 1995] are still an active research area. It is unclear whether one day we will be able to express effectively prior and posterior distributions over neural networks. Less ambitious but potentially worthy, expressing distributions over (conditional) independence might be an alternative that could provide valuable insights into the learnt models. In particular, we believe that graphical normalizing flows combined with an efficient sampling strategy that builds a valid Bayesian network structure from the posterior distribution over (conditional) independence might become an effective modelling strategy.

To conclude, we have once again demonstrated the relevance of drawing connections between distinct classes of models. Here, we have argued that Bayesian networks and normalizing flows share in common the factorisation of a joint density via the Bayes' rule. This new perspective enables exploiting topology discovery algorithms within normalizing to improve the expressivity of their inductive bias. We have demonstrated the relevance of acknowledging independence assumptions when possible and discovering them to avoid overfitting and learn better models.

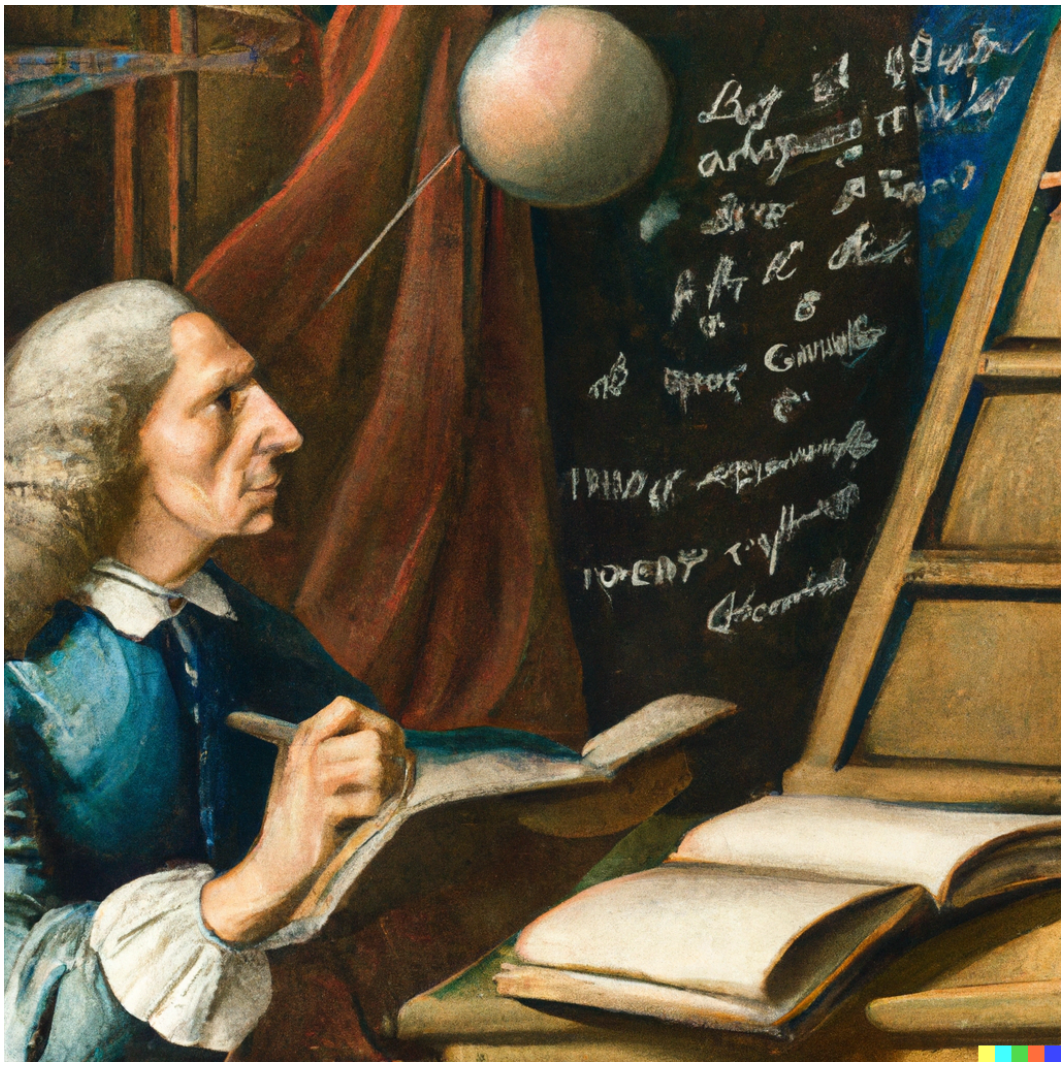


Figure 8.2: *An old painting of Isaac Newton writing the theory of gravity in the language of probability and questioning the role of models in the world.* As seen by DALL · E 2.

If I have seen further, it is by standing on shoulders of giants.

Isaac Newton

9

HYBRID PROBABILISTIC MODELS

Outline

In this chapter we explore deep probabilistic models informed by expert models, hybrid models. We formalise hybrid learning within the probabilistic modelling framework and demonstrate that hybrid models exhibit greater generalisation capabilities than classical machine learning models. Hybrid models reduce the misspecification of expert models with a machine learning (ML) component learned from data. We leverage the insight that the expert model is usually valid even outside the training domain to introduce a hybrid data augmentation strategy termed *expert augmentation*. In contrast to many ML algorithms, the performance guarantees of hybrid models trained with expert augmentation are not limited to the training distribution. We validate the practical benefits of augmented hybrid models on a set of controlled experiments and reflect on the broader impact that hybrid learning may have shortly.

9.1 PROLOGUE

In this thesis, we have presented various algorithms to help practitioners build models from data. The previous chapter has shown that inductive bias is necessary to learn models from medium or high-dimensional data. Following the growing deployments of ML solutions into the real world [Wehenkel, 1998] and the related demand for performance guarantees throughout their lifetime [Lwakatare et al., 2020], the ML research community has gained interest in out-of-distribution robustness [Sehwag et al., 2019; Hendrycks et al., 2021]. Machine learning algorithms are not anymore judged only on their ability to produce faithful models inside the training distribution but also on the behaviour of these models in out-of-distributions scenarios.

Although the concept of out-of-distribution robustness seems appealing, it does not clearly say what we seek. To use this concept rigorously, we argue that we shall first answer the following related questions: • *What is in-distribution?* • *What is out-of-distribution?* • *What is the measure of robustness?* Refusing to answer one of these questions will inevitably lead to an ill-posed ML problem. It is essential to notice the distinction between out-of-distribution and *not* in-distribution. In practice, we might need to restrain ourselves to out-of-distribution settings that correspond to a well-defined subset of what

is not in distribution. Finally, we shall be able to express robustness with a quantitative metric.

Hybrid models are the ones that explicitly combine expert models with a machine learning component. They constitute an excellent alternative to purely data-driven solutions and expert models that rely on assumptions often violated in practice. We expect that hybrid models require less data to achieve performance on par with ML models. They may also exhibit better interpretability. In addition, we argue that hybrid models are particularly well suited to work within the context of out-of-distribution robustness. Indeed, the expert model often depends on a low-dimensional set of parameters for which we can provide an informed definition of in-distribution and out-of-distribution.

In line with this argument, we propose an augmentation strategy based on the expert model that enforces a well-defined notion of robustness for a specific out-of-distribution scenario. We show that existing hybrid learning algorithms may learn effective representation but require an additional augmentation step to exhibit robustness. This chapter demonstrates that informed machine learning may achieve results that are out of reach for uninformed solutions. It also shows, once again, that combining various models may be beneficial.

9.2 ROBUST HYBRID LEARNING WITH EXPERT AUGMENTATION

9.2.1 *Author contributions*

I co-authored this paper during an internship at Apple within the Health AI team led by Guillermo Sapiro. Hsu Hiang initially explored the idea of combining expert models with machine learning under the advisory of Jens Behrmann and Jörn-Henrik Jacobsen. I took over Hsu's work and, together with Jens and Jörn, we explored a probabilistic formulation of hybrid modelling and out-of-distribution robustness. We designed the experiments together, and I wrote the code corresponding to the two hybrid modelling frameworks (APHYNITY and the hybrid-VAE) used in this project. Gilles helped write the paper and design additional experiments to check the robustness of the expert augmentation in different settings. Guillermo gave feedback on the manuscript. Finally, Gilles, Jens, and Jörn gave feedback on the manuscript and helped me improve its writing.

9.2.2 *Reading tips*

The methods explored in the paper are specific to hybrid learning and have not been described in the background. Thus we encourage the reader to carefuly review the complete paper.

Robust Hybrid Learning With Expert Augmentation

Antoine Wehenkel [1 2] **Jens Behrmann** [3] **Hsiang Hsu** [4 2] **Guillermo Sapiro** [3] **Gilles Louppe** [1]
Jörn-Henrik Jacobsen [3]

Abstract

Hybrid modelling reduces the misspecification of expert models by combining them with machine learning (ML) components learned from data. Like for many ML algorithms, hybrid model performance guarantees are limited to the training distribution. Leveraging the insight that the expert model is usually valid even outside the training domain, we overcome this limitation by introducing a hybrid data augmentation strategy termed *expert augmentation*. Based on a probabilistic formalization of hybrid modelling, we show why expert augmentation improves generalization. Finally, we validate the practical benefits of augmented hybrid models on a set of controlled experiments, modelling dynamical systems described by ordinary and partial differential equations.

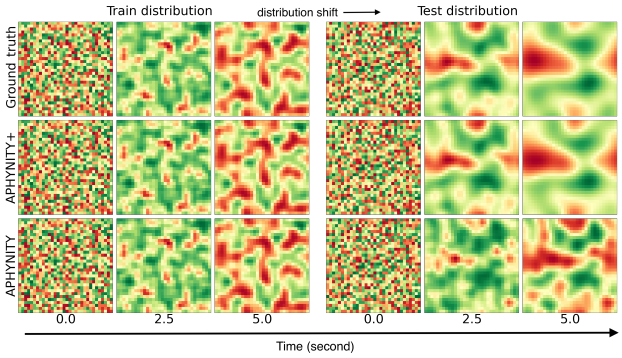


Figure 1. APHYNITY, an existing hybrid modelling strategy, is unable to predict accurately the dynamic of a 2D diffusion reaction for a shifted test distribution although it predicts well configuration that follows the training distribution. On the opposite, APHYNITY+, the same model fine-tuned with our data augmentation, generalizes to shifted distributions as expected from the validity of the underlying physics.

1. Introduction

Generalization to unseen data is a key property of a useful model. When training and test data are independently and identically distributed (IID), one way to check generalization is by evaluating the model on a held out subset of the training data or with k-fold cross validation. Unfortunately, this setting is often unrealistic because the training scenario is rarely fully representative of the test scenario. This has motivated lot of recent research efforts to focus on the robustness of ML models (Gulrajani & Lopez-Paz, 2020; Geirhos et al., 2020; Koh et al., 2021). Common strategies can be broadly grouped in two categories: The first class of methods aims at aligning specific properties of the model (e.g., invariance, equivariance, monotonicity, etc.) with expertise on the problem of interest (Cubuk et al., 2019; Mahmood et al., 2021; Keriven & Peyré, 2019; Silver et al., 2017). The second category is data focused (Sagawa et al., 2019; Arjovsky et al., 2019; Krueger et al., 2021; Creager et al., 2021) and leverages variations present in the training data, e.g. by minimizing worst case sub-group performance, to achieve robustness.

The data oriented methods, which include GroupDRO (Sagawa et al., 2019) and Invariant Risk Minimization (Arjovsky et al., 2019, IRM), can be very appealing because they only require implicit specification of invariances via domains or environments. However, these methods' performance is limited to variations present in the training data and the inductive bias of the ML algorithm. This may be insufficient when the modelling problem is too complex or the variations of interest are not present in the training data. On the other hand, methods based on domain-specific expertise do not suffer from such limitations. Embedding expertise into a model can be done via architectural inductive biases (LeCun et al., 1995; Xu et al., 2018), data augmentation (Cubuk et al., 2019), or a learning objective (Cranmer et al., 2020) that enforces established symmetries of the problem. As an example, simple data augmentation techniques combined with convolutions lead to excellent performance on natural image problems (Cubuk et al., 2019). Another natural approach to embed expertise in ML models, and the one studied in this paper, is called hybrid learning (HyL). HyL combines an expert model (e.g., physics-motivated equations) with a learned component that improves the expert model so that the combination better fits real-world

*Equal contribution [1]University of Liege [2]Work done as an intern at Apple [3]Apple [4]Harvard University. Correspondence to: Antoine Wehenkel <antoine.wehenkel@uliege.be>.

data. A particularity of HyL is the central role played by the expertise, which is supposed to provide a simple and well-grounded parametric description of the process considered. HyL usually considers the expert model as an analytical function, or as a set of equations, that relates the expert parameters to the quantity of interest. The expert model is often motivated by the underlying physics of the system considered. Hence, we will use the terms *expert* model and *physical* model interchangeably.

In recent work (Yin et al., 2021; Takeishi & Kalousis, 2021; Qian et al., 2021; Mehta et al., 2020; Lei & Mirams, 2021; Reichstein et al., 2019), HyL demonstrated success in complementing partial physical models and improving the inference of the corresponding parameters. However, contrarily to the common belief that HyL achieves better generalization than black box ML models, we argue that hybrid models do not meet their promise regarding robustness. Although HyL achieves strong performance on IID test distributions by exploiting the inductive bias of the expert models, we show that their performance collapses when the test domain is not included in the training domain. This is unsatisfactory as the expert model is typically well-defined for a range of parameters that can correspond to realistic data far outside of the training distribution. A test distribution not covered by the training data, but for which an expert model exists, happens often in the real world. As an example, Qian et al. (2021) apply HyL to a pharmacological model describing the effect of a COVID-19 treatment for which only a limited quantity of real-world data is available. In this context, although the underlying biochemical dynamic of treatments is well modelled, data is often scarce and biased. Therefore, the hybrid model does not necessarily generalize to configurations that are well modelled by the pharmacological model but unseen during training.

We introduce *expert augmentations* for training augmented hybrid models (AHMs), a procedure that extends the range of validity of hybrid models and improves generalization as pictured by Figure 1. Our contribution is to first formalise the HyL problem as: 1) Learning a probabilistic model partially defined by the expert model; 2) Performing inference over this probabilistic hybrid model. In this context, we show that HyL is vulnerable to distribution shifts for which the expert model is well defined (see Figure 1, bottom row). Motivated by our analysis, we propose to fine-tune the hybrid model on an expert-augmented dataset that includes distribution shifts (see results of augmentation in Figure 1, middle row). These expert augmentations only rely on the hybrid model itself, leveraging that the expert model is also well-defined outside of the training distribution. Our experiments on various controlled HyL problems demonstrate that AHMs achieve multiple orders of magnitude superior generalization in realistic situations and can be applied to any state-of-the-art HyL algorithm.

