Université de Montréal

Probability Flows in Deep Learning

par Chin-Wei Huang

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Thèse présentée à la Faculté des arts et des sciences
en vue de l'obtention du grade de Philosophiæ Doctor (Ph.D.)
en informatique

October, 2023

© Chin-Wei Huang, 2023.

Université de Montréal
Faculté des arts et des sciences

Cette thèse intitulée:

Probability Flows in Deep Learning

présentée par:

Chin-Wei Huang

a été évaluée par un jury composé des personnes suivantes:

Ioannis Mitliagkas,	président-rapporteur
Aaron Courville,	directeur de recherche
Pierre-Luc Bacon,	membre du jury
David Duvenaud,	examinateur externe

Thèse acceptée le: . . . . . .

To my beloved mom,
to whom I owe an eternal debt of gratitude.

Résumé

Les modèles génératifs basés sur la vraisemblance sont des éléments fondamentaux pour la modélisation statistique des données structurées. Ils peuvent être utilisés pour synthétiser des échantillons de données réalistes, et la fonction de vraisemblance peut être utilisée pour comparer les modèles et déduire diverses quantités statistiques. Cependant, le défi réside dans le développement de modèles capables de saisir avec précision les schémas statistiques présentés dans la distribution des données. Les modèles existants rencontrent souvent des limitations en termes de flexibilité représentationnelle et d'évolutivité computationnelle en raison du choix de la paramétrisation, freinant ainsi la progression vers cet idéal.

Cette thèse présente une exploration systématique des structures appropriées qui peuvent être exploitées pour concevoir des modèles génératifs basés sur la vraisemblance, allant des architectures spécialisées telles que les applications triangulaires et les applications de potentiel convexes aux systèmes dynamiques paramétriques tels que les équations différentielles neuronales qui présentent des contraintes minimales en termes de paramétrisation. Les modèles proposés sont fondés sur des motivations théoriques et sont analysés à travers le prisme du changement de variable associé au processus de génération de données. Cette perspective permet de considérer ces modèles comme des formes distinctes de *probability flows*, unifiant ainsi des classes apparemment non liées de modèles génératifs basés sur la vraisemblance. De plus, des conceptions algorithmiques pratiques sont introduites pour calculer, approximer ou estimer les quantités nécessaires pour l'apprentissage et l'évaluation.

Il est prévu que cette thèse suscite l'intérêt des communautés de modélisation générative et d'apprentissage automatique Bayésien/probabiliste, et qu'elle serve de ressource précieuse et d'inspiration pour les chercheurs et les praticiens du domaine.

Mots-clés : modèles génératifs, estimation de densité, normalizing flow, autoencodeur variationnel, modèle autorégressif, modèle basé sur l'énergie, modèle de diffusion, score matching.

Abstract

Likelihood-based generative models are fundamental building blocks for statistical modeling of structured data. They can be used to synthesize realistic data samples, and the likelihood function can be used for comparing models and inferring various statistical quantities. However, the challenge lies in developing models capable of accurately capturing the statistical patterns presented in the data distribution. Existing models often face limitations in representational flexibility and computational scalability due to the choice of parameterization, impeding progress towards this ideal.

This thesis presents a systematic exploration of suitable structures that can be exploited to design likelihood-based generative models, spanning from specialized architectures like triangular maps and convex potential maps to parametric dynamical systems such as neural differential equations that bear minimal parameterization restrictions. The proposed models are rooted in theoretical foundations and analyzed through the lens of the associated change of variable in the data generation process. This perspective allows for viewing these models as distinct forms of *probability flows*, thereby unifying seemingly unrelated classes of likelihood-based generative models. Moreover, practical algorithmic designs are introduced to compute, approximate, or estimate necessary quantities for training and testing purposes.

It is anticipated that this thesis would be of interest to the generative modeling and Bayesian/probabilistic machine learning communities, and will serve as a valuable resource and inspiration for both researchers and practitioners in the field.

Keywords: generative models, density estimation, normalizing flow, variational autoencoder, autoregressive model, energy-based model, diffusion model, score matching.

Contents

List of figures

List of Tables

Acknowledgements

A story isn't great without some struggles. Back in 2016, my journey in postgraduate education began when I enrolled in a PhD program in Mechanical engineering at Polytechnique Montréal under the supervision of Sofiane Achiche. After completing a few months of coursework in probabilistic graphical models taught by Simon Lacost-Julien, my passion for machine learning grew to a point that I could no longer ignore it. Thanks to the encouragement of Sheng-Chang Lin, I made the decision to drop out and switch my career path. I'd like to express my gratitude to Sofiane for being supportive and understanding throughout the transition.

Months later, I had received many rejections, likely due to my lack of research experience and non-computer science background. Without a plan, I found myself spending my first winter in a foreign country without many friends. However, just a week before Christmas, I received an email from Laurent Charlin, who later offered me an internship at MILA. It was during this internship that I met many colleagues and close friends. I vividly remember many of our conversations at Pavillon André-Aisenstadt and the cantine of HEC. For the first time in awhile, I found myself among a collective of kindred spirits who shared common research interests. I owe Laurent Charlin a huge thank you for opening the door to this community and welcoming me into the MILA family.

The initial excitement of joining MILA quickly gave way to a period of uncertainty as I grappled with the reality of my temporary status as an intern. I felt immense pressure to prove myself in order to secure a more permanent position. However, the project wasn't making sufficient progress to justify publication. I found myself desperately trying to demonstrate my worth through contributions to the project and active participation in my coursework. It was during these moments of great uncertainty that Aaron extended an offer to supervise me for a Masters program (along with Laurent), who later became my PhD advisor. I owe Aaron a debt of gratitude for taking a chance on me, shaping my research approach, and providing various opportunities and ample space to explore and grow as a researcher.

During my PhD, I also had the opportunity to work with Alexandre Lacoste to intern with him at ElementAI (now a research division of ServiceNow). I want to express my gratitude to Alex, for his approachable and easy-going nature, which made working together a pleasure, and for providing me an utmost amount of freedom and resources. During my short stay at Google as a research student, I had the opportunity to work with Dilip Krishnan, Tyler Zhu, and Laurent Dinh. I'd like to thank them for being great hosts and mentors. I'm also grateful to Google for awarding me a PhD fellowship, which not only provided essential financial support during my academic journey but also served as a validation of my efforts.

I would also like to thank all of my collaborators throughout my PhD (in chrological order) — Ahmed Touati, Laurent Dinh, Michal Drozdzal, Mohammad Havaei, David Krueger, Riashat Islam, Ryan Turner, Yikang Shen, Zhouhan Lin, Shawn Tan, Faruk Ahmed, Kundan Kumar, Fan-Yun Sun, Kris Sankaran, Eeshan Dhekane, Tianshi Cao, David Hui, Joseph Paul Cohen, Pascal Vincent, Gintare Karolina Dziugaite, Jae Hyun Lim, Christopher Pal, Brady Neal, Sunand Raghupathi, Christos Tsirigotis, Ricky Chen, Milad Aghajohari, Joey Bose, and Prakash Panangaden. I'd like to give a special thank you to Faruk Ahemd and Joey Bose for giving feedback on some parts of this thesis, and to Samuel Lavoie for proofreading le résumé.

Thank you to friends at MILA who helped make my stay in Montréal memorable, notably Amina Madzhun, Samuel Lavoie, Salem Lahlou, Michael Noukhovitch, Christopher Beckham, Sai Rajeswar, Disha Shrivastava, Gabriele Prato, Andre Cianflone, Ankesh Anand, Bogdan Mazoure, Sébastien Lachapelle, Massimo Caccia, Lucas Caccia, Aristide Baratin, Breandan Considine, Rosemary Ke, Alex Lamb, Alexandre Piché, Alex Fedorov, Evan Racah, Le Caravane Café, Dobe & Andy, and many more whom I may not have listed here.

I'd like to thank my dear friends from Taiwan, who have been nothing less than supportive from the other side of the globe — Pin-I Yu, Wei-Chieh Hsu, and Chia-Yun Chung.

Finally, I want to thank myself, for being fearless in the face of adversity and for following my heart even if it meant I had to take the riskier route.

1 Introduction and background

Generative models play an essential role in machine learning by allowing us to explore and analyze variations in random events that may not be fully explained by available information [Russell and Norvig, 2002]. For example, when provided with text prompts, generative models have the capability to generate a diverse range of images that match the given description [Radford et al., 2021, Ramesh et al., 2021, 2022, Saharia et al., 2022]. In addition, they can probabilistically infer molecular conformations based on intramolecular distances [Topel and Ferguson, 2020], or generate molecules with desired chemical properties [Jin et al., 2018, Xie et al., 2021]. Recently, there has been a surge of interest in applying deep neural networks to model these random events. The neural networks serve as feature extractors, effectively transforming raw data streams into more abstract representations, or as statistical models that can translate the higher-level representations back into the raw data format. These two procedures, known as *inference* and *reconstruction*, are dual transformations of the description of a random observation.

This thesis seeks to provide a unifying view on various types of transformations that can be used as fundamental building blocks in the context of generative modeling. Through this unifying lens, we seek to tackle the common challenges in designing these models. Some of these challenges are **computational** (*e.g.* the computational cost of training the model or generating samples from the model), some are **representational** (*e.g.* whether the "compact" representation of the joint distribution of the variables describing the random events is expressive).

Having an expressive representation of the joint distribution is of paramount importance to general-purpose statistical modeling of structured data. This is in contrast to tradi-

tional ways of modeling in scientific fields such as biology and physics, where domain experts seek to find out the numerical value of some interpretable parameters of a model used to describe the physics of the observations. Oftentimes, we are not faced with a simple system that allows us to make strong assumptions about the data-generating process. Should these erroneous assumptions be made, they would lead to model misspecification, which is a cause of the lack of realism in the generated samples drawn from the model that we hope would resemble the real data.

A natural price to pay for having a highly expressive model is computation. For simpler, less expressive models, most quantities of interest, such as the likelihood of the observations (which is needed for likelihood-based training and model evaluation), can usually be readily computed. However, this is often not the case for model classes that enjoy a greater degree of flexibility. For more complex models, *intractability* is often an issue, which means the computation of certain quantities of interest cannot be carried out within a reasonable amount of time. This necessitates the design of algorithms for estimating or approximating the quantities of interest, at the cost of some tolerable error. We refer to this tension between the two aspects as the *expressivity-scalability tradeoff*, and we aim to design better statistical models in terms of both.

This thesis undertakes a comprehensive investigation of likelihood-based generative models, encompassing specialized structures like triangular and convex potential maps, alongside parametric dynamical systems such as neural ordinary and stochastic differential equations. A key contribution lies in our theoretical exploration, focusing on variable transformations within the data generation process, which unifies these models as distinct probability flows. Additionally, we introduce practical algorithms for efficient computation and estimation of essential quantities, addressing the expressivity-scalability tradeoff. The primary novelty of this thesis is a unified view that bridges disparate generative model classes, offering both theoretical insights and practical solutions to enhance expressiveness while managing computational complexity in generative modeling.

For the rest of the chapter, we will first review the formalism of learning (Section 1.1) and some practical aspects of deep learning (Section 1.2), which will be used as a

black box for function approximation in the later chapters. We will give an overview of likelihood-based generative models in Section 1.3, and come back to the thesis question in Section 1.4.

1.1 FORMALISM OF LEARNING

The objective of learning is to generalize from experience. It is usually assumed a learner has access to finite experience, in the form of some training data that follows an unknown probability distribution. A learner's goal is to make sense of the data by capturing the statistical regularity exhibited by the data when it comes in large numbers.

Formally, denote by $\mathcal{X}$ the data space and $q(x)$ the probability density function of an unknown data distribution, which is something that we would like to approximate. We are given a training set of n data points assumed to be distributed *independently and identically* (*i.i.d.*) from the data distribution, written as $(x_i)_{i=1}^n \sim q(x)^n$. We assume the true data distribution lies within, or close to, a family of density models $\{p_\pi(x) : \pi \in \mathfrak{P}(\mathcal{X})\}$, where $\mathfrak{P}(\mathcal{X})$ is the set of parameters[1] that can be used to sufficiently describe the density function whose domain is $\mathcal{X}$. The goal of learning, in the context of generative modeling, is to select the best candidate $p_{\pi^*}(x)$ from $\{p_\pi(x) : \pi \in \mathfrak{P}(\mathcal{X})\}$ to approximate the data distribution $q(x)$.

Models that admit an explicit form of density that can either be readily computed or approximated are called explicit-density models. This will be the main focus of the thesis. Another family of models that do not admit an explicit density function are called implicit density models, the most famous example being the *Generative Adversarial Networks*, or GANs [Goodfellow et al., 2014]. We do not discuss the formalism of learning for implicit models and refer the readers interested in this topic to Goodfellow [2016], Mohamed and Lakshminarayanan [2016].

[1] We sometimes also use θ to denote the model's parameters.

Applications of explicit-density models abound. More broadly speaking, generative models can be used to augment human creativity [Jordan, 2019]. The data sampler can be used to edit high-dimensional data such as images [Kingma and Dhariwal, 2018], encode the data in a lossy manner [Chen et al., 2016b] and represenent the data with disentangled and interpretable features [Chen et al., 2016a]. It can also be used to tackle inverse problems in digital processing, such as denoising, super-resolution, inpainting, and conditional generation [Asim et al., 2020, Lugmayr et al., 2020, Whang et al., 2020, Song et al., 2021c, Kawar et al., 2021]. A more interactive interface can also be made possible for humans to be involved in guiding the image synthesis [Meng et al., 2021].

Having an explicit likelihood function also has many benefits. For example, likelihood can be used as a measure of an agent's uncertainty about the surroundings, which can then be used to incentivize exploration [Ostrovski et al., 2017]. The learned density model can be used to design distribution-dependent compression algorithms [Kingma et al., 2019, Ho et al., 2019b, Hoogeboom et al., 2019], since the cross-entropy score, *i.e.* the negative average log-likelihood, can be viewed as an upper bound on the optimal code length [Cover, 1999]. Moreover, in numerous fields, including physics, economics, climate science, computational chemistry, etc., likelihood-based models can be used to enhance simulation-based inference [Cranmer et al., 2020], where typically, a simulator is used to enable the prediction of the behavior of a complex system that might not be well-suited for certain statistical inference tasks. Last but not least, learning a probabilistic module is essential in probabilistic programming, which automatically performs inference as a fundamental tool in probabilistic machine learning [van de Meent et al., 2018].

Many learning frameworks, or parameter estimation principles (for obtaining π^*), exist for models with an explicit density, that utilize the density function itself, or the density up to some normalizing constant. A few will be presented throughout the rest of the section.

Maximum Likelihood Estimation Maximum likelihood estimation finds the parameters that maximize the chance of the data being generated by the model. Namely,

$$\hat{\pi} := \arg\max_{\pi \in \mathfrak{P}(\mathcal{X})} \left\{ \sum_{i=1}^{n} \log p_\pi(x_i) \right\} = \arg\max_{\pi \in \mathfrak{P}(\mathcal{X})} \mathbb{E}_x[\log p_\pi(x)], \tag{1.1}$$

where the latter expectation is over the empirical distribution $\hat{q}(x) := \frac{1}{n} \sum_{i=1}^{n} \delta(x - x_i)$. The Dirac delta function δ is a measure that equals 1 if and only if its argument contains the singleton $\{0\}$. $\hat{\pi}$ is known as the maximum likelihood estimate (MLE) for the parameter π. We refer to models trained with the maximum likelihood principle as *likelihood-based models*. We dedicate Section 1.3 to a more detailed account of different types of likelihood-based models.

Maximum likelihood estimators enjoy many desirable properties, the most prominent one being *consistency*. This essentially means that, given infinite training data, $\hat{\pi}$ can get arbitrarily close to the optimal parameters π^* that perfectly describe the density function of the data distribution $p_{\pi^*}(x) = q(x)$ provided $q(x)$ is within the model class.

Traditionally, the inaccuracy of the model $q(x) \not\approx p_{\hat{\pi}}(x)$ comes from two sources: ① the estimation error (variance), and ② the approximation error (bias). This is also commonly known as the *bias-variance tradeoff* in learning. The estimation error comes from the empirical distribution $\hat{q}$, which is constituted of the randomly drawn training data, and the error can be high when we do not have enough data. Caused by model misspecification (*i.e.* if $\{p_\pi(x) : \pi \in \mathfrak{P}(\mathcal{X})\}$ does not contain the data distribution), the approximation error refers to the model's inability to approximate the true underlying data distribution. Normally, the approximation error could be reduced by considering a more flexible family of models, but it could also lead to an increase in the estimation error since the model would be more capable of over-fitting the training data. The art of learning lies in finding the sweet spot somewhere in between to minimize the overall approximation error.

Divergence Minimization Maximizing the likelihood of the data is equivalent to minimizing the forward Kullback-Leibler (KL) divergence between the (empirical) data distribution and the model distribution, assuming we can sample directly from the data distribution q:

$$\begin{aligned}\arg\max_p \mathbb{E}_{x\sim q}[\log p(x)] &= \arg\max_p -\mathbb{E}_{x\sim q}[\log q(x)] + \mathbb{E}_{x\sim q}[\log p(x)] \\ &= \arg\min_p D_{\mathrm{KL}}(q||p),\end{aligned}$$

since the entropy term $-\mathbb{E}_{x\sim q}[\log q(x)]$ is a constant wrt the model's parameters π.

General statistical divergences can be used like a distance function[2], since it can tell us if two distributions are identical; the divergence from one distribution to another is zero if and only if the two distributions are identical. There are other divergences other than KL one can potentially choose to minimize. For example, an f-divergence is characterized by a convex function f with $f(1) = 0$, defined as

$$D_f(q||p) = \int f\left(\frac{q(x)}{p(x)}\right) p(x)\,\mathrm{d}x. \tag{1.2}$$

For example, with $f(t) = t\log t$, we recover the KL divergence. $f(t) = -\log t$ corresponds to the reverse[3] KL divergence $D_{\mathrm{KL}}(p||q)$. If $f(t) = \frac{1}{2}|t-1|$, we have the *total variation distance*; see Appendix B.1 for some background on total variation. Furthermore, f-divergence is invariant to reparameterization (Proposition 36), non-negative, and equal to zero if and only if $p = q$.

Score matching Another learning principle is via matching the score function of the model with the score function of the data distribution. Score matching is convenient when the density of the model is defined up to a normalizing constant. Within the

[2]However, a divergence is not always a *metric*, since it might not satisfy the triangle inequality, nor symmetry, both of which are axioms of a metric.

[3]To clarify, q is the target distribution here, whereas p is the distribution we want to optimize.

context of generative modeling, a score is defined as the gradient of the log-density function wrt the data vector, *i.e.* $\nabla_x \log q(x)$. For simplicity, we sometimes suppress the subscript x when it is clear in the context wrt which variable we are taking the gradient.

Let Λ be a positive-definite matrix. Then it can be used to define a norm $\|x\|_\Lambda := x^\top \Lambda x$. The Λ-explicit score matching (ESM) loss, or just the score matching loss for short, is

$$\mathcal{L}_{\text{ESM}} = \mathbb{E}_{x\sim q}\left[\frac{1}{2}||\mathbf{s}_\theta(x) - \nabla \log q(x)||_\Lambda^2\right], \tag{1.3}$$

where $\mathbf{s}_\theta$ is a parametric score function, which can be taken to be the gradient of a parametric density function if the latter is the trainable degree of freedom, *i.e.* $\mathbf{s}_\theta = \nabla \log p_\theta$. Score matching can also be useful for training energy-based models, where the density is defined up to a normalizing constant via an energy function (see Section 1.3). Or even more loosely, we can directly parameterize a score function, without restricting it to be the gradient field of an energy potential.

The ESM loss is in fact also known as the *Fisher divergence*. But this divergence cannot be directly minimized as we do not have access to the ground-truth score $\nabla \log q$; otherwise score matching is simply a regression problem. A few alternative losses can be used, which are all equal to one another up to a constant, including implicit score matching [Hyvärinen and Dayan, 2005, ISM], sliced score matching [Song et al., 2020, SSM], and denoising score matching [Vincent, 2011, DSM]. The losses are summarized in Table 1.1, and are related through the following identity (see Section A.1 for the derivation):

$$\mathcal{L}_{\text{ESM}} - \frac{1}{2}\mathcal{I}(q(x)) = \mathcal{L}_{\text{ISM}} = \mathcal{L}_{\text{SSM}} = \mathcal{L}_{\text{DSM}} - \frac{1}{2}\mathbb{E}_{x_0}[\mathcal{I}(q(x \mid x_0))], \tag{1.4}$$

where $\mathcal{I}(q) = \mathbb{E}[||\nabla \log q||_\Lambda^2]$ is a constant. It is worth noting that they are all equal to the ESM loss up to some constant (wrt the model $\mathbf{s}_\theta$), and can all be computed or estimated without knowing $\nabla \log q$.

Method	Loss
$\mathcal{L}_{\text{ESM}}$	$\frac{1}{2}\mathbb{E}[\|\|\boldsymbol{s}_\theta(x) - \nabla \log q(x)\|\|^2_\Lambda]$
$\mathcal{L}_{\text{ISM}}$	$\mathbb{E}[\frac{1}{2}\|\|\boldsymbol{s}_\theta(x)\|\|^2_\Lambda + \nabla \cdot (\Lambda^\top \boldsymbol{s}_\theta)]$
$\mathcal{L}_{\text{SSM}}$	$\mathbb{E}[\frac{1}{2}\|\|\boldsymbol{s}_\theta(x)\|\|^2_\Lambda + v^\top \nabla (\Lambda^\top \boldsymbol{s}_\theta) v]$
$\mathcal{L}_{\text{DSM}}$	$\frac{1}{2}\mathbb{E}[\|\|\boldsymbol{s}_\theta(x) - \nabla \log q(x \mid x_0)\|\|^2_\Lambda]$

Table 1.1: Score matching losses. v follows the Rademacher distribution. For the DSM loss, $x, x_0 \sim q(x, x_0)$ live in the same probability space, and $q(x)$ is the marginal distribution. It is usually assumed $q(x \mid x_0)$ takes a simple form such as a Gaussian distribution, in which case the loss becomes the mean squared error for training a denoising autoencoder. See Vincent [2011] or the appendix (section A.1) for more details.

1.2 DEEP LEARNING

Learning with deep neural networks allows one to progressively extract higher levels of features from the raw input, such that the features can represent more abstract and compact information useful for the underlying application. To put it in a somewhat reductionistic, but perhaps pragmatic manner, neural nets can be seen as a black-box function approximator that can be used to approximate any mapping of interest to arbitrary precision. In this section, we review some basic feedforward architectures used as deep neural nets and optimization methods used for training. For a more comprehensive review, we refer the readers to [Goodfellow et al., 2016].

Feed-forward Networks The simplest form of deep neural networks has a feed-forward structure, one that does not form a loop for the output to be fed back as an input. Let $\mathcal{X}$ and $\mathcal{Y}$ be the spaces of the input data and the output. A feed-forward network is a function $f : \mathcal{X} \to \mathcal{Y}$ of the form

$$f = f_L \circ f_{L-1} \circ \cdots \circ f_1, \tag{1.5}$$

where f_l is a layer of the network. Each intermediate (or **hidden**) layer is normally composed of a linear transformation $L_l(x)$ and an elementwise non-linear activation

$\sigma_l(x)$. These non-linear transformations coupled with the linear mappings allow the network to represent complex functions of the input vector. Usually, a single activation function is used repeatedly across the entire network, denoted by σ, except for the last layer, which can be parameterized by simply a linear function (*e.g.* for regression) or be activated by a nonlinear function that constrains the range of the output (*e.g.* for classification, where one would normally use the softmax activation).

Multilayer perceptrons (MLPs) are the vanila neural networks for which $L_l(x) = W_l x + b_l$ is simply a dot product with a weight matrix W_l which is dense. Since the nodes between f_l and f_{l-1} are fully connected, MLPs are also known as the fully connected networks, and each individual layer as a dense layer. The bias term b_l allows for the shifting of the entire curve prior to activation, which is not achievable through the weight alone.

Although MLPs are powerful (as we shall see in the next section) and commonly adopted where no prior knowledge on the input space is known, it does not take into account the characteristics of the domain of the input data.

This leads us to the second class of feed-forward networks called the **convolutional neural networks** (CNNs, or ConvNets, LeCun et al. [1989]). ConvNets are an example of imposing a strong inductive bias for image data in the architectural design of the model that is inspired by neuroscience. It was discovered by Hubel and Wiesel [1959] that some mammals' visual system contained neurons that responded to particular orientations of shapes. If we want to represent elementary concepts of shapes and orientations in our network design, then one way to do it is to measure the similarity between an adaptable template which can be thought of as a lookup table that has a similar topology as the data and the data itself. Mathematically, we define a convolutional layer[4] using the convolution operator $*$ to be

$$C(X)_{i,j} = (X * K)_{i,j} = \sum_m \sum_n X_{i+m-1,j+n-1} K_{m,n},$$

[4]This is technically a *valid* 2D *cross-correlation*, but we refer to them as convolutions, following the convention of some machine learning libraries.

where the indices for the summations m, n range from 1 to the height and width of the convolutional kernel K (usually smaller than the input X). Intuitively, the output value of the convolutional layer will have a large positive value if locally the input is well aligned with the pattern of the kernel. Through training, the kernels can be adapted to recognize edges and lines. Through composition, these kernels can then be used to recognize more complex objects while the *receptive field*, i.e. the "visible" region of the input data X that can be seen by a particular output neuron, gets bigger. Notably, convolutions are *translation-equivariant*, which means if we move the object in the input, its representation output will move by the same amount but its value will remain unchanged.

Universal Approximation Theorem Neural networks are powerful tools for approximating functions. There are many forms of theorems developed to describe the approximating capabilities of neural networks, of which we review a classical one concerning the asymptotic universality of MLPs.

To start with, we need some notion of distance so that we can talk about whether functions are close to each other. Formally, we let $(\mathcal{F}, d)$ be a *metric space*, where $\mathcal{F}$ is the family of functions that we would like to approximate, and d is a distance function, or a metric, that measures the distance between a pair of elements in $\mathcal{F}$, i.e.

$$d : \mathcal{F} \times \mathcal{F} \to [0, \infty).$$

Fixing $f \in \mathcal{F}$, we want to know if for any error tolerance $\epsilon > 0$, there exists an MLP f' of a certain form that satisfies $d(f, f') < \epsilon$. If the neural net f' lies in the family $\mathcal{F}$, then we can say that there exists a sequence of neural networks that converges to the limit f. This is because for any sequence of decreasing ϵ_n that goes to 0, we can find a sequence of neural networks f_n with $d(f, f_n) < \epsilon_n$. Hence $\lim_n d(f, f_n) < \lim_n \epsilon_n = 0$. If this is true for any $f \in \mathcal{F}$, then we say the set of neural networks is *dense* in $\mathcal{F}$. If a subset is dense in a set, then every point in the set is either a point in the subset or a limiting

point of the subset, which means it can be approximated arbitrarily well by the elements in the subset.

We consider the set of continuous functions on $[0,1]^d$, denoted by $C([0,1]^d)$, and the metric induced by the the uniform norm $\|f(x)\| = \sup_{x\in[0,1]^d} |f(x)|$:

$$d(f,g) = \|f(x) - g(x)\| .$$

The following theorem due to Cybenko [1989] establishes the universality of 1-hidden layer MLP in the space of continuous functions on a compact domain.

Theorem 1 (Universal function approximation, UFA). *Let σ be a continuous sigmoidal activation function; i.e. $\sigma(x) \to 1$ as $x \to \infty$ and $\sigma(x) \to 0$ as $x \to -\infty$. Neural networks $f'(x)$ of the form*

$$f'(x) = \sum_{j=1}^{N} u_j \sigma(w_j^\top x + b_j)$$

are dense in $C([0,1]^d)$.

Training Deep Networks Learning can be formulated as an optimization problem, such as the maximum likelihood principle. For simple models, the optimization problem can be solved analytically, and the MLE can be expressed in closed form as a function of the training data. However, in more general cases when the model is given enough capacity to describe a wider family of distributions, this is no longer the case. Therefore, training of deep models necessitates iterative optimization algorithms such as *gradient descent*.

Gradient descent works as follows. Let $l : \Theta \to \mathbb{R}$ be a loss function that we want to minimize, where Θ is the space of parameters. Then starting from θ, the loss function decreases fastest when taking the negative gradient direction $-\nabla l(\theta)$, which can be

computed efficiently using the back-propagation algorithm [Rumelhart et al., 1985], which is also known as the reverse-mode automatic differentiation [Baydin et al., 2018]. When the step size coefficient $\eta > 0$ is small enough, the update rule

$$\theta_{k+1} = \theta_k - \eta \nabla l(\theta_k)$$

is guaranteed to decrease.

For most large-scale problems, however, even gradient descent might not be an option as it scales progressively with the size of the dataset. Commonly, the loss function can be written as an average, *e.g.*

$$l(\theta) = \frac{1}{n} \sum_{i=1}^{n} l_i(\theta).$$

In this case, the computational cost and memory both scale $\mathcal{O}(n)$.

One way to address this scalability issue is to have a constant-cost random estimate of the gradient. For example, the gradient of a uniformly drawn l_i is an unbiased estimate of the actual (full-batch) gradient. A mid-point between the full-batch gradient and the one-sample estimate is the mini-batch gradient estimator, which is

$$\frac{1}{B} \sum_{j=1}^{B} \nabla l_{k_j}(\theta_k),$$

where k_j is a sequence of random integers drawn uniformly between 1 and n. This is known as the (minibatch) *stochastic gradient descent* (SGD) algorithm. With a fixed batch size B, it has a constant memory and compute cost, unlike the full-batch gradient, which allows it to scale to larger-scale problems. Generally, SGD can also be applied to cases where the loss function can be written as an expectation, *i.e.* $l(\theta) = \mathbb{E}_\xi[l(\theta, \xi)]$, where $l(\cdot, \cdot)$ depends on the parameters θ and a random noise ξ. Then under some mild assumption on $l(\cdot, \cdot)$, we can push the gradient through the expectation to obtain $\nabla \mathbb{E}_\xi[l(\theta, \xi)] = \mathbb{E}_\xi[\nabla l(\theta, \xi)]$. With this observation, we can use $\nabla l(\theta, \xi)$ with random ξ as an unbiased estimate of the gradient. With some conditions on the learning rate

and the loss function, SGD is guaranteed to converge to a local minimum [Robbins and Monro, 1951, Bottou, 1998]. See Bottou et al. [2018] for more discussion.

The step size η plays an important role in the convergence speed of SGD (or even just full-batch gradient descent): if it is too big, the algorithm might overshoot; if it is too small, the algorithm may barely make any progress. A practical remedy for this is to use an adaptive learning rate and momentum, including some popular optimizers such as *RMSProp* [Tieleman et al., 2012] and *Adam* [Kingma and Ba, 2015].

Finally, an iterative optimization algorithm also has the additional benefit of controlling how the capacity of the network grows, since in practice the stopping criterion of the optimization algorithm is normally based on the evaluation on a hold-out (validation) set. At initialization, the parametric function tends to be smoother, and its *effective complexity* is limited by the number of updates we are allowed to make in order to adapt its parameters. This can be thought of as a form of inductive bias for simpler and smoother functions. The training algorithm terminates when the function starts to overfit and it exhibits a drop in performance on the validation set. This procedure is known as *early stopping* [Wang et al., 1993].

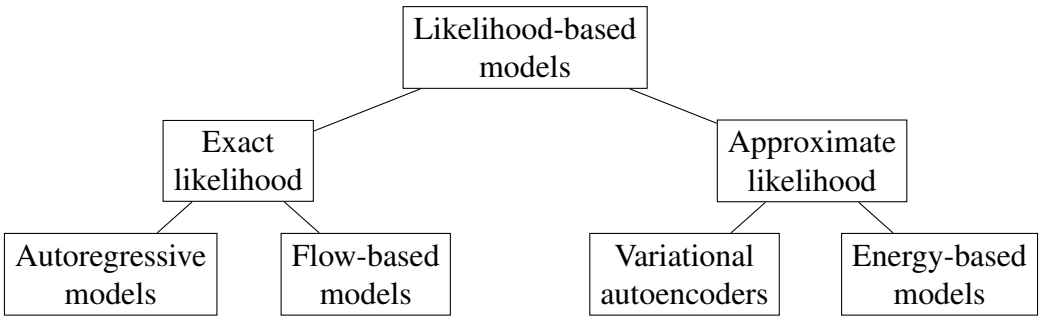


Figure 1.1: A taxonomy of likelihood-based deep generative models.

1.3 LIKELIHOOD-BASED MODELS

Likelihood-based models are trained by maximizing the likelihood of the data $\{x_i : i \in [n]\}$, which are *i.i.d.* realizations of a random variable x. With slight abuse of notation, we let q (resp. p_θ) denote the probability mass function (pmf) of the data distribution (resp. the model) if it is discrete-valued, or the probability density function (pdf) if it is continuous. Suppose x has d dimensions, and each dimension can take on 1 out of K possible values. Then the full joint distribution requires $K^d - 1$ degrees of freedom[5] to describe the probability of all K^d possible values of the pmf. This becomes unmanageable for large d, both from a statistical point of view and in terms of space-and-time complexity, therefore necessitating a more compact representation of high dimensional distributions. This is just for discrete-valued data. If the data is a continuous variable, traditionally we either make excessively simple assumptions on the family of distributions, or resort to the histogram estimator and kernel density estimation (KDE), both of which scale poorly in dimensionality [Wasserman, 2006].

In this section, we review a few generative models that are designed to compactly represent joint probability distributions parameterized by deep neural networks. Depending on whether the exact log-likelihood can be tractably computed or requires approximation, they can be categorized according to Figure 1.1.

[5]The minus one is because the probability sums to one.

Autoregressive Models For any random variable x, we can have the following decomposition known as the chain rule (or product rule) of probability

$$q(x) = \prod_j q(x_j \mid x_{<j}),$$

where $x_{<j}$ is short for the realization $\cap_{j'<j} x_{j'}$. This is the inspiration for the parameterization of *autoregressive models* $p_\theta(x) = \prod_j p_\theta(x_j \mid x_{<j})$. Instead of tackling the joint probability directly, we can more compactly represent it via the conditional distributions $p_\theta(x_j \mid x_{<j})$.

Autoregressive models are commonly used in language modeling. Consider a sequence of tokens (such as a set of common words used in a language), where the set of tokens is of size d. The joint probability can be represented by of table of size K^d, which grows exponentially as the length of the sequence d increases. We can instead model the conditionals $q(x_j \mid x_{<j})$ using $p_\theta(x_j \mid x_{<j})$, where the latter can be parameterized by an autoregressive architecture, such as the *recurrent neural networks* (RNN) [Rumelhart et al., 1986].

A common criticism of RNN is that the computation cost of these conditionals depends on the length of the data, which is problematic for longer sequences especially during training. A parallelizable architecture called MADE [Germain et al., 2015] addresses this issue. MADE is an autoencoder with dense layers and makes use of binary masks to enforce the internal nodes of an MLP to not depend on certain input features, so that the j'th block of output nodes is merely a function of $x_{<j}$, and can be used to parameterize the conditional distribution $p_\theta(x_j \mid x_{<j})$. Other more specialized, parallelizable architectures include PixelCNN [Oord et al., 2016b] for image data, WaveNet for raw audio waveforms [Oord et al., 2016a], and the transformer architecture for large-scale language modeling [Vaswani et al., 2017].

Generative Flows Alternatively, we can define a generative model using invertible maps. Let $Z \in \mathbb{R}^d \sim p_Z$ be a real-valued random vector following a prior distribution

p_Z, and $f : \mathbb{R}^d \to \mathbb{R}^d$ be a differentiable, invertible mapping. Then the pdf of the random variable $X = f^{-1}(Z)$ can be computed by the change of variable formula [Folland, 1999, Theorem 2.47]

$$\boxed{p_X(x) = p_Z(f(x)) \left| \det \frac{\mathrm{d}f(x)}{\mathrm{d}x} \right|} \tag{1.6}$$

where $\frac{\mathrm{d}f}{\mathrm{d}x}$ denotes the Jacobian matrix.[6]

In practice, we usually parameterize the function f that transforms the structured data into the prior space[7], hoping that $f(x)$ would be approximately distributed according to the prior $p_Z(z)$. The log likelihood of a data point x under the density $p_\theta(x) := p_X(x)$ can be readily computed [Dinh et al., 2014, 2017] by (i) evaluating the log-likelihood of $f(x)$ under the prior density p_Z, and (ii) computing the log-determinant of the Jacobian $\mathrm{d}f/\mathrm{d}x$. The first term ensures the encoded data is likely under the prior, and the determinant of the Jacobian penalizes the infinitesimal change of volume due to the change of variable f, making sure that the encoding $f(x)$ does not collapse to a point mass in the prior space. Once this is achieved, we can then sample $z \sim p(z)$ and obtain the synthetic data via $x = f^{-1}(z)$, which is the *generative flow* from the prior to the data.

The main computational requirements of designing generative flows are: (1) f needs to be bijective and differentiable. (2) The log-determinant of $\mathrm{d}f/\mathrm{d}x$ is easy to compute; in general, computing the determinant of a full-rank matrix requires $\mathcal{O}(d^3)$ computation (using the LU-decomposition), which is costly for large matrices. (3) For the model to be used as a generative model, the inversion of f should also be cheap; there exist invertible functions that are not analytically invertible, which necessitates numerical methods such as bisection search or fixed-point methods. As a result, in order to fulfill

[6] We also use ∇ to denote the Jacobian interchangeably when it's clear in the context that f is vector-valued.

[7] An alternative is to parameterize the mapping f^{-1} directly and let f be its inverse. But we would then have to ensure the inverse function f and its gradient can be efficiently computed since they are needed for likelihood evaluation and training according to (1.6).

these requirements, the expressivity of the mapping f is usually severely restricted. Below are some examples.

1. Block-wise Affine Coupling: Dinh et al. [2017] propose the affine coupling:

$$f_\theta(x_a, x_b) = \texttt{concat}(x_a,\ s_\theta(x_a) \odot x_b + m_\theta(x_a)), \tag{1.7}$$

where s_θ and m_θ are parameterized by neural networks, x_a and x_b are two partitioning of the data vector, and `concat` is an operator that concatenates the input vectors. Since the composition of invertible maps is invertible, Dinh et al. [2014, 2017] propose to compose multiple layers of block-wise affine transformations intertwined with a permutation of elements of x. This is known as RealNVP in the literature, which stands for real-valued non-volume-preserving transformation Dinh et al. [2017]. Its predecessor, NICE [Dinh et al., 2014], is a special case with $s_\theta = 1$, which restricts the transformation to be volume-preserving (since the Jacobian determinant is 1). A RealNVP block can be easily inverted, via

$$f_\theta^{-1}(y_a, y_b) = \texttt{concat}(y_a,\ (y_b - m_\theta(y_a))/s_\theta(y_a)),$$

where the division is elementwise. The Jacobian determinant of (1.7) is the product of the elements of $s_\theta(x_a)$, which can be computed in linear time.

2. Inverse Autoregressive Affine Coupling: While RealNVP is computationally efficient for evaluation and for sampling (inversion), the common criticism is that it requires a longer chain of transformations to reach the same performance level as other flow methods [Rezende and Mohamed, 2015, Kingma et al., 2016]. RealNVP can be extended so that all of the data is transformed in one pass [Kingma et al., 2016]:

$$f_\theta(x)_i = s_{\theta,i}(x_{<i}) \cdot x_i + m_{\theta,i}(x_{<i}), \tag{1.8}$$

for i iterating over the indices of the features. Since this transformation can be viewed as the inverse function of an autoregressive model, we refer to it as the *inverse autore-*

gressive flow (IAF)[8]. To see that, we notice the inversion of (1.8) satisfies the following recursive formula

$$f_\theta^{-1}(y)_i = (y_i - m_{\theta,i}(f_\theta^{-1}(y)_{<i}))/s_{\theta,i}(f_\theta^{-1}(y)_{<i}). \tag{1.9}$$

Similar to RealNVP, the Jacobian determinant of (1.8) is $\prod_i s_{\theta,i}(x_{<i})$, which can be computed in linear time. The scale and shift coefficients s_θ and m_θ can be implemented via PixelCNN or MADE so that all of the computations involved in training can be done in parallel.

Variational Autoencoders Another natural way to define a generative model is via latent variables, which allow us to abstractly represent data in a latent space. Let $z \in \mathbb{R}^k \sim p(z)$ be a random vector following a prior $p_Z(z)$ (below we suppress the subscript to avoid cluttering in notation). We assume the data x follows a data generation process given the latent representation z, in the form of a conditional distribution $p(x \mid z)$. Then the marginal pdf of x is

$$p(x) = \int p(x \mid z)p(z)\,\mathrm{d}z, \tag{1.10}$$

which is in general not tractable if the prior $p(z)$ is not conjugate to the conditional $p(x \mid z)$, or if the conditional involves complex nonlinear mappings of z. For example, we will not be able to easily simplify the integration in the marginal density if we parameterize the conditioning by a neural network, *i.e.* $p(x; \mathrm{NN}(z))$. We can instead approximately maximize the log marginal likelihood, by maximizing a *variational lower bound*. First, we define an auxiliary distribution[9] $q(z \mid x)$. Then we have

[8] We refer to IAF as MAF, which stands for *masked autoregressive flow* [Papamakarios et al., 2017], when we use it for generative modeling (or density estimation). We call it IAF if it is used to improve variational inference, which is what it was originally designed for (see the following paragraphs).

[9] It is also known as the variational distribution, the encoder, the recognition model, the inference machine, or the approximate posterior, for a reason that will soon be clear.

$$\log p(x) = \log \int p(x \mid z) p(z) \, \mathrm{d}z \tag{1.11}$$

$$= \log \int \frac{q(z \mid x)}{q(z \mid x)} p(x \mid z) p(z) \, \mathrm{d}z \tag{1.12}$$

$$\geq \int q(z \mid x) \log \frac{p(x \mid z) p(z)}{q(z \mid x)} \, \mathrm{d}z \tag{1.13}$$

$$= \mathbb{E}_{q(z|x)} \left[\log p(x \mid z) \right] - D_{\mathrm{KL}}(q(z \mid x) || p(z)) := \mathcal{L}. \tag{1.14}$$

We inserted the identity q/q to obtain (1.12), which results in a change of measure $p(z) \, \mathrm{d}z \to q(z \mid x) \, \mathrm{d}z$, and then we applied Jensen's inequality wrt this new measure to obtain a lower bound (1.13). As the marginal likelihood is often called the evidence in Bayesian inference, this lower bound is called the Evidence Lower BOund, or ELBO for short.

We also parameterize $q(z \mid x)$ using a neural network, so as to *amortize* the inference process for all data points x [Dayan et al., 1995, Gershman and Goodman, 2014]. Assume q is a conditional Gaussian with diagonal covariance matrix. Then the latent variable z can be reparameterized as $\mu(x) + \sigma(x) \odot e$ where e is a standard normal noise. This allows us to rewrite the ELBO as

$$\mathcal{L} = \underbrace{\mathbb{E}_{e \sim \mathcal{N}(0,I)} [\log p(x \mid \mu(x) + \sigma(x) \odot e)]}_{\text{reconstruction error}} - \underbrace{D_{\mathrm{KL}}(q(z \mid x) || p(z))}_{\text{regularization}}. \tag{1.15}$$

The first term is the reconstruction error of a randomized code $z = \mu(x) + \sigma(x) \odot e$. The second term can be decomposed into the entropy $-\mathbb{E}_q[\log q(z \mid x)]$, which ensures the approximate posterior does not shrink to a point mass, and the cross-entropy $\mathbb{E}_q[\log p(z)]$, which penalizes the latent code if it deviates too far away from the prior. As the ELBO can be seen as the loss function of a regularized, stochastic autoencoder, we call the generative model p along with the inference model q the *variational autoencoder* (VAE) [Kingma and Welling, 2014, Rezende et al., 2014].

The change of variable (aka the *reparameterization trick*) allows us to decouple the noise

source from the parameters we would like to update (parameters of the encoder), so that the gradient of the expectation can be estimated via Monte Carlo[10], enabling standard deep learning optimization methods such as variants of SGD. The change of variable is known as the *law of the unconscious statistician* in the statistics literature (LOTUS) [DeGroot and Schervish, 2012], and the gradient estimator is called the *pathwise derivative*, a fundamental tool in infinitesimal perturbation analysis [Glasserman and Ho, 1991]. See the recent review Mohamed et al. [2020] for more details on gradient estimators.

The gap in the inequality of Equation (1.14) is equal to the KL divergence between the approximate posterior and the true posterior:

$$\log p(x) - \mathcal{L} = D_{\mathrm{KL}}(q(z \mid x)||p(z \mid x)), \tag{1.16}$$

which means maximizing the lower bound to update the model $p(x, z)$ may not necessarily improve the marginal $\log p(x)$ if $q(z \mid x) \neq p(z \mid x)$. This also means q approximates the true posterior, since we are essentially minimizing the KL divergence when we maximize the lower bound by updating q. The bias introduced by the lower bound can be reduced by considering a richer family of variational distributions for q. A flexible choice of parameterization is to use a neural network as a noise generator that transforms a random variable e sampled by some base distribution $q(e)$. However, with arbitrary neural nets, we cannot guarantee the range of the output matches the entire latent space (over which the prior density is defined – usually the real space).

1. Normalizing flows: To induce a richer family of tractable densities, we can leverage the idea of generative flows. Essentially, we can generalize the Gaussian reparameterization to an arbitrary invertible map $f(x, e)$ (f is invertible in e given x), which induces a more flexible density through the change of variable. The invertible map ensures the density is non-zero everywhere in the desired domain, and that the

[10]Note that reparameterization is just one way to estimate the gradients. There are other options such as REINFORCE (also known as the score function estimator) [Williams, 1992]. The reparameterization trick often gives lower variance in practice, although it is not always the case. See 3.1.2 of Gal [2016] for a detailed study.

entropy term in the ELBO (1.14) can be efficiently estimated by

$$-\mathbb{E}_{z\sim q(z|x)}[\log q(z \mid x)] = -\mathbb{E}_{e\sim q(e)}\left[\log q(e)\left|\det \frac{\partial f(x,e)}{\partial e}\right|^{-1}\right] \tag{1.17}$$

$$\approx -\log q(e) + \log\left|\det \frac{\partial f(x,e)}{\partial e}\right|, \tag{1.18}$$

where e is drawn from some base distribution with a pdf $q(e)$. As the Jacobian determinant guarantees that the induced density is normalized (integrating to 1), following the convention in the literature, we call an invertible map used to enhance variational inference *normalizing flow* [Tabak et al., 2010, Tabak and Turner, 2013, Rezende and Mohamed, 2015].

2. Hierarchical inference: Another way to make sure the density is well-defined and nonzero for a variational distribution parameterized by a neural network is to add a bit of perturbation around the output of the network. This amounts to a hierarchical variational posterior; *i.e.* the approximate posterior itself is also a latent-variable model. In general, the density of z of this hierarchical model is also intractable, but its entropy can be lower bounded via another variational inequality

$$\begin{aligned}
-\mathbb{E}_{q(z)}[\log q(z)] &\geq -\mathbb{E}_{q(z)}\big[\log q(z) + D_{\mathrm{KL}}(q(e \mid z)||p(e \mid z))\big] \\
&= -\mathbb{E}_{q(z)}\big[\log q(z) + \mathbb{E}_{q(e|z)}[\log q(e \mid z) - \log p(e \mid z)]\big] \\
&= -\mathbb{E}_{q(z)q(e|z)}\big[\log q(z) + \log q(e \mid z) - \log p(e \mid z)\big] \\
&= -\mathbb{E}_{q(z)q(e|z)}\big[\log q(z \mid e) + \log q(e) - \log p(e \mid z)\big],
\end{aligned}$$

due to the non-negativity of the KL divergence between the "posterior" of the variational distribution $q(e \mid z)$ and its variational approximation $p(e \mid z)$. The tightness of this bound depends on how close $p(e \mid z)$ is to $q(e \mid z)$.

If one just maximizes the first term alone, there is no incentive for $q(z \mid e)$ to use e. In some sense $p(e \mid z)$ forces $q(z \mid e)$ to "specialize" by using the conditioning information e, as the gradient from p will flow through z to affect the parameter updates of $q(z \mid e)$.

Energy-based models Our last family of generative models, which enjoy a great degree of flexibility, is defined via an energy function $E(x)$, which is an unconstrained scalar function of the input data. Energy-based models, or EBMs, are characterized by an unnormalized density $p(x) \propto \exp(-E(x))$. Sampling from EBM can be done with the assistance of Markov chain Monte Carlo methods, such as discretizing the Langevin diffusion

$$\begin{aligned} \mathrm{d}X &= \nabla \log p(X)\,\mathrm{d}t + \sqrt{2}\,\mathrm{d}B_t && (1.19)\\ &= -\nabla E(X)\,\mathrm{d}t + \sqrt{2}\,\mathrm{d}B_t, && (1.20) \end{aligned}$$

where B_t is a Brownian motion. In physics, the Langevin equation is used as a model of molecular dynamics to account for the friction caused by a viscous solvent, and the energy function E in our case is analogous to the interaction potential of the atoms.

We now turn to the maximum likelihood training of EBM. Immediately, we are faced with a computational challenge: computing the likelihood requires normalizing the density

$$p(x) = \frac{1}{Z}\exp(-E(x)) \qquad \text{where } Z = \int \exp(-E(x'))\,\mathrm{d}x'. \tag{1.21}$$

The constant Z, which is known as the *partition function* in statistical mechanics, is to ensure the density p integrates to 1, and is in general computationally intractable. Fortunately, we can still estimate the gradient of the log-likelihood

$$\begin{aligned} \nabla_\theta \log p(x) &= -\nabla_\theta E(x) - \nabla_\theta \log \int \exp(-E(x'))\,\mathrm{d}x' && (1.22)\\ &= -\nabla_\theta E(x) - \frac{1}{Z}\int \exp(-E(x'))(-\nabla_\theta E(x'))\,\mathrm{d}x' && (1.23)\\ &= -\nabla_\theta E(x) + \int \frac{\exp(-E(x'))}{Z}(\nabla_\theta E(x'))\,\mathrm{d}x' && (1.24)\\ &= -\nabla_\theta E(x) + \int p(x')\nabla_\theta E(x')\,\mathrm{d}x'. && (1.25) \end{aligned}$$

Taking the average over the data distribution q (or $\hat{q}$ for the empirical distribution), we

have

$$\nabla_\theta \mathbb{E}_q[\log p(x)] = \mathbb{E}_q[-\nabla_\theta E(x)] + \mathbb{E}_p[\nabla_\theta E(x)]. \tag{1.26}$$

That is, we draw *positive* samples from the data distribution $x \sim q$, and minimize the energy. We also draw *negative* samples from the model $x \sim p$, and maximize the energy. Intuitively, this means only the regions where the data is likely (under q) will have high model density (lower energy), and elsewhere the model will be forced to have low density (higher energy). The model samples are typically drawn via MCMC, and the resulting algorithm for training is called the *contrastive divergence* algorithm [Hinton, 2002].

Contrastive divergence and its variants are notoriously slow and unstable, due to the need to draw negative samples from the model. For this reason, some other learning principles which do not require estimating the log-partition function or its gradient might be preferred, including the score matching principle we have seen in section 1.1.

1.4 THESIS QUESTION AND THESIS STRUCTURE

Having introduced different families of likelihood-based generative models, we might wonder if some models are better than others. If some models are indeed superior in some respects, can we draw inspiration from them to design better models? In general, there are two attributes that we would like to optimize when designing models:

- **Expressivity:**
  We would like to design models that are capable of approximating the data distribution arbitrarily well. This is especially important in the large-data regime, where the statistical estimation error is less of a concern and we would rather spend our resources bringing down the approximation error to achieve a high level of realism in the generated samples or good held-out likelihood estimates.

- **Scalability:**
  After a model class is specified, we need to be able to train the model efficiently. Loosely speaking, the scalability of a model refers to the space-and-time complexity of training, which in the case of gradient-based likelihood optimization pertains to the growth of the memory and computational costs in computing, approximating, or estimating log-likelihood gradients, as we increase the model complexity or consider larger-scale problems. Improving scalability is the first step towards making these computations tractable for high-dimensional problems. Crucially, it will also enable us to consider more powerful model classes that are normally more computationally demanding.

Note that these two aspects are by no means representative of the complete picture of likelihood-based models. For example, besides expressivity, there are other aspects that might be useful for attaining good performance, such as the inductive biases of the architecture, model specification, or even the training algorithm that we use to obtain the parameter estimates. Though we focus on improving the space-time complexity of training, that does not mean inference (such as sample generation) would be automatically more efficient. For example, MAF is efficient for training thanks to parallel computing but slow for sampling.

We focus on these two aspects as they are the key ingredients for a general-purpose statistical model, but oftentimes they are mutually exclusive. More expressive models are usually less scalable, due to the lack of structure to be leveraged to make computation cheaper. For example, computing the determinant of a free-form Jacobian matrix takes $\mathcal{O}(d^3)$ time, whereas the same computation takes $\mathcal{O}(d)$ time for RealNVP and IAF. This makes the latter more suitable for higher-dimensional problems. However, making this compromise would not necessarily lead to a significant gain in performance, as a less expressive model may be limited in the level of realism it can achieve. While a highly expressive model can, in theory, attain better performance given unlimited resources, in reality this is never the case, and we are always bound by finite computing budgets. The ideal direction we would like to move towards in the space of models should lead us to both higher scalability and expressivity (see Figure 1.2).

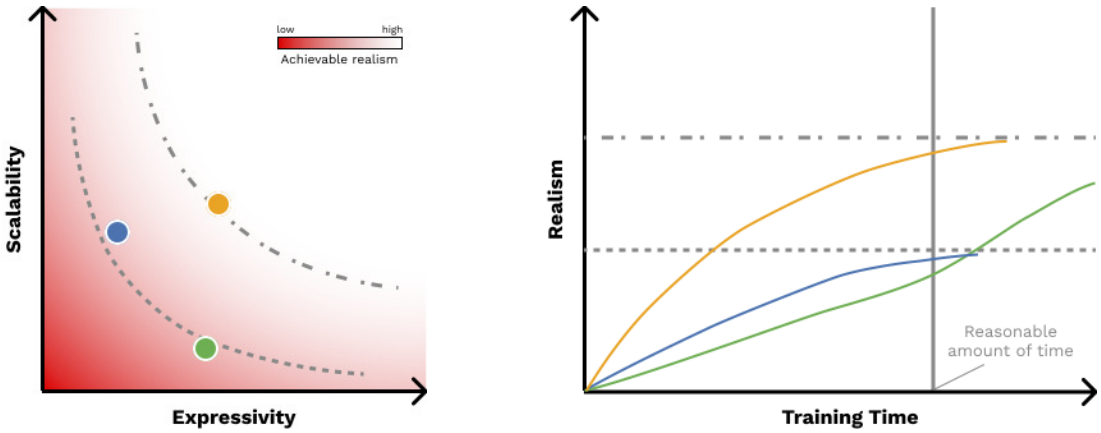


Figure 1.2: (*left*) An illustration of models of different scalability and expressivity, and the levels of realism achievable within a reasonable amount of time. The ideal kind of progress would benefit from the combined improvement made in both aspects. (*right*) An illustration of how models improve throughout training. A highly expressive, less scalable model can, in theory (green line), eventually attain a higher level of realism, but may not be able to do so within a fixed time budget. A more scalable, yet representationally restricted model (blue line) may appear to make more progress early on during training, but will eventually hit a plateau due to limited expressivity. A desirable improvement on the Pareto front should lead to both improved expressivity and scalability, allowing the model (orange line) to achieve superior realism within the time frame.

The aim of the thesis is to design better likelihood-based models in these two aspects. As argued, additional structures usually need to be imposed in exchange for tractability or scalability. We may ask: *what kind of structure is suitable for general-purpose likelihood-based generative modeling?* By suitable, we mean that the expressivity of the model class would not be undermined too much, and we would still be able to approximate the data distribution arbitrarily well. The imposed structure will allow us to compute the quantities of interest scalably. The thesis is organized in three parts to answer this question.

Part I: Universal flows with specialized structure

In this part of the thesis, we are concerned with designing generative flows that are capable of universally approximating a broad class of distributions. We first introduce a design principle that guarantees universality in Chapter 2. Following the principle, we propose a few specialized invertible structures that can be leveraged to largely

reduce the computational cost of likelihood evaluation and training, including *monotone functions* in Chapter 3, *triangular maps* in Chapter 4, and *convex potentials* in Chapter 5. Importantly, the triangular maps connect flow-based generative models with autoregressive models, which allows us to reuse the architectures designed for the latter to improve generative flows.

Part II: Improving expressivity via augmentation

We present an interlude chapter (Chapter 6) to introduce a technique that can be used to enhance the expressivity of any type of generative flow. Notably, we draw a connection between latent variable models such as VAEs and generative flows, marrying the two families of models. The implication is twofold: (1) it allows flow-based models to admit a low-dimensional representation of the data, like VAEs, and (2) it relaxes the typical independence assumption of VAEs by generalizing the encoding and decoding transformations.

Part III: Continuous-time flows

The last part of the thesis centers around the use of continuous-time dynamical systems as generative models, which form an important class of probability flows, as the parameterization only requires very minimal assumptions, unlike the discrete-time flows presented in Part I. In Chapter 7, we review the flow maps induced by *neural ordinary differential equations* [Chen et al., 2018, Neural ODEs], which can be seen as an infinitely deep residual flow [Behrmann et al., 2019, Chen et al., 2019a], and we prove that they are universal approximators for distributions. The cost of expressivity is that we need to resort to numerical solvers during training: the computational cost scales $\mathcal{O}(m)$, where m is the number of function evaluations, typically depending on a pre-specified error tolerance. In Chapter 8, we generalize neural ODEs to *neural stochastic differential equations* (neural SDE). The added degree of freedom as a by-product of variational inference allows us to design algorithms to largely cut down the computational complexity of training to $\mathcal{O}(1)$, making it much more scalable than neural ODEs. Furthermore, we show that by parameterizing the model in a certain way, the likelihood training loss is equal to the score-matching loss typically used for

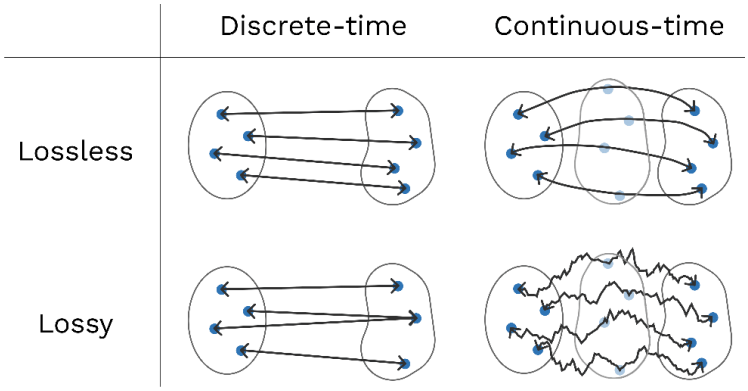


Figure 1.3: (*top left*) discrete-time models with lossless representation include normalizing flows (see Part I) and injective flows [Kumar et al., 2020]. (*bottom left*) discrete-time models with lossy representation such as VAE, or more generally augmented normalizing flows (see Part II) and SurVAE flows [Nielsen et al., 2020]. (*top right*) dynamical models such as the flow map induced by a neural ordinary differential equation (see Chapter 7). (*bottom right*) stochastic continuous-time models, such as diffusion models (see Chapter 8).

training EBMs. This theoretical discovery sheds light on the recent advancements in score-based generative models [Song et al., 2021c].

In all three parts of the thesis, we derive the likelihood of the model(s) via the change of variable associated with the flow map, or more generally the data generating process if the flow is stochastic such as augmented models and SDEs. In fact, this means all likelihood-based models summarized by Figure 1.1 in the previous section are now unified, as they can all be seen as special cases of *probability flows* (hence the title of the thesis). We can further divide the span of these models along two orthogonal dimensions: discrete-time vs continuous-time models, and models with lossless vs lossy representations (see Figure 1.3). We hope that this unification will inspire more cross-over in the design of likelihood-based generative models and more cross-pollination of research ideas from different sub-fields.

Personal contributions This thesis is mainly comprised of the contents of the following papers, majorly reorganized and with the addition of some new results, in

chronological order. I am the main driver of all of the work in terms of theoretical development and experiments, except for the first article for which I share the co-first authorship.

1. *Neural autoregressive flows*
   Chin-Wei Huang, David Krueger, Alexandre Lacoste, Aaron Courville
   International Conference on Machine Learning (ICML), 2018
   Chapter 3 and 4.
   I identified the difficulty of modeling multimodal distributions, came up with DSF/DDSF, developed the universality proof, and ran the synthetic experiments as well as the variational inference experiments.

2. *Augmented normalizing flows: Bridging the gap between generative flows and latent variable models* and *Solving ode with universal flows: Approximation theory for flow-based models*
   Chin-Wei Huang, Laurent Dinh, Aaron Courville
   ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations
   Chapter 2, 6 and 7.
   I developed the main theory of the work and ran all the experiments.

3. *Convex potential flows: Universal probability distributions with optimal transport and convex optimization*
   Chin-Wei Huang, Ricky T. Q. Chen, Christos Tsirigotis, Aaron Courville
   International Conference on Learning Representations (ICLR), 2021
   Chapter 3 and 5.
   I came up with the idea of using gradients of a convex function as normalizing flow, developed the theory of invertibility, universality and optimality, implemented the CG method for stochastic gradient estimation, explored different ICNN variants, and ran the variational inference tasks on the tabular data.

4. *A variational perspective on diffusion-based generative models and score matching*

Chin-Wei Huang, Jae Hyun Lim, Aaron Courville
Neural Information Processing Systems (NeurIPS), 2021
Chapter 8.
I developed the main theory of the work and ran the CIFAR experiments.

Accompaniments In order to maintain a coherent narrative, a few other projects and publications that were conducted and completed during my Ph.D. are not included in the thesis. I list them below for completeness.

1. *Bayesian Hypernetworks*
   David Krueger, **Chin-Wei Huang**, Riashat Islam, Ryan Turner, Alexandre Lacoste, Aaron Courville
   NIPS Workshop on Bayesian Deep Learning, 2017

2. *Neural Language Modeling by Jointly Learning Syntax and Lexicon*
   Yikang Shen, Zhouhan Lin, **Chin-Wei Huang**, Aaron Courville
   International Conference on Learning Representations (ICLR), 2018

3. *Improving explorability in variational inference with annealed variational objectives*
   Chin-Wei Huang, Shawn Tan, Alexandre Lacoste, Aaron Courville
   Neural Information Processing Systems (NeurIPS), 2018

4. *Hierarchical Importance Weighted Autoencoders*
   Chin-Wei Huang, Kris Sankaran, Eeshan Dhekane, Alexandre Lacoste, Aaron Courville
   International Conference on Machine Learning (ICML), 2019

5. *Probability Distillation: A Caveat and Alternatives*
   Chin-Wei Huang, Faruk Ahmed, Kundan Kumar, Alexandre Lacoste, Aaron

Courville
Association for Uncertainty in Artificial Intelligence (UAI), 2019

6. *vGraph: A Generative Model for Joint Community Detection and Node Representation Learning*
Fan-Yun Sun, Meng Qu, Jordan Hoffmann, **Chin-Wei Huang**, Jian Tang
Neural Information Processing Systems (NeurIPS), 2019

7. *Stochastic Neural Network with Kronecker Flow*
Chin-Wei Huang, Ahmed Touati, Pascal Vincent, Gintare Karolina Dziugaite, Alexandre Lacoste, Aaron Courville
International Conference on Artificial Intelligence and Statistics (AISTATS), 2020

8. *AR-DAE: Towards Unbiased Neural Entropy Gradient Estimation*
Jae Hyun Lim, Aaron Courville, Chris Pal, **Chin-Wei Huang**
International Conference on Machine Learning (ICML), 2020

9. *A benchmark of medical out of distribution detection*
Tianshi Cao, **Chin-Wei Huang**, David Yu-Tung Hui, Joseph Paul Cohen
ICML Workshop on Uncertainty and Robustness in Deep Learning, 2020

10. *Bijective-Contrastive Estimation*
Jae Hyun Lim, **Chin-Wei Huang**, Aaron Courville, Chris Pal
Symposium on Advances in Approximate Bayesian Inference, 2020

11. *Problems in the deployment of machine-learned models in health care*
Joseph Paul Cohen, Tianshi Cao, Joseph D Viviano, **Chin-Wei Huang**, Michael Fralick, Marzyeh Ghassemi, Muhammad Mamdani, Russell Greiner, Yoshua Bengio
Canadian Medical Association Journal (CMAJ), 2021

12. *Learning to Dequantise with Truncated Flows*
Shawn Tan, **Chin-Wei Huang**, Alessandro Sordoni, Aaron Courville
International Conference on Learning Representations (ICLR), 2022

13. *Riemannian Diffusion Models*
    Chin-Wei Huang, Milad Aghajohari, Joey Bose, Prakash Panangaden, Aaron Courville
    Neural Information Processing Systems (NeurIPS), 2022

Part I

Universal flows with specialized structure

Universal flows with specialized structure

In this part, we address the research question about the structure needed to be imposed for generative flows to have a tractable density that can be efficiently optimized via maximum likelihood. Recall that we want the flow f to satisfy a few practical requirements and desiderata, which we summarize below:

RD1 f needs to be invertible and sufficiently smooth.

RD2 $\log \left|\det \frac{\partial f}{\partial x}\right|$ and $\nabla_\theta \log \left|\det \frac{\partial f}{\partial x}\right|$ can be easily computed or estimated.

RD3 f^{-1} has an exact formula or can be numerically approximated.

RD4 f is sufficiently flexible.

RD1 is obviously needed to invoke the change of variable formula. **RD2** is a computational requirement since otherwise, it will be intractable to train a flow on high dimensional data. **RD3** is needed when we want to invert the flow for sampling, *i.e.* when the flow is used for generative modeling. One of the main goals of this part of the thesis is to provide a principled way to design flows that not only satisfy **RD1-3**, but are also sufficiently expressive (**RD4**) for the purpose of density modeling. This is especially important if we would like to learn the structure of the data generative process directly from the data when there is little prior knowledge available.

We will first present a design principle for generative flows that are guaranteed to be distributionally universal (for the definition of distributional universality, see Section 2.1). It will allow us to design and analyze **specialized architectures** different from standard deep neural net architectures. Typically, the Jacobian matrix of the flow would have some special structure that can be leveraged to accelerate the computation or estimation of its determinant. This makes it different from the *free-form Jacobian* of an unconstrained neural network (we will see more of this in the part III). Such specialized structure is required to satisfy the above-mentioned requirements and desiderata. While they restrict the function class expressible by the invertible neural net, the design

principle will allow us to show that they are sufficiently powerful in the sense that they can approximate a large family of distributions that might be of interest.

2 Designing universal flows

An essential aspect of this thesis revolves around the expressive power of a model class. How expressive is the model? How can we determine its capacity to approximate any data distribution of interest, provided sufficient computational resources? In this chapter, we address these questions by introducing the concept of distributional universality. We then propose a design principle for *universal flows* – a family of invertible models capable of effectively translating arbitrary probability measures. This principle serves as a foundation for the parameterization of the flows discussed in Chapters 3, 4, and 5 that satisfy some of the requirements and desiderata (RD1-4). Furthermore, it allows us to demonstrate the distributional universality of these models, including the continuous-time flows introduced in Chapter 7.

Setup and notation: Given a base probability measure μ and an invertible map f applied to μ, we denote by $f_{\#}\mu := \mu \circ f^{-1}$ the pushforward measure of μ under f. That is, if $x \sim \mu$ is a random variable distributed according to μ and if we set $y := f(x)$, then the law of y, denoted by ν, satisfies $\nu = f_{\#}\mu$. When μ and ν admit probability densities p_X and p_Y, we also write $p_Y = f_{\#}p_X$.

2.1 DISTRIBUTIONAL UNIVERSALITY

Since our subject of interest is the distribution induced by the flow, we need to introduce a concept of stochastic convergence that tells us the distribution induced by the flow is indeed getting closer to some target distribution that we would like to approximate. Some notion of convergence is needed because it tells us that we can take a sequence of distributions to approximate the target, and the sequence of distributions will be represented by flows parameterized in a certain way.

We first define a set of functions called *test functions*, which can be used to distinguish unique distributions. Test functions should be a large enough class of functions if we want to use them to detect whether two distributions are the same or not. At the same time, we also want them to be easy to work with. For example, we consider $\mathcal{C}_b$, the class of bounded, continuous functions. A classic result in probability theory tells us that two probability measures μ and ν on $(\mathbb{R}^d, \mathcal{B}(\mathbb{R}^d))$ are equal if and only if $\int \phi \, \mathrm{d}\mu = \int \phi \, \mathrm{d}\nu$ for all $\phi \in \mathcal{C}_b$. This motivates the definition of weak convergence of measures.

Definition 2 (Weak convergence). *We say μ_n converges weakly to ν, denoted by $\mu_n \rightharpoonup \nu$, if for any $\phi \in \mathcal{C}_b$, $\lim_{n\to\infty} \int \phi \, \mathrm{d}\mu_n = \int \phi \, \mathrm{d}\nu$.*

The intuition is that if two measures differ on some subsets of $\mathbb{R}^d$, then we can choose ϕ to emphasize those regions so that the equality would not hold. We note that smaller sets of test functions can also be used in place of $\mathcal{C}_b$, such as C_c^∞, the class of compactly supported, smooth functions, and the class of bounded, Lipschitz functions, which will come in handy soon[1]. This definition is also equivalent to the notion of *convergence in distribution* of random variables. The laws of a sequence of random variables converge weakly to some target distribution if their corresponding (cumulative) distribution functions converge at all continuity points.

This definition might not be the easiest to work with. Below we give a few equivalent

[1] This is known as the **Portmanteau Lemma**, which provides multiple equivalent definitions of weak convergence of probability measures. See chapter 2 of Billingsley [1999] for more discussion.

conditions. Let $||\cdot||_{BL}$ denote the bounded Lipschitz norm on functions, *i.e.*

$$||\phi||_{BL} = ||\phi||_\infty + ||\phi||_L,$$

where $||\phi||_\infty = \sup_x |\phi(x)|$ is the uniform norm and $||\phi||_L = \sup\{|\phi(x) - \phi(y)|/||x - y|| : x \neq y\}$ is the Lipschitz seminorm. Let μ and ν be probability measures. We define

$$\rho(\mu, \nu) := \sup\left\{\left|\int \phi \,\mathrm{d}\mu - \int \phi \,\mathrm{d}\nu\right| : ||\phi||_{BL} \leq 1\right\}. \tag{2.1}$$

ρ can be shown to be a metric on the space of probability measures on $\mathbb{R}^d$ (or on any metric space). It is a type of *integral probability metric* [Müller, 1997], known as the Dudley metric. The following theorem shows ρ metrizes the weak convergence of probability measures.

Theorem 3 (**Characterization of weak convergence**, Theorem 11.3.3 of Dudley [2018])**.** *Let $\{\mu_n\}$ and ν be probability measures on $(\mathbb{R}^d, \mathcal{B}(\mathbb{R}^d))$. The following are equivalent:*

1. $\int \phi \,\mathrm{d}\mu_n \to \int \phi \,\mathrm{d}\nu$ *for all* $\phi \in \mathcal{C}_b$ *(weak convergence).*
2. $\int \phi \,\mathrm{d}\mu_n \to \int \phi \,\mathrm{d}\nu$ *for all* $\phi \in \mathcal{BL}$*, where* $\mathcal{BL} := \{\phi : ||\phi||_{BL} < \infty\}$.
3. $\rho(\mu_n, \nu) \to 0$.

Equipped with the notion of weak convergence, we now are ready to define the concept of universal approximation suitable for generative models such as normalizing flows.

Definition 4 (**Distributional universality**)**.** *Let $\mathfrak{M}$ and $\mathfrak{N}$ be two families of probability measures. Given a family of mappings $\mathcal{F}$, we say that $\mathcal{F}$ is $\mathfrak{M}$-$\mathfrak{N}$ distributionally universal if for any $\mu \in \mathfrak{M}$ and $\nu \in \mathfrak{N}$, there exists a sequence $\{f_n\} \subset \mathcal{F}$ such that $(f_n)_\# \mu \rightharpoonup \nu$.*

For consistency, throughout the thesis, we denote by $\mathfrak{P}$ the space of probability measures over $\mathbb{R}^d$, and $\mathfrak{A} \subset \mathfrak{P}$ the space of probability measures that are absolutely continuous

w.r.t. the Lebesgue measure (*i.e.* they have a density). We will show that the main invertible maps appearing in this thesis are $\mathfrak{A} - \mathfrak{P}$ distributionally universal, including neural autoregressive flows (Theorem 10), convex potential flows (Theorem 13), and continuous-time flows induced by neural ODEs (Theorem 18). This subsumes the case of density estimation where we want to transform complicated data distributions in $\mathfrak{M}$ into a simple standard Gaussian prior $\mathfrak{N} = \{\mathcal{N}(0, I)\}$, and the case of variational inference where we want to transform a simple base distribution $\mathfrak{M} = \{\mathcal{N}(0, I)\}$ into complex target posterior distributions in $\mathfrak{N}$.

2.2 APPROXIMATE COUPLING

As normalizing flows require special parameterization to ensure invertibility and efficient training, they are either a sub-family of standard neural networks (such as MLP) or an entirely different family of architectures. This means that (1) they have a limited capacity in terms of the mappings they can represent, and (2) the standard result of UFT such as Theorem 1 does not apply. In particular, unlike standard neural networks, invertible maps cannot approximate arbitrary continuous functions. For example, even if we consider the family of all bijective functions in $\mathbb{R}$, we still won't be able to approximate simple functions such as $x \mapsto |x|$ in the same sense as Theorem 1. However, for applications such as density estimation or variational inference, it is more important to characterize what distributions p_Y can approximate. The goal of this section is to outline a design principle for distributionally universal normalizing flows.

In this section, we introduce the main principle to establish distributional universality, which we summarize below in two steps:

S1 Identify a transport map g between measures μ and ν.

S2 Approximate g using normalizing flows.

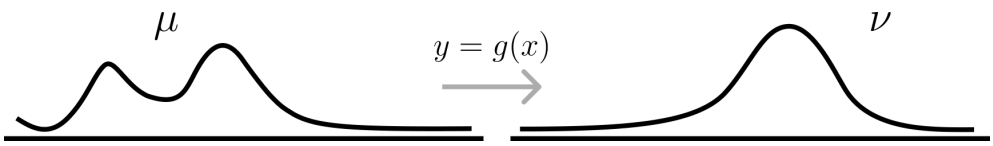


Figure 2.1: A deterministic coupling (transport map) from measure μ to measure ν is such that $g_{\#}\mu = \nu$.

Given probability measures μ and ν, a **transport map** or a (deterministic) **coupling**[2] from μ to ν is a measurable function g that transforms μ into ν; *i.e.* $g_{\#}\mu = \nu$. This means if x and y are distributed by μ and ν respectively, then $g(x)$ and y have the same distribution. See figure 2.1.

Note that we would also require g to be invertible so that they can be approximated by parametric invertible maps. To see that a transport map may not be invertible in general, take g to be an arbitrary non-invertible map. Then g is an non-invertible transport map from μ to $g_{\#}\mu$ by construction.

Suppose for any μ and ν; we can find a universal way to construct a transport map that transforms μ into ν. At this point, we already have a non-parametric universality result, but this non-parametric form will not necessarily be realizable. We would like to establish the universality of specific parametric families of invertible neural networks, hence the need for approximation.

The following lemma shows that if we can find a sequence of invertible maps that approximates a transport map arbitrarily well, then the distributions of the outputs of the invertible maps also get arbitrarily close to the target distribution. In other words, pointwise approximation of a transport map implies distributional approximation (see Figure 2.2).

Lemma 5. *Let μ be a probability measure on $\mathbb{R}^d$ and $\{f_n\}$ be a sequence of functions acting on μ. Assume $f_n \to g$ pointwise almost everywhere. Then $(f_n)_{\#}\mu \rightharpoonup g_{\#}\mu$.*

[2]More generally, a coupling refers to a joint probability measure whose marginals correspond to μ and ν [Lindvall, 1992]. We restrict ourselves to the case where one variable is deterministic given the other, since flow maps are deterministic.

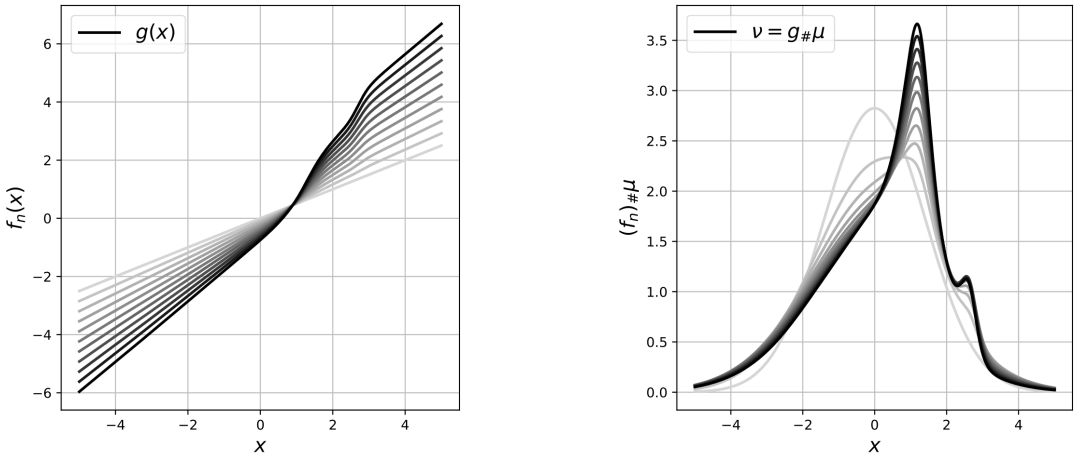


Figure 2.2: Pointwise convergence of functions (*left*) implies convergence of the induced distributions (*right*). Suppose g is the transport map between a Gaussian base distribution μ and a target measure ν; *i.e.* $\nu = g_{\#}\mu$. f_n converges to g as n increases, indicated by the darker lines. When f_n converges to g (dark gray), the induced distribution also gets closer to ν.

Proof. For any $h \in \mathcal{C}_b$, $\mathbb{E}[h(T_n(x))] \to \mathbb{E}[h(T_\infty(x))]$ by the bounded convergence theorem. □

3 CDF transforms and monotone flows

Is there a universal way to construct a transport map for arbitrary probability distributions? What does it look like, and what kind of structure does it possess? We answer these questions for the 1D case in this chapter and propose parameterizations of 1D invertible architectures to approximate the universal transport map. We then generalize it to higher-dimensional cases in the subsequent two chapters.

Our goal is to find a universal way to identify a transport map in 1D and then utilize the structure of the transport map to design invertible flows. In $\mathbb{R}^1$, a simple construction of a transport map is the cumulative distribution functions (CDF) of the probability measures.

CDF transform, inverse CDF transform, and monotone rearrangement

A universal way to construct a transport map is via the distribution function of the random variable [Devroye, 1986]. Let $x \in \mathbb{R} \sim \mu$, and let $\Phi_\mu(a) = \mu((-\infty, a])$ denote the distribution function of μ. As long as μ does not have atoms (*i.e.* Φ_μ is continuous), then $\Phi_\mu(x)$ is uniformly distributed in $[0, 1]$. On the other hand, let $\Phi_\mu^{-1}(b) = \inf\{a : \Phi_\mu(a) > b\}$, which is a μ-a.s. left inverse of Φ_μ. If U is a $[0, 1]$-uniform random variable, then $\Phi_\mu^{-1}(U)$ follows the same distribution as X. We refer to these transport maps as the *CDF transform* and the *inverse CDF transform*. This is visualized in Figure 3.1. Now let $y \sim \nu$ be another random variable. Then $\Phi_\nu^{-1} \circ \Phi_\mu$ is a monotone (in fact, non-decreasing) rearrangement of the input satisfying $\Phi_\nu^{-1}(\Phi_\mu(x)) \overset{d}{=} y$.

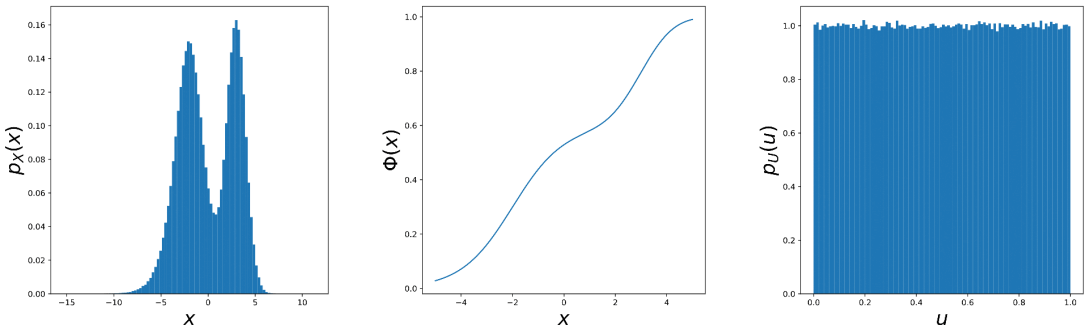


Figure 3.1: CDF transform. The CDF Φ and the inverse CDF Φ^{-1} can be used to translate between a probability distribution p_X and the uniform distribution p_U.

Note that CDFs and inverse CDFs are always increasing monotone functions, and that in $\mathbb{R}^1$, a continuous function is one-to-one if and only if it is strictly monotone. This characterizes invertible functions in $\mathbb{R}^1$, and tells us that it suffices to focus on (increasing) monotone flows. However, usual parametric ways of performing function approximation such as neural networks do not guarantee the learned function to be monotone and therefore invertible. For example, the commonly used ReLU activation function $x \mapsto \max\{0, ax + b\}$ is not invertible, since half of the real line will be mapped to 0. This means additional constraints are needed. We discuss a few ways to parameterize a scalar-valued function that will always be monotone by design.

3.1 DIRECT PARAMETERIZATION

One way of parametering a monotone function is to make sure the derivative of the function does not switch sign. Without loss of generality, we will focus on strictly increasing monotone functions with strictly positive derivatives.