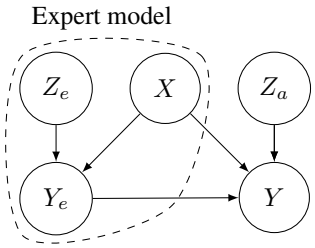


Figure 2. A hybrid probabilistic model which describes the relationship between the input X and the output Y for a configuration of the system as defined by the latent variables Z_e and Z_a. The prescribed expert model defines the conditional density $p(y_e|z_e, x)$, where Y_e is an approximation of Y. Hybrid learning aims at learning the conditional distribution $p(y|z_a, y_e, x)$.

2. Hybrid learning

In order to show that our proposed expert augmentations lead to robust models, we first formalize hybrid learning with the probabilistic model depicted in Figure 2. In this Bayesian network, capital letters denote random variables (e.g., Y) and, in the following, we will use calligraphic letters for the domain of the corresponding realization (e.g., $y \in \mathcal{Y}$). In our formalism, the expert model is a conditional density $p(y_e|x, z_e)$ that describes the distribution of the $expert$ response Y_e to an input x together with a parametric description of the system z_e, denoting expert or physical parameters. We augment the expert model with the *interaction model* which is a conditional distribution $p(y|x, y_e, z_a)$ that describes the distribution of the observation Y given the input x, the expert model response y_e, and a parametric description of the interaction model z_a.

Our final goal is to create a robust predictive model $p(y|x, (x_o, y_o))$ of the random variable Y, given the input x together with independent observations (x_o, y_o) of the same system, where the subscript o denotes an observed quantity. As a concrete example, we consider predicting the evolution of a damped pendulum (described in Section 4.1) given its initial angle and speed ($x = \left[\theta, \dot{\theta}\right]$) and a sequence of observations of the same pendulum. The expert model we assume is able to describe a frictionless pendulum whose dynamic is only characterized by one parameter $z_e := \omega_0$, denoting its fundamental frequency. A perfect description of the system should model the friction with a second parameter $z_a := \alpha$, the damping factor. In this problem, (x_o, y_o) and (x, y) are IID realization of the same pendulum which corresponds, in general terms, to samples from $p(x, y|z_a, z_e)$ for some fixed but unknown values of z_a and z_e. The expert variables z_e (e.g., ω_0) together with z_a (e.g., α) should accurately describe the system that produces Y (e.g., the evolution of the pendulum's angle and speed along time) from X (e.g., the initial pendulum's state). In our setting we assume that we are given a pair (x_o, y_o) (e.g., past observations) from

which we can accurately infer the state of the system (z_a, z_e) as described by the interaction and expert models, and then predict the distribution of Y for a given input x (e.g., forecasting future observations) to the same system. Because the interaction between z_e and y is essentially defined by the expert model, it should be possible, and preferable, to learn an accurate predictive model of Y whose accuracy is independent from the training distribution of the expert variables z_e. Provided all probability distributions in Figure 2 are known, the Bayes optimal hybrid predictor p_B can be written as

$$p_B(y|x,(x_o,y_o)) = \mathbb{E}_{p(z_a,z_e|(x_o,y_o))}\left[p(y|x,z_a,z_e)\right]. \quad (1)$$

We observe that the Bayes optimal predictor explicitly depends on the posterior $p(z_a, z_e|(x_o, y_o))$ which is itself a function of the marginal distribution over z_e. This may preclude the existence of a good predictor that is invariant to shift of $p(z_e)$. However, in the following we will consider that the pair (x_o, y_o) contains enough information about the parameters z_a, z_e. As a consequence, the posterior distribution shrinks around the correct parameters value and the effect of the prior becomes negligible.

2.1. Hybrid generative modelling

We consider expert models that are deterministic; that is, for which $p_\theta(y_e|x, z_e)$ is a Dirac distribution. The expert model describes the system as a function $f_e : \mathcal{X} \times \mathcal{Z}_e \to \mathcal{Y}_e$ that computes the response y_e to an input x, parameterized by expert variables z_e. The goal of hybrid modelling is to augment the expert model with a learned component from data as depicted in Figure 2. Formally, given a dataset $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^N$ of N IID samples, we aim to learn the interaction model $p_\theta(y|x, y_e, z_a)$ that fits the data well but is close to the expert model. For example, we could define closeness via a small L2-distance between expert and hybrid outputs or via a small Kullback-Leibler (KL) divergence between the marginal distributions of Y and Y_e.

Learning a model that is close to the expert model and fits the training data well is a hard problem. However, the APHYNITY algorithm (Yin et al., 2021) and the Hybrid-VAE (Takeishi & Kalousis, 2021, HVAE) are two recent approaches that offer promising solutions to this problem. We now briefly describe these two methods and how they can be used to approximate the Bayes optimal predictor of (1). Our augmentation strategy is compatible (and effective) with both approaches.

APHYNITY. Yin et al. (2021) formulate hybrid learning in a context where the expert model is an ordinary differential equation (ODE). They consider an additive hybrid model that should perfectly fit the data, which is equivalent to assuming the conditional distribution $p_\theta(y|x, y_e, z_a)$ is a Dirac distribution. Formally, they solve the optimization problem

$$\min_{z_e, F_a} ||F_a|| \quad \text{s.t.} \quad \forall (x,y) \in \mathcal{D}, \forall t, \frac{dy_t}{dt} = (F_e + F_a)(y_t) \quad \text{with} \quad y_0 := x, \quad (2)$$

where $||\cdot||$ is a norm operator on the function space, $F_a : \mathcal{Y}_t \times \mathcal{Z}_a \to \mathcal{Y}_t$ is a learned function, $F_e : \mathcal{Y}_t \times \mathcal{Z}_e \to \mathcal{Y}_t$ defines the expert model and $\mathcal{D}$ is a dataset of initial states $x := y_0$ and sequences $y \in \mathcal{Y} := (\mathcal{Y}_t)^k$, where k is the number of observed timesteps. APHYNITY solves this problem with Lagrangian optimization and Neural ODEs (Chen et al., 2018) to compute derivatives. In the context of ODEs, the random variable X is the initial state of the system at t_0 and Y is the observed sequence of k states between t_0 and t_1.

This formulation only considers learning a missing dynamic for one realization of the system described by Figure 2, for a single z_a and z_e. However, we are interested in learning a hybrid model that works for the full set of systems described by Figure 2. As suggested in Yin et al. (2021), we use an encoder network $g_\psi(\cdot,\cdot) : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}_a \times \mathcal{Z}_e$ that corresponds to a Dirac distribution located at g_ψ as the approximate posterior $q_\psi(z_a, z_e|x, y)$. The interaction model is a product of Dirac distributions whose locations correspond to the solution of the ODE

$$\frac{dy_t}{dt} = F_e(y_t, z_e) + F_a(y_t, z_a; \theta), \quad y_0 := x. \quad (3)$$

Hence the corresponding approximate Bayes predictor replaces the parameters (z_a, z_e) in (3) with the prediction of g_ψ and predicts a product of Dirac distributions.

Hybrid-VAE (HVAE). In contrast to APHYNITY, the model proposed by Takeishi & Kalousis (2021) is not limited to additive interactions between the expert model and the ML model, nor to ODEs. Instead, their goal is to learn the generative model described by Figure 2. They achieve this with a variational auto-encoder (VAE) where the decoder specifically follows Figure 2. Similarly to the amortized APHYNITY model, the encoder $g_\psi(x, y)$ predicts a posterior distribution over z_a and z_e, and the model is trained with the classical Evidence Lower Bound on the likelihood (ELBO). Takeishi & Kalousis (2021) observe that relying only on an architectural inductive bias and maximum likelihood training is not enough to ground the generative model to the expert equations. They propose to add three regularizers $R_{PPC}, R_{DA,1}$, and $R_{DA,2}$ that encourage the generative model to rely on the expert model. The final objective is

$$\max_{\theta,\psi} \mathbb{E}_{\mathcal{D}}\left[\text{ELBO}((x,y);\psi,\theta)\right] + \alpha R_{PPC} + \beta R_{DA,1} + \gamma R_{DA,2}. \quad (4)$$

The first regularizer, R_{PPC}, encourages the marginal distribution of samples generated by the complete model to be

close to the marginal distribution that would be only generated by the physical model. The two other regularizers specifically require the encoder network for z_e to be made of two sub-networks. The first network filters the observations to keep only what can be generated by the expert model alone, and the second should map the filtered observations to the posterior distribution over z_e. $R_{DA,1}$ penalizes the objective if the observations generated by the expert model are not close to the filtered observations. Finally, $R_{DA,2}$ relies on data augmentation with the expert model to enforce that the second sub-network correctly identifies the expert variables z_e when the observations are correctly filtered. We refer the reader to Takeishi & Kalousis (2021) for more details on HVAE. For HVAE, the approximate predictor takes the form described by (1) where $p(z_a, z_e|(x_o, y_o))$ is approximated by the encoder $q_\psi(z_a, z_e|x, y)$ and $p(y|x, z_e, z_a)$ by the learned hybrid generative model.

3. Robust hybrid learning

We now formalize our definition of out of distribution (OOD) and robustness. In general, a test scenario is OOD if the joint test distribution $\tilde{p}(x, y)$ is different from the training distribution $p(x, y)$, that is $d(\tilde{p}, p) > 0$ for any properly defined divergence or distance d. In the following, we reduce our discussion to a sub-class of distribution shifts for which the marginal train and test distributions over z_e may be different, $d(p(z_e), \tilde{p}(z_e)) > 0$, but the marginals of z_a and x are constant. As a consequence, the joint distribution of (x, y) pairs is also shifted. Formally, the training and test distributions are respectively defined as

$$p(x, y) := \mathbb{E}_{p(z_e)p(z_a)p(y_e|z_e,x)} \left[p(x)p(y|z_a, x, y_e)\right],$$
$$\tilde{p}(x, y) := \mathbb{E}_{\tilde{p}(z_e)p(z_a)p(y_e|z_e,x)} \left[p(x)p(y|z_a, x, y_e)\right].$$

In this context, we demonstrate, theoretically and empirically, that classical hybrid models fail. To address this failure, we introduce *augmented hybrid models* and show that, under some assumptions, they achieve optimal performance on both the train and test distributions.

Our goal is to learn a predictive model

$$p_{\theta,\psi}(y|x, (x_o, y_o)) = \mathbb{E}_{\substack{q_\psi(z_a,z_e|x_o,y_o) \\ p(y_e|z_e,x)}} \left[p_\theta(y|y_e, x, z_a)\right]$$

that is *exact* on both the train and test domains when they follow the aforementioned training and testing distribution shifts. We say that a learned predictive model $\hat{p}(a|b)$ is $\mathcal{E}$-*exact*, or *exact* on the sample space $\mathcal{E}$, if $\hat{p}(a|b) = p(a|b) \quad \forall (a, b) \in \mathcal{E}$. Here we qualify a predictive model as *robust* to a test scenario if its *exactness* on the training domain is sufficient to ensure exactness on the test domain.

We now define an augmented distribution $\overset{+}{p}(z_e)$ over the expert variables whose support $\overset{+}{\mathcal{Z}}_e$ includes the joint support $\mathcal{Z}_e \cup \tilde{\mathcal{Z}}_e$ between the train and test distribution of the physical parameters. As depicted in Figure 3, we denote the corresponding support over the observation space $\mathcal{X} \times \mathcal{Y}$ as $\overset{+}{\Omega}, \Omega$, and $\tilde{\Omega}$, respectively. In this context, and with **A1**, we may demonstrate that even under perfect learning, classical hybrid learning algorithms do not produce an Ω-*exact* predictor while our augmentation strategy does.

Assumption 1 (A1): *Hybrid modelling learns an interaction model $p_\theta(y|y_e, x, z_a)$ that is $\overset{+}{\Omega}$-exact.*

Although strong, **A1** is consistent with the recent literature on hybrid modelling, which assumes that $p(y_e|x, z_e)$ is an accurate description of the system, thereby $p_\theta(y|y_e, x, z_a)$ should not be overly complex. As an example, we consider an additive interaction model in our experiments for which extrapolation to unseen y_e holds if this assumption is correct. That said, we still notice that the exactness of the interaction model p_θ on $\overset{+}{\Omega}$ is insufficient to prove that the predictive model $p_{\theta,\psi}$ is $\overset{+}{\Omega}$-*exact*. Indeed, the encoder q_ψ is only trained on the training data and cannot rely on a strong inductive bias in contrast to p_θ. Thus, even if the encoder is exact on the training distribution, the corresponding predictive model does not achieve exactness outside Ω.

3.1. Expert augmentation

We propose a data augmentation strategy to improve the robustness of hybrid models to unseen test scenarios. Once trained, the hybrid model is composed of an encoder q_ψ and an interaction model p_θ that are respectively Ω- and $\overset{+}{\Omega}$-*exact*. We may create a new training distribution with a support over $\overset{+}{\Omega}$ by sampling physical parameters z_e from a distribution that covers $\overset{+}{\mathcal{Z}}_e$. We can then train the encoder q_ψ on $\overset{+}{\Omega}$, under perfect training the corresponding predictive model $p_{\theta,\psi}(y|x, (x_o, y_o))$ is $\overset{+}{\Omega}$-*exact*, hence exact on both train and test domains.

Our learning strategy is grounded in existing hybrid modelling algorithms. Here, we focus on APHYNITY and HVAE, but our approach is applicable to other HyL algorithms. We first train an encoder q_ψ and a decoder p_θ with a HyL algorithm. Together with experts we then decide on a realistic distribution $\overset{+}{p}(z_e)$ and create a new dataset $\overset{+}{\mathcal{D}}$ by sampling from the hybrid generative model defined by Figure 2 and the interaction model p_θ. A notable difference between the augmented training set $\overset{+}{\mathcal{D}}$ and the original training set $\mathcal{D}$ is that the former contains ground truth values for the expert's variables z_e. As we assume that the interaction model is $\overset{+}{\Omega}$-*exact*, we freeze it and only fine-tune the encoder q_ψ on $\overset{+}{\mathcal{D}}$. We use a combination of the loss function ℓ of the original HyL algorithm (e.g., (4) for HVAE, and the Lagrangian of (2) for APHYNITY) and a supervision on

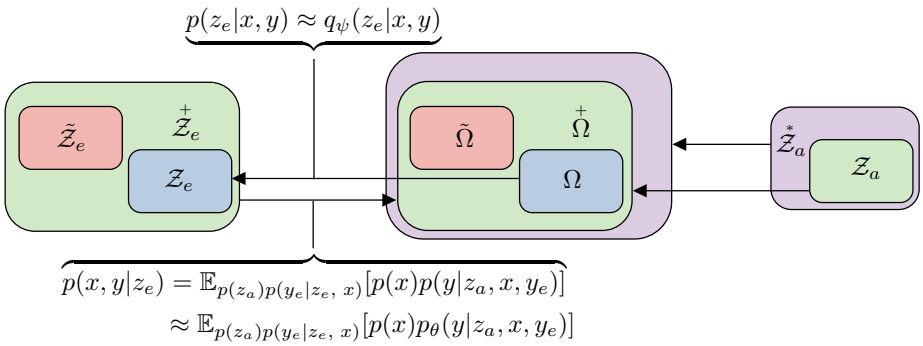


Figure 3. Visualization of the distribution shifts considered in this work. The train support Ω of (x, y) results from $(z_a, z_e) \in \mathcal{Z}_e \times \mathcal{Z}_a$. The test supports (in red) are denoted with a tilde symbols as $\tilde{\mathcal{Z}}_e$ for z_e and $\tilde{\Omega}$ for (x, y). The augmented support $\overset{+}{\Omega}$ (in green) includes both train and test scenarios and corresponds to $(z_a, z_e) \in \overset{+}{\mathcal{Z}}_e \times \mathcal{Z}_a$. The outer violet domain that includes $\overset{+}{\Omega}$ depicts one of our experiment in which the domain of z_a is also shifted. Hybrid modelling algorithms alone may learn a mapping $p_\theta : \overset{+}{\mathcal{Z}}_e \to \overset{+}{\Omega}$ but augmentation is necessary to learn the inverse mapping $q_\psi : \overset{+}{\Omega} \to \overset{+}{\mathcal{Z}}_e$.

the latent variable objective to learn a decoder that solves

$$\psi = \arg\min_\psi \mathbb{E}_{\overset{+}{\mathcal{D}}} \left[\ell(x, y; \theta, \psi) - \log q_\psi(z_e|x, y)\right].$$

In our experiments we chose a Gaussian model for the posterior, which is equivalent to a mean square error (MSE) loss on the physical parameters. We provide a detailed description of the expert augmentation scheme in Appendix A.