Monotone neural network For neural networks, we can constrain the values of the weight parameters and the activation functions to ensure monotonicity. For example, Sill [1997] proposed the following parameterization, which is a form of maxout network[1] [Goodfellow et al., 2013]:

$$f(x) = \min_j \max_i a_{ij}x + b_{ij}, \tag{3.1}$$

with $a_{ij} > 0$. The positive weights a_{ij} ensure the slope is positive, and the minimum and the maximum operations allow the network to implement locally concave and convex curvatures. The result is a piece-wise linear increasing function that can approximate arbitrary differentiable increasing functions [Sill, 1997]. This architecture, however, is not smooth, and therefore will result in a discontinuous density function if we apply the change of variable formula, which poses a problem for likelihood optimization. For this reason, we can choose to directly constrain the parameters of a neural net that is smooth. A simple construction (see Figure 3.2) of a monotone network rely on the following sufficient conditions:

Theorem 6 (**Monotone neural network**)**.** *A multiple-layer perceptron (1.5) is strictly increasing, and therefore* ***injective****, if*

1. *all the activation functions are strictly increasing, and*
2. *all the weight parameters are strictly positive.*

Proof. This follows from the chain rule of the derivative wrt the input variable, which is a product of positive matrices and vectors. □

This only guarantees injectivity though. For a 1D flow to be bijective from $\mathbb{R}$ to $\mathbb{R}$, we need it to map to all of the points in $\mathbb{R}$, *i.e.* we want the flow to be **surjective**. To obtain surjectivity, one needs to make sure the image of $\mathbb{R}$ through the flow is unbounded. Therefore we cannot use bounded activation function such as sigmoid or ELU. We can use, for example, $x \mapsto x + \sigma(x)$ or some soft version of leaky ReLU.

[1] The minimum operator can be implemented by $\min_j\{x_j\} = -\max_j\{-x_j\}$.

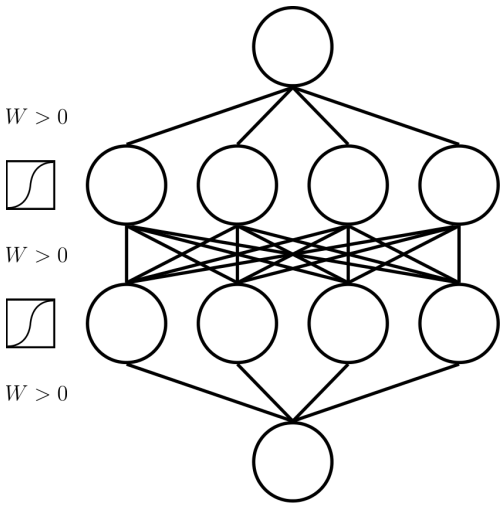


Figure 3.2: Monotone neural network with positive weights and increasing activations.

To be more concrete, lets look at a specific instantiation of monotone network called the *deep sigmoidal flow* (DSF):

$$f(x) = \sigma^{-1}\left(\sum_j w_j \sigma(a_j x + b_j)\right), \tag{3.2}$$

where $0 < w_j < 1$, $\sum_j w_j = 1$ and $a_j > 0$.

Since all of the sigmoid activations are bounded between 0 and 1, so is the final preactivation (which is their convex combination). The complete DSF transformation can starts from mapping the original random variable to a different space through an activation function, where doing affine/linear operations is non-linear in the original variable. We then map it back through the inverse activation, which guarantees the overall transformation is surjective.

When stacking multiple sigmoidal transformation, we realize it resembles an MLP with a bottleneck, as shown by Figure 3.3 (left). We can generalize it to the *deep dense sigmoidal flow* (DDSF), which takes the form of an MLP without a bottleneck (Figure 3.3, right).

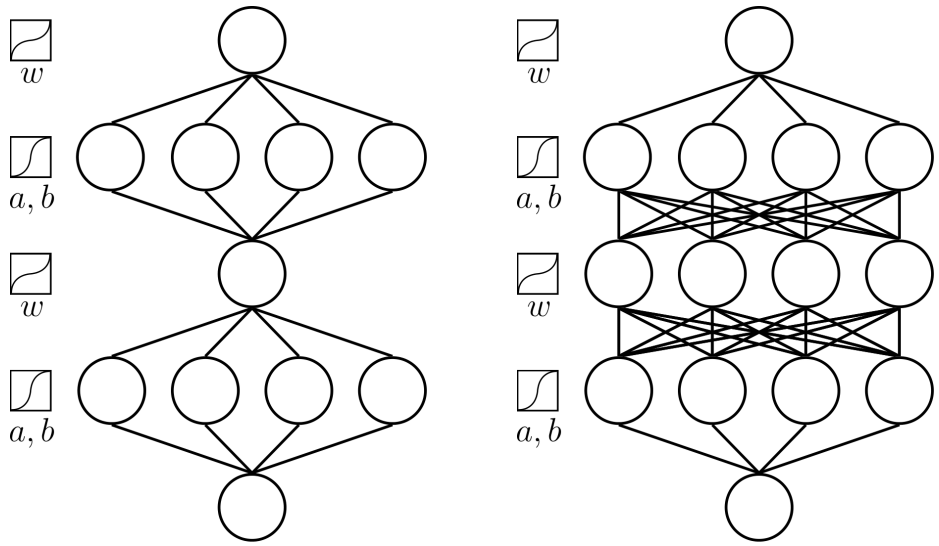


Figure 3.3: Deep sigmoidal flows (DSF) on the left and deep dense sigmoidal flows (DDSF) on the right.

In this 1D case, computing the log-determinant (logdet) can be easily achieved via standard automatic differentiation library by taking the log derivative of the function. However, to avoid numerical issues such as underflow, it is better to incorporate numerical tricks such as log-sum-exp to the chain rule. More details can be found in Appendix C of Huang et al. [2018a].

When sampling is required, one has to invert monotone neural networks numerically since a closed-form expression for inversion does not generally exist. This can be done by applying root-finding algorithms such as bisection search or iterative methods like Newton's method and its variants [Dahlquist and Björck, 2008, Chapter 6].

Lastly, it is essential to address the expressive power of DSF. This discussion serves to remind the reader of the fundamental challenge we face in designing generative models that strike a balance between representational richness and computational feasibility. First, note that when using DSF with a Uniform prior, we do not need the logit transform that maps from $(0, 1)$ to $\mathbb{R}$ in (3.2), since the support over the Uniform distribution is $(0, 1)$. If we drop the logit, the monotone transform is in fact just the CDF of a Logistic mixture distribution. This suggests a connection between mixture models and DSF-like

monotone flows can be made, and that we can study the expressivity of DSF by studying the expressivity of mixture densities.

More generally, let $\sigma : \mathbb{R} \to (0, 1)$ be a continuous, increasing activation function, which is not necessarily the logistic sigmoid function. We consider monotone functions from $\mathbb{R}$ to $(0, 1)$ of the form $x \mapsto \sum_{j=1}^{K} w_j \sigma(a_j x + b_j)$ with $w_j, a_j > 0$ and $\sum_j w_j = 1$, and call flows of this type *finite-sum monotone flows*. If the range of σ is $(0, 1)$, the finite-sum flow can be seen as the CDF of a mixture density. We can show that monotone flows of this form are universal approximators of distributions in 1D. First, we look at how mixture densities can be used to approximate arbitrary density functions.

Let q be a probability density function, and let σ_q be its distribution function $\sigma_q(x) = \int_{-\infty}^{x} q(y)dy$. We will show that with any activation function σ_q constructed this way, finite-sum monotone flows can approximate arbitrary monotonic function with range $(0, 1)$. We let $\mathcal{M}_q$ denote the mixture density of the form[2]

$$\sum_{j=1}^{K} w_j a_j q(a_j x + b_j),$$

with w_j's being a partition of unity, and $a_j > 0$. Let $\mathcal{P}$ denote the set of 1D probability density functions. The following theorem shows that with suitable smoothness condition on q, $\mathcal{M}_q$ is dense in $\mathcal{P}$ wrt the L^1 metric (*i.e.* the total variation of probability measures).

Theorem 7 (Universality of mixture models). *If $q \in C^2 \cap \mathcal{P}$, then* $\mathrm{Cl}(\mathcal{M}_q) = \mathcal{P}$.

This means that we can choose a sequence of mixture densities that gets arbitrarily close to any target probability density function. As finite-sum monotone flows are the integrals of mixture densities, they can be shown to approximate any smooth monotone functions (specifically, CDF functions) pointwise.

[2]The extra a_j term is a result of chain rule when we differentiate the finite-sum monotone flow using σ_q as the activation function, and without it, the resulting density would not integrate to one. It can be seen as the change of volume caused by the linear map $x \mapsto a_j x + b_j$.

Corollary 8 (Universality of finite-sum monotone flows). *If $q \in C^2 \cap \mathcal{P}$, then for any cumulative distribution function g, there exists a sequence of finite-sum monotone flows f_n with activation function σ_q that converges to g pointwise almost everywhere as $n \to \infty$.*

In other words, finite-sum monotone flows (including DSF and DDSF) can universally transform any distribution with a continuous CDF into a uniform distribution. Note that even though the above corollary does not require the CDF to be continuous, we will not be able to transform a discontinuous CDF into a uniform distribution over $[0, 1]$ since it contains atoms[3].

Sum-of-squares polynomial Another parametric family of functions that are classically used for approximation are polynomials. It is well-known that polynomials are universal function approximators, a result known as the Weierstrass approximation theorem [Folland, 1999, Section 4.7], but just like neural networks, constraints need to be imposed to the coefficients to make polynomials strictly increasing. For example, Jaini et al. [2019] proposed to parameterize an increasing polynomial function as follows:

$$f(x) = b + \int_0^x \sum_{j=1}^{k} \left(\sum_{i=0}^{r} a_{i,j} u^i \right)^2 \mathrm{d}u, \tag{3.3}$$

which is known as the *sum-of-squares polynomial* (SOS) flow. As the integrand is nonnegative, SOS flow is necessarily increasing. In practice, this integral can also be computed exactly since it is an integral of a univariate polynomial function, which corresponds to a higher order polynomial with constraints on the coefficients.

[3]However, the density of the generative flow is still capable of approximate point masses, which is implied by Theorem 7 and the fact that point masses are just the limit of Gaussians with vanishing variance.

Splines Polynomial functions can be extended to piecewise polynomials, also called *splines*. This gives the function extra degrees of freedom and allows us to avoid using higher order polynomials which might not have an analytic inverse. The use of monotone piecewise polynomials for parameterizing 1D flows was first proposed in Müller et al. [2019], where the authors explored the idea of piecewise quadratic functions. This was then extended to cubic spline flows proposed by Durkan et al. [2019a], and then to neural spline flows [Durkan et al., 2019b] based on rational quadratic splines with a better numerical stability control.

3.2 PARAMETERIZATION BY INTEGRATION

Besides direct parameterization, one can also parameterize monotone functions indirectly. One way to do so is to notice that the derivative of a strictly increasing function is strictly positive. In light of this insight, Wehenkel and Louppe [2019] proposed to parameterize f as

$$f(x) = b + \int_0^x f'(u)\,\mathrm{d}u, \tag{3.4}$$

where f' is a strictly positive parametric function (*e.g.* a neural network with a softplus activation at the final layer). The derivative needed for computing the likelihood is simply $f'(x)$. This also means we gain one order of smoothness by the integration: if f' is continuous, then the integral flow f will induce a continuous density.

Unlike the sum-of-squares polynomial flow (3.3), which is a special case of this, the integral in general does not have an analytic form. In practice, both the forward evaluation of the function $f(x)$ and the backward evaluation of its gradient wrt parameters would rely on numerical integration methods.

3.3 PARAMETERIZATION BY DIFFERENTIATION

A second indirect parameterization of monotone function is through differentiation, since a monotone function is characterized by a convex integral (antiderivative).

To parameterize a strictly increasing function f, we just need to differentiate a strictly convex function F, which means

$$f(x) = \frac{\mathrm{d}}{\mathrm{d}x} F(x). \tag{3.5}$$

Just like monotone neural networks, this only guarantees injectivity. To ensure surjectivity, we would require the integral F to be *strongly* convex. Let us defer more detailed discussion of this to 5.2, and focus on the parameterization of a convex function F for the moment.

Input-convex neural networks Design of scalar-output neural network architectures convex wrt to the input variable is still an under-explored topic. Networks of this type are called *input-convex neural networks*[4] [Amos et al., 2017, ICNN]. We will address the more general case of convex functions of several variables (which means the input could be a vector). Note that for monotone neural networks, we only need to differentiate a $\mathbb{R} \to \mathbb{R}$ input-convex function. Multivariable input-convex functions can be used to generalize 1D monotone functions, as we will see in 5.2.

The key building blocks of ICNN are operators that preserve convexity [Boyd et al., 2004, Section 3.2]:

C1 **Non-negative weighted sums**: if $F_1, ..., F_k$ are all convex, then $\sum_k^K a_k F_k$ is also convex if the weights a_k are all non-negative.

[4] The term "input-convex" is to contrast convexity wrt the parameters.

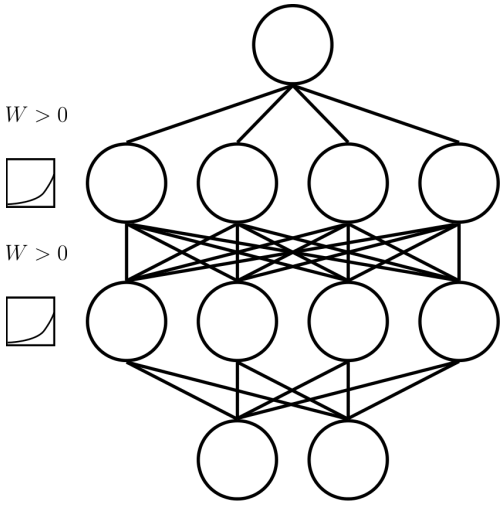


Figure 3.4: Input-convex neural network with positive weights (beyond the first layer) and increasing, convex activations.

C2 Composition: if $F_1 : \mathbb{R}^d \to \mathbb{R}$ and $F_2 : \mathbb{R} \to \mathbb{R}$ are both convex and F_2 is non-decreasing, then $F_2 \circ F_1$ is convex.

Using these building blocks, we can construct an ICNN (see Figure 3.4) as follows:

Theorem 9 (Input-convex neural network). *A multiple-layer perceptron (1.5) is (strictly) convex wrt the input variable if*

1. *all the activation functions are (strictly) increasing and convex, and*
2. *all the weight parameters beyond the first layer are (strictly) positive.*

Proof. This follows from **C1**, **C2** and the fact that the first layer is an affine map, which is convex. □

There are a few possible relaxations of vanilla ICNN satisfying the conditions in Theorem 9: (1) the activation function used for the first layer does not need to be increasing, as long as it is convex, (2) skip connection from the input layer to any preactivation is allowed without constraint on weights, (3) any intermediate layer can

be augmented with skip-connected units from the input layer with regular convex activation, etc. The original ICNN proposed by Amos et al. [2017] explored (2), and (1,3) were explored by Huang et al. [2021a].

Soft-max operation Another well-known operator that preserves convexity is the pointwise maximum operator.

C3 Pointwise maximum: if $F_1, ..., F_K$ are all convex, then $\max_k F_k$ is also convex.

However, the maximum operator is not everywhere differentiable, so the derivative (gradient) is not continuously defined. Furthermore, simply taking the maximum of multiple affine functions will not give us a "strictly" convex function, which makes it undesirable for flows. To fix this problem, we can soften the maximum operator.

An idea is to use a **softmax** operator[5] called log-sum-exp defined as $x \in \mathbb{R}^K \mapsto \log \sum_k \exp(x_k)$. The log-sum-exp operator is a soft version of the pointwise maximum, as the function $x \in \mathbb{R}^K \mapsto \frac{1}{\alpha} \log \sum_k \exp(\alpha x_k)$ converges to $\max_k x_k$, uniformly in x as $\alpha \to \infty$. This is true due to the following inequality:

$$\max_k x_k \leq \frac{1}{\alpha} \log \sum_k \exp(\alpha x_k) \leq \max_k x_k + \frac{1}{\alpha} \log K. \tag{3.6}$$

Finally, lets consider the log-sum-exp as an aggregator of convex functions.

C3' log-sum-exp: if $F_1, ..., F_K$ are all convex, then $\log \sum_k \exp(F_k)$ is also convex.

[5]Note that this differs from the softmax activation. In face, the softmax activation function acts more like a soft version of argmax, and is the gradient of the log-sum-exp.

Proof. We prove it for the case of differentiable F_k. To show log-sum-exp preserves convexity, we check the first order condition. Let $F := (F_k)_k$ denote the vector of convex functions. For any x and y, since log-sum-exp itself is convex,

$$\begin{aligned}\log \sum_k \exp(F_k(x)) - \log \sum_k \exp(F_k(y)) &\geq \nabla_F \left(\log \sum_k \exp(F_k(y)) \right)^\top (F(x) - F(y)) \\ &= \sum_k \left(\frac{e^{F_k}}{\sum_{k'} e^{F_k'}} \cdot (F_k(x) - F_k(y)) \right).\end{aligned}$$

Because $F_k(x) - F_k(y)$ is weighted by a positive coefficient, applying the convexity of F_k yields

$$\geq \sum_k \left(\frac{e^{F_k}}{\sum_{k'} e^{F_k'}} \cdot \nabla_y F_k(y)^\top (x - y) \right).$$

Now note that the first few terms in the RHS is exactly the gradient of the composed function

$$\nabla_y \left(\log \sum_k \exp(F_k(y)) \right) = \sum_k \left(\frac{e^{F_k}}{\sum_{k'} e^{F_k'}} \right) \nabla_y F_k(y).$$

Putting everything together, we have showed that the composed function satisfies the first order condition of convexity:

$$\log \sum_k \exp(F_k(x)) - \log \sum_k \exp(F_k(y)) \geq \nabla_y \left(\log \sum_k \exp(F_k(y)) \right)^\top (x - y).$$

□

C3' tells us that we can substitute log-sum-exp for the max operator in a maxout network [Goodfellow et al., 2013], and restricting the weight parameters beyond the first layer to be strictly positive will give us another ICNN architecture.

3.4 DISCUSSION

Recall that from the beginning of the chapter, we learned that continuous invertible functions in 1D inherently possess monotonicity. Building from this insight, we proposed various ways to parameterize monotone functions. It is noteworthy that these architectures are provably universal in their capacity to approximate arbitrary increasing functions. As suggested by Lemma 5, they induce a universal class of probability distributions in 1D, highlighting the evident expressivity of these architectural choices. In the following two chapters, we will extend these principles to tackle the approximation of high-dimensional couplings, providing a framework to address the scalability challenges posed by the higher dimensions.

4 Triangle maps and neural autoregressive flows

The previous chapter looks at the 1D case and explores a simple coupling using the increasing rearrangement coupling, which can be approximated by monotone neural networks (as well as other parametric monotone functions such as splines). We might ask, how do we extend it to high-dimensional cases?

4.1 Knothe Rosenblatt rearrangement

A straightforward extension of the monotone rearrangement is to look at a conditional rearrangement scheme known as the Knothe-Rosenblatt (KR) Rearrangement [Rosenblatt, 1952, Knothe, 1957]. The KR map is a composition of two invertible maps. The first transforms the source measure into a uniform measure [Hyvärinen and Pajunen, 1999] using the conditional distribution function. The second transforms the uniform measure into the target measure using the inverse of the conditional CDF. An example of the conditional CDF transform can be found in Figure 4.1.

Importantly, the KR map is pointwise invertible if the measures μ and ν have strictly positive densities everywhere, since this implies the conditional CDFs are differentiable with positive derivatives, thus strictly increasing.

Furthermore, the KR map is a triangular map, which means the j'th output of the mapping only depends on the first $j-1$ input variables.

Knothe Rosenblatt rearrangement

Let μ be a probability measure absolutely continuous wrt the Lebesgue measure in $\mathbb{R}^d$. Let ν be a target probability measure we want to transform μ into. Then we can construct a coupling $KR_{\mu\to\nu} : \mathbb{R}^d \to \mathbb{R}^d$ as follows: Let μ_1 and ν_1 denote the marginal measure of the first coordinate. And define the transformation of the first coordinate as $KR_{\mu\to\nu}(x)_1 := \Phi^{-1}_{\nu_1}(\Phi_{\mu_1}(x_1))$. For $j > 1$, let

$$KR_{\mu\to\nu}(x)_j := \Phi^{-1}_{\nu_j(\cdot \,|\, KR_{\mu\to\nu}(x)_{1:j-1})}(\Phi_{\mu_j(\cdot \,|\, x_{1:j-1})}(x_j)), \tag{4.1}$$

where $\mu_j(\cdot \,|\, x_{1:j-1})$ and $\nu_j(\cdot \,|\, y_{1:j-1})$ denote the conditional probability measures. The formal treatment of conditional probability measures requires a concept called the disintegration of measures; see Ambrosio et al. [2008, Section 5.3]. For simplicity, μ_j can be seen as $\int_\cdot p(x_j|x_{1:j-1})\,\mathrm{d}x_j$, where the integrand is the conditional probability density. Furthermore, $KR_{\mu\to\nu}$ is a coupling; that is $(KR_{\mu\to\nu})_{\#}\mu = \nu$. See Santambrogio [2015, Section 2.3] for a proof.
Note that (4.1) can be equivalently defined as $KR_{\mu\to\nu} = KR^{-1}_{\nu\to\lambda} \circ KR_{\mu\to\lambda}$, where $KR_{\mu\to\lambda}$ is a vector field formed by the conditional CDF transforms that map μ into the uniform (Lebesgue) measure λ:

$$KR_{\mu\to\lambda}(x)_j := \Phi_{\mu(\cdot \,|\, x_{1:j-1})}(x_j). \tag{4.2}$$

The inverse of $KR_{\nu\to\lambda}$ might not exist, but we can still define it using the left inverse of the conditional CDFs (see Chapter 3):

$$(KR^{-1}_{\nu\to\lambda})(u)_j := \Phi^{-1}_{\nu_j(\cdot \,|\, (KR^{-1}_{\nu\to\lambda})(u)_{1:j-1})}(u_j). \tag{4.3}$$

It is not hard to show that composing $KR_{\mu\to\lambda}$ with $KR^{-1}_{\nu\to\lambda}$ yields the same mapping as $KR_{\mu\to\nu}$ defined in (4.1) using an induction argument.

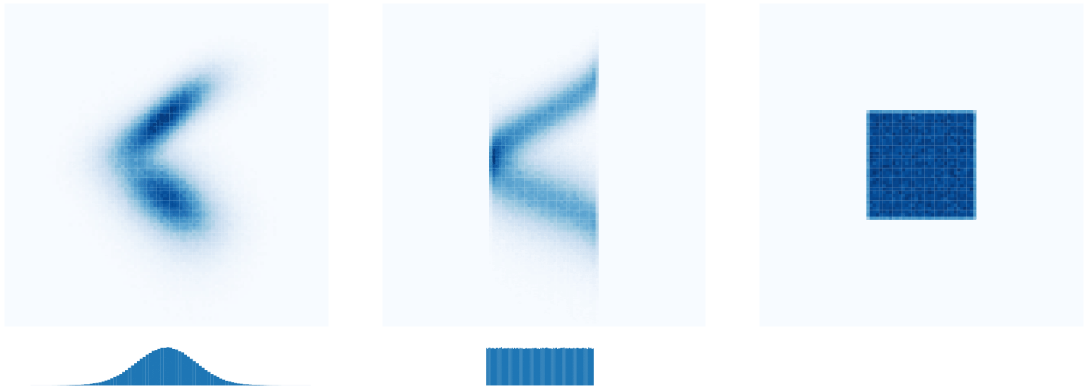


Figure 4.1: KR rearrangement. The distribution on the left is transformed into the middle one by uniformizing its x-coordinate (via the CDF transform). It is then transformed into the distribution on the right by uniformizing the y-coordinate given the x-coordinate (via the conditional CDF transform).

4.2 NEURAL AUTOREGRESSIVE FLOWS

Inspired by the KR map, we propose *Neural Autoregressive Flows* (NAF), an invertible neural network with a triangular Jacobian. NAF generalizes the affine autoregressive flows (1.8) by replacing the conditional affine transforms with more general monotone transforms. Formally, we define an autoregressive flow as[1]

$$f(x)_t = \tau(x_t; c(x_{1:t-1})), \tag{4.4}$$

where τ is a 1D (conditional) monotone **transformer network** introduced in the previous section used to transform the t'th variable, and c is a **conditioner network** that takes all the preceding $x_{1:t-1}$ as inputs and outputs the parameters of the 1D transformer.

Autoregressive conditioning The conditioner network can be parameterized by any autoregressive model that we have seen in 1.3, which will allow the model to apply

[1]We use the index t instead of j here to denote the index for the t'th feature (out of d features in total) since we are reserving j for indexing the hidden units of the monotone transformer, *e.g.* DSF.

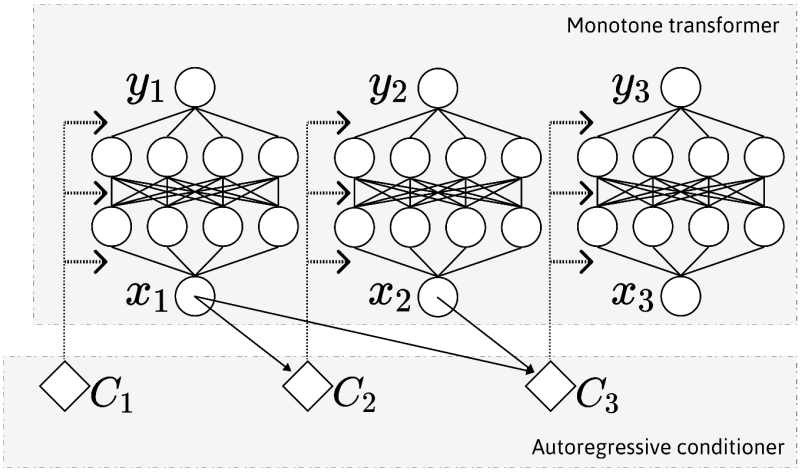


Figure 4.2: Neural autoregressive flow is an invertible map $y = f(x)$ defined by (4.4) consisting of an autoregressive conditioner and a monotone transformer (see Chapter 3).

masks (or a recurrent structure) to induce the autoregressive dependency. As the conditioner outputs the parameters of the transformer (Figure 4.2), NAF falls into the hypernetwork framework [Ha et al., 2017].

A different approach is to integrate the monotone transformer into the autoregressive network. This way, the overall mapping has a more compact representation. De Cao et al. [2020], for instance, introduces block-autoregressive conditioning, where each block of the autoregressive network represents the monotone transformation of one feature. They call the resulting model *Block Neural Autoregressive Flow* (B-NAF).

Comparison to Affine Autoregressive Flows While affine transforms require information about multi-modality in $f(x)_t$ to flow through the autoregressive conditioning (since an affine transform itself cannot change density non-uniformly), NAFs are able to induce multi-modality more easily, via inflection points in the nonlinear monotone transform. This is shown in Figure 4.3. Intuitively, τ is analogous to a CDF, so that its derivative corresponds to a PDF and its inflection points yield local maxima or minima. More formally, inflection points are where the function changes concavity. This means

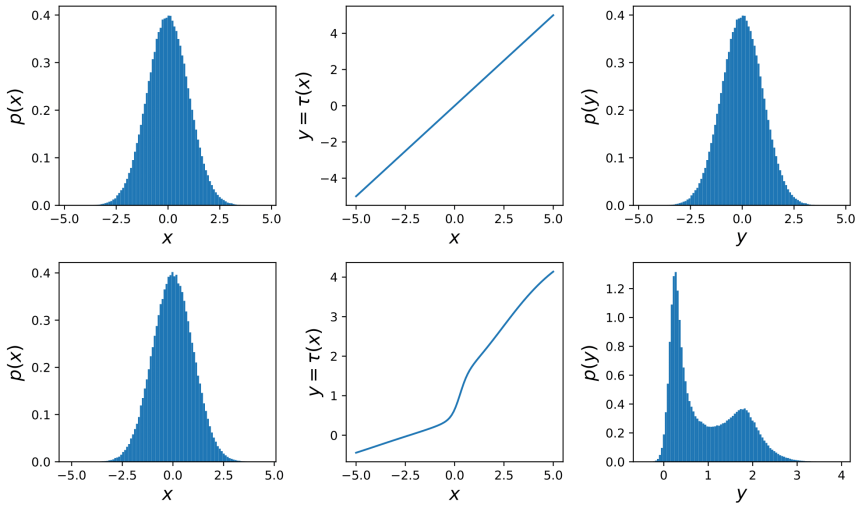


Figure 4.3: Illustration of 1D affine transform (*top*) vs non-linear monotone transform (*bottom*). An affine transformation does not have the cabability to split mode since its curvature (second order derivative) is constant. A non-linear monotone transform on the other hand can split a mode by being steep at some region (*i.e.* expanding the space) and increase the density by decreasing the slope (*i.e.* contracting the space). Therefore, the latter can induce multi-modality.

the nearby input values on the two different sides of the inflection point will be mapped in different directions, thereby causing a maximal local contraction or expansion of volume. We conduct more experiments to demonstrate this point in 4.6.

4.3 INVERTIBILITY AND INVERSION

Like AAF, NAF can be inverted sequentially, but unlike AAF, the conditional monotone transformation might not have an analytic inverse function. This is the case for DSF or DDSF, in particular, where iterative methods are needed for numerical inverting the function. We will also see in 5.3 that if the monotone flow can be parameterized as the derivative of a convex potential, we can invert it via convex optimization.

Another approach is to invert the mapping entirely, by applying a fixed-point type

of algorithm, which is potentially faster than autoregressive sampling. For example, suppose we want to find some x^* such that $y = f(x^*)$. For any x and a learning rate parameter $\alpha > 0$, we have $x^* = x^* - \alpha(f(x^*) - y)$. This means the inversion point x^* is a fixed point of the mapping $T(x) := x - \alpha(f(x) - y)$. The idea is to apply this operator iteratively and hope that the algorithm will converge. Under different assumptions on the structure of f (such as triangular structure, residual structure, and Lipschitzness), various variants of the iterative operator T have been designed with different kinds of convergence guarantees [Behrmann et al., 2019, Song et al., 2019, Wiggers and Hoogeboom, 2020, Song et al., 2021b].

4.4 LIKELIHOOD COMPUTATION

Due to the autoregressive structure of (4.4), we can significantly simplify the computation of the determinant of the Jacobian matrix, which is a triangular matrix now. Specifically, this is because the t'th output $f(x)_t$ depends only on the preceding variables $x_{t'}$ for $t' \leq t$, which means $\frac{\partial}{\partial x_{t'}} f(x)_t = 0$ for $t' > t$. Thus, the upper half of the matrix is filled with zeros, and its determinant is simply the product of the diagonal entries, which correspond to the elementwise derivatives $\frac{\partial}{\partial x_t} f(x)_t = \frac{\partial}{\partial x_t} \tau(x_t; c(x_{1:t-1}))$. This reduces the $\mathcal{O}(d^3)$ computation of the determinant to $\mathcal{O}(d)$.

4.5 UNIVERSALITY

In this section, we prove that NAFs, specifically with DSF as the transformer (which we will call NAF-DSF for brevity), can be used to approximate any probability distribution over real vectors arbitrarily well, given that τ has enough hidden units output by generic

neural networks with autoregressive conditioning. The idea of proving the universal approximation theorem, as described in Chapter 2, is to approximate a universal coupling (which is the KS map in this case) between the input and target probability measures.

Recall NAF-DSF is defined as

$$f(x)_t = \sigma^{-1}\left(\sum_{j=1}^{h} w_{tj}(x_{1:t-1})\sigma(a_{tj}(x_{1:t-1})x_t + b_{tj}(x_{1:t-1}))\right), \tag{4.5}$$

where $h > 0$ is the number of hidden units, and (w_{tj}, a_{tj}, b_{tj}) are output of an autoregressive conditioner network, which we assume satisfies the universal approximation property (*i.e.* Theorem 1). More generally, we consider sigmoid-like activation functions; *i.e.* σ is a strictly monotone C^1 function that goes from 0 to 1.

The following theorem generalizes Corollary 8 to high-dimensional cases.

Theorem 10 (**Universality of NAF-DSF**). *NAF-DSF of the form (4.5) are* $\mathfrak{A} - \mathfrak{P}$ *distributionally universal.*

4.6 EXPERIMENTS

We conduct experiments on variational inference and density estimation to compare with affine autoregressive flows (AAF) such as IAF and MAF.

Density estimation In Figure 4.4, we demonstrate that affine transformations can fail to fit multi-modal distributions due to the lack of expressivity, whereas NAF can easily do so thanks to the use of a non-linear monotone transformer. We choose 2D mixture of Gaussians as the target data distributions. We define the modes of the Gaussians to be laid out on a 2D grid within the range $[-5, 5]$, and consider 2, 5 and 10 modes on each

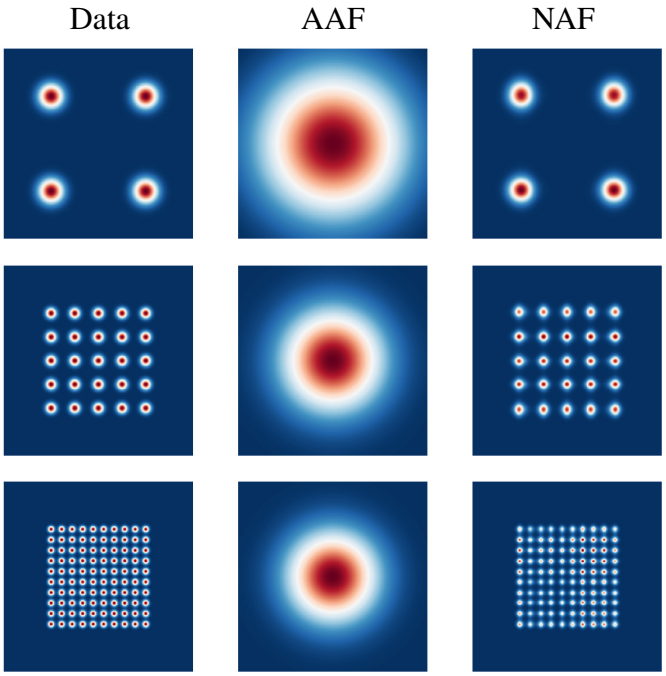


Figure 4.4: Fitting grid-of-Gaussian distributions using maximum likelihood. (*left*) true distribution. (*center*) affine autoregressive flow (AAF). (*right*) neural autoregressive flow (NAF)

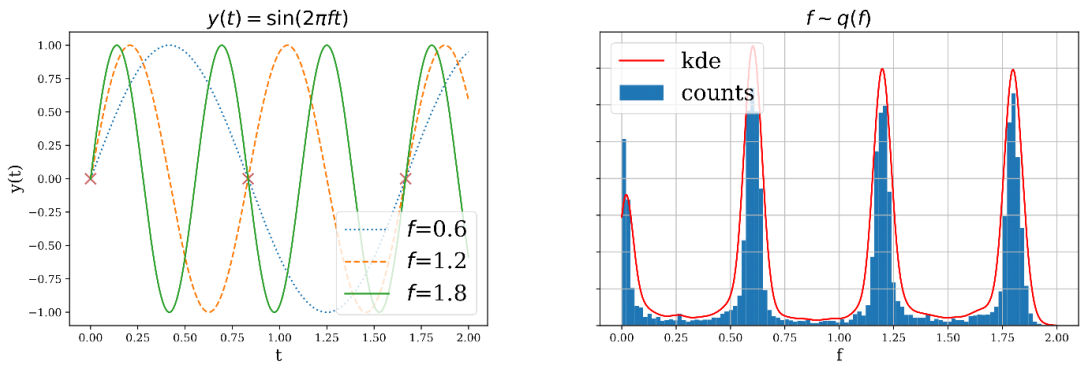


Figure 4.5: The DSF model effectively captures the true posterior distribution over the frequency of a sine wave. Left: The three observations (marked with red x's) are compatible with sine waves of frequency $f \in 0.0, 0.6, 1.2, 1.8$. Right: a histogram of samples from the DSF approximate posterior ("counts") and a Kernel Density Estimate of the distribution it represents (KDE).

dimension. While the affine flow only produces a single mode, the neural flow matches the target distribution quite well even up to a 10x10 grid with 100 modes in total.

To demonstrate NAF is capable of fitting real-world data distributions, we replicate the density estimation tasks proposed in Papamakarios et al. [2017], which consist of the BSDS300 dataset [Martin et al., 2001] as well as 4 UCI datasets [Lichman, 2013] processed the same way as Uria et al. [2013]. We only replace the affine transformer used in MAF with DDSF (recall Figure 3.3) in their best performing architecture for each task. Table 4.1 shows that this results in substantial performance gains, setting a new state-of-the-art for these tasks at the time of the publication of Huang et al. [2018a].

Variational inference Recall from Equation (1.18) that normalizing flows can be used to improve variational inference. Here we demonstrate that DSF is capable of capturing the multi-modal posterior distribution of a Bayesian regression model. To do so, we create a toy experiment where the goal is to infer the posterior over the frequency of a sine wave, given only 3 datapoints. This can be seen as a simpler case of the posterior estimation of a star-exoplanet model given some noisy observation of radial velocity, as done in Gabrié et al. [2021]. The observation function is $y(t) =$

Table 4.1: Test log-likelihood and error bars of 2 standard deviations on the 5 datasets (each with 5 runs). NAFs produce state-of-the-art density estimation results on all 5 datasets. The numbers (5 or 10) in parantheses indicate the number of transformations which were stacked. We also include TAN [Oliva et al., 2018], which was a concurrent work to NAF and focused more on the conditioner architecture design. We include their best results, achieved using different architectures on different datasets. We also include validation results to give future researchers a fair way of comparing their methods with ours during development.

Model	POWER	GAS	HEPMASS	MINIBOONE	BSDS300
MADE MoG	0.40 ± 0.01	8.47 ± 0.02	-15.15 ± 0.02	-12.27 ± 0.47	153.71 ± 0.28
MAF-affine (5)	0.14 ± 0.01	9.07 ± 0.02	-17.70 ± 0.02	-11.75 ± 0.44	155.69 ± 0.28
MAF-affine (10)	0.24 ± 0.01	10.08 ± 0.02	-17.73 ± 0.02	-12.24 ± 0.45	154.93 ± 0.28
MAF-affine MoG (5)	0.30 ± 0.01	9.59 ± 0.02	-17.39 ± 0.02	-11.68 ± 0.44	156.36 ± 0.28
TAN (various architectures)	0.48 ± 0.01	11.19 ± 0.02	-15.12 ± 0.02	-11.01 ± 0.48	157.03 ± 0.07
MAF-DDSF (5)	$\mathbf{0.62 \pm 0.01}$	11.91 ± 0.13	$\mathbf{-15.09 \pm 0.40}$	$\mathbf{-8.86 \pm 0.15}$	$\mathbf{157.73 \pm 0.04}$
MAF-DDSF (10)	0.60 ± 0.02	$\mathbf{11.96 \pm 0.33}$	-15.32 ± 0.23	-9.01 ± 0.01	157.43 ± 0.30
MAF-DDSF (5) valid	0.63 ± 0.01	11.91 ± 0.13	15.10 ± 0.42	-8.38 ± 0.13	172.89 ± 0.04
MAF-DDSF (10) valid	0.60 ± 0.02	11.95 ± 0.33	15.34 ± 0.24	-8.50 ± 0.03	172.58 ± 0.32

Table 4.2: Using DSF to improve variational inference. We report the likelihood and ELBO estimates of affine IAF with our implementation. We note that the negative log likelihood reported by Kingma et al. [2016] is 78.88. The average and standard deviation are carried out with 5 trials of experiments with different random seeds.

Model	-ELBO	$\log p(x)$
VAE	85.00 ± 0.03	81.66 ± 0.05
IAF-affine	82.25 ± 0.05	80.05 ± 0.04
IAF-DSF	$\mathbf{81.92 \pm 0.04}$	$\mathbf{79.86 \pm 0.01}$

$\sin(2\pi f \cdot t)$ and we impose a Uniform prior over the frequency: $p(f) = U([0, 2])$. The task is to infer the posterior distribution $p(f \mid T, Y)$ given an artificial dataset $(T, Y) = ((0, 5/6, 10/6), (0, 0, 0))$, as represented by the red crosses of Figure 4.5 (left). The dataset is chosen so as to create a multi-modal posterior distribution: it is likely generated by a sine wave function of frequency $f \in 0.0, 0.6, 1.2, 1.8$. We assume the data likelihood given the frequency parameter to be $p(y_i \mid t_i, f) = \mathcal{N}(y_i; y_f(t_i), 0.125)$, where the variance $\sigma^2 = 0.125$ represents the inherent uncertainty of the data. Figure 4.5 (right) shows that DSF learns a good posterior in this task.

We also consider a more challenging amortized inference task, by applying DSF to improve the approximate posterior of a VAE. In Table 4.2, we see that the DSF transformer outperforms both the affine transformer used in the standard IAF and the traditional factorized Gaussian posterior by a statistically significant margin.

4.7 DISCUSSION

In this chapter, we expand upon the foundation of monotone neural networks introduced in the preceding chapter to approximate high-dimensional triangular couplings. This extension not only retains the distributional universality but also benefits from an efficient likelihood computation due to the inherent triangular structure in the Jacobian matrix. To illustrate the potential of this framework, we introduce a specific architecture

called the neural autoregressive flow. Through empirical evidence, we showcase its efficacy in addressing complex density estimation and variational inference tasks, consistently achieving competitive state-of-the-art performance.

4.8 IMPACT, RELATED WORK AND RECENT DEVELOPMENTS

When NAF was originally proposed, it was meant to address the poor inductive bias of IAF [Kingma et al., 2016] (and its related affine transform family, including Papamakarios et al. [2017], Dinh et al. [2014, 2017]) for modeling multimodal distributions. The idea of generalizing the affine transform to arbitrary monotone functions was based off the workshop papers Huang et al. [2017b,a]. It is worth mentioning that Goodfellow [2016] also had a few conjectures about the universality of affine coupling flows (see Footnote 2 of the tutorial). Theorem 10 (first presented in Huang et al. [2018a]) is the first distributional universal result for discrete-time flow with an explicit, constrained parameterization that ensures invertibility. Following the workshop paper Huang et al. [2017b], the theorem was originally derived based on an existence result of the non-linear ICA problem [Hyvärinen and Pajunen, 1999], and the connection to the KR map was later made by [Jaini et al., 2019]. Beyond universality, the statistical consistency of triangular flows has been recently studied by Irons et al. [2022]. The tractability of triangle maps should also be attributed to Dinh et al. [2014], where the author made a connection between NICE and autoregressive models, and later on to Kingma et al. [2016] and Papamakarios et al. [2017], where the connection is made clearer. Interestingly, besides autoregressive flows, a connection between latent variable models such as hierarchical VAEs and autoregressive models can also be drawn [Child, 2020].

Being the first general-purpose, universal flow-based density model, NAF has been applied to many different problems. For example, Brouillard et al. [2020] applied NAF to discover the latent causal structured for intervened data. Learning causal structure from static data directly usually requires strong assumptions on the shape

of the distribution (such as unimodality, Gaussianity, etc.), but these assumptions can be relaxed when intervention is permitted, which allows for the use of more flexible families of density models. Within the context of causal estimation, NAF (specifically DSF) has also been used as a black-box generative model for semi-synthetic data simulation for causal estimator benchmarking [Neal et al., 2020].

The monotone neural network used in NAF falls into a greater class of parametric monotone transforms summarized in Chapter 3, and is the first among them to be used for flow-based density modeling, which may have inspired the subsequent works including Müller et al. [2019], Ziegler and Rush [2019], Durkan et al. [2019a,b], Jaini et al. [2019], Wehenkel and Louppe [2019]. Widely speaking, monotone flows have found applications in energy forecasting in power systems [Dumas et al., 2022], gravitational wave analysis [Wang et al., 2021], anomaly detection [Zhang et al., 2021], motion prediction and trajectory planning [Agarwal et al., 2020, Schöller and Knoll, 2021], variational quantum Monte Carlo for Bosonic matrix models [Rinaldi et al., 2021], improving speech vocoding [Gabryś et al., 2021], temporal point processes [Shchur et al., 2019], improving exploration in off-policy reinforcement learning [Ward et al., 2019, Mazoure et al., 2020], etc.

5 Optimal transport maps and convex potential flows

As we have seen in the previous section, one way to extend the CDF transform to high-dimensional problems is via the KR map. We might wonder if there are other couplings—or even better, if there are better couplings—or how couplings are compared in the first place. To answer these questions, we resort to optimal transport theory and draw inspiration from a specific notion of *optimal coupling* to parameterize an invertible map.

5.1 Optimal transport and Brenier's map

To measure the "efficiency" of a coupling, we first define a cost function $c(x, y)$, which measures how far a random particle has been displaced by the transport map. The *Monge problem* [Villani, 2008] pertains to finding the optimal transport map g that realizes the minimal cost on average

$$J_c(p_X, p_Y) = \inf_{\tilde{g}:\tilde{g}(X)\sim p_Y} \mathbb{E}_{X\sim p_X}[c(X, \tilde{g}(X))]. \tag{5.1}$$

A special and important example of the cost function is the squared euclidean distance $c(x, y) = ||x - y||^2$. In this case, the optimal cost is known as the (squared) 2-Wasserstein distance. The Monge problem of this particular cost function has an interesting solution, a celebrated theorem due to Brenier [1987, 1991], which gives us a nice and somewhat convenient characterization of the optimal transport map.

Brenier's theorem, optimal transport map

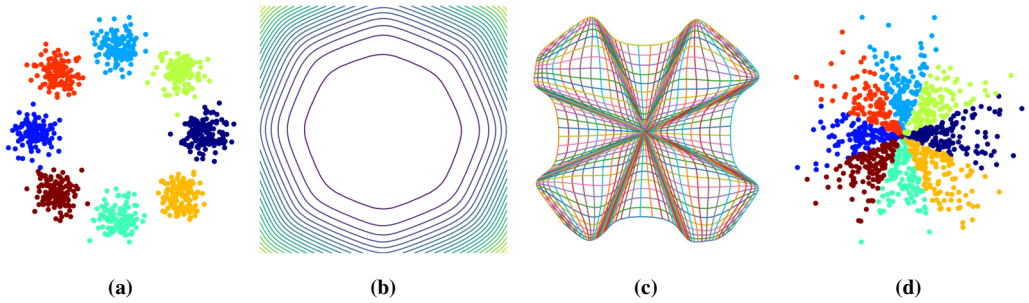
Let μ, ν be probability measures with a finite second moment, and assume μ has a Lebesgue density p_X. Then there exists a convex potential G such that the gradient map $g := \nabla G$ (defined up to a null set) uniquely solves the Monge problem in (5.1) with the quadratic cost function $c(x, y) = ||x - y||^2$ [Santambrogio, 2015, Theorem 1.22]. That is, g satisfies $g_{\#}\mu = \nu$ and

$$\mathbb{E}_{\mu}[\|X - g(X)\|^2] = \inf_{\widetilde{g}:\widetilde{g}_{\#}\mu=\nu} \mathbb{E}_{\mu}[\|X - \widetilde{g}(X)\|^2].$$

The theorem tells us that, to approximate any distribution, we just need to be able to approximate any convex function, since if we can do that well, we can approximate arbitrary optimal transport map by taking the gradient of the convex potential.

5.2 CONVEX POTENTIAL FLOWS

Inspired by Brenier's theorem, we propose to parameterize a flow using the gradient map of a convex potential. This is visualized in Figure 5.1. But how do we guarantee the flow is invertible? Is there a procedure to invert the flow? How do we compute the log-determinant of the Jacobian of the flow and its gradient for model evaluation and optimization? In this section, we answer these questions by casting the problems of model inversion and log-probability (gradient) estimation as convex optimization problems. This allows us to design numerical procedures to compute these quantities of interest. Finally, we provide a few more theoretical analyses, including the universality and optimality of the flow.



(a) (b) (c) (d)

Figure 5.1: Illustration of Convex Potential Flow. (a) Data x drawn from a mixture of Gaussians. (b) Learned convex potential F. (c) Mesh grid distorted by the gradient map of the convex potential $f = \nabla F$. (d) Encoding of the data via the gradient map $z = f(x)$. Notably, the encoding is the *value of the gradient* of the convex potential. When the curvature of the potential function is locally flat, gradient values are small and this results in a contraction towards the origin.

5.3 INVERTIBILITY AND INVERSION

The convex potential F can be parameterized via the ICNN discussed in Section 3.3, which itself is strictly convex. We can make it α-strongly convex by adding a quadratic term $F(x) = \frac{\alpha}{2}||x||_2^2 + \tilde{F}(x)$, where $\tilde{F}$ is the ICNN, such that $H_F \succeq \alpha I \succ 0$. The reason for this parameterization, as shown by the following theorem, is to obtain a bijective gradient map.

Theorem 11 (Invertibility of convex potential flow)**.** *Let $F : \mathbb{R}^d \to \mathbb{R}$ be a C^2 strictly convex mapping. Then $f(x) := \nabla F(x)$ is injective. If F is furthermore strongly convex, then $x \mapsto F(x) - y^\top x$ has a unique minimizer, which implies $f(x)$ is surjective.*

Since F is strongly convex, the gradient map is invertible from $\mathbb{R}^d$ to $\mathbb{R}^d$. We refer to this invertible mapping f as the *convex potential flow*, or the CP-Flow. Also, the potential $F(x) - y^\top x$ has a unique minimizer x^* satisfying the first order condition $\nabla F(x^*) = y$. This implies we can plug in a black-box convex solver to invert the gradient map f, which we summarize in Algorithm 1. Inverting a batch of independent inputs is as simple as summing the convex potential over all inputs: since all of the entries of the scalar l in the minibatch are independent of each other, computing the gradient all l's wrt all x's amounts to computing the gradient of the summation of l's

Algorithm 1 Inverting CP-Flow.

procedure INVERT($F, y,$ `CvxSolver`)
  Initialize $x \leftarrow y$
  def `closure`():
    Compute loss: $l \leftarrow F(x) - y^\top x$
    return l
  $x \leftarrow$ `CvxSolver`(`closure`$, x$)
  return x

wrt all x's. Due to the convex nature of the problem, a wide selection of algorithms can be used with convergence guarantees [Nesterov, 1998]. In practice, we use the *L-BFGS* algorithm [Byrd et al., 1995] as our `CvxSolver`.

5.4 LIKELIHOOD AND LIKELIHOOD GRADIENT ESTIMATIONS

Likelihood estimation Following equation (1.6), computing the log density for CP-Flows requires taking the log determinant of a symmetric positive definite Jacobian matrix (as it is the Hessian of the potential). There exists numerous works on estimating spectral densities (*e.g.* Tal-Ezer and Kosloff, 1984, Silver and Röder, 1994, Han et al., 2018a, Adams et al., 2018), of which this quantity is a special case. See Lin et al. [2016] for an overview of methods that only require access to Hessian-vector products. Hessian-vector products (hvp) are cheap to compute with reverse-mode automatic differentiation [Baydin et al., 2017], which does not require constructing the full Hessian matrix and has the same asymptotic cost as evaluating F_α.

In particular, the log determinant can be rewritten in the form of a generalized trace $\operatorname{Tr} \log H$. Chen et al. [2019a] limit the spectral norm (*i.e.* eigenvalues) of H and directly use the Taylor expansion of the matrix logarithm. Since our H has unbounded eigenvalues, we use a more complex algorithm designed for symmetric matrices, the *stochastic Lanczos quadrature* (SLQ; [Ubaru et al., 2017]). At the core of SLQ is the Lanczos

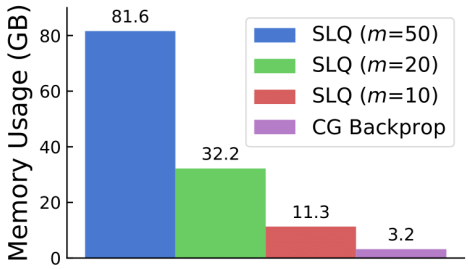


Figure 5.2: Memory for training CIFAR-10.

method, which computes m eigenvalues of H by first constructing a symmetric tridiagonal matrix $T \in \mathbb{R}^{m \times m}$ and computing the eigenvalues of T. The Lanczos procedure only requires Hessian-vector products, and it can be combined with a stochastic trace estimator to provide a stochastic estimate of our log probability. We chose SLQ because it has shown theoretically and empirically to have low variance [Ubaru et al., 2017].

Likelihood gradient estimation We would also like to have an estimator for the *gradient* of the log determinant to enable variants of stochastic gradient descent for optimization. Unfortunately, directly backpropagating through the log determinant estimator is not ideal. Two major drawbacks of directly differentiating through SLQ are that it requires (i) differentiating through an eigendecomposition routine and (ii) storing all Hessian-vector products in memory (see Figure 5.2). Problem (i) is more specific to SLQ, because the gradient of an eigendecomposition is not defined when the eigenvalues are not unique [Seeger et al., 2017]. Consequently, we have empirically observed that differentiating through SLQ can be unstable, frequently resulting in NaNs due to the eigendecomposition. Problem (ii) will hold true for other algorithms that also estimate $\log \det H$ with Hessian-vector products, and generally the only difference is that a different numerical routine would need to be differentiated through. Due to these problems, we do not differentiate through SLQ, but we still use it as an efficient method for monitoring training progress.

Algorithm 2 Surrogate training objective.

procedure SURROGATEOBJ($F, x,$ CG)
  Obtain the gradient $f(x) \triangleq \nabla_x F(x)$
  Sample Rademacher random vector r
  def hvp(v):
    return $v^\top \frac{\partial}{\partial x} f(x)$
  $z \leftarrow$ stop_gradient (CG(hvp, r))
  return hvp$(z)^\top r$

Instead, it is possible to construct an alternative formulation of the gradient as the solution of a convex optimization problem, foregoing the necessity of differentiating through an estimation routine of the log determinant. We adapt the gradient formula from Chen et al. [2019a, Appendix C] to the context of convex potentials. Using Jacobi's formula* and the adjugate representation of the matrix inverse†, for any invertible matrix H with parameter θ, we have the following identity:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\log\det H &= \frac{1}{\det H}\frac{\mathrm{d}}{\mathrm{d}t}\det H \\ &\overset{*}{=} \frac{1}{\det H}\mathrm{Tr}\left(\mathrm{adj}(H)\frac{\partial H}{\partial\theta}\right) \overset{\dagger}{=} \mathrm{Tr}\left(H^{-1}\frac{\partial H}{\partial\theta}\right) = \mathbb{E}_v\left[v^\top H^{-1}\frac{\partial H}{\partial\theta}v\right].\end{aligned} \tag{5.2}$$

In the last equality, we used the Hutchinson trace estimator [Hutchinson, 1989] with a Rademacher random vector v, leading to a $\mathcal{O}(1)$-memory, unbiased Monte Carlo gradient estimator.

Computing the quantity $v^\top H^{-1}$ in (5.2) by constructing and inverting the full Hessian requires d calls to an automatic differentiation routine and is too costly for our purposes. However, we can recast this quantity as the solution of a quadratic optimization problem

$$\arg\min_z \left\{\frac{1}{2}z^\top H z - v^\top z\right\}, \tag{5.3}$$

which has the unique minimizer $z^* = H^{-1}v$ since H is symmetric positive definite.

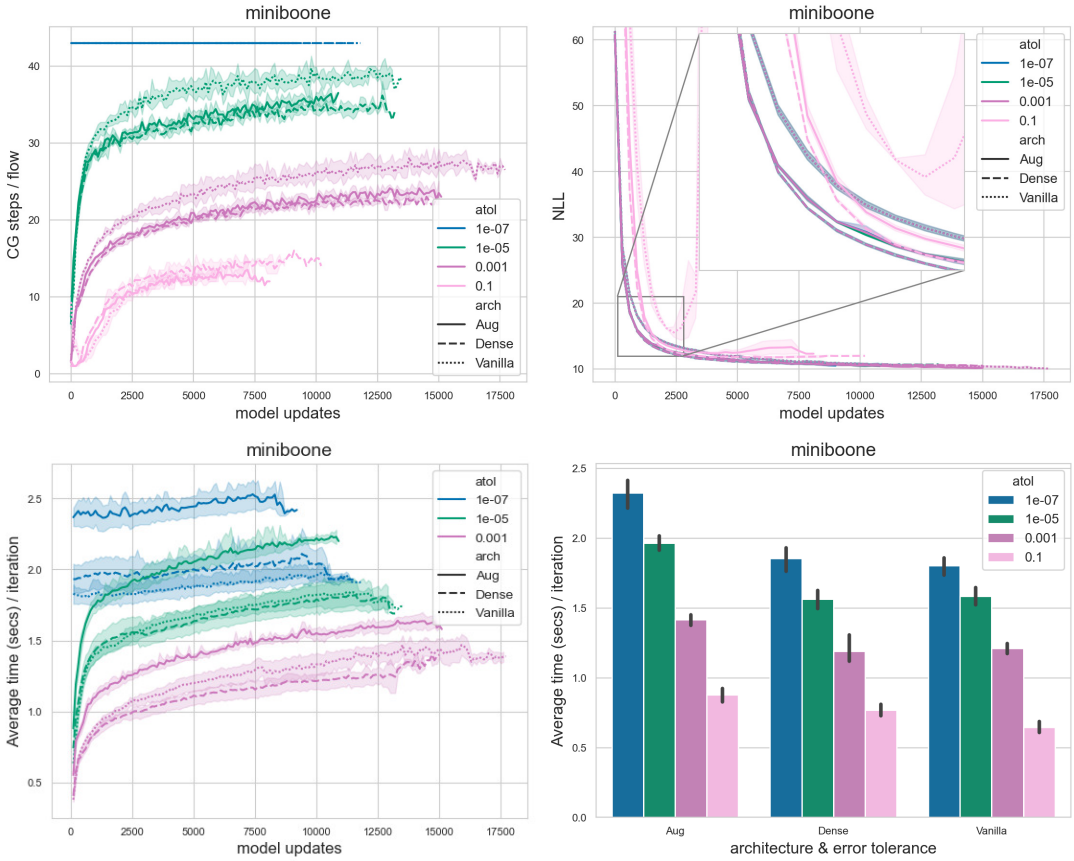


Figure 5.3: Analyzing different ICNN architectures and absolute error tolerance for conjugate gradient. "Vanilla" refers to the original ICNN proposed by Amos et al. [2017] with skip connection from the input layer. "Aug" refers to augmenting each intermediate layer with skip-connected units and "Dense" refers to a densely connected version of ICNN, both explored in Huang et al. [2021a]. The experiments are conducted on the MINIBOONE dataset introduced in Papamakarios et al. [2017]. (*left*) the average number of CG iterates (hvp calls) per flow layer (top row) and the corresponding average time (in seconds) per iteration (bottom row). (*top right*) validation set negative log-likelihood (exact estimate). Notice that, for atol $= 1\mathrm{e}{-7}$, CG iterations cap at 43 per flow layer; this is the dimensionality of the input data in MINIBOONE. (*bottom right*) per-iteration time (in seconds) averaged over all training steps.

We use the *conjugate gradient* (CG) method, which is specifically designed for solving the unconstrained optimization problems in (5.3) with symmetric positive definite H. It uses only Hessian-vector products and is straightforward to parallelize. Conjugate gradient is guaranteed to return the exact solution z^* within d iterations, and the error of the approximation is known to converge exponentially fast $||z^m - z^*||_H \leq 2\gamma^m ||z^0 - z^*||_H$, where z^m is the estimate after m iterations. The rate of convergence $\gamma < 1$ relates to the condition number of H. For more details, see Nocedal and Wright [2006, Ch. 5]. In practice, we terminate CG when $||Hz^m - v||_\infty < \tau$ is satisfied for some user-controlled tolerance. Empirically, we find that stringent tolerance values are unnecessary for stochastic optimization (see Figure 5.3).

Estimating the full quantity in (5.2) is then simply a matter of computing and differentiating a scalar quantity (a surrogate objective) involving another Hessian-vector product: $\frac{\mathrm{d}}{\mathrm{d}\theta}\left((z^m)^\top H v\right)$, where only H is differentiated through (since z^m is only used to approximate $v^\top H^{-1}$ as a modifier of the gradient). We summarize this procedure in Algorithm 2. Similar to inversion, the `hvp` can also be computed in batch by summing over the data index, since all entries are independent.

5.5 UNIVERSALITY, OPTIMALITY, AND CONNECTION TO TRIANGLE MAPS

Since the parameterization of CP-Flow is inspired by the Brenier potential, naturally we would hope to show that (1) CP-Flows are distributionally universal, and that (2) the learned invertible map is optimal in the sense of the average squared distance the input travels $\mathbb{E}[||x - f(x)||^2]$. Proofs of statements made in this section can be found in Section E.2.

Universality To show (1), our first step is to show that ICNNs can approximate arbitrary convex functions. However, convergence of potential functions does not generally imply convergence of the gradient fields. A classic example is the sequence $F_n = \sin(nx)/\sqrt{n}$ and the corresponding derivatives $f_n = \cos(nx)\sqrt{n}$: $F_n \to 0$ as $n \to \infty$ but f_n does not. Fortunately, convexity allows us to control the variation of the gradient map (since the derivative of a convex function is monotone), so our second step of approximation holds.

Theorem 12. *Let $F_n : \mathbb{R}^d \to \mathbb{R}$ be differentiable convex functions and $G : \mathbb{R}^d \to \mathbb{R}$ be a proper convex function. Assume $F_n \to G$. Then for almost every $x \in \mathbb{R}^d$, G is differentiable and $f_n(x) := \nabla F_n(x) \to \nabla G(x) =: g(x)$.*

Combining these two steps and Brenier's theorem, we show that CP-Flow with softplus-type activation function is distributionally universal.

Theorem 13 (Universality of CP-Flow). *Gradient maps of ICNN with softplus-type activation are $\mathfrak{A} - \mathfrak{P}$ distributionally universal.*

Remark 14. *In the theorem we do not require the second moment to be finite, as for arbitrary random variables we can apply the standard truncation technique and redistribute the probability mass so that the new random variables are almost surely bounded. For probability measures with finite second moments, we indeed use the gradient map of ICNN to approximate the optimal transport map corresponding to the Brenier potential.*

Optimality In the following theorem, we show that the optimal transport map is the only such mapping that we can approximate if we match the distributions.

Theorem 15 (Optimality of CP-Flow). *Let G be the Brenier potential of $X \sim \mu$ and $Y \sim \nu$, and let F_n be a convergent sequence of differentiable, convex potentials, such that $\nabla F_n \circ X \to Y$ in distribution. Then ∇F_n converges almost surely to ∇G.*

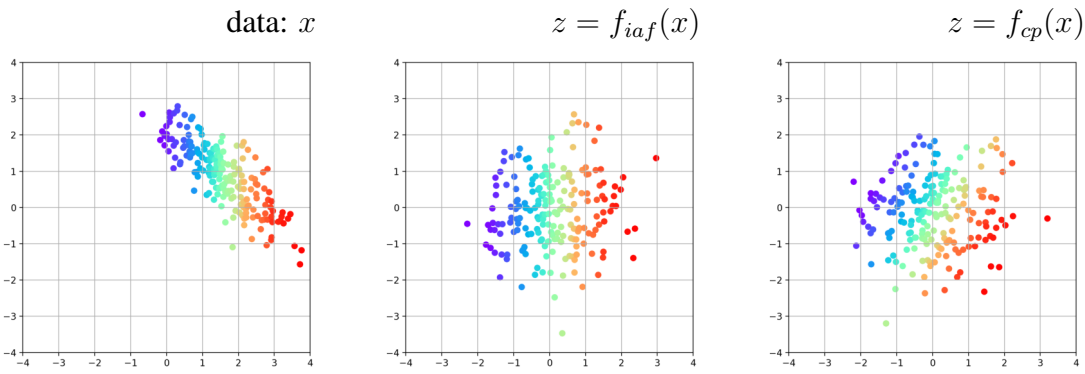


Figure 5.4: Approximating the optimal transport map via maximum likelihood (minimizing KL divergence). The datapoints are colored according to their horizontal values (x_1). The flows f_{iaf} and f_{cp} are trained to transform the data into a standard Gaussian prior. IAF transforms the first coordinate x_1 independently of x_2, and transform x_2 based on the value of the corresponding x_1, which causes rotation. CP-Flow on the other hand is rotation-free, being the gradient of a potential.

The theorem states that in practice, even if we optimize according to some loss that traces the convergence in distribution, our model is still able to recover the optimal transport map, as if we were optimizing according to the transport cost. This allows us to estimate optimal transport maps without solving the constrained optimization in (5.1). See Seguy et al. [2018] for some potential applications of the optimal transport map, such as domain adaptation or domain translation.

Connection to triangle maps As predicted by Theorem 15, CP-Flow is guaranteed to converge to the optimal coupling minimizing the expected quadratic cost. We empirically verify it by learning the Gaussian density and comparing the expected quadratic distance between the input and output of the flow against $J_{||x-y||^2}$ between the Gaussian data and the standard Gaussian prior (as there is a closed-form expression). In Figures 5.4 and 5.5, we see that the transport cost gets closer to the optimal value when the learned density approaches the data distribution (measured by the KL divergence). We compare against the linear inverse autoregressive flow [Kingma et al., 2016], which has the capacity to represent the multivariate Gaussian density, yet it does not learn the optimal coupling.

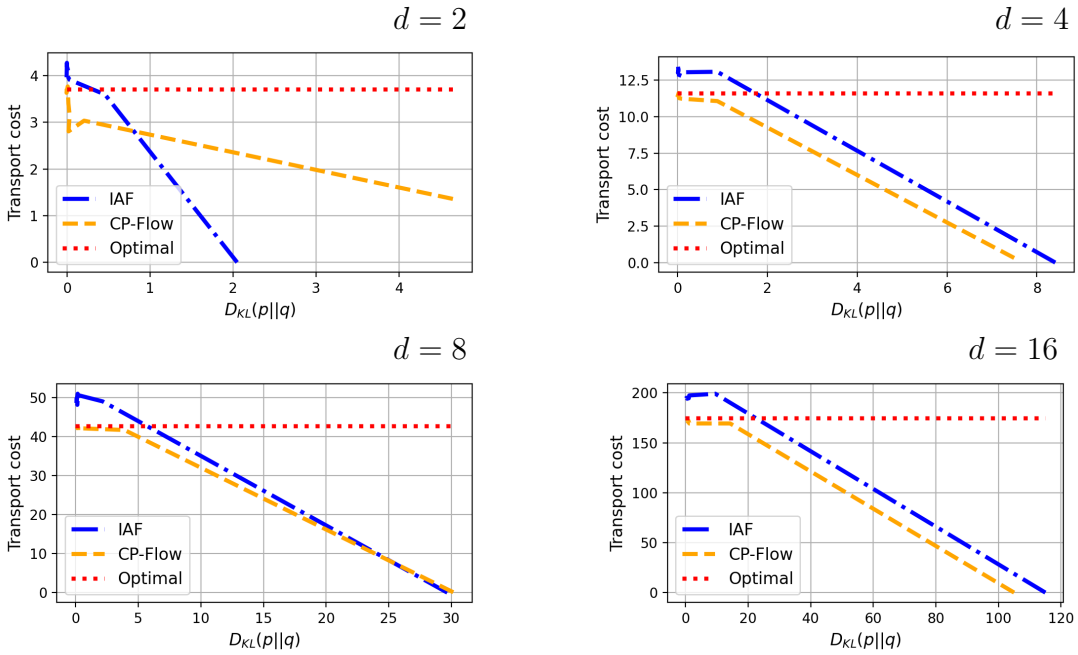


Figure 5.5: Approximating the optimal transport map via maximum likelihood (minimizing KL divergence). We plot the expected quadratic transportation cost versus the KL divergence for different numbers of dimensionality. During training the KL is minimized, so the curves read from the right to the left. The plots show that when KL is minimized, CP-Flow tends to find the transport map that has a smaller transportation cost. The dashed red lines indicate the optimal transport cost.

CP-Flow is to autoregressive flows (such as IAF and NAF) what ZCA whitening is to Cholesky whitening. ZCA whitening applies eigendecomposition to take the square root of the covariance matrix, which leads to a positive semidefinite linear transform (which is the gradient of a quadratic potential), whereas Cholesky whitening takes the Cholesky decomposition, which factorizes the covariance into a product of a lower triangular matrix with itself (which can be seen as an autoregressive linear transform). That is, performing MLE on CP-Flow amounts to performing ZCA whitening, just as performing MLE on linear autoregressive flow amounts to performing Cholesky whitening, up to a re-centering. The latter case has also been discussed in Kingma et al. [2016].

5.6 DISCUSSION

In this chapter, akin to the previous one, we extend the one-dimensional monotone network to higher dimensions. This time, we define the invertible map as the gradient of a convex potential, drawing inspiration from optimal transport theory. This choice naturally extends the notion of monotonicity and demonstrates the capability to approximate the gradient of any convex function, establishing the distributional universality of this class of architectures. Additionally, for efficient training, we introduce a novel constant-memory gradient estimator for log-likelihood using conjugate gradient methods, further enhancing the practicality of this framework.

5.7 IMPACT, RELATED WORK AND RECENT DEVELOPMENTS

Residual Flow For $\alpha = 1$, the gradient map f resembles the residual flow [Behrmann et al., 2019, Chen et al., 2019a]. Behrmann et al. [2019] require the residual block—equivalent to our gradient map f—to be *contractive* (with Lipschitz constant strictly

smaller than 1) as a sufficient condition for invertibility. In contrast, we enforce invertibility by using strongly convex potentials, which guarantees that the inverse of our flow is globally unique. With this, we do not pay the extra compute cost for having to satisfy Lipschitz constraints using methods such as spectral normalization [Miyato et al., 2018]. Our gradient estimator is also derived similarly to that of Chen et al. [2019a], though we have the benefit of using well-studied convex optimization algorithms for computing the gradients.

Sylvester Flow By restricting the architecture of our ICNN to one hidden layer, we can also recover a form similar to Sylvester Flows. For a 1-hidden layer ICNN ($K = 1$) and $\alpha = 1$, we have $F = \frac{1}{2}||x||_2^2 + L_2^+(s(L_1 x)) + L_2(x)$. Setting the weights of L_2 to zero, we have

$$f(x) = \nabla_x F(x) = x + W_1^\top \text{diag}(w_2^+) s'(W_1 x + b_1). \tag{5.4}$$

We notice the above form bears a close resemblance to the Sylvester normalizing flow [Van Den Berg et al., 2018] (with $\boldsymbol{Q}$, $\boldsymbol{R}$ and $\widetilde{\boldsymbol{R}}$ from Van Den Berg et al. [2018] being equal to $W_1^\top$, $\text{diag}(w_2^+)$ and I, respectively). For the Sylvester flow to be invertible, they require that $\boldsymbol{R}$ and $\widetilde{\boldsymbol{R}}$ be triangular and $\boldsymbol{Q}$ be orthogonal, which is a computationally costly procedure. This orthogonality constraint also implies that the number of hidden units cannot exceed d. This restriction to orthogonal matrices and one hidden layer are for applying Sylvester's determinant identity. In contrast, we do not require our weight matrices to be orthogonal, and we can use any hidden width and depth for the ICNN.

Sigmoidal Flow Let s be the softplus activation function and $\sigma = s'$. Then for the 1-dimensional case ($d = 1$) and $\alpha = 0$ (without the residual connection), we have

$$\begin{aligned}\frac{\partial}{\partial x} F_0(x) &= \sum_{j=1} w_{1,j} w_{2,j}^{+} \sigma(w_{1,j} x + b_{1,j}) \\ &= \sum_{j=1} |w_{1,j}| w_{2,j}^{+} \sigma(|w_{1,j}| x + \text{sign}(w_{1,j}) b_{1,j}) + \text{const.}\end{aligned}$$

which is equivalent to the sigmoidal flow we have seen in 3.1, up to a rescaling (since the weighted sum is no longer a convex sum) and a constant shift, and is monotone due to the positive weights. This correspondence is not surprising since a differentiable function is convex if and only if its derivative is monotonically non-decreasing.

Flows with Potential Parameterization Inspired by connections between optimal transport and continuous normalizing flows, some works [Zhang et al., 2018, Finlay et al., 2020a, Onken et al., 2020] have proposed to parameterize continuous-time transformations by taking the gradient of a scalar potential. They do not strictly require the potential to be convex since it is guaranteed to be invertible in the infinitesimal setting of continuous normalizing flows [Chen et al., 2018]. There exist works [Yang and Karniadakis, 2019, Finlay et al., 2020b, Onken et al., 2020] that have applied the theory of optimal transport to regularize continuous-time flows to have low transport cost. In contrast, we connect optimal transport with discrete-time normalizing flows, and CP-Flow is guaranteed by construction to converge pointwise to the optimal mapping between distributions without explicit regularization.

Parameterization by integration Analogous to the parameterization by integration of monotone functions (recall Section 3.2), convex gradients can also be modeled by integrating Hessian matrices (Jacobian of the flow) with a positive-definite structure [Lorraine and Hossain, 2019, Richter-Powell et al., 2021].

Equivariant flows This chapter motivates the parameterization of invertible flows via the optimal transport theory, which allows us to parameterize the flow as the gradient map of a scalar convex potential. This is convenient, not only because the architecture can be more compactly represented, but also because additional symmetry structures can be easily incorporated to design density models that are invariant to various kinds of transformation. Specifically, if the potential F is invariant to a group of unitary transformations $\mathcal{U}$, then for any $U \in \mathcal{U}$,

$$\nabla_x F(Ux) = U^\top \nabla F(Ux) = \nabla F(x),$$

where the first equality is implied by the chain rule, and the second equality is by the invariance assumption. This implies

$$f(Ux) = \nabla F(Ux) = UU^\top \nabla F(x) = Uf(x),$$

which means the flow map is $\mathcal{U}$-equivariant [Papamakarios et al., 2021, Lemma 2]. If furthermore, the prior density p is also invariant under $\mathcal{U}$, then the resulting density induced by the flow is also invariant [Köhler et al., 2020]. Designing invariant/equivariant neural architectures is an active research direction [Cohen and Welling, 2016, Zaheer et al., 2017, Satorras et al., 2021], which has direct impacts on incorporating symmetry structure into flow-based density models [Rezende et al., 2019, Köhler et al., 2020, Bose and Kobyzev, 2021, Garcia Satorras et al., 2021].

Extension to Riemannian manifolds Due to McCann [McCann, 2001] , Brenier's theorem can be generalized to optimal transport problems on Riemannian manifolds. This has motivated the generalization of CP-Flow to model probability distributions on manifolds [Cohen et al., 2021b, Rezende and Racanière, 2021].

Part II

Interlude: Improving expressivity via augmentation

Interlude: Improving expressivity via augmentation

The previous part of this thesis focuses on designing special structures in neural networks that satisfy the requirements & desiderata **RD1-4**; to wit, we want the network to be invertible, admit a cheap-to-compute Jacobian determinant, be easy to invert, and be sufficiently expressive for density approximation. In this part, we exposit a different approach to improve expressivity.

Aside from the unfortunate trade-off between the computational costs of the desired quantities and the expressivity of the invertible mappings, normalizing flows suffer from the limitation of local dependency. Unlike latent variable models such as VAEs and GANs which model the high-dimensional data as coordinates in a latent space, most generative flows model the dependency among features only locally. Dependencies of features far away from each other can only be propagated through composition of mappings, which progressively enlarge the receptive field. Special architectural designs like the attention mechanism are usually needed to address this issue [Ho et al., 2019a]. This is in part due to the requirement that the representation learned by an invertible map needs to preserve the dimensionality of the data. In this part of the thesis, we introduce an *augmentation* technique that allows us to construct invertible maps on an augmented state space, thereby relaxing this requirement, and to partition the augmented state space in creative manners. Specifically, we propose an instantiation of augmented flow that generalizes VAE, which allows us to manipulate and edit the generated samples and the data. We stress that this general augmentation framework is orthogonal to the specialized structure designs discussed in the previous part of the thesis, and they can be used conjunctively to improve the expressivity of the model.

6 State-space augmentation

Recall that normalizing flows are differentiable, bijective maps $f : \mathbb{R}^d \to \mathbb{R}^d$, where d is the dimensionality of the data. We also assume that the data distribution admits a Lebesgue density in $\mathbb{R}^d$ (otherwise the likelihood function can be potentially unbounded[1]). The technical requirement of normalizing flows, specifically the requirement that allows us to use the change-of-variable formula to compute the density of the data is that the input and output spaces of the mapping need to be of the same dimensionality so that volumes can be meaningfully compared. In this chapter, we construct an invertible model on an augmented input space, which when combined with the RealNVP (1.7) satisfies all criteria **RD1-4** from Part I. The motivation is that while regular normalizing flows are restricted to operate on the domain of the data, by making the augmented data more dependent on x, our model effectively represents x in a higher dimensional "lifted" space (see Figure 6.1). This has the additional benefit of sidestepping the topology-preserving property of a diffeomorphism [Dupont et al., 2019]. Moreover, our proposed method is inspired by and generalizes multiple variants of VAEs, and possesses the advantage of transforming the data in a more globally coherent manner via first embedding the data in the augmented state space.

[1] This could happen when the data lies in a lower dimensional manifold. There have been some recent development of designing flows on manifolds; see, for example, Brehmer and Cranmer [2020].

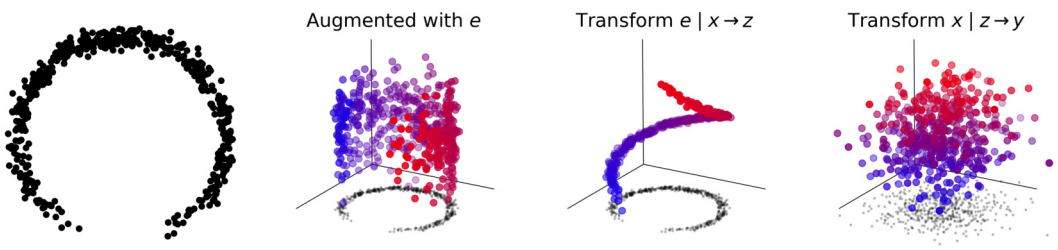


Figure 6.1: Transforming data x (*left*) via augmented normalizing flows: Black dots and blue dots are marginal and joint data points, respectively. *First step*: augment the data x with an independent noise e. *Second step*: transform the augmented data e conditioned on x into z. *Third step*: transform the original data x conditioned on z into y, resulting in a Gaussianized joint distribution of (y, z)

6.1 AUGMENTED MAXIMUM LIKELIHOOD

In this section, we generalize the maximum likelihood principle (recall Section 1.1) to *augmented maximum likelihood*. Estimating the likelihood of the *augmented data* allows us to sidestep the aforementioned topological constraints of invertible maps and to explore a larger family of models for better expressivity and more meaningful representation. For augmented maximum likelihood, we couple each data point with an independent auxiliary random variable $e \in \mathcal{E}$ drawn from $q(e)$ (which is usually taken to be $\mathcal{N}(0, I)$), and consider a family of joint density models $\{p_\pi(x, e) : \pi \in \mathfrak{P}(\mathcal{X} \times \mathcal{E})\}$. Instead of maximizing the marginal likelihood of x_i's, we maximize the expected joint likelihood:

$$\hat{\pi}_{\mathcal{A}} := \underset{\pi \in \mathfrak{P}(\mathcal{X} \times \mathcal{E})}{\arg\max}\ \mathbb{E}_{x,e}[\log p_\pi(x, e)], \tag{6.1}$$

where the expectation is over $(x, e) \sim \hat{q}(x)q(e)$. We refer to this extremum estimator as the *Augmented Maximum Likelihood Estimator* (AMLE). The benefit of maximizing the joint likelihood is that it allows us to make use of the augmented state space to induce structure on the marginal distribution of x in the original input space.

Lower bounding the log marginal likelihood Since the entropy of e is constant wrt the model parameter π, $\hat{\pi}_{\mathcal{A}}$ is equal to the maximizer of

$$\mathcal{L}_{\mathcal{A}}(\pi; x) := \mathbb{E}_e[\log p_\pi(x, e)] + H(e), \tag{6.2}$$

averaged over $x_i \sim \hat{q}(x)$. For any $x \in \mathcal{X}$, the quantity $\log p_\pi(x) - \mathcal{L}_{\mathcal{A}}(\pi; x)$ can be written as the KL divergence:

$$\log p_\pi(x) - \mathcal{L}_{\mathcal{A}}(\pi; x) = D_{\mathrm{KL}}(q(e)||p_\pi(e|x)).$$

Since KL is non-negative, maximizing the exact joint likelihood according to Equation (6.1) is equivalent to maximizing a lower bound on the log marginal likelihood of x. This means (6.2) is an ELBO, being a lower bound on the marginal likelihood (the evidence). We refer to the KL as the *Augmentation Gap*, as it reflects the incapability of the joint density to model e and x independently. As we will show in Section 6.2, $\mathcal{L}_A$ and the augmentation gap are related to the ELBO and the variational gap of VAE, and that e and the latent representation z of a VAE are simply reparameterization of one another. In fact, the expected joint likelihood objective (6.1) can be derived from the variational principle (1.14) by treating e as a latent variable and using q as a fixed approximate posterior [Chen et al., 2020].

Estimating the log marginal likelihood The log marginal likelihood $\log p_\pi(x)$ of the data can be estimated in a way similar to Burda et al. [2015], by drawing K *i.i.d.* samples of $e_j \sim q(e)$ per x to estimate the following stochastic lower bound:

$$\hat{\mathcal{L}}_{\mathcal{A},K}(\pi) := \log \frac{1}{K} \sum_{j=1}^{K} \frac{p_\pi(x, e_j)}{q(e_j)},$$

which can be shown to be a consistent estimator for $\log p_\pi(x)$ and is monotonically tighter in expectation as we increase K.

6.2 AUGMENTED NORMALIZING FLOWS

We now demonstrate how to leverage the augmented input space to model the complex marginal distribution of the data using **augmented normalizing flows (ANF)**. We consider maximizing the joint likelihood of x coupled with an independent random noise $e \sim q(e)$. For simplicity, we can choose $q(e)$ to be a standard Gaussian distribution. Second, we define a joint prior $p(y, z)$, which we also assume to be a standard Gaussian. Assume the data x, e is deterministically generated via an invertible mapping $x, e = F_\pi(y, z)$, with inverse $G_\pi = F_\pi^{-1}$. Then by the change of variable formula, x, e has a joint density

$$p_\pi(x, e) = \mathcal{N}(G_\pi(x, e); 0, I) \left| \det \frac{\partial G_\pi(x, e)}{\partial (x, e)} \right|.$$

In general, ANF generalizes normalizing flows in that if $\frac{\partial y}{\partial e} = 0$, $\frac{\partial z}{\partial x} = 0$ and $\frac{\partial z}{\partial e} = I$, $\mathcal{L}_\mathcal{A}$ is exactly the marginal likelihood $\log p(x)$ (since $p(z) = q(z)$) and $G_\pi(x, e)$ reduces to regular normalizing flow since the transformation $x \mapsto y$ is independent of e. This also means that to improve expressivity via augmentation, we need to harness the augmented state space $\mathcal{E}$ to induce a more complex marginal in $\mathcal{X}$.