As a side note, we would like to emphasize the difference between the data augmentation proposed in this paper and the one from Takeishi & Kalousis (2021). While HVAE also requires to sample new physical parameters z_e, it is only to ensure that a sub-part of the encoder is able to infer correctly z_e given y_e. This augmentation does not contribute to robustness distribution shifts on y in contrast to ours.

4. Experiments

4.1. Problem description

We assess the benefits of expert augmentation on three controlled problems described and simulated by the ODE

$$\frac{dy_t}{dt} = F_e(y_t; z_e) + F_a(y_t; z_a), \tag{5}$$

where $F_e : \mathcal{Y}_t \times \mathcal{Z}_e \to \mathcal{Y}_t$ is the expert model and $F_a : \mathcal{Y}_t \times \mathcal{Z}_a \to \mathcal{Y}_t$ complements it. In our notation X is the initial state y_0 and the response Y is the sequence of states $y_{1:t_1} := [y_{i\Delta t}]_{i=1}^{t_1/\Delta t}$. For all experiments we train the models to maximize $p_{\theta,\psi}(y = y_{1:t_1}|x = y_0)$ on the training data. We validate and test the models on the predictive distribution $p(y = y_{1:t_2}|x = y_0, x_o = y_0, y_o = y_{1:t_1})$, where $t_2 > t_1$ assesses the generalization over time. A brief description of the different problems is provided below.

The damped pendulum is often used as an example in the hybrid modelling literature (Yin et al., 2021; Takeishi & Kalousis, 2021). The system's state at time t is $y_t = \begin{bmatrix}\theta_t & \dot{\theta}_t\end{bmatrix}^T$, where θ_t is the angle of the pendulum at time t and $\dot{\theta}_t$ its angular speed. The evolution of the state over time is described by (5), where $z_e := \omega$, $z_a = \alpha$ and

$$F_e := \begin{bmatrix}\dot{\theta} & -\omega_0^2 \sin\theta\end{bmatrix}^T \quad \text{and} \quad F_a := \begin{bmatrix}0 & -\alpha\dot{\theta}\end{bmatrix}^T. \tag{6}$$

The corresponding systems are defined by the damping factor α and ω_0, the fundamental frequency of the pendulum.

The RLC series circuits are electrical circuits made of 3 electrical components that may model a large range of transfer functions. These models are often used in biology (e.g., the Hodgkin-Huxley class of models (Hodgkin & Huxley, 1952), in photoplethysmography (Crabtree & Smith, 2003)) and in electrical engineering to model the dynamics of various systems. The system's state at time t is $y_t = \begin{bmatrix}U_t & I_t\end{bmatrix}^T$, where U_t is the voltage around the capacitance and I_t the current in the circuit. The evolution of the state over time is described by (5), where $z_e := \{L, C\}$, $z_a = \{R\}$ and

$$F_e := \begin{bmatrix}\frac{I_t}{C} \\ \frac{1}{L}(V(t) - U_t)\end{bmatrix} \quad \text{and} \quad F_a := \begin{bmatrix}0 \\ -\frac{R}{C} I_t\end{bmatrix}. \tag{7}$$

The dynamics described by the RLC circuit is more diverse than for the pendulum and the system can be hard to identify. This system is characterised by the resistance R, capacitance C, and inductance L, provided $V(t)$ is known.

The 2D reaction diffusion was used by Yin et al. (2021) to assess the quality of APHYNITY. It is a 2D FitzHugh-Nagumo on a 32×32 grid. The system's state at time t is a $2 \times 32 \times 32$ tensor $y_t = \begin{bmatrix}u_t & v_t\end{bmatrix}^T$. The evolution of the state over time is described by (5), where $z_e := \{a, b\}$, $z_a = \{k\}$ and

$$F_e := \begin{bmatrix}a\Delta u_t \\ b\Delta v_t\end{bmatrix} \quad \text{and} \quad F_a := \begin{bmatrix}R_u(u_t, v_t; k) \\ R_v(u_t, v_t)\end{bmatrix}, \tag{8}$$

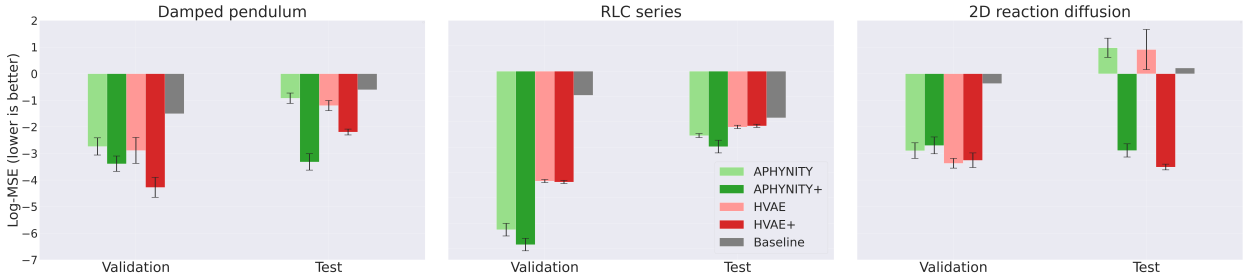


Figure 4. The average log-MSEs over 10 runs for three synthetic problems on the validation and test sets. We compare HVAE (in red) and APHYNITY (in green), in light colours, to their expert augmented versions HVAE+ and APHYNITY+, in darker colours. *On the test sets, AHMs outperform the original models, and by a large margin on the pendulum and diffusion problems. Moreover, augmentation conserves the relatively good performance on the validation set (IID w.r.t. the training set).*

where Δ is the Laplace operator, the local reaction terms are $R_u(u, v; k) = u - u^3 - k - v$ and $R_v(u, v) = u - v$. This model is interesting to study as it considers a state space for which neural architectures may have a real advantage compared to other ML models.

In the following experiments we analyze the effect of our data augmentation strategy on APHYNITY and HVAE. All models explicitly use the assumption that the interaction model follows the structure of (5). For each problem the validation and test sets are respectively IID and OOD with respect to the training distribution. The best models are always selected based on validation performance, that is with samples from Ω. We provide additional details on the different expert models, dataset creation, and neural networks architectures in Appendix B.

4.2. Results

Performance gain from augmentation. *This experiment demonstrates that HVAE and APHYNITY are not robust to OOD test scenarios in opposition to the corresponding AHMs, as shown in Figure 1* for the 2D diffusion problem and in Appendix C for the two other problems. We emphasize that our intention is not to declare a winner between HVAE and APHYNITY. Indeed, both algorithms have already demonstrated superior performance than black box ML models. Hence, we only report a very simple baseline that is the mean value of the signals. We want to compare performance in OOD settings and empirically validate the benefit of AHMs. We compare the predictive performance in Figure 4 (see Table 2 for the raw numbers). Although classical hybrid learning strategies do very well on the IID validation set, they exhibit poor generalization on OOD test sets for all three problems. We also observe some disparity between APHYNITY and HVAE. In addition to different learning strategies, this is probably due to differences in the networks' architectures as they were respectively inspired from the corresponding pendulum experiment in each paper. However, even if one method may outperform the other for some problems, they both benefit from our augmentation strategy (APHYNITY+, HVAE+). Overall, the effect of augmentation goes up to dividing the test error by a factor of $e^{4.6} \approx 100$ in some cases.

Dataset		APH.	HVAE	APH.+	HVAE+
Pendulum	Valid.	$6_{\pm 2}$	$3_{\pm 1}$	$6_{\pm 2}$	$2_{\pm 1}$
	Test	$66_{\pm 9}$	$117_{\pm 10}$	$10_{\pm 4}$	$11_{\pm 2}$
RLC	Valid.	$6_{\pm 3}$	$38_{\pm 2}$	$7_{\pm 5}$	$28_{\pm 1}$
	Test	$17_{\pm 3}$	$25_{\pm 2}$	$5_{\pm 2}$	$12_{\pm 1}$
Diffusion	Valid.	$2_{\pm 0}$	$2_{\pm 0}$	$2_{\pm 0}$	$2_{\pm 0}$
	Test	$27_{\pm 2}$	$32_{\pm 10}$	$3_{\pm 1}$	$2_{\pm 0}$

Table 1. Comparison of mean relative precision (in %, $\pm$ indicates one standard deviation) over 10 runs of predicted physical parameters of different hybrid modelling strategies in validation and OOD test settings. Augmented versions are denoted with a $+$. *While the accuracy of APHYNITY and HVAE is good on the validation set, it collapses on the OOD test set. On the opposite, the augmented versions perform well on both validation and test sets.*

Stability for non-exact models. The empirical results from Figure 4 are very important as they show that even when the decoder is not Ω-*exact* (and hence not $\overset{+}{\Omega}$-*exact*), augmentation is still useful. In particular, Table 1 shows that the encoder does not predict the physical parameters perfectly. This indicates that the encoder is not Ω-*exact* and neither should be the decoder. This table shows the relative error on the physical parameters computed as $\sum_{i=1}^{k} \frac{1}{k} \left| \frac{z_e^i - \mu_\theta^i}{z_e^i} \right|$, where μ_θ^i is the estimated most likely value of the i^{th} component of the physical parameters. We first notice that APHYNITY and HVAE perform differently and their performance depends on the specific problem. While APHYNITY accurately estimates the physical parameters on the IID validation set for the 3 problems, HVAE's performance are mixed on the RLC problem as it makes prediction that are 38% away from the nominal parameter value on average whereas APHYNITY reduces this error to 6%. Interestingly, we observe that the proposed augmentation strategies improve the encoder such that it accurately estimates the physical parameters also on the OOD test set even for HVAE on the RLC problem. This confirms that the augmentation strategy is helpful even when the hybrid

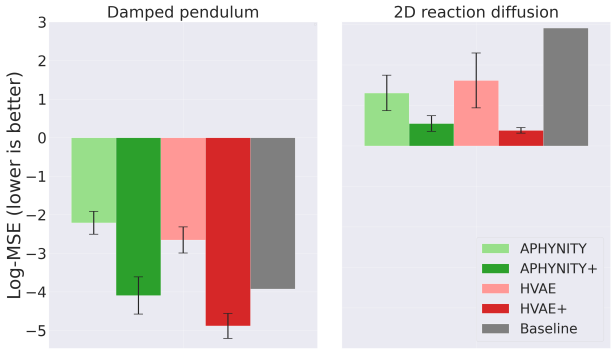


Figure 5. The average log-MSEs over 10 runs for the *damped pendulum* and *2D reaction diffusion* problems on a test distribution for which z_a, in addition to z_e, is also shifted. *AHM achieves better peformance than stand HyL algorithms even when the test distribution support z_a differs from the training.*

model is not Ω-*exact*. As a conclusion, augmented hybrid learning outperforms classical hybrid learning both on the predictive accuracy and at inferring the expert variables.

Effect of out of expertise shift. *This experiment shows that our augmentation strategy may remain beneficial even when the train and test supports of z_a are not identical.* This scenario corresponds to samples (x, y) generated by $(z_a, z_e) \in (\overset{*}{\mathcal{Z}}_a \setminus \mathcal{Z}_a) \times \tilde{\mathcal{Z}}_e$ depicted by the violet domains in Figure 3. In Figure 5 we observe the log-MSE of augmented and non-augmented hybrid models trained for $(z_a, z_e) \in \mathcal{Z}_a \times \mathcal{Z}_e$ on test data that are generated with $(z_a, z_e) \in \tilde{\mathcal{Z}}_a \times \tilde{\mathcal{Z}}_e$. For the pendulum, the support over $z_a = \alpha$ is $[0, 0.3]$ in train and $[0.3, 0.6]$ in test; For the 2D reaction diffusion, $z_a = k$ is $[0.003, 0.005]$ in train and $[0.005, 0.008]$ in test. We observe that augmented models outperform the original models by a large margin. These results suggest that augmentation could be very valuable in practice, even when the distribution shift is also caused by non expert variables. However, if the shift on z_a becomes the dominant effect, augmented models also eventually becomes vulnerable to shifts on z_e as demonstrated by supplementary experiments in Appendix B.

5. Related work

5.1. Hybrid modelling

Hybrid Learning (HyL), or gray box modelling as called in its early days in the 90's (Psichogios & Ungar, 1992; Rico-Martinez et al., 1994; Thompson & Kramer, 1994; Rivera-Sampayo & Vélez-Reyes, 2001; Braun & Chaturvedi, 2002), has been an appropriate method to learn models that are both expressive and interpretable, while also allowing them to be learnt on fewer data. The interest for HyL (Mehta et al., 2020; Lei & Mirams, 2021; Reichstein et al., 2019; Saha et al., 2020; Guen & Thome, 2020; Levine & Stuart, 2021; Espeholt et al., 2021) has greatly renewed since the outbreak of recent neural network architectures that simplify the combination of physical equations within ML models. As an example, Neural ODE (Chen et al., 2018) and convolutional neural networks (LeCun et al., 1995, CNN) are privileged architectures to work with dynamical systems described by ODEs or PDEs. While most of the HyL's literature focus on the predictive performance of hybrid models, recent work have also showed that HyL may help to infer the physical parameter accurately (Yin et al., 2021; Takeishi & Kalousis, 2021). This is aligned with Zyla et al. (2020) (see Section 40.2.2.2) which observe that inference on incomplete models results in a *systematic bias*. Similar to HyL, they extend the model with *nuisance* parameters in order to improve its fidelity, and to reduce the systematic bias.