6.3 AUGMENTED REALNVP

Inspired by the affine coupling (1.7) proposed by Dinh et al. [2017] and how a VAE [Kingma and Welling, 2014] correlates the latent code and the data, we conditionally transform x and e, hoping the structure in the marginal of x can "leak" into $\mathcal{E}$ and make the joint more easily Gaussianized. Concretely, we define two types of affine coupling:

$$\begin{aligned} g_\pi^{\text{enc}}(x, e) &= \texttt{concat}(x,\, s_\pi^{\text{enc}}(x) \odot e + m_\pi^{\text{enc}}(x)), \\ g_\pi^{\text{dec}}(x, e) &= \texttt{concat}(s_\pi^{\text{dec}}(e) \odot x + m_\pi^{\text{dec}}(e),\, e). \end{aligned} \tag{6.3}$$

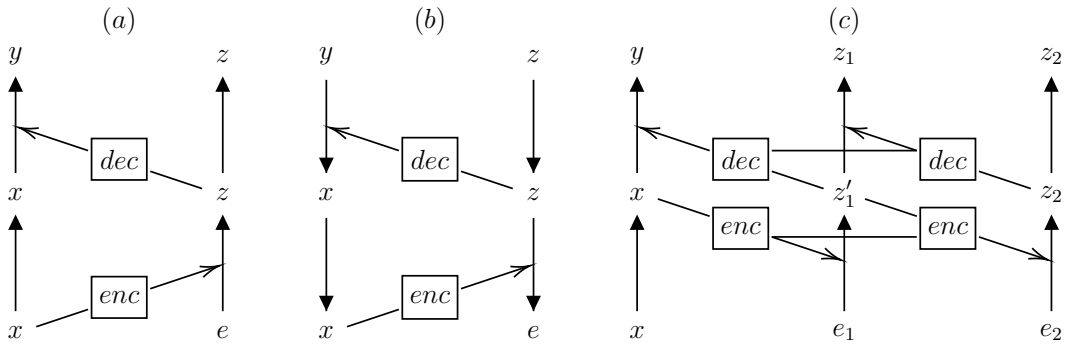


Figure 6.2: (*a*) Augmented normalizing flow with autoencoding transform, *i.e.* augmented RealNVP, and (*b*) the reverse path for generation. (*c*) Hierarchical augmented normalizing flow. The horizontal connections indicate deterministic features that will be concatenated with the stochastic features in the next transform block.

We refer to the pair of encoding transform and decoding transform as the *autoencoding* transform. In practice, using the composability of invertible maps, we stack them up in alternating order, i.e. $G_\pi = g^{\text{dec}}_{\pi_N} \circ g^{\text{enc}}_{\pi_N} \circ ... \circ g^{\text{dec}}_{\pi_1} \circ g^{\text{enc}}_{\pi_1}$ for $N \geq 1$ steps, where $\pi = \{\pi_1, ..., \pi_N\}$ is the set of all parameters. See Figure 6.2-(a,b) for an illustration of a single autoencoding transform. Like regular normalizing flows, we can also stack them up to compose the invertible functions to form a more complex, flexible invertible map (such as the case of Figure 6.3). This particular choice of G_π inherits properties **RD1-3** mentioned in Section I from RealNVP, and the augmentation trick allows us to improve the expressivity of the model, as required by **RD4**. In contrast, the specialized architectures, CP-Flow and NAF, from the previous chapters might not admit a closed form inversion formula and require numerical treatments.

VAE as one-step augmented RealNVP Variational Autoencoders are a special case of augmented normalizing flows with only "one step" of encoding and decoding transform [Dinh et al., 2014]. To see this, assume the decoding distribution $p_\theta(x|z)$ is a factorized Gaussian with mean $\mu_\theta(z)$ and standard deviation $\sigma_\theta(z)$. By letting $z = \mu_\phi(x) + \sigma_\phi(x) \cdot e$ and $y = (x - \mu_\theta(z))/\sigma_\theta(z)$ and applying the change of variable

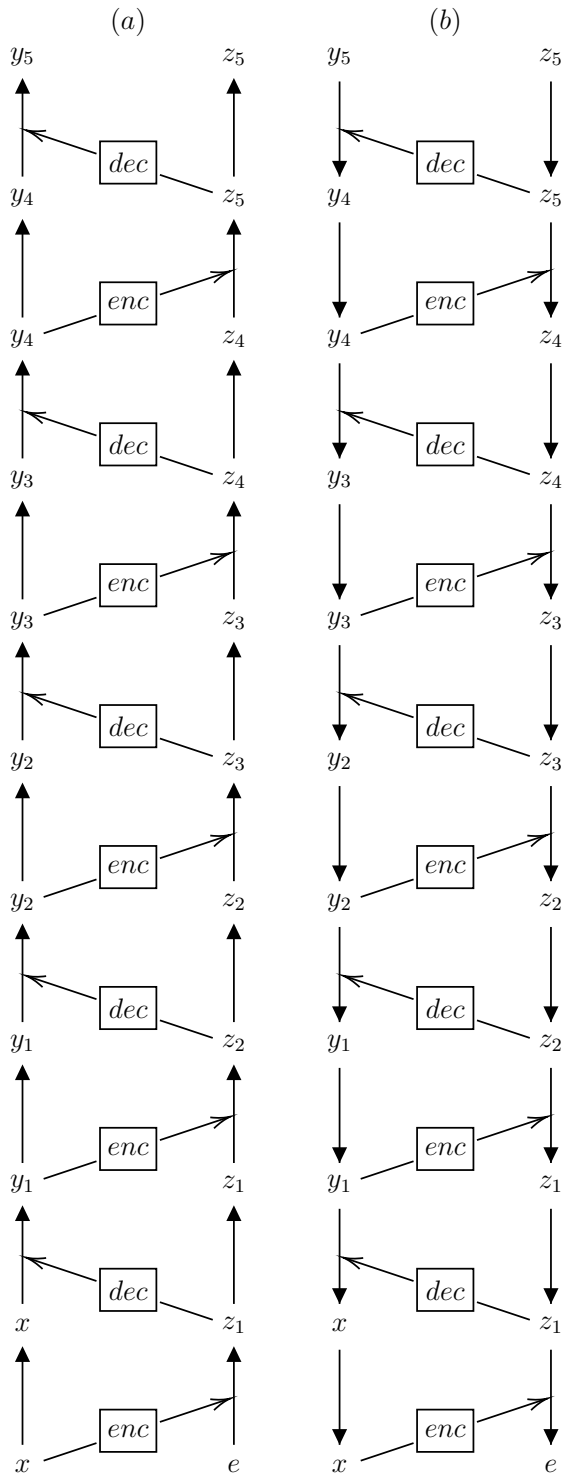


Figure 6.3: Stacked augmented NealNVP with 5 autoencoding transforms. Similarly to Figure 6.2, (*a*) and (*b*) correspond to the forward and reverse paths, respectively.

formula to both $q_\phi(z|x)$ and $p_\theta(x|z)$, we get from Equation (1.15)

$$\mathcal{L} = \mathbb{E}_{e\sim q(e)}\left[\log \mathcal{N}(y; 0, I) - \textstyle\sum_i \log \sigma_{\theta,i}(z) + \log \mathcal{N}(z; 0, I) + \sum_j \log \sigma_{\phi,j}(x)\right] + H(e).$$

Averaging over the data distribution $\hat{q}(x)$, we obtain the expected joint likelihood (up to the constant $H(e)$)

$$\mathbb{E}_{x,e\sim\hat{q}(x)q(e)}\left[\log \mathcal{N}((y, z); 0, I)\left|\det \frac{\partial(y, z)}{\partial(x, e)}\right|\right].$$

The variational gap between the log marginal likelihood and the evidence lower bound is equal to the augmentation gap since the KL divergence is invariant under the transformation between $e \longleftrightarrow z$:

$$KL(q(z|x)||p(z|x)) = D_{\mathrm{KL}}(q(e)||p(e|x)).$$

This gives us an alternative interpretation of *inference suboptimality* [Cremer et al., 2018]: the inaccuracy of inferring the true posterior $p(z|x)$ can be attributed to the incapability of the joint density to model the augmented data $q(e)$.

As an illustration, we model the density of a one dimensional mixture of Gaussian (1D MoG). In Figure 6.4 (left), we plot the density histograms of the MoG distribution (blue) and a one-step ANF, which is a VAE with Gaussian encoder and decoder (orange), trained on the MoG samples. Not surprisingly, the latter fails to represent two well separated modes of probability mass. In Figure 6.4 (right), we visualize the joint density of the augmented data $x, e \sim q(x)q(e)$ throughout the transformation. We see that the transformed data $y, z = g^{dec}_{\pi_1}(g^{enc}_{\pi_1}(x, e))$ is not perfectly Gaussianized. In fact, if we project it horizontally we can see that the "aggregated posterior" (marginal of z) does not match the prior distribution $p(z)$. As a result, the pushforward $x, e = g^{enc,-1}_{\pi_1}(g^{dec,-1}_{\pi_1}(y, z))$ of $y, z \sim p(y, z)$ does not follow the augmented data distribution $q(x)q(e)$ well. When we fix different values of x, we have different slices of density

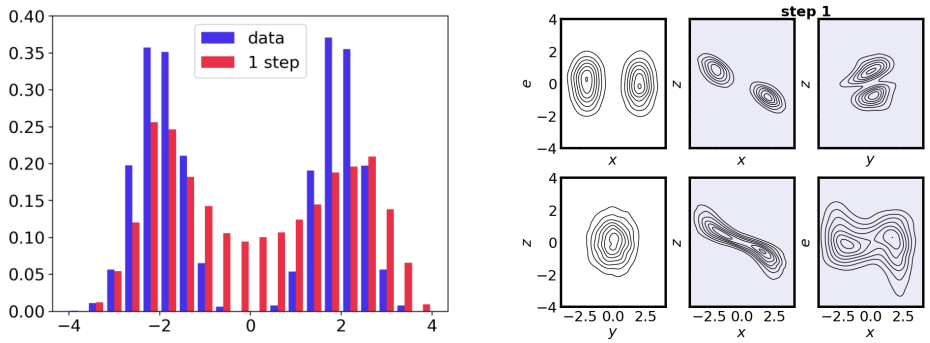


Figure 6.4: Density modeling of 1D MoG with VAE (aka 1-step ANF). (*left*) marginal distribution in the $\mathcal{X}$-space. (*right*) joint distribution in the $\mathcal{X} \times \mathcal{E}$-space. The first row is the inference path, where the joint data density $q(x)q(e)$ is mapped by an encoding transform (transforming e into z conditioned on x) followed by a decoding transform (transforming x into y conditioned on z). The second row is the generation path, where the joint prior density $p(y)p(z)$ is transformed by the inverse decoding (transforming y into x) followed by the inverse encoding (transforming z into e).

functions for e, indicating that e and x are dependent and that $p_\pi(e|x)$ deviates from $q(e)$.

We carry out the same experiment on 1D MoG with multiple flow layers, which generalizes the VAE with Gaussian encoder and decoder. We set the number of flow layers (*i.e.* steps) to be 5. Figure 6.3 provides a visualization of the computational graph of the forward and the reverse paths. To furthermore demonstrate the benefit of transformation composition, we also tie the parameters of each encoder and decoder step, separately. That is, the same set of parameters are used at different steps of encoding and decoding to make sure capacity stays constant. The conditional independence assumption in VAE is relaxed, and the augmented data is more successfully Gaussianized, as can be seen in Figure 6.5. The generated samples also follow the target joint density more closely.

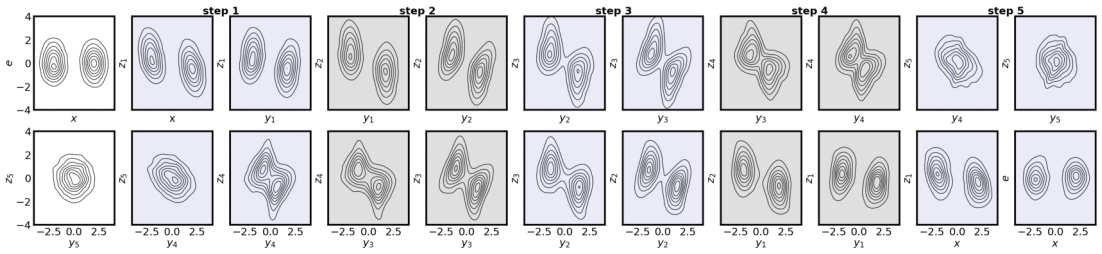


Figure 6.5: 5-step ANF on 1D MoG. In the inference path (*top*), we start with an encoding transform that maps e to z_1 conditioned on x, followed by a decoding transform that maps x into y_1 conditioned on z_1. We reuse the same encoder and decoder to refine the joint variable repeatedly to obtain y_5 and z_5. In the generative path (*bottom*), we reverse the process, starting with the inverse transform of the decoding, followed by the inverse transform of the encoding, etc.

6.4 REPRESENTATION LEARNING WITH HIERARCHICAL AUGMENTED FLOWS

The information flow of the encoding-decoding transform just described is limited to the size of the random vector e, which makes it hard to optimize for more complex settings such as natural images. We thus propose a second architecture by emulating the hierarchical variational autoencoder, which is defined by two pairs of joint distributions

$$p(x, z_1, ..., z_L) = p(x|z_1, ..., z_L) \prod_{l=1}^{L} p(z_l|z_{l+1}, ..., z_L)$$

$$q(z_1, ..., z_L|x) = \prod_{l=1}^{L} q(z_l|z_1, ..., z_{l-1}, x).$$

This particular factorization of the variational distribution is known as bottom-up inference, as opposed to top-down inference [Kingma et al., 2016] and bidirectional inference [Maaløe et al., 2019]. When all the conditionals are Gaussian distributions, the corresponding ELBO can be similarly rearranged to be the loss function of an ANF (see Figure 6.2-(c)). The encoding transform for each e_l is conditioned on the "transformed" preceding variables $s^e_{\pi,l}(x, z_{<l}) \odot e_l + m^e_{\pi,l}(x, z_{<l})$ due to the conditioning in $q(z_l|z_{<l}, x)$. This is a form of autoregressive flow, as the Jacobian matrix is block-triangular. The decoding transform on the other hand is conditioned on the

Figure 6.6: Lossy reconstruction. (*left*) original data. (*right*) reconstruction from the topmost representation. Similar to training, the auxiliary variables are all randomly generated. After encoding, the lower level representations are all resampled with independent noise drawn from the prior, which is then combined with the topmost representation for joint decoding.

"original" preceding variables $s^d_{\pi,l}(e_{>l}) \odot e_l + m^d_{\pi,l}(e_{>l})$, which is block-wise inverse autoregressive [Kingma et al., 2016].

As a hierarchical model, ANF with this particular invertible architecture can be used to perform inference for the higher level representation, and sample the lower level details for reconstruction. This is because when the conditional mappings are convolutional, the lower level transformation preserves information of the input locally, which is then combined with the deterministic path of the decoding that "sees" more of the input, thus containing information of more high-level semantics of the data. We can then perform *lossy reconstruction* by sampling $e_1, ..., e_L$, obtaining the corresponding $y, z_1, ..., z_L \leftarrow G_\pi(x, e_1, ..., e_L)$, randomizing all but the last representations $y', z'_1, ..., z'_{L-1} \sim \mathcal{N}(0, I)$, and reconstructing from the new joint representation $x', e'_1, ..., e'_L \leftarrow G^{-1}_\pi(y', z'_1, ..., z'_{L-1}, z_L)$. Similar to other hierarchical models [Gulrajani et al., 2017, Belghazi et al., 2018], hierarchical ANF is also capable of retaining global, semantic information of the raw data stored in its higher level code; this is shown in Figure 6.6. The model is trained on the CelebA dataset [Liu et al., 2015].

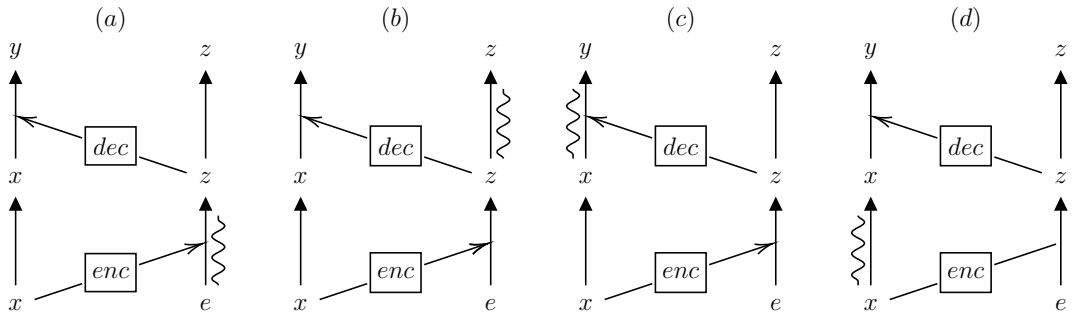


Figure 6.7: VAEs with flow components as augmented flows with non-affine transforms (wavy curves). (*a*) VAE with an amortized flow as the encoder. (*b*) VAE with a flow prior. (*c, d*) VAE with a flow-based decoder using a conditional invertible map and a conditional base distribution.

More details can be found in Huang et al. [2020b].

6.5 FLOWIFICATION

The augmented view also allows us to look at different variants of VAE as augmented flows with non-affine conditional or unconditional transforms. For example, as explained in Section 1.3, normalizing flows can be used to improve variational inference to bridge the variational gap (1.16), as done in Rezende and Mohamed [2015], Kingma et al. [2016], Van Den Berg et al. [2018] for instance. Generative flows can also be used as a trainable prior to fill the pockets in the aggregated posterior [Chen et al., 2016b, Huang et al., 2017b]. Other works have explored replacing the Gaussian decoder with a generative flow [Agrawal and Dukkipati, 2016, Ma et al., 2021], or using the decoder likelihood as the base distribution for the generative flow [Morrow and Chiu, 2020]. These models are schematically illustrated in Figure 6.7, which makes it easier to compare them.

6.6 AUGMENTED GLOW/FLOW++

While augmented RealNVP and its hierarchical extension make use of the partition of the data and the auxiliary variables $(x, e_1, \cdots)$, we can in fact leverage other architectures that disregard this structure. One benefit of this is that it makes it easier to see the benefit of improved expressivity via augmentation. For example, we can replace the autoencoding transform with a Glow-like architecture [Kingma and Dhariwal, 2018], which extends the permutation of RealNVP to a general linear transformation (in the case of image data, this is implemented with a 1×1 convolution).

Concretely, suppose x is a 3D tensor representing an image data of size $3 \times h \times w$, where each slice of the first axis corresponds to one color channel. A Glow block typically contains a 1×1 convolution followed by a RealNVP-type of block-wise affine coupling (1.7), which performs an affine transform on the second half of the channels conditioned on the first half. Usually, Glow blocks are intertwined with a split-and-squeeze operator that reduces the resolution in half and doubles the remaining channels that will undergo further transformations, which is known as the *multiscale architecture* [Dinh et al., 2017]. For the sake of simplicity, lets assume we are not performing this operation. For augmented Glow, we pad the image data with d_{aug} channels of white noise, *i.e.* $\mathcal{E} = \mathbb{R}^{d_{aug} \times h \times w}$.

Augmentation allows us to build invertible maps on a higher dimensional lifted space, but the model can only be optimized by maximizing the lower bound (6.2). It still remains unclear whether an augmented model trained with the lower bound can surpass an unaugmented one. We argue that any series of Glow blocks can be equivalently implemented by an augmented Glow model. This will imply the augmented family possesses greater expressivity. We focus on the case of fully connected networks, and assume the distribution of the augmented data $q(e)$, the priors $p(x)$ and $p(e)$ are all standard Gaussian. Suppose the first invertible linear layer of the Glow is characterized by the matrix $V := \begin{bmatrix} V_1; & V_2 \end{bmatrix}$. We can let the first invertible linear transformation of the augmented Glow implement the mapping $(x_1, x_2, e_1, e_2) \mapsto (V_1 \begin{bmatrix} x_1 & x_2 \end{bmatrix}, e_1, V_2 \begin{bmatrix} x_1 & x_2 \end{bmatrix}, e_2)$. And the subsequent affine coupling just needs to neglect the e_1 in the first half of the parti-

tioning for conditioning, and only transform x_2 in the second half (by outputting an identity map for e_2). Similarly, for the second flow block and beyond, if the invertible linear layer of the Glow block is characterized by a matrix W, we can let the linear map of the augmented model implement $(x_1, e_1, x_2, e_2) \mapsto (W_1 \begin{bmatrix} x_1 & x_2 \end{bmatrix}, e_1, W_2 \begin{bmatrix} x_1 & x_2 \end{bmatrix}, e_2)$, and likewise, let the coupling ignore e_1 and e_2. Since after all of the transforms, e_1 and e_2 remain the same, $q(e)$ and $p(e)$ will cancel out in the lower bound, which means the lower bound of the augmented Glow is tight and is actually equal to the likelihood of the Glow model. More formally,

Proposition 16. *Let g be a Glow model, i.e. each flow layer consists of an invertible linear map and a block-wise affine coupling (1.7) on $\mathbb{R}^d$. Denote its likelihood function by $\log p_g$. For any augmented dimension $d_{aug} > 0$, we can find a Glow model G on $\mathbb{R}^{d+d_{aug}}$ such that its marginal likelihood and ELBO (6.2) satisfy $\log p_G(x) = \mathcal{L}_{\mathcal{A}}(G; x) = \log p_g(x)$.*

The result can be easily extended to convolutional models by replacing the linear maps with a 1×1 convolution. Furthermore, the same can be said about RealNVP with alternating directions of conditioning, which has been shown to be able to represent any invertible linear map with a constant number of flow layers [Koehler et al., 2021].

6.7 IMPACT, RELATED WORK AND RECENT DEVELOPMENTS

Augmented flows Much of the work presented in this chapter is inspired by Dinh et al. [2014], who first interpreted VAE as NICE, and by Dupont et al. [2019], who discussed the representational limitation of neural ODE and proposed to augment it. Independently and concurrently to ours, Chen et al. [2020] also explored the idea of variational augmentation. Nielsen et al. [2020] generalized augmentation and interpreted it as a surjective map via the projection operation involved in marginalization. Grcić et al. [2021] applies the augmentation technique incrementally and couple the features via

densely connected layers. In Chapter 8, we explore continuous-time diffusion models, which can be interpreted a kind of augmented neural ODE [Chen et al., 2018], which injects noise to every time step, unlike Dupont et al. [2019]. Dockhorn et al. [2022] further augment the state space with a velocity variable to accelerate the convergence of the diffusion dynamics.

Theoretical result The improved expressivity result is from a concurrent work by Chen et al. [2020], which better fits the narrative of the thesis. In the original preprint Huang et al. [2020b], we also proved the distributional universality of augmented RealNVP, but the construction results in degeneracy; the approximate flows have a singular Jacobian, which means they are not invertible. The transport map used therein is an accelerated version of the probability flows induced by the overdamped Langevin diffusion from Taghvaei and Mehta [2019], Wang and Li [2019]. It was later improved by Lee et al. [2021], who approximated the probability flow of the underdamped Langevin dynamics [Ma et al., 2019]. They focus more on the conditioning of the Jacobian and therefore require restriction to log-concave data distributions. Aside from augmented RealNVP, universality of RealNVP-type of coupling has also been studied in the work of Teshima et al. [2020a], Koehler et al. [2021].

Applications The augmentation framework is also compatible with certain structures that are desirable for different applications. For example, Dibak et al. [2021] explored augmented flows to sample from equilibrium states of many-body systems with different temperature parameters, which they termed *temperature steerable flows*. Rezende et al. [2019] proposed to learn an augmented Hamiltonian neural ODE to incorporate symmetries as inductive bias for learning invariant densities. Rasul et al. [2019] uses augmentation within the context of learning permutation invariant densities of exchangeable sets. Similar to the hierarchical augmented flows in Section 6.4, Ma et al. [2021] introduced latent variables to decouple global and local representations of

image data. Ho et al. [2021] improved the compression efficiency of VAE using ANF. More recently, ANF has also been applied to model coarse-grained potential function for molecular simulation [Köhler et al., 2022], emulate long time-scale molecular dynamics [Klein et al., 2023], and incorporate SE(3) symmetry in modeling small peptides [Midgley et al., 2023].

Part III

Continuous-time flows

Continuous-time flows

Recall that in part I, we have looked at a few ways to construct probability flows that are provably flexible, by designing the invertible map to approximate certain types of coupling. This means a single layer of flow is guaranteed to be universal given enough capacity in the invertible architecture, *e.g.* the autoregressive monotone transformation and the input convex neural network. Another way to improve expressivity is by stacking the flows. We can do this since the composition of invertible maps is still invertible, and the overall Jacobian determinant can be decomposed into a product of Jacobian determinants of all of the flow layers. A natural parameterization of a flow of this type is to consider the dynamic of an *ordinary differential equation* (ODE), which we will see in Chapter 7. These continuous-time flows are very flexible, as they can be parameterized by neural networks with free-form Jacobian matrices directly, and they can be seen as an infinitely deep invertible model.

The cost of having greater expressivity by parameterizing the flow this way is that evaluating likelihood requires resorting to numerical solvers which are not compatible with parallel computing. In Chapter 8, we instead look at the probability flow induced by a *stochastic differential equation*, *i.e.* a diffusion process, which generalizes an ODE to include a stochastic integral. There are many benefits of this stochastic generalization. Predominantly, it allows us to design a likelihood estimation scheme that can be computed in $\mathcal{O}(1)$ time, which makes training much more scalable. An interesting reparameterization of the model also reveals a non-trivial connection to score matching. As a diffusion model is essentially an infinitely deep VAE, this unifies gradient parameterization of EBM with generative flows and VAE (recall the likelihood-based generative model taxonomy in Table 1.1).

7 Deterministic continuous-time flows

In this chapter, we review deterministic continuous-time flows, which is an important family of flows defined as the solution of an ordinary differential equation. We also provide theoretical analysis of the distributional universality of continuous-time flows.

7.1 Neural ordinary differential equations and flow maps

An ordinary differential equation (ODE) has the following form

$$\mathrm{d}X_t = v(X_t, t)\,\mathrm{d}t, \tag{7.1}$$

where v can be interpreted as the velocity at which a particle travels at a particular location and time, starting from some initial position $X_0 = x_0$. We assume v is t-uniformly Lipschitz in x, which implies this initial value problem is guaranteed to have a unique solution [Coddington and Levinson, 1955]. The solution of the ODE is then the curve X_t of positions the particle travels through, starting from x_0 at $t = 0$.

Neural ODEs are parametric ODEs (where v is parameterized by an $\mathbb{R}^d \times [0, T] \to \mathbb{R}^d$ neural network) that can be trained via gradient-based optimization. This is typically done by using a continuous analog of back-propagation known as the adjoint method [Chen et al., 2018], which amounts to solving another reverse-time ODE. The ODE can be solved by using any black-box numerical solver. Suppose the numerical solver

requires m steps ($\mathcal{O}(m)$ time) to converge or terminate; m either is a hyperparameter, or implicitly depends on the step size of the solver or a pre-specified error tolerance for adaptive methods. A crucial property of the adjoint method for computing gradients is that we do not need to store all the intermediate (hidden) values of X_t, which means the space complexity is $\mathcal{O}(1)$. See Chen et al. [2018] for more details.

The constant-memory cost for the adjoint method used in Chen et al. [2018] is owing to the fact that all X_t can be recovered by solving (7.1) reversed in time. Let $f(x, t)$ be the value of X_t with initial condition $X_0 = x$. It can be shown that f is a bijective map from $\mathbb{R}^d$ to $\mathbb{R}^d$, which makes it suitable for modeling the transformation between densities. Furthermore, f and its inverse f^{-1} are both Lipschitz continuous, which means f is a bi-Lipschitz homeomorphism [Santambrogio, 2015, Box 4.1].

As $f(x, t)$ is a continuum of invertible maps, it induces a probability flow starting from the initial law of X_0. That is, if $x_0 \sim p(x, 0)$, then $f(\cdot, t)$ induces a family of probability densities $p(x, t)$ indexed by t. In the rest of the section, we will spend a bit of time on this change of density from two perspectives. First, we adopt the *Eulerian* formalism, drawing a parallel to the mass flow rate in fluid dynamics by looking at the fluid motion at a specific location and time [Deen, 1998]. The result of the derivation is a partial differential equation (PDE) summarizing the temporal change in $p(x, t)$. We then convert it to the *Lagrangian* framework, which allows us to trace the change of density of a "moving particle". This gives us a computational advantage, as the change of density can be readily computed by solving an augmented ODE. We emphasize that the derivation is not rigorous, and that we will refer to standard references for the formal treatment. The point of the discussion made in this section is to provide a physical intuition of continuous-time flow and its change of density.

Lastly, the probability flow induced by an ODE is immensely flexible. We will study the distributional universality of continuous-time flows in Section 7.5.

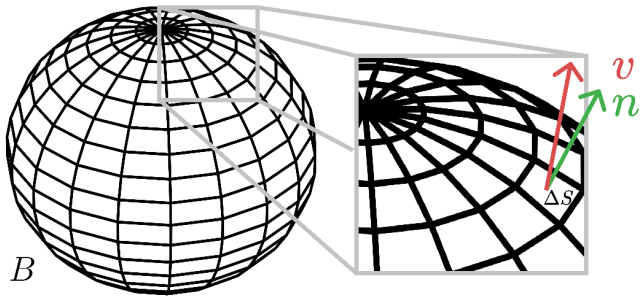


Figure 7.1: The mass flow rate $pv \cdot n\Delta S$ through an infinitesimal patch on a sphere.

7.2 CONSERVATION OF MASS AND CONTINUITY EQUATION

For illustration, we consider a three dimensional space filled with a fluid. Fix some location $x \in \mathbb{R}^3$ and time $t \in (0, T)$. We look at the density of the fluid passing through x at time t. Let $p(x, t)$ be the mass density and assume the total mass is 1 (so that we can interpret it as a probability later on). Let B be a fixed subregion in $\mathbb{R}^3$ (*e.g.* an open ball), and $m_t(B)$ be the total amount of mass in B at time t, defined as

$$m_t(B) := \int_B p(x, t)\,\mathrm{d}x. \tag{7.2}$$

By definition, the rate of change of mass in B is equal to

$$\frac{\partial m_t(B)}{\partial t} = \int_B \frac{\partial p}{\partial t}\,\mathrm{d}x. \tag{7.3}$$

We make the assumption that *mass cannot be created or destroyed*. This implies the change of mass in B must be due to the particles leaving or entering B through its boundary ∂B, since X_t moves continuously. Let n be an outward normal of ∂B. The *volumetric flow rate* of the fluid through ∂B per unit of area is $v \cdot n$. Let ΔS be the area of an infinitesimal patch on ∂B. See figure 7.1. Then the *mass flow rate* through the patch is $pv \cdot n\Delta S$, and the total change of mass per unit of time in B can be

approximated by

$$-\sum pv \cdot n\Delta S, \tag{7.4}$$

where the minus sign is because n points outward, and the quantity pv is known as the *mass flux*, *i.e.* the change in mass per unit of area and time. By refining the partitioning of B (taking $\Delta S \to 0$), this converges to the surface integral. That is,

$$\int_B \frac{\partial p}{\partial t}\,\mathrm{d}x = -\oiint_{\partial B} (pv) \cdot \mathrm{d}S. \tag{7.5}$$

Formally, this means that *the rate of increase of mass in B is equal to the rate at which mass enters B through ∂B*. Applying the divergence theorem to the RHS, we have

$$\int_B \frac{\partial p}{\partial t}\,\mathrm{d}x = -\int_B \nabla \cdot (pv)\mathrm{d}x. \tag{7.6}$$

Since this identity holds for any B with sufficiently smooth boundary, we have

$$\boxed{\frac{\partial p}{\partial t} = -\nabla \cdot (pv)} \tag{7.7}$$

which is known as the differential form of the *law of conservation of mass*, or the *continuity equation*.

We note that this derivation is more of a heuristic. A more general version of the continuity equation also exists, where we do not even require the initial law of X_0 to have a density (*e.g.* it can be discrete- or mixed-valued). For a formal treatment, see Santambrogio [2015, Propositions 4.2 and 4.3]. The continuity equation is a special case of the Fokker-Planck equation associated with a stochastic differential equation with zero diffusion, also known as the Liouville equation. We will see more of this in the next chapter.

7.3 INSTANTANEOUS CHANGE OF VARIABLE

The continuity equation is a first-order PDE that governs the temporal change in the density by relating it to the spatial derivative of the flux. We can relate it to the *instantaneous* change of the density around a moving particle X_t, as opposed to a fixed position x. To that end, we apply the product rule of the divergence operator and obtain

$$-\nabla \cdot (pv) = -v \cdot \nabla p - p \nabla \cdot v,$$

which means

$$\frac{\partial p}{\partial t} + v \cdot \nabla p = -p \nabla \cdot v. \tag{7.8}$$

The first term of the LHS is due to the temporal change of the density and the second term is driven by the particle's motion, *i.e.* convection. Overall the LHS is called the *material derivative* of p, which accounts for the fact that the particle is moving. Mathematically, this is simply the total derivative of p evaluated at X_t and time t; therefore

$$\frac{\mathrm{d}}{\mathrm{d}t} p(X_t, t) = -p \nabla \cdot v. \tag{7.9}$$

This leads to the **instantaneous change of variable formula**

$$\boxed{\frac{\mathrm{d}}{\mathrm{d}t} \log p(X_t, t) = -\nabla \cdot v} \tag{7.10}$$

or in the integral form

$$\boxed{p(X_t, t) = p(X_0, 0) e^{-\int_0^t \nabla \cdot v(X_r, r)\, \mathrm{d}r}} \tag{7.11}$$

Compared to the continuity equation, the instantaneous change of variable provides a Lagrangian view of the density function—it traces the change of density along the

trajectory of X_t. This also makes it useful for computing the change in density induced by the flow. To evaluate the log-likelihood $\log p(x, T)$ for some data point x, one just needs to (1) take x as the terminal condition, solve the ODE reversed in time, (2) at the same time accumulate the infinitesimal log-change of volume $-\nabla \cdot v$ by augmenting the ODE with a new state that represents the delta in $\log p$, and (3) evaluate the resulting "initial" state under the prior $p(\cdot, 0)$.

7.4 DIVERGENCE ESTIMATOR

Naive computation of the divergence of the vector field can be costly, as it scales $\mathcal{O}(d^2)$ in time. As a practical remedy proposed by Grathwohl et al. [2019], it can be unbiasedly estimated using the Hutchinson trace estimator [Hutchinson, 1989] along with the reverse mode automatic differentiation for the computation of vector-Jacobian products. This estimation scheme costs $\mathcal{O}(d)$ time.

Concretely, since the divergence operator is simply the trace of the Jacobian, we have

$$\nabla \cdot v = \mathrm{Tr}(\nabla v) = \mathbb{E}[z^\top \nabla v z] \approx \frac{1}{m} \sum_{i=1}^{m} z_i^\top \nabla v z_i,$$

for *i.i.d.* random vectors z_i satisfying $\mathbb{E}[z_i] = 0$ and $\mathbb{E}[z_i z_i^\top] = I$. To compute this quantity, we can first differentiate the inner product $z_i^\top v$ (or equivalently differentiate the vector-valued function v while letting the "reverse accumulation" of the gradient be z_i) to obtain $z_i^\top \nabla v$ and then inner product with z_i again.

If z is a Rademacher random vector (*i.e.* the symmetric Bernoulli $z_i = \pm 1$ with equal probability), for a general matrix A, the variance of the estimator $z^\top A z$ is [Hutchinson,

1989, Proposition 1]

$$\mathrm{Var}(z^\top Az) = \frac{1}{2}\|A + A^\top\|_F^2 - \frac{1}{2}\mathrm{Tr}(A + A^\top)^2 \le 2\|A\|_F^2 - 2\mathrm{Tr}(A)^2 = \mathcal{O}(\|A\|_F^2). \tag{7.12}$$

This suggests this estimator has smaller variance when the off-diagonal entries have smaller values. The variance vanishes if the off-diagonal terms are all zeros.

7.5 UNIVERSALITY

Continuous time flows are extremely expressive, in that the constraints on the flow maps are very mild, which is manifested by the fact that there is no strong restriction on the Jacobian.[1] This also makes it easier to analyze its expressive power.

Let p and q be two probability densities. From part I, we have seen a few ways to construct a coupling $g : \mathbb{R}^d \to \mathbb{R}^d$ such that $g_\# p = q$. Our goal here is to construct an ODE whose terminal flow map is a coupling; this is known as a *dynamical coupling*.

Suppose we can find a smooth interpolation between g and the identity map $g(x, t)$ which satisfies

$$g(x, 0) = x \qquad \text{and} \qquad g(x, 1) = g(x). \tag{7.13}$$

Lets also assume $x \mapsto g(x, t)$ is diffeomorphic for all $t \in [0, 1]$. Then we can extract a vector field by viewing $g(x, t)$ as the flow map generated by the vector field, *i.e.*

$$v(g(x, t), t) = \frac{\partial g(x, t)}{\partial t}, \tag{7.14}$$

[1] In the literature, this is known as free-form Jacobian [Grathwohl et al., 2019]

which means[2]

$$v(x,t) = \frac{\partial g(g^{-1}(x,t),t)}{\partial t}, \tag{7.15}$$

where $g^{-1}(x,t)$ is the inverse of $g(x,t)$.

Take the Brenier map (the optimal transport map from Section 5.1) as an example, which can be written as $g = \nabla G$ for some convex potential G. Now set $g(x,t)$ to be the *linear* interpolation between the identity map and $g(x)$, *i.e.*

$$g(x,t) = tg(x) + (1-t)x, \tag{7.16}$$

which by Theorem 11 is injective since $g(x,t)$ can be seen as the gradient map of $\frac{1-t}{2}\|x\|^2 + tG(x)$, which is strongly convex. This is also known as the dynamical optimal coupling [Villani, 2003, Theorem 5.5] between p and q. In this case, the particle will be traveling at a constant speed, since

$$v(x,t) = g(g^{-1}(x,t)) - g^{-1}(x,t), \tag{7.17}$$

which can be identified by $g(x_0) - x_0$ where $x_0 = g^{-1}(x,t)$ is the initial value of the particle that ends up at x at time t.

As another example, consider the KR map $KR_{p\to q}$ (4.1). The linear interpolation between the identity map and the KR map is still invertible since

$$g(x,t)_j = (1-t)x_j + tKR_{p\to q}(x)_j \tag{7.18}$$

is invertible in x_j given $x_{1:j-1}$ and does not depend on $x_{>j}$.

These give us dynamical couplings and their associated velocity fields. What we need to do next is to show that we can approximate the dynamical couplings by approximating the velocity fields, which is the trainable degree of freedom.

[2]Note that the partial derivative is wrt the second t.

Given a uniformly Lipschitz velocity field $v(x,t)$ of an ODE $\mathrm{d}x = v(x,t)\,\mathrm{d}t$, we let $\phi_v(x,t)$ denote the associated flow map, *i.e.* $\phi_v(x,0) = x$ and $\frac{\partial}{\partial t}\phi_v(x,t) = v(\phi_v(x,t),t)$. The following theorem says that parametric ODE with a universal velocity field (such as neural ODE) is a universal approximator of flow maps.

Theorem 17 (**Universality of parametric flow map**). *Let $\mathcal{V}$ be a family of (parametric) Lipschitz functions that is is dense (wrt the uniform metric) in $C(K \times [0,T]; \mathbb{R}^d)$ for any compact domain $K \subset \mathbb{R}^d$. Let $v(x,t) : \mathbb{R}^d \times [0,T] \to \mathbb{R}^d$ be a velocity field uniformly Lipschitz in x for all t, and continuous in t. Then for any compact set $K \in \mathbb{R}^d$, we can find a (parametric) velocity $v' \in \mathcal{V}$ such that $\sup_{x \in K, t \in [0,T]} ||\phi_{v'}(x,t) - \phi_v(x,t)|| \leq \epsilon$.*

We note that a similar result has been proved by Teshima et al. [2020b] (see their Lemma 2), as our time-dependent ODE can be converted into an autonomous ODE. However, we stress that the enclosing domain K' that we choose in the proof is tighter than theirs and shrinks to K as $\epsilon \to 0$, and that our proof is more constructive, whereas theirs is a proof by contradiction.

Combining this theorem and the dynamical couplings, we deduce that neural ODEs are distributionally universal.

Theorem 18 (**Universality of Neural ODE**). *Flow maps of neural ODE are $\mathfrak{A} - \mathfrak{P}$ distributionally universal.*

7.6 DISCUSSION

In this chapter, we look at a different family of invertible maps induced by ODEs. We start by deriving the likelihood function from the bystander's view (the Eulerian formalism), which shows the change of density in time at any location is related to the net mass flux out of the point. To allow for tractable likelihood computation, we then adopt the Lagrangian formalism which provides a perspective following the system's

particles evolving through time and space, leading to the instantaneous change-of-variable formula [Chen et al., 2018].