In this work, we decided to study Yin et al. (2021) and Takeishi & Kalousis (2021) for two reasons that distinguish them from the rest of the HyL literature. First, these are notable examples of HyL algorithms that can be applied to a broad class of problems in contrast to papers that focus on specific applications (Lei & Mirams, 2021; Reichstein et al., 2019). Second, those methods also learn a reliable inference model for the physical parameters, suggesting that the expert model is used properly in the generative model, which is a key assumption for our augmentation. While Takeishi & Kalousis (2021) claim to achieve robustness with HyL, we argue that this statement is incomplete as HVAE fails in OOD settings. In particular, their approach is only able to generalize with respect to unseen time or initial state if the model correctly identifies the latent variables z_a, z_e.

5.2. Combining hybrid modelling and data augmentation

Close to our idea is the one proposed in Shrivastava et al. (2017) where they train a GAN model that improves the realism of a simulated image while conserving its semantic content (e.g. eyes colour) as modeled by the simulation parameters. The generated data with their annotations may then be used for a downstream task, such as inferring the properties of real images that corresponds to simulation parameters. The GAN objective from Shrivastava et al. (2017) requires that the two distributions induced by the semantic content of real and simulated data are identical. On the opposite, we consider training data that corresponds to expert parameters with limited diversity, and overcome this scarcity with expert augmentation. Another line of work similar to ours is Sim2Real, which considers the task of transferring a model trained on simulated data to real world (Doersch & Zisserman, 2019; Sadeghi et al., 2018; 2017). Robust HyL, as a way to enhance simulations, could be used for Sim2Real.

5.3. Robust ML and Invariant Learning

Various statistical methods have been introduced to ensure models generalize under distribution shift. Domain-adversarial objectives aimed at learning (conditionally) invariant predictors (Ganin et al., 2016; Zhang et al., 2017; Li et al., 2018), GroupDRO (Sagawa et al., 2019) optimizing for worst-case loss over multiple domains and IRM (Arjovsky et al., 2019) as well as sub-group calibration (Wald et al., 2021) aiming to satisfy calibration or sufficiency constraints to learn features invariant across domains. Extensions, able to infer domain labels from training data have been proposed as well (Lahoti et al., 2020; Creager et al., 2021), partially inspired by fairness objectives (Hébert-Johnson et al., 2018; Kim et al., 2019). In contrast to AHM, all of these methods rely on the variation of interest being present in the training data.

6. Discussion

We now examine the assumptions we made to derive our augmentation strategy and discuss potential limitations.

Erroneous interaction model. The exactness of the hybrid component $p_\theta(y|x, y_e, z_a)$ is a critical assumption underlying our expert-based augmentation strategy. Unfortunately, this component is learned from training data only, hence we cannot prove its exactness on the test domain, which corresponds to a different domain $\mathcal{Y}_e$. However, we argue that soft assumptions on the class of interaction model may alleviate this problem. As an example, when we consider an additive hybrid model, as in APHYNITY (Yin et al., 2021), and embed this hypothesis into the interaction model, generalization to unseen y_e follows. When this assumption is too strong, we could still expect generalization of $p_\theta(y|x, y_e, z_a)$ because HyL drives y samples from p_θ to be close to y_e. It implies that the corresponding function approximator is smooth, which helps generalization to unseen scenarios. This contrasts with the encoder q_ψ for which a a good inductive bias usually is not available.

Diagnostic. While crucial, we cannot guarantee the exactness of the decoder p_θ in general because we only evaluate the encoder and the decoder jointly on data points (x, y, x_o, y_o). However, in some cases we can detect model misspecification by observing that the predictive model $p_{\theta,\psi}(y|x, x_o, y_o)$ is imperfect. Making this observation is not always simple as it requires prior knowledge on the expected accuracy of an exact model. However, when the system is deterministically identifiable, we may argue that the accuracy should be only limited by the intrinsic noise between x and y given z_a and z_e.

Relaxing exactness. Even with a strong inductive bias on the decoder, achieving exactness is hopeless in practical settings. However, our experiments demonstrate that expert-augmentation works in practice. We can explain this by taking a look at Figure 3. If the generative model that maps x and (z_a, z_e) is incorrect, the mapping from $\mathcal{Z}_a$ and $\mathcal{Z}_e$ could be slightly off from $\overset{+}{\Omega}$. However, this does not preclude the set of augmented samples to be closer to $\overset{+}{\Omega}$ than Ω and to induce a better predictive model on $\overset{+}{\Omega}$ than the original model trained only on Ω.

Limitations We considered expert models that are parameterized by a small number of parameters, which can be covered densely via sampling. Covering densely a higher dimensional parameter space with the augmentation strategy becomes quickly impossible, hence a smarter sampling strategy would be required, such as worst-case sampling. Another difficulty is to choose a plausible range of parameters that contains both the train and the test support, this will often require a human expert in the loop. Finally, we assume that the train distribution of z_a should be representative of the test distribution, we empirically observed that a softer version of this assumption could be enough. However, performance will eventually decline as the support of the test distribution for z_a is far from the training domain.

7. Conclusion

In this work, we describe HyL with a probabilistic model in which one component of the latent process, denoted the expert model, is known. In this context, we establish that state-of-the-art HyL algorithms are vulnerable to distribution shifts even when the expert model is well defined for such configurations. Grounded in this formalisation, we derive that expert augmentations induce robustness to OOD settings. We discuss how our assumptions can transfer to real-world settings and describe how to diagnose potential shortcomings. Finally, empirical evidence asserts that expert augmentations may be beneficial even when one of our assumptions on the class of distribution shift is violated.

Our augmentation is applicable to a large class of hybrid models, hence it should benefit from future progress in HyL. Thus, we believe research in HyL and formally defining its targeted objectives is an important direction for further improving the robustness of hybrid models. As an example, the minimal description length principle (Grünwald, 2007) could be a great resource to investigate the balance between the model's capacity and robustness. Finally, robust ML models must eventually translate to real-world applications, hence a next step would be to apply AHMs to real-wold data. Paving the way to future research combining AHM with robust ML methods.

Acknowledgements

We would like to acknowledge Andy Miller, Dan Busbridge, Jason Ramapuram, Joe Futoma, and Mark Goldstein for providing useful feedback on this manuscript or an earlier version of it.

References

Arjovsky, M., Bottou, L., Gulrajani, I., and Lopez-Paz, D. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

Braun, J. E. and Chaturvedi, N. An inverse gray-box model for transient building load prediction. *HVAC&R Research*, 8(1):73–99, 2002.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. Neural ordinary differential equations. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 6572–6583, 2018.

Crabtree, V. P. and Smith, P. R. Physiological models of the human vasculature and photoplethysmography. *Electronic Systems and Control Division Research, Department of Electronic and Electrical Engineering, Loughborough University*, pp. 60–63, 2003.

Cranmer, M., Greydanus, S., Hoyer, S., Battaglia, P., Spergel, D., and Ho, S. Lagrangian neural networks. *arXiv preprint arXiv:2003.04630*, 2020.

Creager, E., Jacobsen, J.-H., and Zemel, R. Environment inference for invariant learning. In *International Conference on Machine Learning*, pp. 2189–2200. PMLR, 2021.

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 113–123, 2019.

Doersch, C. and Zisserman, A. Sim2real transfer learning for 3d human pose estimation: motion to the rescue. *Advances in Neural Information Processing Systems*, 32: 12949–12961, 2019.

Espeholt, L., Agrawal, S., Sønderby, C., Kumar, M., Heek, J., Bromberg, C., Gazen, C., Hickey, J., Bell, A., and Kalchbrenner, N. Skillful twelve hour precipitation forecasts using large context neural networks. *arXiv preprint arXiv:2111.07470*, 2021.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.

Geirhos, R., Jacobsen, J.-H., Michaelis, C., Zemel, R., Brendel, W., Bethge, M., and Wichmann, F. A. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020.

Grünwald, P. D. *The minimum description length principle*. MIT press, 2007.

Guen, V. L. and Thome, N. Disentangling physical dynamics from unknown factors for unsupervised video prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11474–11484, 2020.

Gulrajani, I. and Lopez-Paz, D. In search of lost domain generalization. *arXiv preprint arXiv:2007.01434*, 2020.

Hébert-Johnson, U., Kim, M., Reingold, O., and Rothblum, G. Multicalibration: Calibration for the (computationally-identifiable) masses. In *International Conference on Machine Learning*, pp. 1939–1948. PMLR, 2018.

Hodgkin, A. L. and Huxley, A. F. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117 (4):500–544, 1952.

Keriven, N. and Peyré, G. Universal invariant and equivariant graph neural networks. *Advances in Neural Information Processing Systems*, 32:7092–7101, 2019.

Kim, M. P., Ghorbani, A., and Zou, J. Multiaccuracy: Black-box post-processing for fairness in classification. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pp. 247–254, 2019.

Koh, P. W., Sagawa, S., Marklund, H., Xie, S. M., Zhang, M., Balsubramani, A., Hu, W., Yasunaga, M., Phillips, R. L., Gao, I., et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning*, pp. 5637–5664. PMLR, 2021.

Krueger, D., Caballero, E., Jacobsen, J.-H., Zhang, A., Binas, J., Zhang, D., Le Priol, R., and Courville, A. Out-of-distribution generalization via risk extrapolation (rex). In *International Conference on Machine Learning*, pp. 5815–5826. PMLR, 2021.

Lahoti, P., Beutel, A., Chen, J., Lee, K., Prost, F., Thain, N., Wang, X., and Chi, E. H. Fairness without demographics through adversarially reweighted learning. *arXiv preprint arXiv:2006.13114*, 2020.

LeCun, Y., Bengio, Y., et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

Lei, C. L. and Mirams, G. R. Neural network differential equations for ion channel modelling. *Frontiers in Physiology*, pp. 1166, 2021.

Levine, M. E. and Stuart, A. M. A framework for machine learning of model error in dynamical systems. *arXiv preprint arXiv:2107.06658*, 2021.

Li, Y., Tian, X., Gong, M., Liu, Y., Liu, T., Zhang, K., and Tao, D. Deep domain generalization via conditional invariant adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 624–639, 2018.

Mahmood, O., Mansimov, E., Bonneau, R., and Cho, K. Masked graph modeling for molecule generation. *Nature communications*, 12(1):1–12, 2021.

Mehta, V., Char, I., Neiswanger, W., Chung, Y., Nelson, A. O., Boyer, M. D., Kolemen, E., and Schneider, J. Neural dynamical systems: Balancing structure and flexibility in physical prediction. *arXiv preprint arXiv:2006.12682*, 2020.

Psichogios, D. C. and Ungar, L. H. A hybrid neural network-first principles approach to process modeling. *AIChE Journal*, 38(10):1499–1511, 1992.

Qian, Z., Zame, W. R., van der Schaar, M., Fleuren, L. M., and Elbers, P. Integrating expert odes into neural odes: Pharmacology and disease progression. *arXiv preprint arXiv:2106.02875*, 2021.

Reichstein, M., Camps-Valls, G., Stevens, B., Jung, M., Denzler, J., Carvalhais, N., et al. Deep learning and process understanding for data-driven earth system science. *Nature*, 566(7743):195–204, 2019.

Rico-Martinez, R., Anderson, J., and Kevrekidis, I. Continuous-time nonlinear signal processing: a neural network based approach for gray box identification. In *Proceedings of IEEE Workshop on Neural Networks for Signal Processing*, pp. 596–605. IEEE, 1994.

Rivera-Sampayo, R. and Vélez-Reyes, M. Gray-box modeling of electric drive systems using neural networks. In *Proceedings of the 2001 IEEE International Conference on Control Applications (CCA'01)(Cat. No. 01CH37204)*, pp. 146–151. IEEE, 2001.

Sadeghi, F., Toshev, A., Jang, E., and Levine, S. Sim2real view invariant visual servoing by recurrent control. *arXiv preprint arXiv:1712.07642*, 2017.

Sadeghi, F., Toshev, A., Jang, E., and Levine, S. Sim2real viewpoint invariant visual servoing by recurrent control. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4691–4699, 2018.

Sagawa, S., Koh, P. W., Hashimoto, T. B., and Liang, P. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. In *International Conference on Learning Representations*, 2019.

Saha, P., Dash, S., and Mukhopadhyay, S. Physics-incorporated convolutional recurrent neural networks for source identification and forecasting of dynamical systems. *arXiv preprint arXiv:2004.06243*, 2020.

Shrivastava, A., Pfister, T., Tuzel, O., Susskind, J., Wang, W., and Webb, R. Learning from simulated and unsupervised images through adversarial training. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2107–2116, 2017.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

Takeishi, N. and Kalousis, A. Physics-integrated variational autoencoders for robust and interpretable generative modeling. *Advances in Neural Information Processing Systems*, 34, 2021.

Thompson, M. L. and Kramer, M. A. Modeling chemical processes using prior knowledge and neural networks. *AIChE Journal*, 40(8):1328–1340, 1994.

Wald, Y., Feder, A., Greenfeld, D., and Shalit, U. On calibration and out-of-domain generalization. *arXiv preprint arXiv:2102.10395*, 2021.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Yin, Y., Le Guen, V., Dona, J., de Bézenac, E., Ayed, I., Thome, N., and Gallinari, P. Augmenting physical models with deep networks for complex dynamics forecasting. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12):124012, 2021.

Zhang, Y., Barzilay, R., and Jaakkola, T. Aspect-augmented adversarial networks for domain adaptation. *Transactions of the Association for Computational Linguistics*, 5:515–528, 2017.

Zyla, P. et al. Review of Particle Physics. *PTEP*, 2020(8): 083C01, 2020. doi: 10.1093/ptep/ptaa104.

A. Additional description of expert augmentation

We provide the procedure to do expert augmentation for robust HyL as the sequence of steps below.

1. Train both the encoder $q_\psi(z_a, z_e|x, y)$ and the interaction model $p_\theta(y|x_o, z_a, y_e)$ with a HyL algorithm, by minimizing the corresponding loss $\mathcal{L}(\psi, \theta) = \mathbb{E}_{\mathcal{D}}\left[\ell(x, y; \theta, \psi)\right]$ on the training set $\mathcal{D}$;

2. Decide on an augmented distribution $p(\overset{+}{z_e})$ for z_e that contains both train and test scenarios;

3. Reproduce the following steps to generate a dataset $\overset{+}{\mathcal{D}}$ of observations and expert variables $(x, y, z_e) \sim \mathbb{E}_{p(z_a)p(y_e|z_e,x,y)}\left[p(z_e)p(x)p_\theta(y|y_e, z_a, x)\right]$:
   (a) Sample (x_o, y_o) from the data;
   (b) Sample z_a from the posterior $q_\psi(z_a|x_o, y_o)$;
   (c) Sample z_e from $p(\overset{+}{z_e})$;
   (d) Push forward x, z_a and z_e in the generative model as $y_e \sim p(y_e|x_o, z_e)$ and $y \sim p_\theta(y|x_o, z_a, y_e)$;
   (e) Add the triplet (x_o, y, z_e) to the augmented training set $\overset{+}{\mathcal{D}}$.