To highlight the expressive power of continuous-time flows, we show that the flow maps induced by neural ODEs can approximate the dynamical optimal coupling. However, we can in fact use other dynamical couplings, such as the linearized KR map, the probability flow of the Langevin dynamics (see the next chapter where we discuss the notion of marginal equivalency, and how to extract an *equivalent* ODE out of an SDE), the Dacorogna-Moser map [Dacorogna and Moser, 1990, Rozen et al., 2021], etc. This shows the desirable flexibility of Neural ODEs. However, training a neural ODE still requires numerical integration. In practice, this is equivalent to having hundreds or thousands of layers, depending on the complexity of the dynamics, which makes training expensive. One remedy for this is to regularize the dynamics to encourage the ODE to have simpler trajectories [Finlay et al., 2020b]. In the next chapter, we explore a different approach, by injecting noise to the deterministic dynamics, which results in a diffusion process. We will derive a variational framework for likelihood estimation of diffusion models, which has profound connections to generative flows like neural ODE, VAEs, and score matching. The extra degree of freedom introduced by the stochastic perturbation allows us to design inference machines that can query the representation at any time step without recourse to numerical integration, thereby enabling simulation-free training at scale.

8 Stochastic continuous-time flows

In this section, we generalize the deterministic dynamics described in the previous section to the setting where an infinitesimal amount of white noise is added at every time step, to create a *diffusion process*. This additional degree of freedom will allow us to design the model and training procedure to further simplify the computation of likelihood for training so that we will not have to rely on numerical integration as in the case of a neural ODE (see Section 8.6). This will be the key to obtaining a *scalable* density model that is not just *tractable*.

The main development of the variational framework is split into Sections 8.3 and 8.4. We also derive theoretical connections to VAE in 8.5 and score matching for energy-based models in Section 8.7.

8.1 Neural stochastic differential equations and diffusion processes

Let X_t be a diffusion process induced by the following Itô SDE [Øksendal, 2003]:

$$\mathrm{d}X = \mu(X, t)\,\mathrm{d}t + \sigma(X, t)\,\mathrm{d}B_t, \tag{8.1}$$

where B_t is a standard Brownian motion. We assume the dynamics has a stochastic initial condition $X_0 \sim p_0$, namely the prior, which induces a family of densities

$X_t \sim p(\cdot, t)$. We refer to this SDE as the generative SDE, and we are interested in computing $\log p(x, T)$ for maximum likelihood.

The rigorous treatment of the Itô SDE can be found in Øksendal [2003], Karatzas and Shreve [2014], Protter [2005]. Intuitively, it can be thought of as the limit of the following discrete-time dynamics (by taking the step size parameter Δt to 0, Milshtein 1975):

$$X_{i+1} = X_i + \Delta t \mu(X_i, i\Delta t) + \sqrt{\Delta t}\sigma(X_i, i\Delta t)\epsilon_i, \tag{8.2}$$

where ϵ_i's are *i.i.d.* standard Gaussian noise. This is known as the Euler-Maruyama (EM) method for numerically integrating an Itô SDE.

In the discrete-time case, we already know that diffusion models do not come with a convenient change-of-variable formula since noise is injected. Consequently, maximizing likelihood becomes a challenging task. To address this issue, we approach it directly from a continuous-time perspective in the following sections. In Section 8.3, we expand upon the previous discussion on deterministic dynamics in Section 7.3 and extend the instantaneous change-of-variable formula to encompass stochastic dynamics Similar to a latent variable model, the resulting change-of-variable formula involves an intractable integral due to the stochasticity of the model. Considering this complexity, we proceed to derive an ELBO for continuous-time diffusion models in Section 8.4, which bears resemblance to the ELBO used in VAEs.

8.2 KOLMOGOROV FORWARD (FOKKER-PLANCK) EQUATION

The density $p(x,t)$ can be shown to follow the *Kolmogorov forward* (or the *Fokker-Planck*) *equation* [Gardiner, 2009, Pavliotis, 2014][1]:

$$\partial_t p(x,t) = -\sum_j \partial_{x_j}[\mu_j(x,t)\, p(x,t)] + \sum_{i,j} \partial^2_{x_i,x_j}[D_{ij}(x,t)\, p(x,t)], \tag{8.3}$$

with the initial value $p(\cdot,0) = p_0(\cdot)$, where $D = \frac{1}{2}\sigma\sigma^T$ is the diffusion matrix.

Fokker-Planck equation is an extension of the continuity equation we have seen in Section 7.2. It describes the temporal change of the density function via the spatial variations, as the change in density is caused by the deterministic flux and the stochastic perturbation. Heuristically, if we assume the diffusion matrix is constant and $x \in \mathbb{R}$, then the probability will dissipate locally where the density is concave, and the mass will flow into regions where the density is convex. When $\sigma = 0$, this boils down to the continuity equation (7.7).

Solving the Fokker-Planck PDE in general is non-trivial since μ and D can be very complex, non-linear functions of x and t. One possibility is to resort to numerical solutions, but numerical methods often fail in the face of the curse of dimensionality [Han et al., 2018b]. Thus, we approach the Fokker-Planck PDE in a different manner.

8.3 STOCHASTIC INSTANTANEOUS CHANGE OF VARIABLE

Inspired by the derivation of the instantaneous change of variable formula for an ODE, we would like to simplify the PDE-solving problem by adopting a Lagrangian view. However, the additional stochastic term makes it impossible to revert the dynamics directly, since the Brownian motion is not an observed quantity. In other words, we

[1] See G.1 for a summary of notation used in this chapter for clarification.

cannot extract the velocity out of the Fokker-Planck PDE to account for the change in density contributed by convection (the particle's motion)[2] like (7.8). This suggests we need to consider an ensemble of trajectories leading to a particular observation x.

We start by inspecting the Fokker-Planck PDE by expanding it using the product rule and rearranging the terms so that all the coefficients of the same order of differentiation are grouped together:

$$\begin{aligned}\partial_t p(x,t) = &\left[-\nabla\cdot\mu(x,t) + \sum_{i,j}\partial^2_{x_i,x_j}D_{ij}(x,t)\right]p(x,t) \; + \\ &\sum_i\left[-\mu_i(x,t) + 2\sum_j \partial_{x_j}D_{ij}(x,t)\right]\partial_{x_i}p(x,t) \; + \sum_{i,j}D_{ij}(x,t)\partial^2_{x_i,x_j}p(x,t),\end{aligned} \tag{8.4}$$

For simplicity, we assume the diffusion term σ is independent of x throughout this chapter (except in the appendix, we provide a general formula that extends the following derivation). With this simplification, (8.4) reduces to

$$\partial_t p(x,t) = -\left(\nabla\cdot\mu(x,t)\right)\, p(x,t) - \mu(x,t)^\top \nabla p(x,t) + D(t) : H_p(x,t), \tag{8.5}$$

where : denotes the Frobenius inner product between matrices.

Even with this simplification, solving (8.5) is still not trivial. Fortunately, we notice that (8.4) and (8.5) are a specific kind of second-order linear PDE called *parabolic* PDE. The solution to this type of PDE has a convenient probabilistic representation, known as the *Feynman-Kac formula*.

Theorem 19 (**Feynman-Kac representation**, Chapter 5.7 of Karatzas and Shreve [2014])**.** *Let $T > 0$. Let y and ς be the spatial and temporal arguments to the function*

[2]We will see in 8.7 that we can in fact still find an ODE by merging the second-order term into the first order term, but the velocity of this ODE will involve an intractable score function of the probability flow.

F-K	F-P
$v(y,\varsigma)$	$p(y,T-\varsigma)$
$c(y,\varsigma)$	$-\nabla\cdot\mu(y,T-\varsigma)$
$b(y,\varsigma)$	$-\mu(y,T-\varsigma)$
$\eta(y,\varsigma)$	$\sigma(T-\varsigma)$
$g(y)$	$p_0(y)$

Table 8.1: Feynman-Kac (F-K) coefficients for solving the Fokker-Planck (F-P) equation.

$v\in C^{2,1}(\mathbb{R}^d\times[0,T])$ *solving*

$$\partial_\varsigma v + cv + b^\top\nabla v + A : H_v = 0, \tag{8.6}$$

with the terminal condition $v(y,T)=h(y)$*, where* $A=\frac{1}{2}\eta\eta^\top$ *for some matrix-valued function* $\eta(y,\varsigma)$*. Assume there exist some constants* $B_h, B_v>0$ *and* $p_h,p_v\geq 1$ *such that* $h\in C^0(\mathbb{R}^d)$ *and* $v\in C^{2,1}(\mathbb{R}^d\times[0,T])$ *satisfy*

$$|h(y)|\leq B_h\left(1+\|y\|^{2p_h}\right) \text{ or } h(y)\geq 0 \tag{8.7}$$

$$\max_{0\leq\varsigma\leq T}|v(y,\varsigma)|\leq B_v\left(1+\|y\|^{2p_v}\right). \tag{8.8}$$

Then v *can be written as*

$$v(y,\varsigma)=\mathbb{E}\left[h(Y_T)\exp\left(\int_\varsigma^T c(Y_s,s)\,\mathrm{d}s\right)\,\middle|\,Y_\varsigma=y\right], \tag{8.9}$$

where

$$\mathrm{d}Y = b(Y,s)\,\mathrm{d}s+\eta(Y,s)\,\mathrm{d}B'_s, \tag{8.10}$$

with the initial datum $Y_\varsigma=y$*, and* B'_s *is a Brownian motion.*

To apply the representation formula (8.9) to the density $p(\cdot,T)$ solving the PDE (8.5), we can apply the change of variable $p(x,t):=v(x,T-t)$ to turn it into a terminal

value problem, and let the Feynman-Kac (F-K) coefficients correspond to their Fokker-Planck (F-P) counterparts according to Table 8.1. This way, solving (8.6) backward is equivalent to solving (8.5) forward, and we have the following representation of the marginal density at T:

$$\boxed{p(x,T) = \mathbb{E}\left[p_0(Y_T)\exp\left(\int_0^T -\nabla\cdot\mu(Y_s, T-s)\,\mathrm{d}s\right)\,\middle|\, Y_0 = x\right]} \tag{8.11}$$

where Y_s is a diffusion process solving

$$\mathrm{d}Y = -\mu(Y, T-s)\,\mathrm{d}s + \sigma(T-s)\,\mathrm{d}B'_s. \tag{8.12}$$

Remark 20 (Marginalization & mixture density). *Equation (8.11) allows us to interpret diffusion models as mixtures of continuous-time flows induced by a neural ODE. Essentially we can treat the Brownian motion B as a latent variable. Let B be given, and we are interested in how the density evolves following the dynamics (8.1). We can discretize time into multiple infinitesimal intervals and view the continuous-time dynamics as the limit of applying infinitely many invertible maps of the form $x \mapsto x + \mu(x,t)\Delta t + \sigma(t)\Delta B_i$, where $\Delta B_i := B_{(i+1)\Delta t} - B_{i\Delta t}$ is the Brownian increment. When the step size decreases to 0, this should converge to the Itô integral. Furthermore, when Δt is small enough, under the assumption that μ is uniformly Lipschitz, the finite approximation will be invertible for all steps.*

Now, to see how this relates to continuous-time flows of ODEs, since the diffusion term is independent of the spatial variable, it can be seen as a constant additive transformation, which is volume-preserving, so it will not be taken into account when computing the change of density. The only contribution to the change of density will be from $\mathrm{id} + \mu\Delta t$. Concretely, the determinant of the Jacobian of the overall transformation is just the

product of determinant of each step:

$$\begin{aligned}\prod_i \det\left(\nabla\left(x+\mu(x,t)\Delta t+\sigma(t)\Delta B_i\right)\right) &= \prod_i \det\left(\mathbf{I}+\Delta t\nabla\mu\right)\\ &= \prod_i \left(1+\Delta t\mathrm{Tr}(\nabla\mu)+\mathcal{O}(\Delta t^2)\right)\\ &= \exp\left(\sum_i \log\left(1+\Delta t\nabla\cdot\mu+\mathcal{O}(\Delta t^2)\right)\right)\\ &= \exp\left(\sum_i \Delta t\nabla\cdot\mu+\mathcal{O}(\Delta t^2)\right)\\ &\to \exp\left(\int\nabla\cdot\mu\right) \text{ as } \Delta t\to 0.\end{aligned}$$

This leads to the same derivation for the instantaneous change of variable formula for continuous-time flow [Chen et al., 2018], but the argument of μ will be the solution to the reverse-time SDE involving an Itô integral, which is not a deterministic dynamics.

This will be the conditional density given the entire sample path $\{B_t : t \geq 0\}$, and marginalizing it out results in the expectation in (8.11).

This framework also works with the general case where σ depends on x, but the formulae need to be adapted to account for the spatial partial derivatives. Following a similar conversion as in Table 8.1, we have the following general representation formula for the Fokker-Planck equation (8.4).

Theorem 21 (Probabilistic representation of the Fokker-Planck equation). *Let p be a time-indexed density function solving the Fokker Planch equation (8.4), and let $p(\cdot,0)=p_0(\cdot)$ be the initial value. Then p has the following representation:*

$$p(x,T) = \mathbb{E}\left[p_0(Y_T)\exp\left(\int_0^T -\nabla\cdot\mu(Y_s,T-s)+\sum_{i,j}\partial^2_{x_i,x_j}D_{ij}(Y_s,T-s)\,\mathrm{d}s\right)\,\middle|\,Y_0=x\right] \tag{8.13}$$

where Y_s solves

$$\mathrm{d}Y = -\tilde{\mu}(Y, T-s)\,\mathrm{d}s + \sigma(Y, T-s)\,\mathrm{d}B'_s, \tag{8.14}$$

where $\tilde{\mu}(y,s)_i := \mu_i(y,s) - 2\sum_j \partial_{x_j} D_{ij}(y,s)$.

Lastly, note that this section and the previous one mirror the discussion made in Sections 7.2 and 7.3—the Fokker-Planck equation describes the change in density at a fixed position (a Eulerian view), whereas the stochastic instantaneous change-of-variable formula tells us how the density evolves along an ensemble of trajectories (a Lagrangian view), which results in the marginalization. This is also more convenient computationally, since we just need to solve an SDE instead of a PDE. See Remark 24 below for more details on numerical integration.

8.4 CONTINUOUS TIME ELBO

As our goal is to estimate likelihood, we would like to compute the log density value using (8.11), or (8.13) more generally. However, this involves integrating out all possible Brownian paths, which is intractable. To resolve this, we view the Brownian motion as a latent variable and perform inference by assigning a higher probability to sample paths that are more likely to generate the observation. In this section, we follow the recipe of deriving the ELBO for a VAE (1.11-1.14), except we have an infinite dimensional latent vector since the Brownian motion is a continuously indexed family of random variables.

Formally, let $(\Omega, \mathcal{F}, \mathbb{P})$ be the underlying probability space for which B'_s is a Brownian motion. Suppose $\mathbb{Q}$ is another probability measure on $(\Omega, \mathcal{F})$ *equivalent* to $\mathbb{P}$; that is, $\mathbb{P}$ and $\mathbb{Q}$ are similar in the sense that they have the same measure zero sets. This allows us to apply the change-of-measure trick and lower bound the log-likelihood with a finite

quantity using Jensen's inequality:

$$\log p(x,T) \geq \mathbb{E}_{\mathbb{Q}}\left[\log \frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}} + \log p_0(Y_T) - \int_0^T \nabla \cdot \mu \,\mathrm{d}s \,\middle|\, Y_0 = x\right]. \tag{8.15}$$

Note that $\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}$ is the *Radon-Nikodym derivative* of $\mathbb{P}$ wrt $\mathbb{Q}$. When both measures are absolutely continuous wrt a third measure, say Lebesgue, then the derivative can be expressed as the ratio of the two densities, like (1.12). However, since we are dealing with an infinite dimensional space, we are immediately faced with the following problems:

1. Is there a measure $\mathbb{Q}$ (equiv. to $\mathbb{P}$) for which $\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}$ can be easily computed, or at least numerically approximated?
2. Can we find a reparameterization (similar to the Gaussian reparameterization (1.15)) of B'_s under the new law $\mathbb{Q}$ to estimate the gradient needed for training?

We resort to the *Girsanov theorem*, which describes a general framework for dealing with the change of measure of Gaussian random variables under additive perturbation. It allows us to consider the law of a diffusion process as $\mathbb{Q}$. See Appendix G.2 for an explanation using the more familiar notion of probability densities.

Theorem 22 (**Girsanov theorem**, Theorem 8.6.3 of Øksendal [2003])**.** *Let $\hat{B}_s$ be an Itô process solving*

$$\mathrm{d}\hat{B}_s = -a(\omega, s)\,\mathrm{d}s + \mathrm{d}B'_s, \tag{8.16}$$

for $\omega \in \Omega$, $0 \leq s \leq T$ and $\hat{B}_0 = 0$, where $a(\omega, s)$ satisfies the Novikov's condition

$$\mathbb{E}\left[\exp\left(\frac{1}{2}\int_0^T \|a\|_2^2 \,\mathrm{d}s\right)\right] < \infty.$$

Then $\hat{B}_s$ is a Brownian motion wrt $\mathbb{Q}$ where

$$\frac{\mathrm{d}\mathbb{Q}}{\mathrm{d}\mathbb{P}}(\omega) := \exp\left(\int_0^T a(\omega, s) \cdot \mathrm{d}B'_s - \frac{1}{2}\int_0^T \|a(\omega, s)\|_2^2 \, \mathrm{d}s\right). \tag{8.17}$$

Equation (8.16) provides a standarization formula of B'_s under $\mathbb{Q}$, which means we can "invert" it to reparameterize B'_s. This leads to the following lower bound.

Theorem 23 (Continuous-time ELBO). *Let $\mathbb{Q}$ be defined via the density (8.17). Then the RHS of (8.15) can be rewritten as*

$$\mathbb{E}\left[-\frac{1}{2}\int_0^T \|a(\omega, s)\|_2^2 \, \mathrm{d}s + \log p_0(Y_T) - \int_0^T \nabla \cdot \mu \, \mathrm{d}s \,\middle|\, Y_0 = x\right] =: \mathcal{E}^\infty, \tag{8.18}$$

where the expectation is taken wrt the Brownian motion $\hat{B}_s$, and Y_s solves[3]

$$\mathrm{d}Y = (-\mu + \sigma a)\,\mathrm{d}s + \sigma \mathrm{d}\hat{B}_s. \tag{8.19}$$

We call Y_s solving (8.19) the inference SDE, and $\mathcal{E}^\infty$ the continuous-time ELBO (CT-ELBO).

Remark 24 (Computation). *This lower bound can be numerically estimated by using any black box SDE solver, by augmenting the dynamic of y with the accumulation of $\|a\|^2$ and $\nabla \cdot \mu$. Computing the divergence term $\nabla \cdot \mu$ directly can be expensive, but it can be efficiently estimated using the Hutchinson trace estimator [Hutchinson, 1989] along with reverse-mode automatic differentiation, similar to Grathwohl et al. [2019]. As the parameters of both the generative and inference models are decoupled from the random variable $\hat{B}_s$, their gradients can be estimated via the reparameterization trick [Kingma and Welling, 2014, Rezende et al., 2014]. Furthermore, backpropagation can be computed using an adjoint method with a constant memory cost [Li et al., 2020].*

Remark 25 (Drift a). *(i) In general, the drift term of the approximate posterior to the latent Brownian motion can be made conditional, so that it will encode the information*

[3] Note that μ and σ run backward in time from T, whereas a runs forward.

of an individual datum x. (ii) The regularization $\|a\|^2$ ensures that a is kept close to 0, since it represents the deviation of the measure it induces (i.e. $\mathbb{Q}$) from the classical Wiener measure (which is a centered Gaussian measure). (iii) When the diffusion coefficient σ is 0, the inference SDE reduces to the reverse dynamic of the generative ODE, and if $a \equiv 0$ in this case, the lower bound is tight. (iv) There is generally no constraint on the form of $a(\omega, s)$, so one can potentially augment it with additional dimensions to have a non-Markovian inference SDE. For simplicity, we let the inference SDE be a Markovian model, i.e. $a = a(y, s)$. This is justified by the following theorem.

Theorem 26 (**Variational gap and optimal inference SDE**). *The variational gap can be written as*

$$\log p(x, T) - \mathcal{E}^{\infty} = \int_0^T \mathbb{E}\left[\left\|a(\omega, s) - \sigma^{\top}\nabla \log p(Y_s, T - s)\right\|^2\right] \mathrm{d}s. \tag{8.20}$$

In particular, $\mathcal{E}^{\infty} = \log p(x, T)$ if and only if $a(\omega, s)$ can be written as $a(\omega, s) = a(Y_s(\omega), s)$ for almost every $s \in [0, T]$ and $\omega \in \Omega$, and $a(y, s) = \sigma^{\top}\nabla \log p(y, T - s)$ almost everywhere.

Remark 27 (**Variational gap**). *Even though the inference SDE seemingly takes a simple form, it is sufficiently flexible in that this type of variational problem can be generally solved by taking the supremem over all progressively measurable processes $a(\omega, s)$ [Boué et al., 1998]. In fact, the above theorem shows that $\mathcal{E}^{\infty} = \log p(x, T)$ if and only if $a(y, s) = \sigma^{\top}\nabla \log p(y, T - s)$, which does not depend on x. This means an unconditional Markovian inference process is powerful enough, and one can use the same $a(y, s)$ for all data points x.*

For the above reasoning, for the rest of the chapter, we assume a takes the form $a(y, s)$, which will be parameterized by a deep neural net.

8.5 DISCRETE-TIME APPROXIMATION: AN INFINITELY DEEP VAE

The ELBO derived in the previous section is for a continuous-time model. In this section, we show that we can derive the same bound by extending the ELBO of a discrete-time model, which allows us to we formally address the common belief that "diffusion models can be viewed as the continuous limit of hierarchical VAEs" [Tzen and Raginsky, 2019]. We do so by inspecting the ELBO of a hierarchical VAE defined as discretized generative and inference SDEs. Following the Euler-Maruyama (EM) scheme, we assume the generative model (*i.e.* the decoder) follows the transition probabilities

$$p(x_{i+1} \mid x_i) = \mathcal{N}(x_{i+1}; \tilde{\mu}_i(x_i), \tilde{\sigma}_i^2) \tag{8.21}$$

$$\tilde{\sigma}_i^2 = \Delta t \sigma^2(i\Delta t), \tag{8.22}$$

where $\Delta t = T/L$ is the step size and L is the number of layers. For the inference model (*i.e.* the encoder), we assume

$$q(x_i \mid x_{i+1}) = \mathcal{N}(x_i; \hat{\mu}_{i+1}(x_{i+1}), \hat{\sigma}_{i+1}^2) \tag{8.23}$$

$$\hat{\mu}_i(x) = x + \Delta t(-\mu(x, i\Delta t) + \sigma(i\Delta t) a(x, T - i\Delta t)) \qquad \hat{\sigma}_i^2 = \Delta t \sigma^2(i\Delta t). \tag{8.24}$$

These transition kernels constitute a hierarchical variational autoencoder of L stochastic layers, whose marginal likelihood can be lower bounded by

$$\log p(x_L) \geq \mathbb{E}_q \left[\log p(x_0) + \sum_{i=0}^{L-1} \log \frac{p(x_{i+1} \mid x_i)}{q(x_i \mid x_{i+1})} \right] =: \mathcal{E}^L, \tag{8.25}$$

which we refer to as the discrete-time ELBO (DT-ELBO). The reconstruction error of the stochastic layer can be seen as some form of finite difference approximation to differentiation, which gives rise to $\nabla \cdot \mu$ in the CT-ELBO in the infinitesimal limit (as Δt approaches 0). The regularization of $\|a\|^2$ pops up when we compare the difference

between $\tilde{\mu}_i$ and $\hat{\mu}_i$ using the Gaussian reparameterization to compute the reconstruction error. We formalize this idea in the following theorem.

Theorem 28 (Consistency). *Assume μ, σ, σ^{-2}, a, $\|a\|^2$ and their derivatives up to the fourth order are all bounded and continuous, and that σ is non-singular. Then $\mathcal{E}^L \to \mathcal{E}^\infty$ as $L \to \infty$.*

This theorem tells us that the CT-ELBO we derive for continuous-time diffusion models is not that different from the traditional ELBO, and that maximizing the CT-ELBO can be seen as training an infinitely deep hierarchical VAE. We present the proof in Appendix G.3, which formalizes the above intuition, using Taylor's theorem to control the polynomial approximation error, which will go to 0 as the step size Δt vanishes when the number of layers L increases to infinity.

8.6 SCALABLE TRAINING BY RANDOMIZING TIME INTEGRAL

Estimating the lower bound (8.18) still involves numerically solving the variational inference SDE, which can be time consuming and has the same complexity as evaluating the instantaneous change of variable of a neural ODE. In this section, we simplify the computation to make it more scalable. Our strategy is to derive a Monte Carlo estimator of this lower bound that does not require numerical integration, by making additional assumptions on the model.

First, by Tonelli's theorem (for $\|a\|^2$) and Fubini's theorem (for $\nabla \cdot \mu$, which is usually bounded for neural nets are Lipschitz), we can swap the time integral with the expectation:

$$\mathcal{E}^\infty = \mathbb{E}_{Y_T}[\log p_0(Y_T) \,|\, Y_0 = x] - \int_0^T \mathbb{E}_{Y_s}\left[\tfrac{1}{2}\,\|a(Y_s, s)\|_2^2 + \nabla \cdot \mu(Y_s, T-s) \,\middle|\, Y_0 = x\right] \mathrm{d}s. \tag{8.26}$$

This way, we can treat the time index s inside of the integral as a random variable[4], and draw it uniformly at random between 0 and T. As for the expectation, we only have Y_s left, since we can marginalize out all $Y_{s'}$ for $s' \neq s$. This means we only need to sample Y_s conditioned on Y_0, whose distribution, denoted by $q(y, s \mid y_0)$, is induced by the inference dynamics (8.19). Generally, $q(y, s \mid y_0)$ may not take a simple form, due to the potential nonlinearity of μ, σ and a, and we have to numerically solve the SDE to obtain a sample following $q(y, s \mid y_0)$. For the sake of computational simplicity, we assume the coefficients $f := -\mu + \sigma a$ and $g := \sigma$ of the inference SDE allow us to sample from $q(y, s \mid y_0)$ directly, so that we can avoid numerical integration. This can be done, for example, by letting the inference SDE take the following linear form:

$$\mathrm{d}Y_s = \underbrace{(A(s)Y_s + a(s))}_{f}\,\mathrm{d}s + g(s)\,\mathrm{d}\hat{B}_s. \tag{8.27}$$

Using the product rule, it can be shown that

$$Y_s = \Phi(s)\left[Y_0 + \int_0^s \Phi(r)^{-1}a(r)\,\mathrm{d}r + \int_0^s \Phi(r)^{-1}g(r)\mathrm{d}\hat{B}_r\right] \tag{8.28}$$

solves (8.27) [Karatzas and Shreve, 2014, Section 5.6], where $\Phi(s)$ is a matrix function satisfying

$$\frac{\mathrm{d}}{\mathrm{d}t}\Phi(s) = A(s)\Phi(s), \quad \Phi(0) = I. \tag{8.29}$$

For simplicity, lets assume A and g are both diagonal matrices so that any matrix multiplication is commutative. Then from (8.28) and Itô's isometry, we know that

$$Y_s \mid Y_0 \sim \mathcal{N}\left(\Phi(s)\left[Y_0 + \int_0^s \Phi(r)^{-1}a(r)\,\mathrm{d}r\right], \Phi(s)^2\int_0^s \Phi(r)^{-2}g^2(r)\,\mathrm{d}r\right). \tag{8.30}$$

An example is the temporally nonlinear, zero-reverting (*i.e.* $a = 0$) Ornstein-Uhlenbeck

[4] More generally, we can apply importance sampling to estimate the time integral to reduce variance.

(OU) process:

$$\mathrm{d}Y_s = -\theta(s)Y_s\,\mathrm{d}s + g(s)\,\mathrm{d}\hat{B}_s, \tag{8.31}$$

where $\theta(s)$ and $g(s)$ are both strictly positive, scalar-matrix functions[5]. The solution to (8.29) in this case is $\Phi(s) = \exp(-\int_0^s \theta(r)\,\mathrm{d}r)$, which means the mean value of $Y_s \mid Y_0$ is

$$\mathbb{E}[Y_s \mid Y_0] = \exp\left(-\int_0^s \theta(r)\,\mathrm{d}r\right) Y_0. \tag{8.32}$$

And the variance is

$$\mathrm{Var}(Y_s \mid Y_0) = \exp\left(-\int_0^s 2\theta(r)\,\mathrm{d}r\right)\int_0^s \exp\left(\int_0^r 2\theta(r')\,\mathrm{d}r'\right) g^2(r)\,\mathrm{d}r. \tag{8.33}$$

Since the prior distribution p_0 is typically standard Gaussian, we can set $g(s) = \sqrt{2\theta(s)}$ so that the stationary distribution of the inference SDE is standard Gaussian. We refer to this process as the *standard (generalized) OU process*. This way we have

$$\begin{aligned}
\mathrm{Var}(Y_s \mid Y_0) &= \exp\left(-\int_0^s 2\theta(r)\,\mathrm{d}r\right)\int_0^s \exp\left(\int_0^r 2\theta(r')\,\mathrm{d}r'\right)\cdot 2\theta(r)\,\mathrm{d}r && (8.34)\\
&= \exp\left(-\int_0^s 2\theta(r)\,\mathrm{d}r\right)\int_0^s \frac{\mathrm{d}}{\mathrm{d}r}\exp\left(\int_0^r 2\theta(r')\,\mathrm{d}r'\right)\,\mathrm{d}r && (8.35)\\
&= \exp\left(-\int_0^s 2\theta(r)\,\mathrm{d}r\right)\cdot \exp\left(\int_0^r 2\theta(r')\,\mathrm{d}r'\right)\Big|_0^s && (8.36)\\
&= 1 - \exp\left(-\int_0^s 2\theta(r)\,\mathrm{d}r\right). && (8.37)
\end{aligned}$$

From (8.32) and (8.37), we know that $q(y, s \mid y_0)$ will get closer to $N(0, I)$ as s increases, no matter what the initial value y_0 is. In fact the "aggregated posterior" $q(y, T) = \int q(y, T \mid y')q(y', 0)\,\mathrm{d}y'$ gets arbitrarily close to $N(0, I)$ by taking $T \to \infty$.

Theorem 29 (Convergence of the OU process)**.** *If* $\lim_{T\to\infty}\int_0^T \theta(r)\,\mathrm{d}r = \infty$*, then* $q(y, T)$ *following the standard OU process converges to* $\mathcal{N}(0, I)$ *in total variation as*

[5] In what follows we also use the scalar notation for simplicity.

$T \to \infty$.

As a result, as long as $\int_0^T \theta(r)\,\mathrm{d}r$ is sufficiently large, the aggregated posterior will be close to the prior distribution. Furthermore, we can sample Y_s conditioned on Y_0 according to (8.30, 8.32, 8.37) without needing to numerically integrate the inference SDE.

Computation-estimation trade-off We compare the training time of the continuous-time flows induced by a neural ODE and a diffusion model, fitting the distribution of a swissroll dataset. Neural ODE corresponds to zero diffusion $\sigma = 0$, and only the drift (or velocity) coefficient is parameterized by a neural network. We use the Hutchinson trace estimator to estimate the likelihood following [Grathwohl et al., 2019]. For the diffusion model, we use the variance-preserving SDE [Ho et al., 2020, Song et al., 2021c] as the inference SDE, which is an instantiation of the standard OU process and allows us to sample Y_s using a closed-form formula. We parameterize a using a neural net. We experiment with two different estimators for the divergence term $\nabla \cdot \mu$ in the loss functional (8.26), where $\mu = ga - f$: (1) the Hutchinson estimator (referred to as *slice*), and (2) the denoising score matching estimator (referred to as *denoising*). See the next section for more details.

The trained models are visualized in Figure 8.1 (ODE on the left, diffusion model with a fixed OU inference process on the right), the learning curves presented in Figure 8.2. From the learning curve figures, we see that neg-likelihood decreases rapidly for the continuous-time flow in the number of parameter updates. However, once the x-axis is normalized by runtime, the convergence speed becomes almost indistinguishable. This is because for continuous-time flows, numerical integration takes time, whereas for the diffusion models, we train on a random time step s; that is, within a fixed amount of time the latter can make more parameter updates at the cost of noisier gradients. Note that both models have constant memory cost (wrt T or L, the number of integration steps), so a large batch size can be used to reduce variance for training. The benefit for

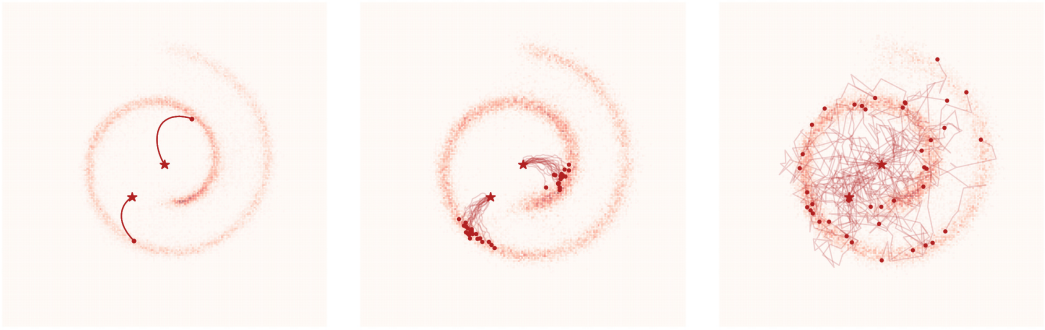


Figure 8.1: Three special cases of generative SDEs. The stars indicate the initial values, followed by some random sample paths. (*left*) trained with no diffusion $\sigma = 0$ (*i.e.* neural ODE). (*center*) trained with some fixed diffusion $\sigma > 0$. (*right*) trained with a fixed OU process as the inference SDE.

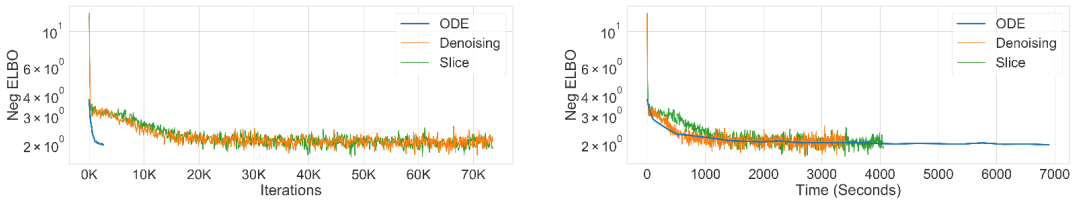


Figure 8.2: Neural ODE vs diffusion model with a fixed OU inference process (using denoising or slice score matching estimator). The learning curves are presented as a function of iterations (*left*) and runtime (*right*) to emphasize the computational distinction between the two families of models.

estimating the time index would be more drastic for higher dimensional, larger scale experiments, where numerical solvers would require more function evaluations. This makes diffusion models the more scalable choice between the two, and we can also use higher capacity networks to parameterize the model through the variational degree of freedom a.

8.7 SCORE-BASED GENERATIVE MODELLING

Maximum likelihood training of diffusion models is also deeply connected to score matching for energy-based models (see Sections 1.1 and 1.3). To see that, we redefine

the variational a function as $a = g^\top \boldsymbol{s}_\theta$, where $\boldsymbol{s}_\theta$ is the trainable variable (which we will see soon is an approximate score function). Following the reparameterization $f := -\mu + \sigma a$ and $g := \sigma$, the generative SDE and inference SDE can be written as

$$\mathrm{d}X = (gg^\top \boldsymbol{s}_\theta - f)\,\mathrm{d}t + g\,\mathrm{d}B_t \quad \text{and} \quad \mathrm{d}Y = f\,\mathrm{d}s + g\,\mathrm{d}\hat{B}_s. \tag{8.38}$$

The generative SDE can be seen as an approximate, parametric version of the *reverse SDE* [Anderson, 1982][6]:

$$\mathrm{d}X = (gg^\top \nabla \log q(y, T-t) - f)\,\mathrm{d}t + g\,\mathrm{d}B_t, \tag{8.39}$$

which induces the same probability law over the sample paths as the inference SDE if the prior matches the aggregated posterior $p(\cdot, 0) = q(\cdot, T)$. Therefore, we know that if $\boldsymbol{s}_\theta$ approximates the score $\nabla \log q$ well, then we can simulate realistic data samples by substituting the approximate score function into (8.39), which becomes the generative SDE in (8.38), and numerically integrating the dynamics. For this reason, we refer to this reparameterized generative SDE as the *plug-in reverse SDE*.

The CT-ELBO (8.26) associated with the generative and inference SDEs (8.38) can be written as

$$\mathcal{E}^\infty = \mathbb{E}_{Y_T}[\log p_0(Y_T) \mid Y_0 = x] - \int_0^T \mathbb{E}_{Y_s}\left[\tfrac{1}{2}\,\|\boldsymbol{s}_\theta\|^2_{gg^\top} + \nabla \cdot (gg^\top \boldsymbol{s}_\theta - f) \,\middle|\, Y_0 = x\right] \mathrm{d}s. \tag{8.40}$$

Comparing the integrand to the ISM loss in Table 1.1, we immediately see that the network $\boldsymbol{s}_\theta$ approximates $\nabla \log q(y, s)$, the score function of the marginal density of Y_s. That is, *matching the score of $q(y, t)$ amounts to maximizing the lower bound on the marginal likelihood of the plug-in reverse SDE.*

[6] We note that the description of the original reverse SDE in Anderson [1982] is more stringent as they require the independence property of the "reverse" Brownian motion, and the reversal is "pointwise". As we only concern about the statistical property of the process, we stick to our notation of Brownian motions where X and Y do not even need to lie in the same probability space.

Equivalently, from (A.4), we can rewrite the second term of (8.40) using the DSM loss

$$
\begin{aligned}
&-\int_0^T \mathbb{E}_{Y_s}\left[\frac{1}{2}\left\|\boldsymbol{s}_\theta\right\|^2_{gg^\top} - \boldsymbol{s}_\theta^\top gg^\top \nabla \log q(Y_s, s \mid Y_0) - \nabla \cdot f \,\middle|\, Y_0 = x\right] \mathrm{d}s \qquad (8.41)\\
&= -\int_0^T \mathbb{E}_{Y_s}\left[\frac{1}{2}\left\|\boldsymbol{s}_\theta - \nabla \log q(Y_s, s \mid Y_0)\right\|^2_{gg^\top} - \frac{1}{2}\left\|\nabla \log q\right\|^2_{gg^\top} - \nabla \cdot f \,\middle|\, Y_0 = x\right] \mathrm{d}s.
\end{aligned}
$$

This means we can avoid calculating the divergence term by replacing it with the mean-squared-error between the approximate score and the conditional score of the data. In the case of the OU process, the mean-squared-error can be seen as the reconstruction error of a denoising autoencoder [Vincent, 2011] since the conditional distribution of the data is just Gaussian.

More generally, since we only care about the probability flow induced by the stochastic dynamics, we can generalize the reverse SDE (8.39) by looking at the marginal probability induced by the inference SDE. First of all, lets define a notion of marginal equivalent SDE:

Definition 30 (**Marginally equivalent processes / SDEs**). *Let Y_s, $\tilde{Y}_s$ and X_t be stochastic processes for $0 \leq s, t \leq T$. If Y_s and $\tilde{Y}_s$ have the same distribution for all s, then they are said to be marginally equivalent. If X_t and Y_{T-t} have the same distribution for all t, then we say X_t is a marginally equivalent reverse process. Two SDEs are equivalent if the processes they induce are equivalent. Two SDEs are equivalent reverse of each other if the processes they induce are equivalent reverse of one another.*

Note that when talking about the equivalency between SDEs, the dependency on an initial condition is implied. As in the following context, it is clear that we are always interested in equivalent processes or SDEs that have the same marginal distributions, we will also omit saying "marginally" repeatedly. Now, we show how to construct a family of equivalent (reverse) SDEs generalizing (8.39). Let Y_s solve

$$
\mathrm{d}Y = f\,\mathrm{d}s + g\,\mathrm{d}\hat{B}_s.
$$

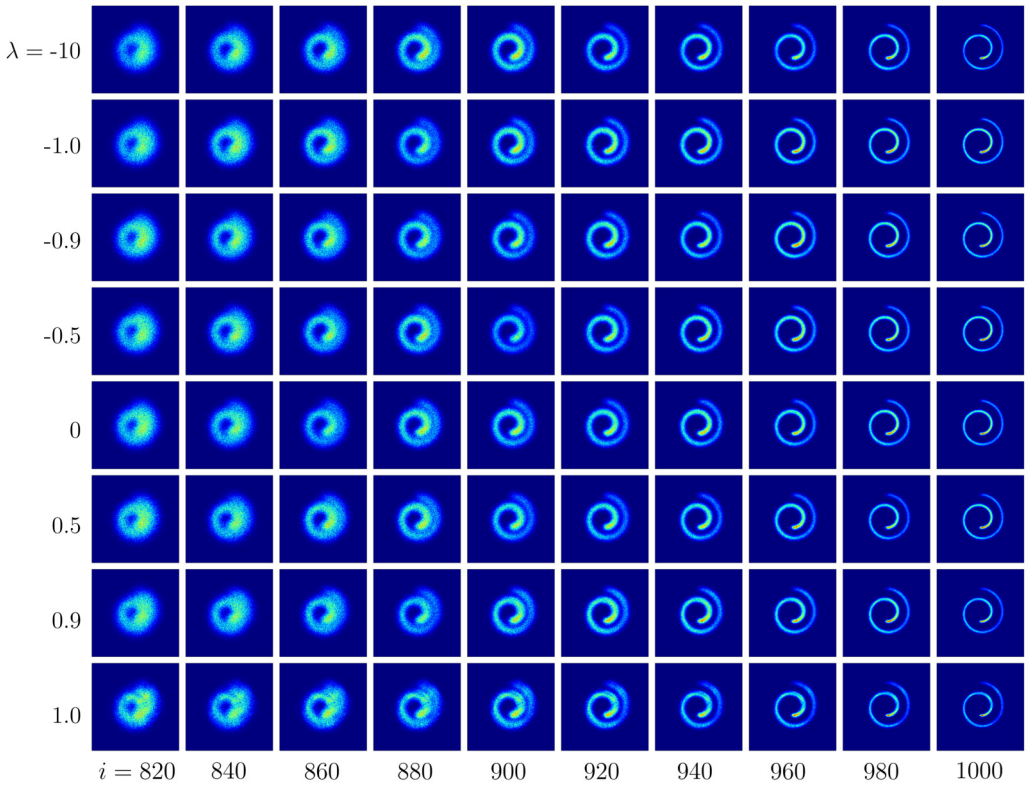


Figure 8.3: Samples from plug-in reverse SDEs with different λ values (rows). We use the same score function s_θ trained on the Swiss roll dataset, and use it to define the λ-plug-in reverse SDE (8.46). For generation, we use the Euler Maruyama method with a step size of $\Delta t = 1/1000$. We visualize the samples for the i-th iterates (columns). As the figure shows, different rows correspond to different values of λ, but they all induce the same marginal density per column.