4. Freeze the interaction model, and fine-tune the encoder $q_\psi(z_a, z_e|x, y)$ on the augmented dataset $\overset{+}{\mathcal{D}}$ by minimizing $\overset{+}{\mathcal{L}}(\psi, \theta) = \mathbb{E}_{\overset{+}{\mathcal{D}}}\left[\ell(x, y; \theta, \psi) - \log q_\psi(z_e|x, y)\right]$.

B. Additional details on experiments

B.1. Damped pendulum

Datasets. We use Neural Ordinary Differential Equations (NODE) (Chen et al., 2018) to solve the ODE ruling the damped pendulum. Each sample is simulated for $t_0 = 0s$, $t_1 = 5s$, and $t_2 = 20s$, with a time resolution equal to 0.1 second. The models are trained with only the realizations between t_0 and t_1. At test and validation time, the pair $(x_o, y_o) = (y_0, [y_{i\Delta t}]_{i=1}^{t_1/\Delta t})$, $x = y_{t_1}$ and the model predicts $y = [y_{i\Delta t}]_{i=t_2/\Delta t+1}^{t_2/\Delta t}$. The initial angular speed is always 0 and $\theta_0 \sim \mathcal{U}(-\frac{\pi}{2}, \frac{\pi}{2})$.

The training set is made of 1000 samples and the validation set of 100 samples. They are both generated by sampling uniformly $z_a := \alpha$ from $\mathcal{Z}_a := [0, 0.6]$ and $z_e := \omega_0$ from $\mathcal{Z}_e := [1.5, 3.1]$. The shifted test set contains 100 samples generated by sampling uniformly z_a in $\mathcal{Z}_a$ and z_e in $\tilde{\mathcal{Z}}_e := [0.5, 1.5]$.

APHYNITY. Our model is composed of a 1-layer RNN with 128 units that encodes the input signal $y_{0:t_1}$ as $h(y_{0:t_1}) \in \mathbb{R}^{128}$. An MLP with 3 layers of 150 units and ReLU activations maps h to $\mathbb{R}_+$ to predict ω_0. The function $f_a : \mathbb{R}^{128} \times \mathbb{R}^2$ is an MLP with 3 layers of 50 units and ReLU activations (no activation for the last layer). The models are trained for 50 epochs with Adam with no weight decay and a learning rate equal to 0.0005. For the Lagrangian optimization we use $N_{iter} = 5, \lambda_0 = 10, \tau_2 = 5$ (see (Yin et al., 2021). The augmented data are generated by sampling uniformly $z_e \in \overset{+}{\mathcal{Z}}_e := [0.5, 3.5]$ and z_a from the marginal predictive prediction of the model, that is we use the training dataset to infer values of z_a and use these as samples. The batch size is 100.

HVAE. We use the notations from Takeishi & Kalousis (2021) to describe the architecture of the VAE. The network $g_{p,1} : \mathbb{R}^2 \times \mathbb{R}^{d_a}$, where $d_a = 1$ is the size of the latent space for the interaction model, is supposed to filter the observations so that they can be generated by the expert model. It has 2 hidden layers with 128 units, $g_{p,2}$ is an MLP with the following hidden layers $[128, 128, 256, 64, 32]$ and takes the full sequence of filtered states and predicts the mean and variance of a normal distribution that parameterize the posterior $p_\theta(z_e|x, y, z_a)$. Another network, g_a takes the sequence of observations and predict the posterior distribution of z_a as a normal distribution. This network has the following hidden layers $[256, 256, 128, 32]$. All networks have SeLU activations. In general the decoder of HVAE can be anything that combines the expert model in order to produce samples in the observation space, as we made the hypothesis that the ODE is just missing an additive term, the decoder is a NODE where the function is the sum of f_e and f_a a two hidden layers MLP with 64 units and SeLU activation (except for the last layer that has no activation). The likelihood model is also Gaussian

with the mean being predicted by the NODE and the variance learned but shared for all observations. For additional details on our architecture and implementation details we encourage the interested reader to check our code.

The networks are trained jointly for 1000 epochs with Adam optimizer, with a learning rate equal to 0.0005, weight decay equal to 0.000001 and batch size 200. The other parameters are set to $\gamma = 1$, $\alpha = 0.01$ and $\beta = 0.01$. The HVAE also relies on some augmentation during training and in order to compare fairly our model to theirs we use the same distribution for our augmentation and theirs that is $z_a \sim \mathcal{N}(0, I)$ and $z_e \sim \mathcal{U}(0.5, 3.5)$.

B.2. RLC series

Datasets. Similar to the damped pendulum, we use NODE to solve the ODE ruling the RLC circuit. Each sample is simulated for $t_0 = 0s$, $t_1 = 5s$, and $t_2 = 20s$, with a time resolution equal to 0.1 second. The models are trained with only the realizations between t_0 and t_1. At test and validation time, the pair $(x_o, y_o) = (y_0, [y_{i\Delta t}]_{i=1}^{t_1/\Delta t})$, $x = y_{t_1}$ and the model predicts $y = [y_{i\Delta t}]_{i=t_2/\Delta t+1}^{t_2/\Delta t}$. In all experiments, the initial value for $U_0 \sim \mathcal{N}(0, 1)$ and $I_0 = 0$, the voltage source delivers a AC + DC tension $V(t) = 2.5\sin(4\pi t) + 1$.

The training set is made of 2000 samples and the validation set of 100 samples. They are both generated by sampling uniformly $z_a := R$ from $\mathcal{Z}_a := [1, 3]$ and $z_e := [L, C]$ from $\mathcal{Z}_e := [1, 3] \times [0.5, 1.5]$. The shifted test set contains 100 samples and is generated by sampling uniformly z_a in $\mathcal{Z}_a$ and z_e in $\tilde{\mathcal{Z}}_e := [3, 5] \times [1., 2.5]$.

APHYNITY. Our model is composed of a 1-layer RNN with 128 units that encodes the input signal $y_{0:t_1}$ as $h(y_{0:t_1}) \in \mathbb{R}^{128}$. An MLP with 3 layers of 200 units and ReLU activations maps h to $\mathbb{R}_+^2$ that predicts L and C. The function $f_a : \mathbb{R}^{128} \times \mathbb{R}^2$ is an MLP with 3 layers of 150 units and ReLU activations (no activation for the last layer). The models are trained for 50 epochs with Adam with no weight decay and a learning rate equal to 0.0005. For the Lagrangian optimization we use $N_{iter} = 5, \lambda_0 = 10, \tau_2 = 5$ (see (Yin et al., 2021)). The augmented data are generated by sampling uniformly $z_e \in \overset{+}{\mathcal{Z}}_e := [1, 5] \times [0.5, 2.5]$ and z_a from the marginal predictive prediction of the model, that is we use the training dataset to infer values of z_a and use these as samples. The batch size is 100.

HVAE. We use the same networks' architectures than for the damped pendulum experiment. Except that $g_{p,1}$ is has 3 hidden layers with 100 units.

The networks are trained jointly for 1000 epochs with Adam optimizer, with a learning rate equal to 0.0005, weight decay equal to 0.000001 and batch size 100. The other parameters are set to $\gamma = 1$, $\alpha = 0.01$ and $\beta = 0.01$. The HVAE also relies on some augmentation during training and in order to compare fairly our model to theirs we use the same distribution for our augmentation and theirs that is $z_a \sim \mathcal{N}(0, I)$ and $z_e \sim \mathcal{U}(1, 5) \times \mathcal{U}(0.5, 2.5)$.

B.3. 2D reaction diffusion

Datasets. Similar to the damped pendulum, we use NODE to solve the PDE ruling the reaction diffusion. We closely follow the experimental setting described in Yin et al. (2021) and approximate the Laplace operator with a 3×3 discrete version of the operator. Each sample is simulated for $t_0 = 0s$, $t_1 = 1s$, and $t_2 = 5s$, with a time resolution equal to 0.1 second. The models are trained with only the realizations between t_0 and t_1. At test and validation time, the pair $(x_o, y_o) = (y_0, [y_{i\Delta t}]_{i=1}^{t_1/\Delta t})$, $x = y_{t_1}$ and the model predicts $y = [y_{i\Delta t}]_{i=t_2/\Delta t+1}^{t_2/\Delta t}$. The initial state is sampled from a uniform distribution in $[0, 1]$.

The training set is made of 2000 samples and the validation set of 100 samples. They are both generated by sampling uniformly $z_a := k$ from $\mathcal{Z}_a := [0.003, 0.005]$ and $z_e := [a, b]$ from $\mathcal{Z}_e := [0.001, 0.002] \times [0.003, 0.007]$. The shifted test set contains 100 samples and is generated by sampling uniformly z_a in $\mathcal{Z}_a$ and z_e in $\tilde{\mathcal{Z}}_e := [0.002, 0.004] \times [0.001, 0.1]$.

APHYNITY. Our model is composed of a deep CNN that encodes the input sequence of 10 images. The exact architecture can be found in the code. The dimension of z_a is equal to 10. Similarly to Yin et al. (2021) the function f_a is a 3-layers CNN with ReLU activations. The models are trained for 500 epochs with Adam with no weight decay and a learning rate equal to 0.0005. For the Lagrangian optimization we use $N_{iter} = 1, \lambda_0 = 10, \tau_2 = 5.$. The augmented data are generated by sampling uniformly $z_e \in \overset{+}{\mathcal{Z}}_e := [0.001, 0.004] \times [0.001, 0.01]$ and z_a from the marginal predictive prediction of the model, that is we use the training dataset to infer values of z_a and use these as samples. The batch size is 100.

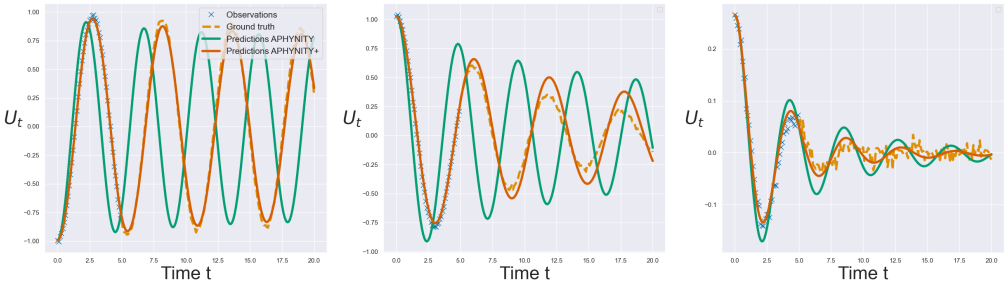


Figure 6. Comparison of the predictions made by APHYNITY and APHYNITY+ on the damped pendulum problem for 3 diverse test examples. It is important to mention that the support of the test distribution is disjoint from the training support. *We clearly observe the beneficial effect of augmentation which lead to more accurate predictions.*

B.3.1. HVAE

We use the notations from Takeishi & Kalousis (2021) to describe the architecture of the VAE. The network $g_{p,1} : \mathbb{R}^{2\times32\times32} \times \mathbb{R}^{d_a}$ is a conditional U-net, where $d_a = 10$ is the size of the latent space for the interaction model, is supposed to filter the observation so that they can be generated by the expert model. The networks $g_{p,1}$ and g_a share a common backbone CNN and are, in addition, respectively parameterized by 2 3-layers MLPs. All networks have ReLU activations. In general the decoder of HVAE can be anything that combines the expert model in order to produce samples in the observation space, as we made the hypothesis that the ODE is just missing an additive term, the decoder is a NODE where the function is the sum of f_e and f_a a 3-layers CNN. The likelihood model is also Gaussian with the mean being predicted by the NODE and the variance learned but shared for all observations. For additional details on our architecture and implementation details we encourage the interested reader to check our code.

The networks are trained jointly for 1000 epochs with Adam optimizer, with a learning rate equal to 0.0005, weight decay equal to 0.00001 and batch size 100. The other parameters are set to $\gamma = 1$, $\alpha = 0.01$ and $\beta = 0.01$. The HVAE also relies on some augmentation during training and in order to compare fairly our model to theirs we use the same distribution for our augmentation and theirs that is $z_a \sim \mathcal{N}(0, I)$ and $z_e \sim \mathcal{U}(0.001, 0.004) \times \mathcal{U}(0.001, 0.01)$.

C. Supplementary results

We now provide additional results for AHM versus standard HyL models.

C.1. Log-mses on the 3 synthetic problems

Dataset		APH.	HVAE	APH.+	HVAE+
Pendulum	Val.	$-2.7_{\pm0.3}$	$-2.9_{\pm0.5}$	$-3.4_{\pm0.3}$	$-2.9_{\pm0.6}$
	Test	$-0.9_{\pm0.2}$	$-1.2_{\pm0.2}$	$-3.3_{\pm0.3}$	$-3.1_{\pm0.3}$
RLC	Val.	$-6.3_{\pm0.2}$	$-4.3_{\pm0.1}$	$-6.8_{\pm0.2}$	$-3.8_{\pm1.5}$
	Test	$-2.5_{\pm0.1}$	$-2.2_{\pm0.1}$	$-3.0_{\pm0.3}$	$-2.1_{\pm0.3}$
Diffusion	Val.	$-2.9_{\pm0.3}$	$-3.4_{\pm0.2}$	$-2.7_{\pm0.3}$	$-3.3_{\pm0.3}$
	Test	$1.0_{\pm0.4}$	$0.9_{\pm0.8}$	$-2.9_{\pm0.2}$	$-3.5_{\pm0.1}$

Table 2. Comparison of the log-mse of different hybrid modelling strategies in validation and OOD test settings. *Except on RLC, AHMs always outperform the corresponding HyL models on the test sets. Good performance on the validation set are conserved with augmentation.*

C.2. Distribution shift visualization

Similar to Figure 1, Figure 6 and Figure 7 showcase the behaviour of APHYNITY and APHYNITY+ for OOD test samples.

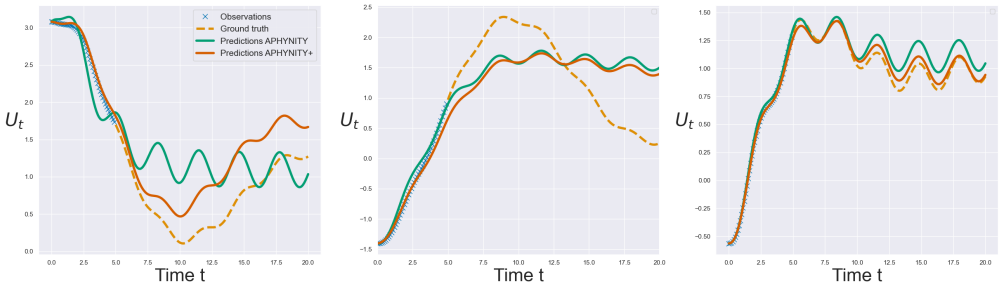


Figure 7. Comparison of the predictions made by APHYNITY and APHYNITY+ on the RLC series problem for 3 diverse test examples. It is important to mention that the support of the test distribution is disjoint from the training support. *We can perceive the beneficial effect of augmentation which lead to more accurate predictions in some cases. However both models are inaccurate. This indicates that the RLC series parameters are not easily identifiable, hence the generative model is not exact and augmentation is not as useful as for the diffusion and the pendulum.*

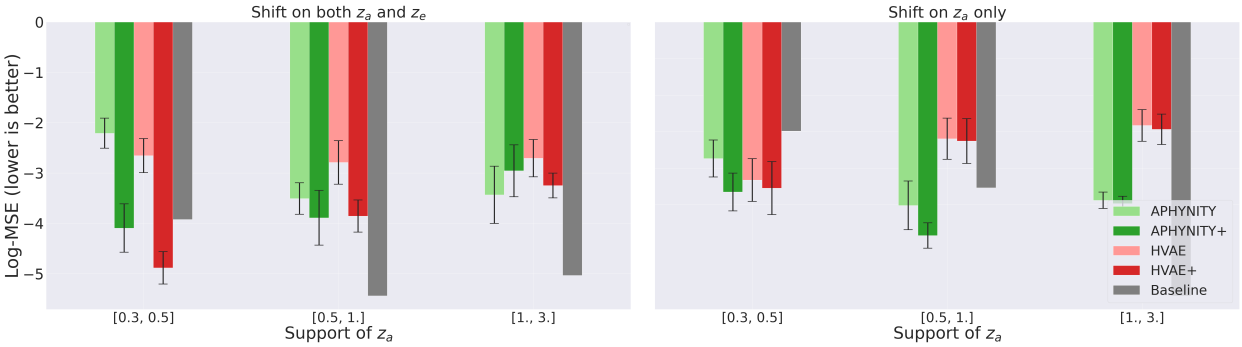


Figure 8. **Damped pendulum.** Effect of a distribution shift on the latent variable z_a of the interaction model. When the shift of z_a is reasonable (less than 1), the augmented models outperforms standard HyL even when the shift is only on z_a.

C.3. On the effect of out of expertise shift

The additional results in Figure 8, Figure 9 and Figure 10 demonstrate that our augmentations is mostly always beneficial. Although the benefit of augmentation decreases with the gap between the support of the distributions of z_a and train and test times, it still performs either better or on par with non-augmented HyL models.

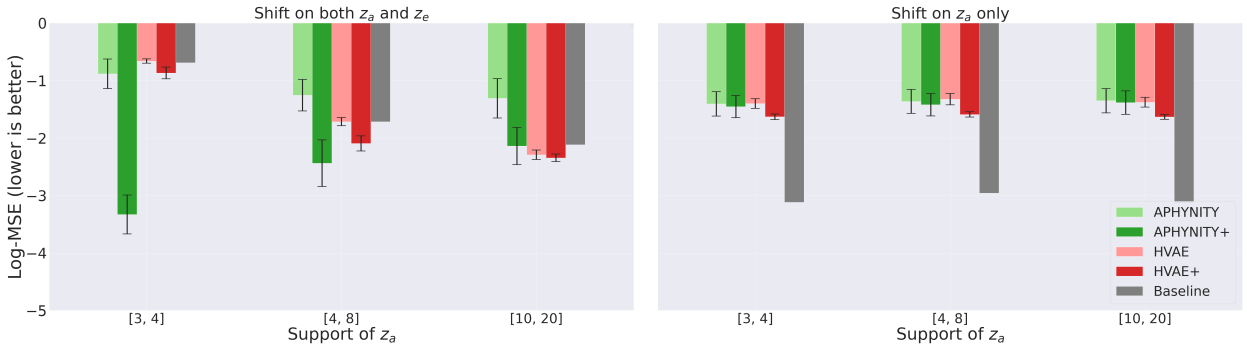


Figure 9. **RLC series.** Effect of a distribution shift on the latent variable z_a of the interaction model. We observe that augmentation is always beneficial, even when the shift is only on z_a. As the dynamics of the RLC series systems depends on the values of all 3 parameters R, L, C, we observe that some distribution shift can even lead to improved performance for the augmented models as for APHYNITY+ when $R \in [3, 4]$

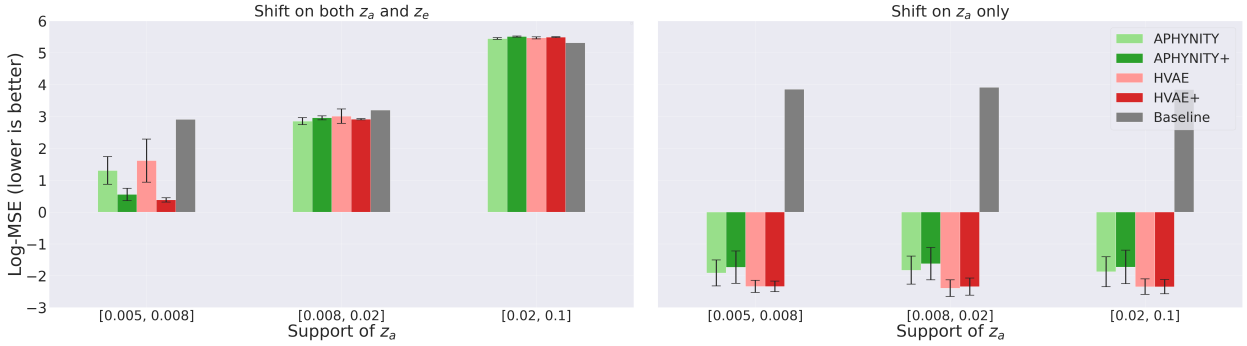


Figure 10. **2D diffusion reaction.** Effect of a distribution shift on the latent variable z_a of the interaction model. When the shift of z_a is reasonable ($k < 0.008$), the augmented models outperforms standard HyL even when the shift is only on z_a.

9.3 EPILOGUE

9.3.1 *Contribution*

The paper demonstrates that existing hybrid learning algorithms are sensitive to distribution shifts, even when they only concern the parameters of the expert model. Expert augmentation addresses this issue when the interaction model is identifiable from data and shows hybrid learning may construct more robust predictive models than uninformed machine learning. The main contributions are i) to provide a simplified description of two hybrid learning algorithms within a common probabilistic modelling framework; ii) to describe a class of out-of-distribution settings for which hybrid learning is relevant; iii) to introduce a simple strategy that enforces robustness in these settings.

When the hybrid model is an auto-encoder, expert augmentation is a simple yet effective strategy to improve the model's performance in unseen scenarios. We did not observe a negative impact of expert augmentation in our experiments. ML practitioners should be aware that guaranteeing robustness to specific out-of-distribution scenarios is feasible. It only requires an expert model that describes the gap between these scenarios and the training data. In contrast to classical augmentation strategies which describe the out-of-distribution data themselves, Augmented hybrid models only require understanding a sub-part of the process related to the distribution shift.

We believe that hybrid learning can change the way measurement devices work. When engineers design a new measurement device, they first start by modelling how the signal of interest relates to first-principles physical effects for which efficient measurement tools exist, such as temperature, light, or sound.

For instance, a speed camera sends light at a given frequency against cars. The reflected light undergoes a frequency shift which can be accurately measured with an appropriate photosensor. Then the device estimates the vehicle's speed with the Doppler effect that relates frequency change and speed to each other. This is only a simplified description; in reality, engineers have developed many strategies to improve the robustness and accuracy of speed cameras. They use multiple light frequencies and elaborate signal processing methods and the camera must be calibrated cautiously. This is what makes these devices expensive; they rely on costly sensors and engineering efforts.

Soon, hybrid learning might unlock the development of new measurement devices at a reduced cost. Many practical settings exist for which we know a model of how the signal of interest and the sensor relate to each other. However, in most cases, the model considers an ideal setting which is free from aggressors which exist in the real world. For example, engineers had either to develop signal processing strategies or elaborate sensors to ensure speed cameras are insensitive to other lights than the one sent by the speed camera itself. Developing better sensors often requires years of research and development in contrast to signal processing. We speculate that hybrid learning might help finding

better signal processing strategies and could reduce the development costs of new sensors in the future.

9.3.2 *Beyond hybrid learning*

Symbolic model discovery Most modern prediction or measurement tools are still free of machine learning solutions. They lean on a profound understanding of the underlying processes described with simple mathematical formulas. Nevertheless, we cannot deny the increasing impact of machine learning on the world. In some sense, symbolic model discovery [Schmidt and Lipson, 2009; Kusner et al., 2017; Sahoo et al., 2018, SMD] reconciles these two observations. It aims at discovering simple mathematical rules that accurately describe data.

Recent SMD techniques [Cranmer et al., 2020b] first learn a deep probabilistic model from a large amount of data and then fit a simple mathematical formula to the learnt model via classical symbolic regression tools [Schmidt and Lipson, 2009]. This two-step strategy benefits from the practical inductive bias of modern deep learning architectures to generalise better than techniques that directly fit a symbolic model to the data. Cranmer et al. [2020b] demonstrated that this strategy is effective for retrieving non-trivial cosmology and might be relevant for interpreting neural networks and discovering novel physics. In addition to their interpretability, symbolic models usually exhibit better generalisation than the corresponding neural network. Symbolic models typically correspond to simpler models that are less prone to overfitting – compressing the neural networks into a few equations acts as a powerful regularisation strategy.

We believe that progress in symbolic model discovery might eventually improve hybrid learning algorithms. Applying SMD to extract a short mathematical description of the interaction model might unlock efficient model discovery grounded on the partial understanding of the phenomenon described by the expert model. In some cases, simple formulas would not be expressive enough to describe the gap between the expert model and reality accurately. We imagine a hybrid model of three components: i) the expert model; ii) a minimal length formula describing the most important part of the misspecification; iii) a deep learning model accounting for the remaining gap.

Inference under misspecification The hybrid learning algorithms considered in the paper jointly build an encoder network that identifies the parameters of the expert model and an interaction model that accounts for the misspecification of the expert model. After training, we can apply the encoder to unseen data and obtain an estimation of the expert model's parameters. However, it is unclear whether we should believe in such estimators as the hybrid model might modify the meaning of the expert parameters.

Another problem of the hybrid model's encoders is their incapacity to reflect uncertainty faithfully. Simulation-based inference [Cranmer et al., 2020a, SBI] methods do not acknowledge model misspecifications but provide an accurate estimation of the uncer-

tainty of the parameters' value [Cannon et al., 2022]. These methods provide efficient algorithmic solutions to perform inference over the parameters of a simulator, even when it is not differentiable. However, the guarantees of classical SBI collapse if the model is misspecified. Recently, inspired by robust Bayesian inference [Chérief-Abdellatif and Alquier, 2020; Knoblauch et al., 2019], robust SBI [Dellaporta et al., 2022] has acknowledged that even complex simulators are misspecified. Still, we believe that robust SBI may be inefficient, and machine learning techniques inspired by hybrid learning might lead to efficient solutions for robust SBI. We believe that the inductive bias of machine learning models combined with a large amount of data should outperform existing robust SBI techniques that rely on classical statistical arguments. One challenge to achieving this is to develop solutions compatible with non-differentiable simulators and efficiently benefit from a large amount of unlabeled data.

9.3.3 *Conclusion and opportunities*

We have observed that hybrid models have robustness properties that are out of reach for purely data-driven machine learning models. In contrast to the classical ML setting, hybrid learning methods embed more than an inductive bias. They start from the assumption that a large part of the phenomenon observed can be described with an expert model. We formulate and achieve a notion of robustness that concerns the effects encoded by the expert model; our simple yet effective augmentation strategy unlocks this robustness in existing hybrid models. Our experiments show the benefit of the augmentation both concerning the parameter identification quality and the hybrid model's predictive accuracy.

There is arguably a significant potential for future development and applications for hybrid models. For example, we foresee the application of hybrid learning to accelerate simulations by augmenting a simplified expert model with a fast machine learning component to close the gap between the fast and inaccurate expert model and the expensive and precise simulator.

We must also acknowledge that model discovery is a challenging problem and inevitably requires some level of causal intervention. We should be careful about when and how we use hybrid learning. In particular, some information about the expert model misspecification and how it relates to the training data is necessary to apply hybrid learning successfully. For instance, the information that some data points correspond to the same physical parameters might suffice. In this case, the inference network should predict parameter estimates consistent within groups of attributes. We argue that the inductive bias of the interaction model is crucial when data is scarce.

This work has only explored existing solutions that focus on differentiable expert models. Both methods considered, APHYNITY and the hybrid-VAE, provide a generic solution that does not require supervision. In the future, hybrid learning algorithms should be compatible with other settings. For example, the differentiability requirement still

limits the range of direct applications of hybrid learning. Moreover, the genericity of existing algorithms might prevent their data efficiency in settings where we have some information about the expert parameters of the training data. There it would make sense to formulate hybrid learning as a semi-supervised machine learning problem rather than an unsupervised one to benefit from the additional structure in the data.

At a higher level, this chapter has demonstrated another benefit of combining probabilistic models: generalisation capabilities that defy results from a naive interpretation of learning theory. Over the past few years, expert models have shifted from black-box languages (e.g., C++ or Matlab) to differentiable probabilistic frameworks. In this context, we anticipate excellent opportunities for hybrid learning. This transition shall streamline interactions between deep probabilistic and expert models. This paradigm motivates further theoretical and practical developments in hybrid learning. For instance, developing new algorithms and the corresponding conditions under which the hybrid model outperforms data-driven solutions is a relevant goal to help practitioners solve real-world problems with hybrid models.

Part IV

CONCLUSION

We know the past but cannot control it. We control the future but cannot know it.

Claude Shannon

10 CONCLUSION

The way Nature drives the world around us appears chaotic at first glance. However, a proper perspective reveals patterns in this illusive disorder. Our ability to discover and exploit these patterns is what we call intelligence. Encoding these structures into mathematical models eventually reduces intelligent reasoning to computing operations and gives rise to artificial intelligence. This dissertation has studied methods for automated model discovery, i.e., machine learning: artificial intelligence that produces intelligence.

In Part i, we argued for a probabilistic modelling approach. We provided an accessible treatment of probabilistic modelling in Chapter 2. We discussed and drew connections between several topics related to probabilistic models, such as maximum likelihood estimation, Bayesian inference and machine learning. Then, Chapter 3 introduced probabilistic graphical models in which graphs serve to express probabilistic statements. We discussed the benefits and limitations of directed and undirected representations and presented practical inference and learning algorithms. Finally, in Chapter 3, we introduced deep neural networks as an effective parameterisation of probability distributions. This parameterisation enables gradient-based optimisation. Hence, it directly translates classical results from statistics and probability into learning and inference algorithms.

The objectives of Part ii were to study and improve deep probabilistic models. In Chapter 5, we established the complementarity of diffusion models and variational autoencoders. Then, Chapter 6 saw normalizing flows as Bayesian networks and highlighted that affine transformations limit the expressivity of normalizing flows. In Chapter 7, we addressed this limitation by introducing a universal parameterisation of monotonic transformations. This new architecture has had an impact outside of normalizing flows; various applications such as model calibration or distributional reinforcement learning have employed them. Overall, Part ii demonstrated that automatic model discovery benefits from improvements and a better understanding of existing deep probabilistic models.