We assume g is position-independent and diagonal for simplicity. Let $\lambda \leq 1$, We can rearrange the Fokker-Planck equation to get

$$\partial_s q = -\nabla \cdot (fq) + \frac{1}{2} g^2 : H_q = -\nabla \cdot \left(\left(f - \frac{\lambda}{2} g^2 \nabla \log q \right) q \right) + \frac{1-\lambda}{2} g^2 : H_q. \tag{8.42}$$

Now let $f_\lambda := f - \frac{\lambda}{2} g^2 \nabla \log q$, and $g_\lambda := \sqrt{1-\lambda} g$. Then the SDE $\mathrm{d}Y = f_\lambda \, \mathrm{d}s + g_\lambda \, \mathrm{d}\hat{B}_s$ has the same Fokker-Planck equation as (8.42), which means the SDEs defined this way form a family of equivalent SDEs[7].

To construct an equivalent reverse SDE, we rearrange the Fokker-Planck of this new SDE,

$$\partial_s q = -\nabla \cdot (f_\lambda q) + \frac{1}{2} g_\lambda^2 : H_q = -\nabla \cdot \left(\left(f_\lambda - g_\lambda^2 \nabla \log q \right) q \right) - \frac{1}{2} g_\lambda^2 : H_q. \tag{8.43}$$

Now let $\mu_\lambda(x,t) := g_\lambda^2(x, T-t) \nabla \log q(x, T-t) - f_\lambda(x, T-t)$ and $\sigma_\lambda = g_\lambda(x, T-t)$. Then the SDE $\mathrm{d}X = \mu_\lambda \, \mathrm{d}t + \sigma_\lambda \, \mathrm{d}B_t$ with the initial condition $X_0 \sim q(\cdot, T)$ is an equivalent reverse SDE, since its marginal density p solving the Fokker-Planck equation

$$\partial_t p = -\nabla \cdot (\mu_\lambda p) + \frac{1}{2} \sigma_\lambda^2 : H_p = \nabla \cdot \left(\left(f_\lambda - g_\lambda^2 \nabla \log q \right) p \right) + \frac{1}{2} g_\lambda^2 : H_p \tag{8.44}$$

is the time-reversal of the marginal density q solving (8.43). This also means there is a family of λ-plug-in reverse SDEs parameterized by λ and $\boldsymbol{s}_\theta$:

$$\mathrm{d}X = (g_\lambda^2 \boldsymbol{s}_\theta - f_\lambda) \, \mathrm{d}t + \sigma_\lambda \, \mathrm{d}B_t \tag{8.45}$$

$$= \left(\left(1 - \frac{\lambda}{2} \right) g^2 \boldsymbol{s}_\theta - f \right) \mathrm{d}t + \sqrt{1-\lambda} g \, \mathrm{d}B_t. \tag{8.46}$$

The plug-in reverse SDE (8.38) corresponds to $\lambda = 0$, and setting $\lambda = 1$ gives us an equivalent (plug-in) reverse ODE. We illustrate different marginally equivalent SDEs and ODE in Figure 8.3 using a learned score function.

To summarize, using f_λ, g_λ, μ_λ and σ_λ, we can define the following generative and

[7] Note that more generally the same would also hold if we let λ be a time-dependent function.

inference pair

$$\mathrm{d}X = \left(\left(1-\tfrac{\lambda}{2}\right) g^2 \boldsymbol{s}_\theta - f\right)\,\mathrm{d}t + \sqrt{1-\lambda}g\,\mathrm{d}B_t \;\text{ and }\; \mathrm{d}Y = \left(f - \tfrac{\lambda}{2} g^2 \nabla \log q\right)\,\mathrm{d}s + \sqrt{1-\lambda}g\,\mathrm{d}\hat{B}_s. \tag{8.47}$$

We show that maximizing the ELBO of this family of plug-in reverse SDEs is also equivalent to performing score matching.

Theorem 31 (Plug-in reverse SDE ELBO). *Assume the generative and inference SDEs follow (8.47). For $\lambda < 1$, then the CT-ELBO (denoted by $\mathcal{E}_\lambda^\infty$) can be written as*

$$\begin{aligned}\mathcal{E}_\lambda^\infty = \mathbb{E}_{Y_T}[\log p_0(Y_T)\,|\,Y_0 = x] - \int_0^T \Bigg(1-\frac{\lambda}{2}\Bigg)\,\mathbb{E}_{Y_s}\left[\frac{1}{2}\left\|\boldsymbol{s}_\theta\right\|_{g^2}^2 + \nabla\cdot\left(g^2\boldsymbol{s}_\theta - \left(\frac{2}{2-\lambda}\right) f\right)\,\middle|\, Y_0 = x\right] \\ + \frac{\lambda}{2}\,\mathbb{E}_{Y_s}\left[\frac{1}{2}\left\|\boldsymbol{s}_\theta\right\|_{g^2}^2 - g^2\boldsymbol{s}_\theta^\top \nabla \log q(Y_s, s)\,\middle|\, Y_0 = x\right] \\ + \frac{\lambda^2}{4(1-\lambda)}\,\mathbb{E}_{Y_s}\left[\frac{1}{2}\left\|\boldsymbol{s}_\theta - \nabla\log q(Y_s,s)\right\|_{g^2}^2\,\middle|\, Y_0 = x\right]\,\mathrm{d}s.\end{aligned}$$

Furthermore, averaging the ELBO over the data distribution yields

$$\begin{aligned}\mathbb{E}_{Y_0}[\mathcal{E}_\lambda^\infty] &= \mathbb{E}_{Y_T}[\log p_0(Y_T)] - \int_0^T \left(1 + \frac{\lambda^2}{4(1-\lambda)}\right)\,\mathbb{E}_{Y_s}\left[\frac{1}{2}\left\|\boldsymbol{s}_\theta\right\|_{g^2}^2 + \nabla\cdot\left(g^2\boldsymbol{s}_\theta\right)\right]\,\mathrm{d}s \\ &\quad + Const. \qquad (8.48) \\ &= \mathbb{E}_{Y_0}[\mathcal{E}_0^\infty] - \left(\frac{\lambda^2}{4(1-\lambda)}\right)\int_0^T \mathbb{E}_{Y_s}\left[\frac{1}{2}\left\|\boldsymbol{s}_\theta(Y_s,s) - \nabla\log q(Y_s,s)\right\|_{g^2}^2\right]\,\mathrm{d}s. \qquad (8.49)\end{aligned}$$

Before concluding this section with some experiments and algorithmic innovation, we first make a few remarks:

1. Setting $\lambda = 0$, this ELBO will reduce to the special case of (8.40).
2. Equation (8.48) tells us that *by minimizing the score matching loss weighted by g^2, we implicitly maximize the likelihood of a continuum of plug-in reverse SDEs*, which is visualized in Figure 8.4.

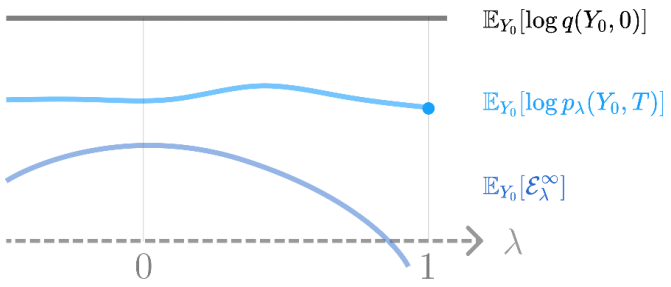


Figure 8.4: Lower bound on the marginal likelihood of a continuum of plug-in reverse SDEs. The lower bound is optimized when the score matching loss is minimized, which will push up the entire dark blue curve, the expected log-likelihood.

3. Equation (8.49) tells us that the average CT-ELBO is maximized when $\lambda = 0$ (also see Figure 8.4).

4. The theorem excludes the case where $\lambda = 1$, *i.e.* the equivalent ODE, since otherwise there will be a division-by-zero problem. But an ODE can be seen as having λ very close to 1, which will make the SDE effectively deterministic in practice. This suggests the likelihood of the equivalent ODE can be improved by minimizing the score matching loss, as the ODE's likelihood will be close to plug-in reverse SDEs with $\lambda \approx 1$. In practice, we can only estimate the ELBO of the case $\lambda = 0$ since otherwise there will be some constants we do not have access to, but their gradients wrt θ can all be estimated via score matching. The exact likelihood of $\lambda = 1$ can be estimated using the instantaneous change of variable and Hutchinson's trace estimator, as discussed in section 7.3.

Bias and variance trade-off The integral in equation (8.40) can be estimated by sampling (Y_s, s), and using the Hutchinson trace estimator to estimate the divergence, which corresponds to implicit score matching. However, in practice the variance of this estimator is very high when the norm of the Jacobian $\nabla \boldsymbol{s}_\theta$ is large (recall Section 7.4).

Another popular approach is to use the denoising estimator (8.41), which is equal to

$$\mathbb{E}_{Y_s}\left[\frac{1}{2}\left\|\boldsymbol{s}_\theta(Y_s, s) - \nabla \log q(Y_s, s \mid Y_0)\right\|^2_{gg^\top} \,\middle|\, Y_0 = x\right], \tag{8.50}$$

up to some normalizing constant. The conditional density $q(y, s \mid y_0)$ is simply Gaussian by (8.30) if the inference SDE is linear, which also allows us to sample $Y_s \mid Y_0$ easily. Denote by μ_s, σ_s^2 the mean and variance of the Gaussian, where μ_s and σ_s are functions of Y_0 and s. In this case, if we reparameterize $Y_s = \mu_s + \sigma_s\epsilon$ where $\epsilon \sim \mathcal{N}(0, I)$, then the score becomes $\nabla \log q = -\frac{\epsilon}{\sigma_s}$. Since $\sigma_s \to 0$ as $s \to 0$, this estimator normally has unbounded variance. Song and Ermon [2019], Song et al. [2021c] propose to remedy this by multiplying the DSM loss by σ_s^2/g^2 (assuming g is a scalar for simplicity), so that the target has constant magnitude on average $\mathbb{E}[\frac{1}{2}\left\|\sigma_s\boldsymbol{s}_\theta + \epsilon\right\|^2]$, which would result in a biased gradient estimate with much smaller variance. We can debias this estimator by performing importance sampling, *i.e.* sampling the random time index according to a density $q(s)$ that is proportional to g^2/σ_s^2 (recall that the non-uniform sampling is to cancel out the non-uniform weighting of the losses, so that the resulting estimator becomes unbiased; see [Owen, 2013] for some background on importance sampling). This ratio, however, is usually not normalizable in practice (as it integrates to ∞). As an alternative, we consider the following unnormalized density $\tilde{q}_\epsilon(s) = g^2(s_\epsilon)/\sigma_{s_\epsilon}^2$ for $s \in [0, s_\epsilon]$, and $\tilde{q}_\epsilon(s) = g^2(s)/\sigma_s^2$ for $s \in [s_\epsilon, T]$. We experiment with this debiased procedure by sampling $s \sim q_\epsilon \propto \tilde{q}_\epsilon$, for f and g chosen to be the variance-preserving SDE. s_ϵ is small so that the bias is negligible.

We train the model on MNIST [LeCun et al., 1998] and CIFAR10 [Krizhevsky et al., 2009]. We present the learning curves and the standard error of the estimate of the ELBO in Figure 8.5. The lower bound is estimated using the Hutchinson trace estimator with s sampled uniformly from $[0, T]$, with the same batch size, so the only thing that will affect the dispersion is the magnitude of $\nabla\boldsymbol{s}_\theta$. Since smaller values of s are more likely to be sampled under q_ϵ, the debiased model will see samples with less perturbation more often. On the contrary, sampling s uniformly will bias the model to learn from noisier data, causing the learned score to be smoother. We also experiment with parameterizing $\boldsymbol{s}_\theta$ vs parameterizing a. We find the latter parameterization to work

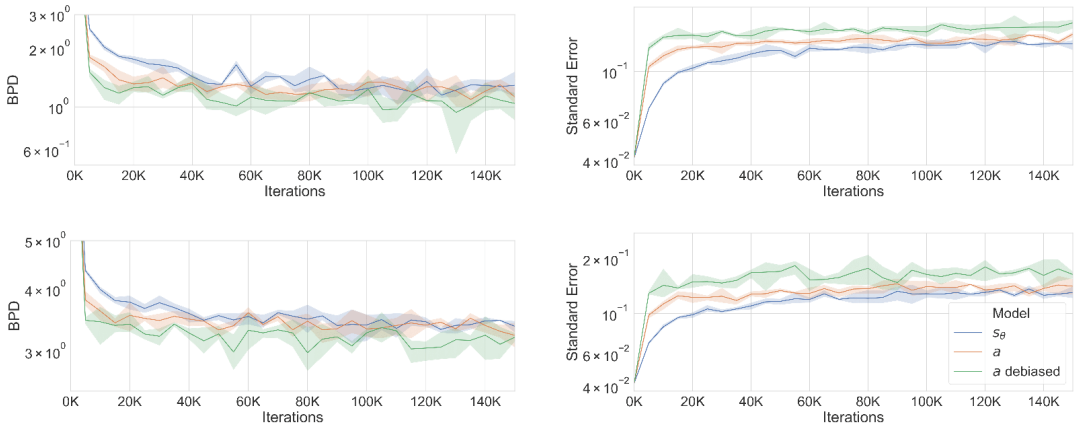


Figure 8.5: Likelihood estimation on MNIST (*top*) and CIFAR10 (*bottom*). $\boldsymbol{s}_\theta$ and a denote which model we parameterize. Y-axes are bits-per-dim (BPD) and the standard error of BPD of the test set. The debiased curves improve upon the original biased gradient estimator [Song et al., 2021c] since it maximizes a proper ELBO. Shaded area reflects the uncertainty estimated by 3 random seeds.

slightly better since the relationship $\boldsymbol{s}_\theta = g^{-1}a$ has the effect of negating the multiplier σ_s in the reweighted loss, *i.e.* $\mathbb{E}[\frac{1}{2}\left\|\frac{\sigma_s}{g}a + \epsilon\right\|^2]$. This is similar to the noise conditioning technique introduced in Song and Ermon [2020].

8.8 DISCUSSION

In this chapter, we generalize deterministic dynamical systems by injecting Brownian noise. Mirroring the deterministic case in the preceding chapter, we examine changes in density from both Eulerian and Lagrangian perspectives. While the Eulerian view involves the Fokker-Planck equation, the Lagrangian perspective yields a stochastic extension of the instantaneous change-of-variable formula using the Feynman-Kac theorem. Unlike the deterministic case, this representation is not immediately tractable as it involves the challenge of marginalizing Brownian noise. To address this, we employ Girsanov's theorem to provide a variational approximation to the Brownian process, leading to an ELBO reminiscent of VAE. For improved training scalability,

we establish a fixed inference process and redefine the generative process through reparameterization [Sohl-Dickstein et al., 2015, Ho et al., 2020]. Lastly, we establish a formal connection to score matching [Hyvärinen and Dayan, 2005, Song et al., 2021c] through another reparameterization of the variational degree of freedom.

8.9 IMPACT, RELATED WORK AND RECENT DEVELOPMENTS

Diffusion models Generative modeling inspired by diffusion processes was first explored by Sohl-Dickstein et al. [2015] in the discrete-time setting, which was then refined and scaled up by Ho et al. [2020]. Different from Ho et al. [2020], which shows the ELBO of discrete-time diffusion process can be likened to DSM (Section 3.2 of the paper), we show that ISM loss naturally arises from the Fokker-Planck equation of the marginal density, via the Feynman-Kac representation and the Girsanov change of measure. Similarly to ours, Kingma et al. [2021] proposed to maximize the continuous limit of DT-ELBO, which they show to be invariant to noise scheduling for a particular choice of inference family.

Diffusion models have been successfully applied to modeling high-dimensional natural images [Dhariwal and Nichol, 2021, Saharia et al., 2021], audio [Kong et al., 2021, Chen et al., 2021], 3D point cloud [Cai et al., 2020, Zhou et al., 2021], and discrete data [Hoogeboom et al., 2021, Austin et al., 2021].

One benefit of the continuous-time formulation is that stochastic calculus can be naturally married with differential geometry to model non-Euclidean data. The variational framework presented in this chapter has been extended to various Riemannian manifolds [Huang et al., 2022]. Taking one step further, Benton et al. [2022] greatly generalized the theory to account for general state space using continuous-time Markov models (termed denoising Markov models by the authors), which subsumes the Riemannian diffusion models [Huang et al., 2022] and continuous-time Markov chains for discrete data [Campbell et al., 2022].

Score matching and generative modeling Score matching was originally proposed in the context of learning an unnormalized density, *i.e.* an EBM [Hyvärinen and Dayan, 2005]. Sampling from an EBM is typically very costly, as it relies on iterative methods such as MCMC (recall the discussion on EBM in Section 1.1). This further complicates the process of model development and debugging, as it is hard to tell if the unsuccessful generation of samples should be attributed to the model itself or the sampling procedure.

Song and Ermon [2019] first proposed to directly parameterize the score as the trainable degree of freedom, instead of the unnormalized density or the energy. Second, they also proposed to learn a sequence of score functions corresponding to different amounts of noise being injected to perturb the data distribution. The Langevin dynamic used for generating samples was also adapted to account for the noise scheduling, resulting in an annealed Langevin dynamics. This framework was subsequently extended to continuous time [Song et al., 2021c], where data generation was framed as an approximate reversal of the diffusion process used to perturb the data.

After the connection between SDE and score-based generative models were drawn, Durkan and Song [2021] attempted to establish an equivalency between maximum likelihood and score matching, by showing that KL divergence can be represented as an integral of weighted Fisher divergence:

$$D_{\mathrm{KL}}(q(y,0)||r(y,0)) = \frac{1}{2}\int_0^T \mathbb{E}_{q(\cdot,s)}\left[\|\nabla \log r(Y_s,s) - \nabla \log q(Y_s,s)\|^2_{gg^\top}\right] \mathrm{d}s, \tag{8.51}$$

where $r(y,s)$ is the density of Y_s solving the same inference SDE with the initial condition $y_0 \sim r(\cdot,0)$, assuming $q(y,T) = r(y,T)$. However, it is inaccurate to claim that score matching is equivalent to maximum likelihood. This is because if we simply let $r(y,0) = p(y,T)$, *i.e.* the density of the generative SDE evaluated at y, $r(y,s)$ will not necessarily be the same as either $p(y,T-s)$ or $\boldsymbol{s}_\theta(y,s)$. This means the KL divergence is not equal to the integral of the weighted score matching loss $\mathbb{E}[\frac{1}{2}\|\boldsymbol{s}_\theta - \nabla \log q\|^2_{gg^\top}]$. In fact, the latter corresponds to a lower bound on the likelihood (the cross-entropy term of the KL) up to some constant, as equation (8.40)

suggests. Concurrently to our work, [Song et al., 2021a] later on rectified this by proving the same bound as we derived, *i.e.* (8.40).

Equation (8.51) is a generalization of Lyu [2009], where the inference perturbation is a simple Brownian motion. This type of formulas fall into the category of de Bruijn's identity [Cover, 1999] for relative entropy. A similar differential form result can be found in Wibisono et al. [2017].

More recently, Dockhorn et al. [2022] proposed to augment the dynamic with a velocity (momentum) variable, inspired by statistical mechanics. This has the benefit of an accelerated dynamics and a smoother score function.

Learning SDEs Tzen and Raginsky [2019], Li et al. [2020] also propose to learn a neural SDE by applying Girsanov's theorem. The key difference is that they treat the SDE entirely as a latent variable, with an additional emission probability, whereas we use the Feynman-Kac formula to directly express the marginal density as an expectation, side-stepping the need to smooth out the density using the emission probability (which will be a Dirac point mass in our case). In their case, the inference direction is the same as the generative direction, since they infer the latent SDE directly, whereas we apply Girsanov to the Feynman-Kac diffusion (opposite the generative direction). Xu et al. [2022] further apply neural SDE as an infinitely deep Bayesian neural network.

Optimal transport Diffusion models are related to optimal transport (recall Chapter 5) through the Schrödinger bridge (SB) problem – which is about finding the process that connects two prescribed marginal measures and at the same time minimizes the KL divergence from a reference measure over the sample paths. When the reference measure is taken to be the classical Wiener process for Brownian motion, the static version of the SB problem can be shown to be equivalent to entropy regularized optimal transport. Like optimal transport, Solving the SB problem in general is very non-trivial.

De Bortoli et al. [2021] proposed to solve this problem by iteratively matching the score functions of the diffusion processes corresponding to the generative and inference SDEs. Chen et al. [2022] presented a different computational framework based on the forward-backward SDE theory in stochastic control [Ma et al., 1999], which allows them to derive likelihood objectives for SB that generalize (8.40).

9 Conclusion

In this thesis, we embarked on a quest to design versatile likelihood-based generative models, aiming for a delicate balance. Our primary goal was to introduce the right level of structure, ensuring tractability and scalability while maintaining distributional universality.

We began with flow-based methods, starting from the foundational 1D case. Here, we characterized 1D flows as monotone functions and proposed various flexible monotone architectures. Subsequently, we extended these architectures to the high-dimensional spaces, leveraging triangle maps and convex potential maps. This led us to establish connections with autoregressive models and the optimal transport theory.

Along the way, we introduced an augmentation technique. This innovation expanded the design space of flow-based methods, enhancing their expressivity. Notably, the augmentation lifted the representation to a higher-dimensional state space, allowing us to encode data in an augmented latent vector akin to the latent space of a VAE. It is worth noting that recent works have further explored the potential of this technique, underscoring its significance in the field.

In the latter part of our journey, we explored using neural differential equations as generative models, unveiling an even more flexible class of probability flows. In both the deterministic and the stochastic case, we tackled the change in likelihood from two viewpoints: that of a bystander and the perspective of particles navigating through space and time. This dual approach not only enhanced our understanding of the models' behavior but also made likelihood computation tractable.

Furthermore, our theoretical analysis showcased the universality of continuous-time flows. Unlike the limitations of triangular maps and convex potential maps, continuous-time flows demonstrated the ability to approximate any target coupling, underlining their flexibility and potential.

In the stochastic case, we improved the scalability of the model by adopting a linear Gaussian inference process. At the end, we established a formal connection to the concept of score matching, further enhancing our model's versatility and utility.

With these contributions, we conclude our journey towards the design of likelihood-based generative models, unified by a commitment to distributional universality, tractability, and scalability, by analyzing the probability flows associated with the generative process.

Index

References

Ryan P Adams, Jeffrey Pennington, Matthew J Johnson, Jamie Smith, Yaniv Ovadia, Brian Patton, and James Saunderson. Estimating the spectral density of large implicit matrices. *arXiv preprint arXiv:1802.03451*, 2018.

Shubhankar Agarwal, Harshit Sikchi, Cole Gulino, and Eric Wilkinson. Imitative planning using conditional normalizing flow. *arXiv preprint arXiv:2007.16162*, 2020.

Siddharth Agrawal and Ambedkar Dukkipati. Deep variational inference without pixel-wise reconstruction. *arXiv preprint arXiv:1611.05209*, 2016.

Luigi Ambrosio, Nicola Gigli, and Giuseppe Savaré. *Gradient flows: in metric spaces and in the space of probability measures*. Springer Science & Business Media, 2008.

Brandon Amos, Lei Xu, and J Zico Kolter. Input convex neural networks. In *International Conference on Machine Learning*, pages 146–155. PMLR, 2017.

Brian DO Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

Muhammad Asim, Max Daniels, Oscar Leong, Ali Ahmed, and Paul Hand. Invertible generative models for inverse problems: mitigating representation error and dataset bias. In *International Conference on Machine Learning*, pages 399–409. PMLR, 2020.

Jacob Austin, Daniel Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. 34, 2021.

Atılım Günes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *The Journal of Machine Learning Research*, 18(1):5595–5637, 2017.

Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *Journal of Marchine Learning Research*, 18: 1–43, 2018.

Jens Behrmann, Will Grathwohl, Ricky T. Q. Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks. In *International Conference on Machine Learning*, pages 573–582, 2019.

Mohamed Ishmael Belghazi, Sai Rajeswar, Olivier Mastropietro, Negar Rostamzadeh, Jovana Mitrovic, and Aaron Courville. Hierarchical adversarially learned inference. *arXiv preprint arXiv:1802.01071*, 2018.

Joe Benton, Yuyang Shi, Valentin De Bortoli, George Deligiannidis, and Arnaud Doucet. From denoising diffusions to denoising markov models. *arXiv preprint arXiv:2211.03595*, 2022.

Patrick Billingsley. *Convergence of probability measures*. Wiley Series in Probability and Statistics: Probability and Statistics. John Wiley & Sons Inc., New York, second edition, 1999. ISBN 0-471-19745-9. A Wiley-Interscience Publication.

Patrick Billingsley. *Probability and measure*. John Wiley & Sons, 2008.

Avishek Joey Bose and Ivan Kobyzev. Equivariant discrete normalizing flows. *arXiv preprint arXiv:2110.08649*, 2021.

Léon Bottou. Online learning and stochastic approximations. *On-line learning in neural networks*, 17(9): 142, 1998.

Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018.

Michelle Boué, Paul Dupuis, et al. A variational representation for certain functionals of brownian motion. *The Annals of Probability*, 26(4):1641–1659, 1998.

Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Johann Brehmer and Kyle Cranmer. Flows for simultaneous manifold learning and density estimation. *Advances in Neural Information Processing Systems*, 33, 2020.

Yann Brenier. Décomposition polaire et réarrangement monotone des champs de vecteurs. *CR Acad. Sci. Paris Sér. I Math.*, 305:805–808, 1987.

Yann Brenier. Polar factorization and monotone rearrangement of vector-valued functions. *Communications on pure and applied mathematics*, 44(4):375–417, 1991.

Haim Brezis and Haim Brézis. *Functional analysis, Sobolev spaces and partial differential equations*, volume 2. Springer, 2011.

Philippe Brouillard, Sébastien Lachapelle, Alexandre Lacoste, Simon Lacoste-Julien, and Alexandre Drouin. Differentiable causal discovery from interventional data. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. In *International Conference on Learning Representations*, 2015.

Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on scientific computing*, 16(5):1190–1208, 1995.

Ruojin Cai, Guandao Yang, Hadar Averbuch-Elor, Zekun Hao, Serge Belongie, Noah Snavely, and Bharath Hariharan. Learning gradient fields for shape generation. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part III*, 2020.

Andrew Campbell, Joe Benton, Valentin De Bortoli, Tom Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. *arXiv preprint arXiv:2205.14987*, 2022.

Jianfei Chen, Cheng Lu, Biqi Chenli, Jun Zhu, and Tian Tian. Vflow: More expressive generative flows with variational data augmentation. In *International Conference on Machine Learning*, pages 1660–1669. PMLR, 2020.

Nanxin Chen, Yu Zhang, Heiga Zen, Ron J Weiss, Mohammad Norouzi, and William Chan. Wavegrad: Estimating gradients for waveform generation. In *International Conference on Learning Representations*, 2021.

Ricky T. Q. Chen, Jens Behrmann, David Duvenaud, and Jörn-Henrik Jacobsen. Residual flows for invertible generative modeling. In *Advances in Neural Information Processing Systems*, 2019a.

Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, 2018.

Tianrong Chen, Guan-Horng Liu, and Evangelos A Theodorou. Likelihood training of schr\" odinger bridge using forward-backward sdes theory. In *International Conference on Learning Representations*, 2022.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pages 2172–2180, 2016a.

Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder. *arXiv preprint arXiv:1611.02731*, 2016b.

Yize Chen, Yuanyuan Shi, and Baosen Zhang. Optimal control via neural networks: A convex approach. In *International Conference on Learning Representations*, 2019b.

Rewon Child. Very deep vaes generalize autoregressive models and can outperform them on images. In *International Conference on Learning Representations*, 2020.

Earl A Coddington and Norman Levinson. *Theory of ordinary differential equations*. Tata McGraw-Hill Education, 1955.

Joseph Paul Cohen, Tianshi Cao, Joseph D Viviano, Chin-Wei Huang, Michael Fralick, Marzyeh Ghassemi, Muhammad Mamdani, Russell Greiner, and Yoshua Bengio. Problems in the deployment of machine-learned models in health care. *CMAJ*, 193(35):E1391–E1394, 2021a.

Samuel Cohen, Brandon Amos, and Yaron Lipman. Riemannian convex potential maps. In *International Conference on Machine Learning*, 2021b.

Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999, 2016.

Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

Kyle Cranmer, Johann Brehmer, and Gilles Louppe. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 117(48):30055–30062, 2020.

Chris Cremer, Xuechen Li, and David Duvenaud. Inference suboptimality in variational autoencoders. In *International Conference on Machine Learning*, 2018.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4), 1989.

Bernard Dacorogna and Jürgen Moser. On a partial differential equation involving the jacobian determinant. In *Annales de l'Institut Henri Poincaré C, Analyse non linéaire*, volume 7, pages 1–26. Elsevier, 1990.

Germund Dahlquist and Åke Björck. *Numerical methods in scientific computing, volume i*. SIAM, 2008.

Peter Dayan, Geoffrey E Hinton, Radford M Neal, and Richard S Zemel. The helmholtz machine. *Neural computation*, 7(5):889–904, 1995.

Valentin De Bortoli, James Thornton, Jeremy Heng, and Arnaud Doucet. Diffusion schr\" odinger bridge with applications to score-based generative modeling. In *Advances in Neural Information Processing Systems*, 2021.

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

William Murray Deen. *Analysis of transport phenomena*, volume 2. Oxford university press New York, 1998.

Morris H DeGroot and Mark J Schervish. *Probability and statistics*. Pearson Education, 2012.

Luc Devroye. *Non-Uniform Random Variate Generation*. Springer-Verlag, 1986.

Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis. In *Advances in Neural Information Processing Systems*, 2021.

Manuel Dibak, Leon Klein, and Frank Noé. Temperature steerable flows and boltzmann generators. *arXiv preprint arXiv:2108.01590*, 2021.

Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

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

Tim Dockhorn, Arash Vahdat, and Karsten Kreis. Score-based generative modeling with critically-damped langevin diffusion. In *International Conference on Learning Representations*, 2022.

Richard M Dudley. *Real analysis and probability*. CRC Press, 2018.

Jonathan Dumas, Antoine Wehenkel, Damien Lanaspeze, Bertrand Cornélusse, and Antonio Sutera. A deep generative model for probabilistic energy forecasting in power systems: normalizing flows. *Applied Energy*, 305:117871, 2022.

Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented neural odes. In *Neural Information Processing Systems*, 2019.

Conor Durkan and Yang Song. On maximum likelihood training of score-based generative models. *arXiv preprint arXiv:2101.09258*, 2021.

Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Cubic-spline flows. 2019a.

Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. *Advances in Neural Information Processing Systems*, 32:7511–7522, 2019b.

Rick Durrett. *Probability: theory and examples*, volume 49. Cambridge university press, 2019.

James F Epperson. *An introduction to numerical methods and analysis*. John Wiley & Sons, 2013.

Chris Finlay, Augusto Gerolin, Adam M Oberman, and Aram-Alexandre Pooladian. Learning normalizing flows from entropy-kantorovich potentials. In *ICML 2020 Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*, 2020a.

Chris Finlay, Jörn-Henrik Jacobsen, Levon Nurbekyan, and Adam M Oberman. How to train your neural ode: the world of jacobian and kinetic regularization. In *International Conference on Machine Learning*, 2020b.

Gerald B Folland. *Real analysis: modern techniques and their applications*, volume 40. John Wiley & Sons, 1999.

Marylou Gabrié, Grant M Rotskoff, and Eric Vanden-Eijnden. Efficient bayesian sampling using normalizing flows to assist markov chain monte carlo methods. In *ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*, 2021.

Adam Gabryś, Yunlong Jiao, Viacheslav Klimkov, Daniel Korzekwa, and Roberto Barra-Chicote. Improving the expressiveness of neural vocoding with non-affine normalizing flows. *arXiv preprint arXiv:2106.08649*, 2021.

Yarin Gal. Uncertainty in deep learning. 2016.

Victor Garcia Satorras, Emiel Hoogeboom, Fabian Fuchs, Ingmar Posner, and Max Welling. E (n) equivariant normalizing flows. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Crispin Gardiner. *Stochastic methods*, volume 4. Springer Berlin, 2009.

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

Samuel Gershman and Noah Goodman. Amortized inference in probabilistic reasoning. In *Proceedings of the annual meeting of the cognitive science society*, volume 36, 2014.

Paul Glasserman and Yu-Chi Ho. *Gradient estimation via perturbation analysis*, volume 116. Springer Science & Business Media, 1991.

Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

Ian Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. In *International conference on machine learning*, pages 1319–1327. PMLR, 2013.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Will Grathwohl, Ricky TQ Chen, Jesse Betterncourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. In *International Conference on Learning Representations*, 2019.

Matej Grcić, Ivan Grubišić, and Siniša Šegvić. Densely connected normalizing flows. *Advances in Neural Information Processing Systems*, 34, 2021.

Ishaan Gulrajani, Kundan Kumar, Faruk Ahmed, Adrien Ali Taiga, Francesco Visin, David Vazquez, and Aaron Courville. Pixelvae: A latent variable model for natural images. In *International Conference on Learning Representations*, 2017.

David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *International Conference on Learning Representations*, 2017.

Insu Han, Haim Avron, and Jinwoo Shin. Stochastic chebyshev gradient descent for spectral optimization. In *Advances in Neural Information Processing Systems*, pages 7386–7396, 2018a.

Jiequn Han, Arnulf Jentzen, and E Weinan. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018b.

Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. *arXiv preprint arXiv:1902.00275*, 2019a.

Jonathan Ho, Evan Lohn, and Pieter Abbeel. Compression with flows via local bits-back coding. In *Advances in Neural Information Processing Systems*, pages 3874–3883, 2019b.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in neural information processing systems*, 2020.

Yung-Han Ho, Chih-Chun Chan, Wen-Hsiao Peng, Hsueh-Ming Hang, and Marek Domański. Anfic: Image compression using augmented normalizing flows. *IEEE Open Journal of Circuits and Systems*, 2:613–626, 2021.

Emiel Hoogeboom, Jorn Peters, Rianne van den Berg, and Max Welling. Integer discrete flows and lossless compression. In *Advances in Neural Information Processing Systems*, pages 12134–12144, 2019.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Towards non-autoregressive language models. In *Advances in Neural Information Processing Systems*, 2021.

Chin-Wei Huang, David Krueger, and Aaron Courville. Facilitating multimodality in normalizing flows. In *NIPS 2017 Workshop on Bayesian Deep Learning*, 2017a.

Chin-Wei Huang, Ahmed Touati, Laurent Dinh, Michal Drozdzal, Mohammad Havaei, Laurent Charlin, and Aaron Courville. Learnable explicit density for continuous latent space and variational inference. In *ICML 2017 Workshop on Principled Approaches to Deep Learning*, 2017b.

Chin-Wei Huang, David Krueger, Alexandre Lacoste, and Aaron Courville. Neural autoregressive flows. In *International Conference on Machine Learning*, 2018a.

Chin-Wei Huang, Shawn Tan, Alexandre Lacoste, and Aaron C Courville. Improving explorability in variational inference with annealed variational objectives. In *Advances in Neural Information Processing Systems*, pages 9701–9711, 2018b.

Chin-Wei Huang, Kris Sankaran, Eeshan Dhekane, Alexandre Lacoste, and Aaron Courville. Hierarchical importance weighted autoencoders. In *International Conference on Machine Learning*, 2019.

Chin-Wei Huang, Faruk Ahmed, Kundan Kumar, Alexandre Lacoste, and Aaron Courville. Probability distillation: A caveat and alternatives. In *Uncertainty in Artificial Intelligence*, pages 1212–1221. PMLR, 2020a.

Chin-Wei Huang, Laurent Dinh, and Aaron Courville. Augmented normalizing flows: Bridging the gap between generative flows and latent variable models. *arXiv preprint arXiv:2002.07101*, 2020b.

Chin-Wei Huang, Laurent Dinh, and Aaron Courville. Solving ode with universal flows: Approximation theory for flow-based models. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020c.

Chin-Wei Huang, Ahmed Touati, Pascal Vincent, Gintare Karolina Dziugaite, Alexandre Lacoste, and Aaron Courville. Stochastic neural network with kronecker flow. In *International Conference on Artificial Intelligence and Statistics*, pages 4184–4194. PMLR, 2020d.

Chin-Wei Huang, Ricky TQ Chen, Christos Tsirigotis, and Aaron Courville. Convex potential flows: Universal probability distributions with optimal transport and convex optimization. In *International Conference on Learning Representations*, 2021a.

Chin-Wei Huang, Jae Hyun Lim, and Aaron Courville. A variational perspective on diffusion-based generative models and score matching. In *Advances in neural information processing systems*, 2021b.

Chin-Wei Huang, Milad Aghajohari, Avishek Joey Bose, Prakash Panangaden, and Aaron Courville. Riemannian diffusion models. In *Advances in Neural Information Processing Systems*, 2022.

David H Hubel and Torsten N Wiesel. Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology*, 148(3):574–591, 1959.

Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.

Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

Aapo Hyvärinen and Petteri Pajunen. Nonlinear independent component analysis: Existence and uniqueness results. *Neural Networks*, 12(3), 1999.

Nicholas J Irons, Meyer Scetbon, Soumik Pal, and Zaid Harchaoui. Triangular flows for generative modeling: Statistical consistency, smoothness classes, and fast rates. In *International Conference on Artificial Intelligence and Statistics*, pages 10161–10195. PMLR, 2022.

Priyank Jaini, Kira A Selby, and Yaoliang Yu. Sum-of-squares polynomial flow. In *International Conference on Machine Learning*, pages 3009–3018. PMLR, 2019.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning*, pages 2323–2332. PMLR, 2018.

Michael I. Jordan. Artificial intelligence—the revolution hasn't happened yet. *Harvard Data Science Review*, 1(1), 7 2019. doi: 10.1162/99608f92.f06c6e61. URL https://hdsr.mitpress.mit.edu/pub/wot7mkc1. https://hdsr.mitpress.mit.edu/pub/wot7mkc1.

Ioannis Karatzas and Steven Shreve. *Brownian motion and stochastic calculus*, volume 113. springer, 2014.

Bahjat Kawar, Gregory Vaksman, and Michael Elad. Snips: Solving noisy inverse problems stochastically. In *Advances in neural information processing systems*, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Diederik P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pages 10215–10224, 2018.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

Diederik 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.

Diederik P Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. In *Advances in Neural Information Processing Systems*, 2021.

Friso H Kingma, Pieter Abbeel, and Jonathan Ho. Bit-swap: Recursive bits-back coding for lossless compression with hierarchical latent variables. *arXiv preprint arXiv:1905.06845*, 2019.

Leon Klein, Andrew YK Foong, Tor Erlend Fjelde, Bruno Mlodozeniec, Marc Brockschmidt, Sebastian Nowozin, Frank Noé, and Ryota Tomioka. Timewarp: Transferable acceleration of molecular dynamics by learning time-coarsened dynamics. In *International Conference on Machine Learning*, 2023.

Herbert Knothe. Contributions to the theory of convex bodies. *Michigan Mathematical Journal*, 4(1): 39–52, 1957.

Frederic Koehler, Viraj Mehta, and Andrej Risteski. Representational aspects of depth and conditioning in normalizing flows. In *International Conference on Machine Learning*, pages 5628–5636. PMLR, 2021.

Jonas Köhler, Leon Klein, and Frank Noé. Equivariant flows: exact likelihood generative learning for symmetric densities. In *International Conference on Machine Learning*, pages 5361–5370. PMLR, 2020.

Jonas Köhler, Yaoyi Chen, Andreas Krämer, Cecilia Clementi, and Frank Noé. Force-matching coarse-graining without forces. *arXiv preprint arXiv:2203.11167*, 2022.

Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. In *International Conference on Learning Representations*, 2021.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

David Krueger, Chin-Wei Huang, Riashat Islam, Ryan Turner, Alexandre Lacoste, and Aaron Courville. Bayesian hypernetworks. In *NIPS 2017 Workshop on Bayesian Deep Learning*.

Abhishek Kumar, Ben Poole, and Kevin Murphy. Regularized autoencoders via relaxed injective probability flow. In *International Conference on Artificial Intelligence and Statistics*, pages 4292–4301. PMLR, 2020.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Holden Lee, Chirag Pabbaraju, Anish Sevekari, and Andrej Risteski. Universal approximation for log-concave distributions using well-conditioned normalizing flows, 2021.