In the last part of this thesis, we sidestepped the expressivity considerations that were the focus of Part ii. Part iii studied informed probabilistic models which embed prescribed domain expertise into deep probabilistic models. Chapter 8 introduced graphical normalizing flows as new explicit probabilistic models. This model class combines the benefits of the Bayesian networks' representation of independencies with the efficient learning algorithm of normalizing flows. Graphical normalizing flows are less prone to overfitting than non-regularised explicit models and benefit from the Bayesian networks' most-attractive features. Finally, in Chapter 9, we discussed probabilistic models informed by a partial physical understanding of the studied phenomenon. We showed that these models outperform the generalisation capabilities of non-informed models.

Probabilistic modelling is a powerful framework. It emphasises the necessity for nuanced answers and reminds us not to carve knowledge in stone. It is also a practical tool. Probabilistic modelling brings the rigour of mathematics to answer questions in the real world. The availability of modern computers, data, and scientific expertise combined with recent algorithmic developments only broadens the impact of probabilistic modelling on the world.

Over four years of research, we have witnessed and, to a certain extent, participated in a true disruption of probabilistic modelling. Among catalysers of this revolution, programming languages natively equipped with automatic differentiation and non-deterministic operations have arguably played an important role. It has allowed the development of new algorithms for training complex probabilistic models on large datasets. In the context of this thesis, this paradigm has strongly supported our answer to the research question – **How to automate the discovery of probabilistic models with deep learning algorithms?** *By acknowledging the connections between distinct model classes.* Without these programming languages developing hybrid models such as in Chapter 9 or diffusion models into auto-encoders as in Chapter 5 would have been impossible.

Our answer to our research question is incomplete and we are still far from resolving the automation of model discovery. Nevertheless, it paves the way for future research and new tools we deem essential to develop.

How we build models is changing. In the future, combining models contextualised by different data sources and able to represent distinct aspects of the entire phenomenon we aim to model shall get simpler. An internet of open-sourced models and an effective search tool to retrieve the models of interest shall be part of this future. Some models would be nearly uninformed, and we would use data to contextualise them to the task of interest. Others would encode our understanding of physics. Finally, some models would be pretrained and represent a phenomenon for which others have already created a faithful model. Open sourcing models and allowing their combination would skyrocket our modelling capabilities.

Achieving this modelling coalescence will require algorithmic and theoretical developments. For instance, it is unclear whether simple gradient-based algorithms are sufficient to train models that combine diverse components. While contextualisation of small models corresponds to computing a posterior over the few parameters, we only infer parameters values from maximum likelihood estimation for deep probabilistic models. The modelling unification will require reconciling these two training paradigms and to create new inference algorithms.

Another issue that holds us away from this long-term objective is our inability to express subtle assumptions about model interactions. For example, it is hard in hybrid models to prevent a neural network from learning something already modelled by the physical equations. It is unclear whether making independence assumptions between parts of a model is an effective strategy or if more subtility could help.

Finally, some models rely on non-deterministic operations or are expressed as systems of equations. In these cases, among others, gradient-based algorithms may be ineffective. Re-expressing these models into functions for which gradient computation is straightforward would allow these models to be part of the coveted probabilistic model's library.

Overall, this thesis argues for reconsidering the artificial distinction between various probabilistic models. Some models, such as deep probabilistic models, can represent various phenomena. These models are relevant when large datasets contextualise them. Others, such as scientific models, are very narrow; they only depend on a few parameters with a prescribed meaning and plausible values. These models only work in specific contexts; they are quasi-static and only necessitate a few data points, if any, for contextualisation. The link from the former class to the latter is clear, training. Similarly to scientific models, deep probabilistic models become tied the a specific context as defined by the training data. The distinction between model classes is often unnecessary.

Part V

APPENDIX

REFERENCES

J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds, et al. Flamingo: a visual language model for few-shot learning. *arXiv preprint arXiv:2204.14198*, 2022.

alexjc. The impact of dallÂ·e on creative work. URL https://app.subsocial.network/@creativeai/the-impact-of-dall-e-on-creative-work-610.

S.-i. Amari. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4-5):185–196, 1993.

B. D. Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

V. I. Arnold. On functions of three variables. *Collected Works: Representations of Functions, Celestial Mechanics and KAM Theory, 1957–1965*, pages 5–8, 2009.

S. Bai, J. Z. Kolter, and V. Koltun. Deep equilibrium models. *Advances in Neural Information Processing Systems*, 32, 2019.

S. Balgi, J. M. Peña, and A. Daoud. Counterfactual analysis of the impact of the imf program on child poverty in the global-south region using causal-graphical normalizing flows. *arXiv preprint arXiv:2202.09391*, 2022a.

S. Balgi, J. M. Pena, and A. Daoud. Personalized public policy analysis in social sciences using causal-graphical normalizing flows. *Assoc. Adv. Artif. Intell. AI Soc. Impact Track*, 2022b.

D. G. Barrett and B. Dherin. Implicit gradient regularization. *arXiv preprint arXiv:2009.11162*, 2020.

P. L. Bartlett, D. J. Foster, and M. J. Telgarsky. Spectrally-normalized margin bounds for neural networks. *Advances in neural information processing systems*, 30, 2017.

J. Behrmann, P. Vicol, K.-C. Wang, R. Grosse, and J.-H. Jacobsen. Understanding and mitigating exploding inverses in invertible neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 1792–1800. PMLR, 2021.

R. v. d. Berg, L. Hasenclever, J. M. Tomczak, and M. Welling. Sylvester normalizing flows for variational inference. *arXiv preprint arXiv:1803.05649*, 2018.

C. M. Bishop. Mixture density networks. 1994.

D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

A. J. Bose, H. Ling, and Y. Cao. Adversarial contrastive estimation. *arXiv preprint arXiv:1805.03642*, 2018.

L. Bottou. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pages 421–436. Springer, 2012.

J. Brehmer and K. Cranmer. Flows for simultaneous manifold learning and density estimation. *Advances in Neural Information Processing Systems*, 33:442–453, 2020.

P. Brouillard, S. Lachapelle, A. Lacoste, S. Lacoste-Julien, and A. Drouin. Differentiable causal discovery from interventional data. *Advances in Neural Information Processing Systems*, 33:21865–21877, 2020.

T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

P. Cannon, D. Ward, and S. M. Schmon. Investigating the impact of model misspecification in neural simulation-based inference. 2022.

CERN. Cern data centre passes the 200-petabyte milestone. URL https://home.cern/news/news/computing/cern-data-centre-passes-200-petabyte-milestone.

C. Ceylan and M. U. Gutmann. Conditional noise-contrastive estimation of unnormalised models. In *International Conference on Machine Learning*, pages 726–734. PMLR, 2018.

T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, pages 6571–6583, 2018.

B.-E. Chérief-Abdellatif and P. Alquier. Mmd-bayes: Robust bayesian estimation via maximum mean discrepancy. In *Symposium on Advances in Approximate Bayesian Inference*, pages 1–21. PMLR, 2020.

K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE transactions on Information Theory*, 14(3):462–467, 1968.

G. F. Cooper and E. Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine learning*, 9(4):309–347, 1992.

K. Cranmer, J. Pavez, and G. Louppe. Approximating likelihood ratios with calibrated discriminative classifiers. *arXiv preprint arXiv:1506.02169*, 2015.

K. Cranmer, J. Brehmer, and G. Louppe. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 117(48):30055–30062, 2020a.

M. Cranmer, A. Sanchez Gonzalez, P. Battaglia, R. Xu, K. Cranmer, D. Spergel, and S. Ho. Discovering symbolic models from deep learning with inductive biases. *Advances in Neural Information Processing Systems*, 33:17429–17442, 2020b.

W. Dabney, M. Rowland, M. Bellemare, and R. Munos. Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

M. Dax, S. R. Green, J. Gair, J. H. Macke, A. Buonanno, and B. Schölkopf. Amortized bayesian inference of gravitational waves with normalizing flows.

B. Dayma, S. Patil, P. Cuenca, K. Saifullah, T. Abraham, P. LÃª Khá¹¯c, L. Melas, and R. Ghosh. DallÂ·e mini, 7 2021. URL https://github.com/borisdayma/dalle-mini.

N. De Cao, W. Aziz, and I. Titov. Block neural autoregressive flow. In *Uncertainty in Artificial Intelligence*, pages 1263–1273. PMLR, 2020.

A. Delaunoy, A. Wehenkel, T. Hinderer, S. Nissanke, C. Weniger, A. R. Williamson, and G. Louppe. Lightning-fast gravitational wave parameter inference through neural amortization. *arXiv preprint arXiv:2010.12931*, 2020.

A. Delaunoy, J. Hermans, F. Rozet, A. Wehenkel, and G. Louppe. Towards reliable simulation-based inference with balanced neural ratio estimation. *arXiv preprint arXiv:2208.13624*, 2022.

C. Dellaporta, J. Knoblauch, T. Damoulas, and F.-X. Briol. Robust bayesian inference for simulator-based models via the mmd posterior bootstrap. In *International Conference on Artificial Intelligence and Statistics*, pages 943–970. PMLR, 2022.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

L. Devroye, L. Györfi, and G. Lugosi. *A probabilistic theory of pattern recognition*, volume 31. Springer Science & Business Media, 2013.

B. Dey, D. Zhao, J. A. Newman, B. H. Andrews, R. Izbicki, and A. B. Lee. Calibrated predictive distributions for photometric redshifts.

L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real nvp. In *International Conference in Learning Representations*, 2017.

T. Dockhorn, A. Vahdat, and K. Kreis. Score-based generative modeling with critically-damped langevin diffusion. *arXiv preprint arXiv:2112.07068*, 2021.

A. Dosovitskiy and T. Brox. Generating images with perceptual similarity metrics based on deep networks. *Advances in neural information processing systems*, 29, 2016.

S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid monte carlo. *Physics letters B*, 195(2):216–222, 1987.

J. Dumas, C. Cointe, A. Wehenkel, A. Sutera, X. Fettweis, and B. Cornélusse. A probabilistic forecast-driven strategy for a risk-aware participation in the capacity firming market. *IEEE Transactions on Sustainable Energy*, 2021.

J. Dumas, A. Wehenkel, D. Lanaspeze, B. Cornélusse, and A. Sutera. A deep generative model for probabilistic energy forecasting in power systems: normalizing flows. *Applied Energy*, 305:117871, 2022.

C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios. Neural spline flows. In *Advances in Neural Information Processing Systems*, pages 7509–7520, 2019.

P. Esser, R. Rombach, and B. Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883, 2021.

A. Faragó and G. Lugosi. Strong universal consistency of neural network classifiers. *IEEE Transactions on Information Theory*, 39(4):1146–1151, 1993.

A. Fischer and C. Igel. Bounding the bias of contrastive divergence learning. *Neural computation*, 23(3):664–673, 2011.

R. A. Fisher. On the mathematical foundations of theoretical statistics. *Philosophical transactions of the Royal Society of London. Series A, containing papers of a mathematical or physical character*, 222(594-604):309–368, 1922.

D. Freedman. *Brownian motion and diffusion*. Springer Science & Business Media, 2012.

R. Gao, E. Nijkamp, D. P. Kingma, Z. Xu, A. M. Dai, and Y. N. Wu. Flow contrastive estimation of energy-based models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7518–7528, 2020.

D. Geiger, T. Verma, and J. Pearl. d-separation: From theorems to algorithms. In *Machine Intelligence and Pattern Recognition*, volume 10, pages 139–148. Elsevier, 1990.

M. C. Gemici, D. Rezende, and S. Mohamed. Normalizing flows on riemannian manifolds. *arXiv preprint arXiv:1611.02304*, 2016.

M. Germain, K. Gregor, I. Murray, and H. Larochelle. Made: Masked autoencoder for distribution estimation. In *International conference on machine learning*, pages 881–889. PMLR, 2015.

C. J. Geyer. Practical markov chain monte carlo. *Statistical science*, pages 473–483, 1992.

J. W. Gibbs. *Elementary principles in statistical mechanics: developed with especial reference to the rational foundations of thermodynamics*. C. Scribner's sons, 1902.

W. R. Gilks, S. Richardson, and D. Spiegelhalter. *Markov chain Monte Carlo in practice*. CRC press, 1995.

M. Gillon, A. H. Triaud, B.-O. Demory, E. Jehin, E. Agol, K. M. Deck, S. M. Lederer, J. De Wit, A. Burdanov, J. G. Ingalls, et al. Seven temperate terrestrial planets around the nearby ultracool dwarf star trappist-1. *Nature*, 542(7642):456–460, 2017.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

W. Grathwohl, R. T. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud. FFJORD: Free-form continuous dynamics for scalable reversible generative models. In *International Conference on Machine Learning*, 2018.

A. Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

D. S. Greenberg, M. Nonnenmacher, and J. H. Macke. Automatic posterior transformation for likelihood-free inference. International Conference on Machine Learning(2019): 2404–2414, 2019.

U. Grenander and M. I. Miller. Representations of knowledge in complex systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 56(4):549–581, 1994.

S. Gruber and F. Buettner. Trustworthy deep learning via proper calibration errors: A unifying approach for quantifying the reliability of predictive uncertainty. *arXiv preprint arXiv:2203.07835*, 2022.

C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR, 2017.

M. U. Gutmann and A. Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of machine learning research*, 13(2), 2012.

T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

D. Hendrycks, S. Basart, N. Mu, S. Kadavath, F. Wang, E. Dorundo, R. Desai, T. Zhu, S. Parajuli, M. Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8340–8349, 2021.

J. Hermans, A. Delaunoy, F. Rozet, A. Wehenkel, and G. Louppe. Averting a crisis in simulation-based inference. *arXiv preprint arXiv:2110.06581*, 2021.

G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. 2020.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

C.-W. Huang, D. Krueger, A. Lacoste, and A. Courville. Neural autoregressive flows. In *International Conference on Machine Learning*, pages 2083–2092, 2018.

C.-W. Huang, L. Dinh, and A. Courville. Augmented normalizing flows: Bridging the gap between generative flows and latent variable models. *arXiv preprint arXiv:2002.07101*, 2020.

A. Hyvärinen and P. Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

P. Jaini, K. A. Selby, and Y. Yu. Sum-of-squares polynomial flow. 2019.

D. Kalatzis, J. Z. Ye, J. Wohlert, and S. Hauberg. Multi-chart flows. *arXiv preprint arXiv:2106.03500*, 2021.

M. E. Khan and H. Rue. The bayesian learning rule. *arXiv preprint arXiv:2107.04562*, 2021.

I. Khemakhem, R. P. Monti, R. Leech, and A. HyvÃ€rinen. Causal autoregressive flows. 2020.

J. Kim and J. Pearl. A computational model for causal and diagnostic reasoning in inference systems. In *International Joint Conference on Artificial Intelligence*, pages 0–0, 1983.

R. Kindermann. Markov random fields and their applications. *American mathematical society*, 1980.

D. P. Kingma and P. Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pages 10236–10245, 2018.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations (ICLR)*, 2013.

D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751, 2016.

J. Knoblauch, J. Jewson, and T. Damoulas. Generalized variational inference: Three arguments for deriving new posteriors. *arXiv preprint arXiv:1904.02063*, 2019.

I. Kobyzev, S. Prince, and M. Brubaker. Normalizing flows: An introduction and review of current methods. pages 1–1, 2020-08. URL https://doi.org/10.1109/tpami.2020.2992934. Publisher: Institute of Electrical and Electronics Engineers (IEEE) _eprint: 1908.09257.

F. Koehler, V. Mehta, and A. Risteski. Representational aspects of depth and conditioning in normalizing flows. In *International Conference on Machine Learning*, pages 5628–5636. PMLR, 2021.

J. Köhler, A. Krämer, and F. Noé. Smooth normalizing flows. *Advances in Neural Information Processing Systems*, 34:2796–2809, 2021.

D. Koller and N. Friedman. *Probabilistic graphical models: Principles and techniques*. MIT press, 2009.

A. Kolmogorov. On the representation of continuous functions of several variables by superpositions of continuous functions of lesser variable count. In *Dokl. Akad. Nauk SSSR*, volume 108, 1956.

A. N. Kolmogorov and A. T. Bharucha-Reid. *Foundations of the theory of probability: Second English Edition*. Courier Dover Publications, 2018.

Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*, 2020.

A. Krogh and J. Hertz. A simple weight decay can improve generalization. *Advances in neural information processing systems*, 4, 1991.

M. J. Kusner, B. Paige, and J. M. Hernández-Lobato. Grammar variational autoencoder. In *International conference on machine learning*, pages 1945–1954. PMLR, 2017.

Y. LeCun, Y. Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

P. Leroy, D. Ernst, P. Geurts, G. Louppe, J. Pisane, and M. Sabatelli. Qvmix and qvmix-max: Extending the deep quality-value family of algorithms to cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2012.12062*, 2020.

C. Li. Fisher, wright, and path coefficients. *Biometrics*, pages 471–483, 1968.

G. Louppe. Understanding random forests: From theory to practice. *arXiv preprint arXiv:1407.7502*, 2014.

L. E. Lwakatare, A. Raj, I. Crnkovic, J. Bosch, and H. H. Olsson. Large-scale machine learning systems in real-world industrial settings: A review of challenges and solutions. *Information and software technology*, 127:106368, 2020.

S. Lyu. Interpretation and generalization of score matching. *arXiv preprint arXiv:1205.2629*, 2012.

D. J. MacKay. Bayesian neural networks and density networks. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 354(1):73–80, 1995.

D. J. MacKay et al. Introduction to gaussian processes. *NATO ASI series F computer and systems sciences*, 168:133–166, 1998.

E. Mathieu and M. Nickel. Riemannian continuous normalizing flows. *Advances in Neural Information Processing Systems*, 33:2503–2515, 2020.

R. J. McEliece, D. J. C. MacKay, and J.-F. Cheng. Turbo decoding as an instance of pearl's" belief propagation" algorithm. *IEEE Journal on selected areas in communications*, 16(2):140–152, 1998.

M. Minderer, J. Djolonga, R. Romijnders, F. Hubis, X. Zhai, N. Houlsby, D. Tran, and M. Lucic. Revisiting the calibration of modern neural networks. *Advances in Neural Information Processing Systems*, 34:15682–15694, 2021.

S. L. Morgan and C. Winship. *Counterfactuals and causal inference*. Cambridge University Press, 2015.

J. Mouton and S. Kroon. Graphical residual flows. *arXiv preprint arXiv:2204.11846*, 2022a.

J. Mouton and S. Kroon. Siren-vae: Leveraging flows and amortized inference for bayesian networks. *arXiv preprint arXiv:2204.11847*, 2022b.

R. M. Neal. Slice sampling. *The annals of statistics*, 31(3):705–767, 2003.

R. M. Neal et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.

I. Ng, S. Lachapelle, N. R. Ke, S. Lacoste-Julien, and K. Zhang. On the convergence of continuous constrained optimization for structure learning. In *International Conference on Artificial Intelligence and Statistics*, pages 8176–8198. PMLR, 2022.

A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016a.

A. v. d. Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu. Conditional image generation with pixelcnn decoders. 2016b.

G. Papamakarios, T. Pavlakou, and I. Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pages 2338–2347, 2017.

G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. 2019a.

G. Papamakarios, D. C. Sterratt, and I. Murray. Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. In *22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019b.

G. Parisi. Correlation functions and computer simulations. *Nuclear Physics B*, 180(3): 378–384, 1981.

J. Pearl. Bayesian networks: A model of self-activated memory for evidential reasoning. In *Proceedings of the 7th conference of the Cognitive Science Society, University of California, Irvine, CA, USA*, pages 15–17, 1985.

J. Pearl. Distributed revision of composite beliefs. *Artificial Intelligence*, 33(2):173–215, 1987.

J. Pearl. A probabilistic calculus of actions. In *Uncertainty Proceedings 1994*, pages 454–462. Elsevier, 1994.

J. Pearl. *Causality*. Cambridge university press, 2009.

J. Pearl. Bayesian networks. 2011.

J. Pearl. Reverend bayes on inference engines: A distributed hierarchical approach. In *Probabilistic and Causal Inference: The Works of Judea Pearl*, pages 129–138. 2022.

A. Pesah, A. Wehenkel, and G. Louppe. Recurrent machines for likelihood-free inference. *arXiv preprint arXiv:1811.12932*, 2018.

J. Peters, D. Janzing, and B. Schölkopf. *Elements of causal inference: foundations and learning algorithms*. 2017.

A. Rahimi, A. Shaban, C.-A. Cheng, R. Hartley, and B. Boots. Intra order-preserving functions for calibration of multi-class neural networks. *Advances in Neural Information Processing Systems*, 33:13456–13467, 2020.

A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International conference on machine learning*, pages 4295–4304. PMLR, 2018.

A. Razavi, A. van den Oord, and O. Vinyals. Generating diverse high-resolution images with vq-vae. 2019a.

A. Razavi, A. Van den Oord, and O. Vinyals. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 32, 2019b.

D. Rezende and S. Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538. PMLR, 2015.

D. J. Rezende, G. Papamakarios, S. Racaniere, M. Albergo, G. Kanwar, P. Shanahan, and K. Cranmer. Normalizing flows on tori and spheres. In *International Conference on Machine Learning*, pages 8083–8092. PMLR, 2020.

S. Rissanen, M. Heinonen, and A. Solin. Generative modelling with inverse heat dissipation. *arXiv e-prints*, pages arXiv–2206, 2022.

R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

S. J. Russell. *Artificial intelligence a modern approach*. Pearson Education, Inc., 2010.

C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. Denton, S. K. S. Ghasemipour, B. K. Ayan, S. S. Mahdavi, R. G. Lopes, et al. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*, 2022.

S. Sahoo, C. Lampert, and G. Martius. Learning equations for extrapolation and control. In *International Conference on Machine Learning*, pages 4442–4450. PMLR, 2018.

S. Sanner and E. Abbasnejad. Symbolic variable elimination for discrete and continuous graphical models. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

V. G. Satorras, E. Hoogeboom, and M. Welling. E (n) equivariant graph neural networks. In *International conference on machine learning*, pages 9323–9332. PMLR, 2021.

M. Schmidt and H. Lipson. Distilling free-form natural laws from experimental data. *science*, 324(5923):81–85, 2009.

L. Schwartz. On bayes procedures. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, 4(1):10–26, 1965.

V. Sehwag, A. N. Bhagoji, L. Song, C. Sitawarin, D. Cullina, M. Chiang, and P. Mittal. Analyzing the robustness of open-world machine learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, pages 105–116, 2019.

A. Sinha, J. Song, C. Meng, and S. Ermon. D2c: Diffusion-decoding models for few-shot conditional generation. *Advances in Neural Information Processing Systems*, 34: 12533–12548, 2021.

S. L. Smith, B. Dherin, D. G. Barrett, and S. De. On the origin of implicit regularization in stochastic gradient descent. *arXiv preprint arXiv:2101.12176*, 2021.

J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

Y. Song and S. Ermon. Generative modeling by estimating gradients of the data distribution. In *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems*, 2019.

Y. Song and D. P. Kingma. How to train your energy-based models. *arXiv preprint arXiv:2101.03288*, 2021.

Y. Song, S. Garg, J. Shi, and S. Ermon. Sliced score matching: A scalable approach to density and score estimation. In *Uncertainty in Artificial Intelligence*, pages 574–584. PMLR, 2020a.

Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020b.

Y. Song, C. Durkan, I. Murray, and S. Ermon. Maximum likelihood training of score-based diffusion models. *Advances in Neural Information Processing Systems*, 34:1415–1428, 2021.

D. A. Sorensen, S. Andersen, D. Gianola, and I. Korsgaard. Bayesian inference in threshold models using gibbs sampling. *Genetics Selection Evolution*, 27(3):229–249, 1995.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

E. G. Tabak and C. V. Turner. A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, 66(2):145–164, 2013.

E. G. Tabak and E. Vanden-Eijnden. Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences*, 8(1):217–233, 2010.

Y. W. Teh, M. Welling, S. Osindero, and G. E. Hinton. Energy-based models for sparse overcomplete representations. *Journal of Machine Learning Research*, 4(Dec):1235–1260, 2003.

T. Théate, A. Wehenkel, A. Bolland, G. Louppe, and D. Ernst. Distributional reinforcement learning with unconstrained monotonic neural networks. *arXiv preprint arXiv:2106.03228*, 2021.

T. Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th international conference on Machine learning*, pages 1064–1071, 2008.

S. T. Tokdar and R. E. Kass. Importance sampling: a review. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(1):54–60, 2010.

J. M. Tomczak. Deep generative modeling.

I. Tsamardinos, L. E. Brown, and C. F. Aliferis. The max-min hill-climbing bayesian network structure learning algorithm. *Machine learning*, 65(1):31–78, 2006.

A. Vahdat and J. Kautz. Nvae: A deep hierarchical variational autoencoder. 2020.

A. Vahdat, K. Kreis, and J. Kautz. Score-based generative modeling in latent space. *Advances in Neural Information Processing Systems*, 34:11287–11302, 2021.

A. Van Den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. WaveNet: A generative model for raw audio. In *9th ISCA Speech Synthesis Workshop*, pages 125–125, 2016.

A. Van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, et al. Conditional image generation with pixelcnn decoders. *Advances in neural information processing systems*, 29, 2016.

A. Van Den Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. Driessche, E. Lockhart, L. Cobo, F. Stimberg, and others. Parallel WaveNet: Fast high-fidelity speech synthesis. In *International Conference on Machine Learning*, pages 3915–3923, 2018.

A. Van Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. In *International conference on machine learning*, pages 1747–1756. PMLR, 2016.

M. Vandegar, M. Kagan, A. Wehenkel, and G. Louppe. Neural empirical bayes: Source distribution estimation and its applications to simulation-based inference. In *International Conference on Artificial Intelligence and Statistics*, pages 2107–2115. PMLR, 2021.

N. Vecoven, D. Ernst, A. Wehenkel, and G. Drion. Introducing neuromodulation in deep neural networks to learn adaptive behaviours. *PloS one*, 15(1):e0227922, 2020.

P. Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

A. Virmaux and K. Scaman. Lipschitz regularity of deep neural networks: analysis and efficient estimation. *Advances in Neural Information Processing Systems*, 31, 2018.

U. von Luxburg and O. Bousquet. Distance-based classification with lipschitz functions. *Journal of Machine Learning Research*, 5(Jun):669–695, 2004.

M. J. Vowels, N. C. Camgoz, and R. Bowden. D'ya like dags? a survey on structure learning and causal discovery. *ACM Computing Surveys (CSUR)*, 2021.

A. Wehenkel and G. Louppe. Unconstrained monotonic neural networks. In *Advances in Neural Information Processing Systems*, pages 1543–1553, 2019.

A. Wehenkel and G. Louppe. You say normalizing flows i see bayesian networks. 2020.

A. Wehenkel and G. Louppe. Diffusion priors in variational autoencoders. In *ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*, 6 2021a.

A. Wehenkel and G. Louppe. Graphical normalizing flows. In *International Conference on Artificial Intelligence and Statistics*, pages 37–45. PMLR, 04 2021b.

A. Wehenkel, A. Mukhopadhyay, J.-Y. Le Boudec, and M. Paolone. Parameter estimation of three-phase untransposed short transmission lines from synchrophasor measurements. *IEEE Transactions on Instrumentation and Measurement*, 69(9):6143–6154, 2020.

A. Wehenkel, J. Behrmann, H. Hsu, G. Sapiro, G. Louppe, and J.-H. Jacobsen. Robust hybrid learning with expert augmentation. *arXiv preprint arXiv:2202.03881*, 2022.

L. A. Wehenkel. *Automatic learning techniques in power systems*. Number 429. Springer Science & Business Media, 1998.

M. Welling and Y. W. Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688. Citeseer, 2011.

T.-W. Weng, H. Zhang, P.-Y. Chen, J. Yi, D. Su, Y. Gao, C.-J. Hsieh, and L. Daniel. Evaluating the robustness of neural networks: An extreme value theory approach. *arXiv preprint arXiv:1801.10578*, 2018.

A. V. Werhli and D. Husmeier. Reconstructing gene regulatory networks with bayesian networks by combining expression data with multiple sources of prior knowledge. *Statistical applications in genetics and molecular biology*, 6(1), 2007.

N. Wermuth. Linear recursive equations, covariance selection, and path analysis. *Journal of the American Statistical Association*, 75(372):963–972, 1980.

N. Wiener and P. Masani. The prediction theory of multivariate stochastic processes. *Acta Mathematica*, 98(1):111–150, 1957.

C. S. Wong and W. K. Li. On a mixture autoregressive model. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 62(1):95–115, 2000.

S. Wright. Systems of mating. i. the biometric relations between parent and offspring. *Genetics*, 6(2):111, 1921.

S. Wright. The method of path coefficients. *The annals of mathematical statistics*, 5(3): 161–215, 1934.

S. Wright. Path coefficients and path regressions: alternative or complementary concepts? *Biometrics*, 16(2):189–202, 1960.

K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

J. Yu, X. Li, J. Y. Koh, H. Zhang, R. Pang, J. Qin, A. Ku, Y. Xu, J. Baldridge, and Y. Wu. Vector-quantized image modeling with improved vqgan. *arXiv preprint arXiv:2110.04627*, 2021.

P. Yu, S. Xie, X. Ma, B. Jia, B. Pang, R. Gao, Y. Zhu, S.-C. Zhu, and Y. N. Wu. Latent diffusion energy-based model for interpretable text modeling. *arXiv preprint arXiv:2206.05895*, 2022.

D. Yurk and Y. Abu-Mostafa. County-specific, real-time projection of the effect of business closures on the covid-19 pandemic. *medRxiv*, 2021.

R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

X. Zheng, B. Aragam, P. K. Ravikumar, and E. P. Xing. Dags with no tears: Continuous optimization for structure learning. *Advances in Neural Information Processing Systems*, 31, 2018.