David A Levin and Yuval Peres. *Markov chains and mixing times*, volume 107. American Mathematical Soc., 2017.

Xuechen Li, Ting-Kam Leonard Wong, Ricky TQ Chen, and David Duvenaud. Scalable gradients for stochastic differential equations. In *International Conference on Artificial Intelligence and Statistics*, pages 3870–3882. PMLR, 2020.

M. Lichman. UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml.

Jae Hyun Lim, Aaron Courville, Christopher Pal, and Chin-Wei Huang. Ar-dae: Towards unbiased neural entropy gradient estimation. In *International Conference on Machine Learning*, pages 6061–6071. PMLR, 2020a.

Jae Hyun Lim, Chin-Wei Huang, Aaron Courville, and Christopher Pal. Bijective-contrastive estimation. In *Third Symposium on Advances in Approximate Bayesian Inference*, 2020b.

Lin Lin, Yousef Saad, and Chao Yang. Approximating spectral densities of large matrices. *SIAM review*, 58(1):34–65, 2016.

T Lindvall. Lectures on the coupling method, wiley series in probability and mathematical statistics. *Probab. Math. Statist.*, 1992.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Jonathan Lorraine and Safwan Hossain. Jacnet: Learning functions with structured jacobians. In *ICML 2019 Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*, 2019.

Andreas Lugmayr, Martin Danelljan, Luc Van Gool, and Radu Timofte. Srflow: Learning the super-resolution space with normalizing flow. In *European Conference on Computer Vision*, pages 715–732. Springer, 2020.

Siwei Lyu. Interpretation and generalization of score matching. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 359–366, 2009.

Jin Ma, J-M Morel, and Jiongmin Yong. *Forward-backward stochastic differential equations and their applications*. Number 1702. Springer Science & Business Media, 1999.

Xuezhe Ma, Xiang Kong, Shanghang Zhang, and Eduard Hovy. Decoupling global and local representations via invertible generative flows. In *International Conference on Learning Representations*, 2021.

Yi-An Ma, Niladri Chatterji, Xiang Cheng, Nicolas Flammarion, Peter Bartlett, and Michael I Jordan. Is there an analog of nesterov acceleration for mcmc? *arXiv preprint arXiv:1902.00996*, 2019.

Lars Maaløe, Marco Fraccaro, Valentin Liévin, and Ole Winther. Biva: A very deep hierarchy of latent variables for generative modeling. In *Advances in Neural Information Processing Systems*, 2019.

David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 2. IEEE, 2001.

Bogdan Mazoure, Thang Doan, Audrey Durand, Joelle Pineau, and R Devon Hjelm. Leveraging exploration in off-policy algorithms via normalizing flows. In *Conference on Robot Learning*, pages 430–444. PMLR, 2020.

Robert J McCann. Polar factorization of maps on riemannian manifolds. *Geometric & Functional Analysis GAFA*, 11(3):589–608, 2001.

Chenlin Meng, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Image synthesis and editing with stochastic differential equations. *arXiv preprint arXiv:2108.01073*, 2021.

Laurence I Midgley, Vincent Stimper, Javier Antorán, Emile Mathieu, Bernhard Schölkopf, and José Miguel Hernández-Lobato. Se (3) equivariant augmented coupling flows. In *Advances in Neural Information Processing Systems*, 2023.

GN Milshtein. Approximate integration of stochastic differential equations. *Theory of Probability & Its Applications*, 19(3):557–562, 1975.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

Shakir Mohamed and Balaji Lakshminarayanan. Learning in implicit generative models. *arXiv preprint arXiv:1610.03483*, 2016.

Shakir Mohamed, Mihaela Rosca, Michael Figurnov, and Andriy Mnih. Monte carlo gradient estimation in machine learning. *J. Mach. Learn. Res.*, 21(132):1–62, 2020.

Rogan Morrow and Wei-Chen Chiu. Variational autoencoders with normalizing flow decoders. *arXiv preprint arXiv:2004.05617*, 2020.

Alfred Müller. Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, 29(2):429–443, 1997.

Thomas Müller, Brian McWilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. Neural importance sampling. *ACM Transactions on Graphics (TOG)*, 38(5):1–19, 2019.

Brady Neal, Chin-Wei Huang, and Sunand Raghupathi. Realcause: Realistic causal inference benchmarking. *arXiv preprint arXiv:2011.15007*, 2020.

Yurii Nesterov. Introductory lectures on convex programming volume i: Basic course. *Lecture notes*, 3 (4):5, 1998.

Didrik Nielsen, Priyank Jaini, Emiel Hoogeboom, Ole Winther, and Max Welling. Survae flows: Surjections to bridge the gap between vaes and flows. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

Bernt Øksendal. Stochastic differential equations. In *Stochastic differential equations*, pages 65–84. Springer, 2003.

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 3898–3907. PMLR, 2018.

Derek Onken, Samy Wu Fung, Xingjian Li, and Lars Ruthotto. Ot-flow: Fast and accurate continuous normalizing flows via optimal transport. *arXiv preprint arXiv:2006.00104*, 2020.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016a.

Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International Conference on Machine Learning*, 2016b.

Georg Ostrovski, Marc G Bellemare, Aäron van den Oord, and Rémi Munos. Count-based exploration with neural density models. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2721–2730. JMLR. org, 2017.

Art B. Owen. *Monte Carlo theory, methods and examples*. 2013.

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. *Journal of Machine Learning Research*, 22(57):1–64, 2021.

Grigorios A Pavliotis. *Stochastic processes and applications: diffusion processes, the Fokker-Planck and Langevin equations*, volume 60. Springer, 2014.

Philip E. Protter. *Stochastic Integration and Differential Equations*, volume 21 of *Stochastic Modelling and Applied Probability*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. ISBN 9783642055607 9783662100615. doi: 10.1007/978-3-662-10061-5. URL http://link.springer.com/10.1007/978-3-662-10061-5.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

Kashif Rasul, Ingmar Schuster, Roland Vollgraf, and Urs Bergmann. Set flow: A permutation invariant normalizing flow. *arXiv preprint arXiv:1909.02775*, 2019.

Danilo J Rezende and Sébastien Racanière. Implicit riemannian concave potential maps. *arXiv preprint arXiv:2110.01288*, 2021.

Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, 2014.

Danilo Jimenez Rezende, Sébastien Racanière, Irina Higgins, and Peter Toth. Equivariant hamiltonian flows. *arXiv preprint arXiv:1909.13739*, 2019.

Jack Richter-Powell, Jonathan Lorraine, and Brandon Amos. Input convex gradient networks. *arXiv preprint arXiv:2111.12187*, 2021.

Enrico Rinaldi, Xizhi Han, Mohammad Hassan, Yuan Feng, Franco Nori, Michael McGuigan, and Masanori Hanada. Matrix model simulations using quantum computing, deep learning, and lattice monte carlo. *arXiv preprint arXiv:2108.02942*, 2021.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

R Tyrrell Rockafellar. *Convex analysis*. Number 28. Princeton university press, 1970.

Murray Rosenblatt. Remarks on a multivariate transformation. *The annals of mathematical statistics*, 23 (3):470–472, 1952.

Noam Rozen, Aditya Grover, Maximilian Nickel, and Yaron Lipman. Moser flow: Divergence-based generative modeling on manifolds. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Stuart Russell and Peter Norvig. Artificial intelligence: a modern approach. 2002.

Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement. *arXiv preprint arXiv:2104.07636*, 2021.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, et al. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*, 2022.

Filippo Santambrogio. Optimal transport for applied mathematicians. *Birkäuser, NY*, 55(58-63):94, 2015.

Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E (n) equivariant graph neural networks. In *International Conference on Machine Learning*, 2021.

Christoph Schöller and Alois Knoll. Flomo: Tractable motion prediction with normalizing flows. *arXiv preprint arXiv:2103.03614*, 2021.

Matthias Seeger, Asmus Hetzel, Zhenwen Dai, Eric Meissner, and Neil D Lawrence. Auto-differentiating linear algebra. *arXiv preprint arXiv:1710.08717*, 2017.

Vivien Seguy, Bharath Bhushan Damodaran, Remi Flamary, Nicolas Courty, Antoine Rolet, and Mathieu Blondel. Large scale optimal transport and mapping estimation. In *International Conference on Learning Representations*, 2018.

Oleksandr Shchur, Marin Biloš, and Stephan Günnemann. Intensity-free learning of temporal point processes. *arXiv preprint arXiv:1909.12127*, 2019.

Yikang Shen, Zhouhan Lin, Chin-wei Huang, and Aaron Courville. Neural language modeling by jointly learning syntax and lexicon. In *International Conference on Learning Representations*, 2018.

Ralph E Showalter. *Hilbert space methods in partial differential equations*. Courier Corporation, 2010.

Joseph Sill. Monotonic networks. *Advances in Neural Information Processing Systems*, 10:661–667, 1997.

RN Silver and H Röder. Densities of states of mega-dimensional hamiltonian matrices. *International Journal of Modern Physics C*, 5(04):735–753, 1994.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in neural information processing systems*, 2019.

Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. *arXiv preprint arXiv:2006.09011*, 2020.

Yang Song, Chenlin Meng, and Stefano Ermon. Mintnet: Building invertible neural networks with masked convolutions. *Advances in Neural Information Processing Systems*, 32:11004–11014, 2019.

Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced score matching: A scalable approach to density and score estimation. In *Uncertainty in Artificial Intelligence*, pages 574–584. PMLR, 2020.

Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum likelihood training of score-based diffusion models. In *Advances in Neural Information Processing Systems*, pages arXiv–2101, 2021a.

Yang Song, Chenlin Meng, Renjie Liao, and Stefano Ermon. Accelerating feedforward computation via parallel nonlinear equation solving. In *International Conference on Machine Learning*, pages 9791–9800. PMLR, 2021b.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021c.

Fan-Yun Sun, Meng Qu, Jordan Hoffmann, Chin-Wei Huang, and Jian Tang. vgraph: a generative model for joint community detection and node representation learning. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 514–524, 2019.

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.

Amirhossein Taghvaei and Prashant Mehta. Accelerated flow for probability distributions. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6076–6085, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

Hillel Tal-Ezer and Ronnie Kosloff. An accurate and efficient scheme for propagating the time dependent schrödinger equation. *The Journal of chemical physics*, 81(9):3967–3971, 1984.

Shawn Tan, Chin-Wei Huang, Alessandro Sordoni, and Aaron Courville. Learning to dequantise with truncated flows. In *International Conference on Learning Representations*, 2022.

Takeshi Teshima, Isao Ishikawa, Koichi Tojo, Kenta Oono, Masahiro Ikeda, and Masashi Sugiyama. Coupling-based invertible neural networks are universal diffeomorphism approximators. In *Advances in Neural Information Processing Systems*, 2020a.

Takeshi Teshima, Koichi Tojo, Masahiro Ikeda, Isao Ishikawa, and Kenta Oono. Universal approximation property of neural ordinary differential equations. *arXiv preprint arXiv:2012.02414*, 2020b.

Tijmen Tieleman, Geoffrey Hinton, et al. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

Maximilian Topel and Andrew L Ferguson. Reconstruction of protein structures from single-molecule time series. *The Journal of Chemical Physics*, 153(19):194102, 2020.

Belinda Tzen and Maxim Raginsky. Neural stochastic differential equations: Deep latent gaussian models in the diffusion limit. *arXiv preprint arXiv:1905.09883*, 2019.

Shashanka Ubaru, Jie Chen, and Yousef Saad. Fast estimation of tr(f(A)) via Stochastic Lanczos Quadrature. *SIAM Journal on Matrix Analysis and Applications*, 38(4):1075–1099, 2017.

John Urbas. On the second boundary value problem for equations of monge-ampère type. 1997.

Benigno Uria, Iain Murray, and Hugo Larochelle. Rnade: The real-valued neural autoregressive density-estimator. In *Advances in Neural Information Processing Systems*, 2013.

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. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, pages 393–402. Association For Uncertainty in Artificial Intelligence (AUAI), 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

Cédric Villani. *Topics in optimal transportation*, volume 58. American Mathematical Soc., 2003.

Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

Changfeng Wang, Santosh Venkatesh, and J Judd. Optimal stopping and effective machine complexity in learning. *Advances in neural information processing systems*, 6, 1993.

He Wang, Zhoujian Cao, Yue Zhou, Zong-Kuan Guo, and Zhixiang Ren. Sampling with prior knowledge for high-dimensional gravitational wave data analysis. *Big Data Mining and Analytics*, 5(1):53–63, 2021.

Yifei Wang and Wuchen Li. Accelerated information gradient flow, 2019.

Patrick Nadeem Ward, Ariella Smofsky, and Avishek Joey Bose. Improving exploration in soft-actor-critic with normalizing flows policies. *arXiv preprint arXiv:1906.02771*, 2019.

Larry Wasserman. *All of nonparametric statistics*. Springer Science & Business Media, 2006.

Antoine Wehenkel and Gilles Louppe. Unconstrained monotonic neural networks. *Advances in Neural Information Processing Systems*, 32:1545–1555, 2019.

Jay Whang, Erik M Lindgren, and Alexandros G Dimakis. Approximate probabilistic inference with composed flows. *arXiv preprint arXiv:2002.11743*, 2020.

Andre Wibisono, Varun Jog, and Po-Ling Loh. Information and estimation in fokker-planck channels. In *2017 IEEE International Symposium on Information Theory (ISIT)*, pages 2673–2677. IEEE, 2017.

Auke Wiggers and Emiel Hoogeboom. Predictive sampling with forecasting autoregressive models. In *International Conference on Machine Learning*, pages 10260–10269. PMLR, 2020.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

Tian Xie, Xiang Fu, Octavian-Eugen Ganea, Regina Barzilay, and Tommi Jaakkola. Crystal diffusion variational autoencoder for periodic material generation. *arXiv preprint arXiv:2110.06197*, 2021.

Winnie Xu, Ricky TQ Chen, Xuechen Li, and David Duvenaud. Infinitely deep bayesian neural networks with stochastic differential equations. In *International Conference on Artificial Intelligence and Statistics*, 2022.

Liu Yang and George Em Karniadakis. Potential flow generator with l_2 optimal transport regularity for generative models. *arXiv preprint arXiv:1908.11462*, 2019.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in neural information processing systems*, pages 3391–3401, 2017.

Jiaxin Zhang, Kyle Saleeby, Thomas Feldhausen, Sirui Bi, Alex Plotkowski, and David Womble. Self-supervised anomaly detection via neural autoregressive flows with active learning. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

Linfeng Zhang, Lei Wang, et al. Monge-ampère flow for generative modeling. *arXiv preprint arXiv:1809.10188*, 2018.

Linqi Zhou, Yilun Du, and Jiajun Wu. 3d shape generation and completion through point-voxel diffusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021.

Zachary Ziegler and Alexander Rush. Latent normalizing flows for discrete sequences. In *International Conference on Machine Learning*, pages 7673–7682. PMLR, 2019.

A Appendix of Chapter 1

A.1 Score matching loss identity

In this section, we prove the score matching loss identity (1.4) for completeness. These proofs are adapted from Hyvärinen and Dayan [2005], Song et al. [2020], Vincent [2011] with slight modifications since we project the score onto the eigen-basis of Λ.

Expanding the ESM loss, we have

$$\mathcal{L}_{\text{ESM}} = \mathbb{E}_x \left[\frac{1}{2} ||\boldsymbol{s}_\theta(x)||_\Lambda^2 - \boldsymbol{s}_\theta(x)^\top \Lambda \nabla \log q(x) + \frac{1}{2} ||\nabla \log q(x)||_\Lambda^2 \right]. \tag{A.1}$$

Moving $\mathcal{I}(q(x))$ from the RHS to the LHS gives us

$$\mathcal{L}_{\text{ESM}} - \frac{1}{2}\mathcal{I}(q(x)) = \mathbb{E}_x \left[\frac{1}{2} ||\boldsymbol{s}_\theta(x)||_\Lambda^2 - \boldsymbol{s}_\theta(x)^\top \Lambda \nabla \log q(x) \right]. \tag{A.2}$$

Implicit score matching: Now to derive the ISM loss, we apply integration by parts and the general Stokes' theorem (with mild regularity condition on $\boldsymbol{s}_\theta$) to the inner product term to obtain

$$\begin{aligned}
\int q(x) \boldsymbol{s}_\theta(x)^\top \Lambda \nabla \log q(x) \, \mathrm{d}x &= \int \boldsymbol{s}_\theta(x)^\top \Lambda \nabla q(x) \, \mathrm{d}x \\
&= \cancelto{0}{\int \nabla \cdot \left(q \Lambda^\top \boldsymbol{s}_\theta \right) \mathrm{d}x} - \int \nabla \cdot \left(\Lambda^\top \boldsymbol{s}_\theta \right) q \, \mathrm{d}x \\
&= -\mathbb{E}_x \left[\nabla \cdot \left(\Lambda^\top \boldsymbol{s}_\theta \right) \right].
\end{aligned}$$

Sliced score matching: For the SSM loss, use the Hutchinson trace estimator [Hutchinson, 1989] to replace the divergence operator, which is simply the trace of the Jacobian matrix. See Section 7.4 for more discussion on computation time and estimation error.

Denoising score matching: For DSM, similarly we first look at the inner product term

$$\begin{aligned}
\int q(x)\boldsymbol{s}_\theta(x)^\top \Lambda \nabla \log q(x)\,\mathrm{d}x &= \int \boldsymbol{s}_\theta(x)^\top \Lambda \nabla q(x)\,\mathrm{d}x \\
&= \int \boldsymbol{s}_\theta(x)^\top \Lambda \nabla \int q(x \mid x_0) q(x_0)\,\mathrm{d}x_0\,\mathrm{d}x \\
&= \int\int q(x_0)\boldsymbol{s}_\theta^\top \Lambda \nabla q(x \mid x_0)\,\mathrm{d}x\,\mathrm{d}x_0 \\
&= \int\int q(x_0) q(x \mid x_0)\boldsymbol{s}_\theta^\top \Lambda \nabla \log q(x \mid x_0)\,\mathrm{d}x\,\mathrm{d}x_0 \\
&= \mathbb{E}_{x_0,x}\left[\boldsymbol{s}_\theta^\top \Lambda \nabla \log q(x \mid x_0)\right],
\end{aligned}$$

where $q(x \mid x_0)$ denotes the conditional density of x given x_0. Combining this with $\mathbb{E}_x[||\boldsymbol{s}_\theta||_\Lambda^2]$, we have

$$\begin{aligned}
&\mathbb{E}_{x_0,x}\left[\frac{1}{2}||\boldsymbol{s}_\theta||_\Lambda^2 - \boldsymbol{s}_\theta^\top \Lambda \nabla \log q(x \mid x_0)\right] = \\
&\mathbb{E}_{x_0,x}\left[\frac{1}{2}||\boldsymbol{s}_\theta - \nabla \log q(x \mid x_0)||_\Lambda^2\right] - \frac{1}{2}\mathbb{E}_{x_0}[\mathcal{I}(q(x \mid x_0))].
\end{aligned}$$

In fact, the identity between the DSM loss and the other score matching losses is more fine-grained. Take ISM for example. From the above derivation, the equality

$$\int q(x_0)\,\mathbb{E}_{q(x|x_0)}\left[\nabla \cdot (\Lambda^\top \boldsymbol{s}_\theta) + \boldsymbol{s}_\theta^\top \Lambda \nabla \log q(x \mid x_0)\right]\,\mathrm{d}x_0 = 0 \tag{A.3}$$

holds for any marginal distribution $q(x_0)$. This implies

$$\mathbb{E}_{q(x|x_0)}\left[\nabla \cdot (\Lambda^\top \boldsymbol{s}_\theta) + \boldsymbol{s}_\theta^\top \Lambda \nabla \log q(x \mid x_0)\right] = 0. \tag{A.4}$$

This will come in handy in Chapter 8.

B Appendix of Chapter 2

B.1 Useful lemmas

Asymptotic indistinguishability The following lemma is a slight generalization of the previous one, as it allows us to look at random variables that may not exhibit any pointwise convergence behavior, as long as they are asymptotically indistinguishable from another sequence of random variables that converges in distribution. An example of random variables that do not converge pointwise but in distribution is $x_n = (-1)^n x$ for $x \in \mathcal{N}(0,1)$. x_n does not converge pointwise, but $x_n \overset{d}{=} x$. This lemma will come in handy when we want to strengthen the universality result.

Lemma 32. *Let x_∞, $(x_n : n \geq 0)$ and $(y_n : n \geq 0)$ be random variables. If $x_n \to x_\infty$ in distribution and if $||x_n - y_n|| \to 0$ almost surely as $n \to \infty$, then $y_n \to x_\infty$ in distribution.*

Proof. Let $\Lambda : \mathbb{R}^d \to \mathbb{R}$ be an arbitrary *bounded* and *Lipschitz continuous* function. Then

$$\begin{aligned} |\mathbb{E}\left[\Lambda\left(x_\infty\right) - \Lambda\left(y_n\right)\right]| &\leq |\mathbb{E}\left[\Lambda\left(x_\infty\right) - \Lambda(x_n) + \Lambda(x_n) - \Lambda\left(y_n\right)\right]| \\ &\leq |\mathbb{E}\left[\Lambda\left(x_\infty\right) - \Lambda(x_n)\right]| + \mathbb{E}\left[|\Lambda\left(x_n\right) - \Lambda\left(y_n\right)|\right]. \end{aligned}$$

First, since $x_n \to x_\infty$ in distribution and since Λ is bounded and continuous, by the Portmanteau Lemma the first term of the RHS converges to 0 as $n \to \infty$. Second, since y_n is almost surely asymptotically indistinguishable from x_n (let Ω be the almost sure set), and since the Lipschitzness of Λ implies uniform continuity, the following are true

- For all $\epsilon > 0$, there exists a $\delta > 0$ such that $||x - y|| \leq \delta$ implies $|\Lambda(x) - \Lambda(y)| \leq \epsilon$.
- For any $\delta > 0$, there exists a integer $N > 0$ such that for all $n \geq N$, $||x_n - y_n|| \leq \delta$ for all $\omega \in \Omega$.

These imply $||\Lambda(x_n) - \Lambda(y_n)|| \to 0$ on Ω. Then

$$\mathbb{E}\left[|\Lambda\left(x_n\right) - \Lambda\left(y_n\right)|\right] = \underbrace{\mathbb{E}_{\Omega}\left[|\Lambda\left(x_n\right) - \Lambda\left(y_n\right)|\right]}_{E_1} + \underbrace{\mathbb{E}_{\Omega^c}\left[|\Lambda\left(x_n\right) - \Lambda\left(y_n\right)|\right]}_{E_2}$$

converges to 0, since (1) boundedness of Λ and the *Bounded Convergence Theorem* imply $E_1 \to 0$ and (2) $\sup_x \Lambda(x) < \infty$ implies $E_2 \leq 2 \sup_x \Lambda(x) \mathbb{P}(\Omega^c) = 0$. Finally, since Λ is arbitrary, by the Portmanteau Lemma again, y_n converges in distribution to x_∞ as $n \to \infty$. □

Total variation distance We review the total variation of signed measures, which induces a natural distance metric on the space of probability measures. It will also make it easier to establish weak convergence of measures.

Definition 33 (Total variation distance). *Let μ and ν be probability measures defined on a measurable space $(\Omega, \mathcal{F})$. The total variation distance (metric) between μ and ν is defined as*

$$d_{TV}(\mu, \nu) = \sup_{A \in \mathcal{F}} |\mu(A) - \nu(A)|. \tag{B.1}$$

Total variation distance is a metric on the space of probability measures. This is easy to check: First, it is obviously symmetric. Second, if $\mu = \nu$ then $d_{TV}(\mu, \nu) = 0$ and if $d_{TV}(\mu, \nu) = 0$, $|\mu(A) - \nu(A)| = 0$ for all $A \in \mathcal{F}$, which means $\mu = \nu$. Finally, for

any measures μ, ν and ρ,

$$\begin{aligned} d_{TV}(\mu,\nu) &= \sup_{A\in\mathcal{F}} |\mu(A)-\nu(A)| \\ &\leq \sup_{A\in\mathcal{F}} |\mu(A)-\rho(A)| + |\rho(A)-\nu(A)| \\ &\leq \sup_{A\in\mathcal{F}} |\mu(A)-\rho(A)| + \sup_{B\in\mathcal{F}} |\rho(B)-\nu(B)| \\ &= d_{TV}(\mu,\rho) + d_{TV}(\rho,\nu). \end{aligned}$$

In addition, d_{TV} is bounded by 1.

Definition 34 (Equivalent definitions). *Let $\mathcal{C}(\Omega)$ be the collection of all partitions of Ω into countable disjoint subsets; i.e. if $C \in \mathcal{C}$, then $\cup_{A\in C} A = \Omega$, and $A, B \in C$ implies $A \cap B = \emptyset$. Let $m\mathcal{F}([0,1])$ denote the collection of all $\mathcal{F}$-measurable $f : \Omega \to [0,1]$. And let $\Gamma(\mu,\nu)$ be the set of measures on $\Omega \times \Omega$ with marginal μ and ν. Then we have the following equivalent definitions of the total variation distance:*

$$\text{(a)} \qquad d_{TV}(\mu,\nu) = \frac{1}{2} \sup_{C\in\mathcal{C}(\Omega)} \sum_{A\in C} |\mu(A)-\nu(A)| \qquad \text{(b)}$$

$$= \sup_{f\in m\mathcal{F}([0,1])} |\mathbb{E}_\mu[f] - \mathbb{E}_\nu[f]| \qquad \text{(c)}$$

$$= \inf_{\pi\in\Gamma(\mu,\nu)} \mathbb{E}_{X,Y\sim\pi}[1_{X\neq Y}]. \qquad \text{(d)}$$

N.B.*: the variational form* (c) *can be replaced with any bounded function by normalizing it to be $f/(\sup f - \inf f)$.*

The proof of (d) is inspired by Levin and Peres [2017], where they have a countable state space which forms a natural partition that maximizes the summation of (b). This also allows us to write the total variation as a sum if Ω is countable

$$d_{TV}(\mu,\nu) = \frac{1}{2} \sum_{\omega\in\Omega} |\mu(\omega)-\nu(\omega)|. \tag{B.2}$$

Similarly, we can write the total variation as an explicit integral if μ and ν are absolutely

continuous wrt some reference λ

$$d_{TV}(\mu,\nu) = \frac{1}{2}\int_{\Omega}\left|\frac{d\mu}{d\lambda} - \frac{d\nu}{d\lambda}\right| \mathrm{d}\lambda. \tag{B.3}$$

This should be reminiscent of the f-divergence (1.2) formulation of the total variation distance. To see that, let q and p be the Lebesgue densities of μ and ν over $\mathbb{R}^d$. Suppose $p, q > 0$. Then

$$d_{TV}(\mu,\nu) = \frac{1}{2}\int_{\mathbb{R}^d}|q(x)-p(x)|\,\mathrm{d}x = \frac{1}{2}\int_{\mathbb{R}^d}\left|\frac{q(x)}{p(x)} - 1\right| p(x)\,\mathrm{d}x = D_f(q||p). \tag{B.4}$$

Proof. (a = b): Let P and N be the Hahn decomposition [Folland, 1999, Theorem 3.3] of the signed measure $\mu - \nu$ satisfying $P \cup N = \Omega$, $P \cap N = \emptyset$, $\mu(A) - \nu(A) \geq 0$ for any measurable subset $A \subseteq P$, and $\mu(A) - \nu(A) \leq 0$ for any measurable subset $A \subseteq N$.

Let C be a partition of Ω, and let $C_{\cap P} = \{A \cap P : A \in C\}$. Then by triangle inequality and countable additivity of measure, we have

$$\begin{aligned}\sum_{A\in C}|\mu(A)-\nu(A)| &\leq \sum_{A\in C_{\cap P}}|\mu(A)-\nu(A)| + \sum_{A\in C_{\cap N}}|\mu(A)-\nu(A)| \\ &= \mu(P)-\nu(P)+\nu(N)-\mu(N).\end{aligned}$$

This shows P and N are the partition maximizing the summation in (b).

Now for any $B \in \mathcal{F}$,

$$\begin{aligned}\mu(B)-\nu(B) &= \mu(B\cap P)-\nu(B\cap P)+\mu(B\cap N)-\nu(B\cap N) \\ &\leq \mu(B\cap P)-\nu(B\cap P) \\ &= \mu(P)-\nu(P)-(\mu(B^c\cap P)-\nu(B^c\cap P)) \\ &\leq \mu(P)-\nu(P).\end{aligned}$$

Likewise, $\nu(B) - \mu(B) \leq \nu(N) - \mu(N)$. Since

$$\mu(P) - \nu(P) - (\nu(N) - \mu(N)) = \mu(P) + \mu(N) - (\nu(P) + \nu(N)) = 1 - 1 = 0,$$

we can write $|\mu(B) - \nu(B)| \leq \mu(P) - \nu(P)$, which implies a $\leq$ b. And since $P \in \mathcal{F}$, the equality is attained.

(a $\leq$ c): Since the characteristic functions of the type 1_A for $A \in m\mathcal{F}$ are measurable and bounded in $[0, 1]$,

$$d_{TV}(\mu, \nu) = \sup_{A \in \mathcal{F}} |\mu(A) - \nu(A)| = \sup_{A \in \mathcal{F}} |\mathbb{E}_\mu[1_A] - \mathbb{E}_\nu[1_A]| \leq \sup_{f:\Omega \to [0,1]} |\mathbb{E}_\mu[f] - \mathbb{E}_\nu[f]|.$$

(c $\leq$ d): Now, since if f takes value in $[0, 1]$, then $f(x) - f(y) \leq 1$ generally, and $f(x) - f(y) = 0$ if $x = y$, which implies

$$\mathbb{E}_\mu[f] - \mathbb{E}_\nu[f] = \mathbb{E}_{X,Y \sim \pi}[f(X) - f(Y)] \leq \mathbb{E}_{X,Y \sim \pi}[1_{X \neq Y}],$$

for any $\pi \in \Gamma(\mu, \nu)$.

Now taking the infimum over $\pi \in \Gamma(\mu, \nu)$ on the RHS and then taking the supremum over $f \in m\mathcal{F}([0, 1])$ on the LHS yield

$$\sup_{f:\Omega \to [0,1]} |\mathbb{E}_\mu[f] - \mathbb{E}_\nu[f]| \leq \inf_{\pi \in \Gamma(\mu,\nu)} \mathbb{E}_\pi[1_{X \neq Y}].$$

(d = a): We let $\rho_0 := \nu(P) + \mu(N)$. The goal is to construct a coupling (X, Y) with $X = Y$ as often as possible so as to minimize the cost $1_{X \neq Y}$. We will let ρ_0 be the chance that $X = Y$ (note that $1 - \rho_0 = d_{TV}(\mu, \nu)$).

Concretely, with probability ρ_0, we let $X = Y$ follow the probability measure

$$\rho_{XY}(A) = \frac{\nu(P \cap A) + \mu(N \cap A)}{\rho_0}.$$

Otherwise, X and Y will be independent, following the product measure of

$$\rho_X(A) = \frac{\mu(P \cap A) - \nu(P \cap A)}{d_{TV}(\mu, \nu)} \quad \text{and} \quad \rho_Y(A) = \frac{\nu(N \cap A) - \mu(N \cap A)}{d_{TV}(\mu, \nu)}.$$

Then we have

$$\rho_0 \rho_{XY}(A) + (1 - \rho_0)\rho_X(A) = \mu(A) \quad \text{and} \quad \rho_0 \rho_{XY}(A) + (1 - \rho_0)\rho_Y(A) = \nu(A),$$

which means the marginals of X and Y are μ and ν, respectively. Let π be this mixture measure. With probability $1 - \rho_0$, X and Y take value on disjoint subsets of Ω (as $P \cap N = \emptyset$) and with probability ρ, $X = Y$, which means

$$\mathbb{E}_\pi[1_{X \neq Y}] = \pi(X \neq Y) = 1 - \rho_0 = d_{TV}(\mu, \nu).$$

Thus, there exists a coupling which attains the lower bound. □

Furthermore, convergence in total variation implies weak convergence. This makes it a useful tool for analysis.

Proposition 35. *Let μ_n and μ be such that $d_{TV}(\mu_n, \mu) \to 0$. Then $\mu_n \rightharpoonup \mu$.*

Proof. We use the variational form of the total variation (c). For any bounded, continuous f,

$$|\mathbb{E}_{\mu_n}[f(x_t)] - \mathbb{E}_\mu[f(x_\infty)]| \leq d_{\mathrm{TV}}(\mu_n, \mu) \cdot \|f\|_\infty$$

which converges to 0 as $n \to \infty$, since $\|f\|_\infty$ is finite. □

Finally, like the other f-divergences, total variation distance is invariant to reparameterization. We provide the general statement for all f-divergence below.

Proposition 36. *Let D_f be an f-divergence, p and q be two density functions, and h be a diffeomorphism. Suppose $p > 0$ whenever $q > 0$. Then $D_f(h_\# q || h_\# p) = D_f(q||p)$.*

Proof. Using the change-of-variable formula for $y = h(x)$, we have

$$2D_f(h_{\#}q||h_{\#}p) = \int_{\operatorname{supp} h_{\#}p} \left|\frac{h_{\#}q}{h_{\#}p}\right| h_{\#}p \,\mathrm{d}y \tag{B.5}$$

$$= \int_{\operatorname{supp} p} \left|\frac{q \,\mathrm{d}h/\mathrm{d}x}{p \,\mathrm{d}h/\mathrm{d}x}\right| p \,\mathrm{d}x \tag{B.6}$$

$$= \int_{\operatorname{supp} p} \left|\frac{q}{p}\right| p \,\mathrm{d}x = 2D_f(q||p). \tag{B.7}$$

□

Note that, in general, the absolute continuity requirement for total variation can be relaxed, since we can partition the space into the region where p and q are overlapping, and regions where one of the two has non-zero density, and we only need to apply the above proof technique to the first case.

C Appendix of Chapter 3

C.1 Proofs

In this section, we provide a proof of Theorem 7 and Corollary 8.

Let p be a density we want to approximate. Fix x, and let ϵ be a random variable, and define $y_n = x + \epsilon/n$, so that $y_n \to x$ a.s. as $n \to \infty$. If p is continuous, then $p(y_n) \to p(x)$. Denote by $q(y_n; x, 1/n^2)$ the density[1] of y_n. If we assume p is bounded, then by the *dominated convergence theorem*,

$$\begin{aligned} p(x) &= p(\lim y_n) = \mathbb{E}[p(\lim y_n)] = \lim \mathbb{E}[p(y_n)] \\ &= \lim_{n\to\infty} \int q(y_n; x, 1/n^2) p(y_n) \, \mathrm{d}y_n =: \lim g_n(x) \\ &\approx \lim_{n\to\infty} \sum_j^{M_n} \frac{1}{B_n} q(z_j; x, 1/n^2) p(z_j) =: \lim f_n(x), \end{aligned}$$

where $B_n = \sum_j p(z_j)$. Note that this renormalization guarantees f_n is a proper probability density function (of x), since we can rewrite it as

$$\begin{aligned} f_n(x) &= \sum_j^{M_n} \frac{p(z_j)}{B_n} q(-x; -z_j, 1/n^2) \\ \Rightarrow \int f_n(x) \, \mathrm{d}x &= \sum_j^{M_n} \frac{p(z_j)}{B_n} \int q(-x; -z_j, 1/n^2) \, \mathrm{d}x = 1. \end{aligned}$$

[1] If ϵ has a finite second moment and is standardized, x and $1/n^2$ would correspond to the mean and variance of y_n, respectively.

By the above heuristic, we know that with a large enough number of mixture components M_n, f_n can be used to approximate g_n, which will then converge to p by taking the bandwidth $1/n^2$ to 0. Note that M_n needs to grow sufficiently fast (depending on the shape of p and $q(\cdot; 0, 1)$, and n). Otherwise the numerical integration will fail as q will become too peaky.

We formalize this observation in Theorem 7, which is restated below. Note that f_n is an equivalent way to write a mixture density that we defined in Section 4.5.

Theorem 7 (Universality of mixture models). *If $q \in C^2 \cap \mathcal{P}$, then* $\text{Cl}(\mathcal{M}_q) = \mathcal{P}$.

Proof. We assume p is a C^∞ density function with a compact support, e.g. $\{x : p(x > 0)\} \subseteq [a, b]$. We will relax this condition later using the standard mollification technique. First, we decompose the error into two parts

$$|f_n(x) - p(x)| \leq \underbrace{|f_n(x) + g_n(x)|}_{A} + \underbrace{|g_n(x) + p(x)|}_{B}. \tag{C.1}$$

The first term describes the error due to numerical integration and renormalization, and the second term describes the error of the convolved density. Define

$$\tilde{f}_n(x) := \sum_j^{M_n} \frac{1}{C_n} h(z_n, x; n),$$

where $h(z, x; n) = q(z; x, 1/n^2)p(z)$ and $C_n = M_n/(b - a)$. This allows us to further decompose A as

$$A \leq \underbrace{|f_n(x) - \tilde{f}_n(x)|}_{A_1} + \underbrace{|\tilde{f}_n(x) - g_n(x)|}_{A_2}.$$

A_1 is the renormalization error, which vanishes asymptotically as B_n/C_n is just the

Riemann sum of $p(x)$, if z_j's are placed uniformly on the interval $[a, b]$. Precisely

$$A_1 \leq \sum_j^{M_n} \left| \frac{1}{B_n} - \frac{1}{C_n} \right| h(z_j, x; n) \tag{C.2}$$

$$= \sum_j^{M_n} \frac{|1 - B_n/C_n|}{B_n} \cdot h(z_j, x; n) \leq \frac{M_n}{B_n} \cdot \left| 1 - \frac{B_n}{C_n} \right| \cdot H_n, \tag{C.3}$$

where $H_n := \sup_{z,x} h(z, x; n)$. Since p is smooth, $|p''(x)|$ is bounded by some constant $K > 0$ in $[a, b]$, and by the midpoint rule (see 5.4 of Epperson [2013]), we have

$$\left| 1 - \frac{B_n}{C_n} \right| = \left| \int p(x)dx - \frac{B_n}{C_n} \right| \leq \frac{K(b-a)}{24C_n^2} = \frac{K(b-a)^3}{24M_n^2}.$$

Thus for any $0 < \eta < 1$, we can choose a big enough M_n such that $\left| 1 - \frac{B_n}{C_n} \right| \leq \eta$, which then implies $\frac{C_n}{B_n} \leq \frac{1}{1-\eta}$, and thus

$$A_1 \leq \frac{\eta}{1-\eta} \cdot H_n \cdot (b-a).$$

Choosing[2] $\eta = \min\{\frac{1}{2}, \frac{1}{2nH_n}\}$, we have $A_1 \leq \frac{b-a}{n}$.

Now applying midpoint rule again to A_2, we have

$$A_2 = |\tilde{f}_n(x) - g_n(x)| \leq \frac{G_n(b-a)^3}{24M_n^2},$$

where $G_n := \sup_{x \in [a,b]} |g_n''(x)|$. We can then set M_n to be big enough such that both A_1 and A_2 are $\mathcal{O}(1/n)$.

Now we turn to the error of convolution. By compactness of $[a, b]$ and smoothness, p is uniformly continuous. Thus, for any $e' > 0$, we can find a small enough $\delta > 0$ such

[2]Setting η implicitly determines a lower bound on M_n, since M_n needs to be big enough such that this inequality holds.

that $|p(x) - p(y)| \leq e'$ whenever $|x - y| \leq \delta$. By Jensen's inequality,

$$
\begin{aligned}
B &= |\mathbb{E}[p(y_n)] - p(x)| && \text{(C.4)}\\
&\leq \mathbb{E}[|p(y_n) - p(x)|] && \text{(C.5)}\\
&= \mathbb{E}[|p(y_n) - p(x)|; |x - y_n| \leq \delta] + \mathbb{E}[|p(y_n) - p(x)|; |x - y_n| > \delta] && \text{(C.6)}\\
&\leq e' + 2 \sup_y p(y) \cdot \mathbb{P}(|x - y_n| > \delta) && \text{(C.7)}\\
&\leq e' + 2 \sup_y p(y) \cdot \mathbb{P}(|\epsilon| > n\delta). && \text{(C.8)}
\end{aligned}
$$

This shows $f_n \to p$ uniformly as $n \to \infty$.

Now for $p \in L^1$ with bounded support, let p_ϵ be the mollified function $p_\epsilon(x) = \int p(x - y)\phi_\epsilon(y)\,\mathrm{d}y$, where ϕ_ϵ is a mollifier. Since $p_\epsilon \in C_c^\infty$, the above approximation scheme applies and Scheffe's lemma then implies $||f_{n(\epsilon)} - p_\epsilon||_1 \leq \epsilon$ for a sufficiently large integer $n(\epsilon)$. Taking $\epsilon \to 0$, we have

$$||f_{n(\epsilon)} - p||_1 \leq ||f_{n(\epsilon)} - p_\epsilon||_1 + ||p_\epsilon - p||_1 \to 0,$$

since $p \in L^1$ [Showalter, 2010, Lemma 1.2 of Chapter 2]).

If p has an unbounded support, let $Y_n := X1_{|X| \leq n} + U_n 1_{|X|>n}$ with U_n uniformly distributed between $\pm n$. Y_n has a density $p_n(y) = p(y) + \frac{\mathbb{P}(|X|>n)}{2n}$ for $y \in [-n, n]$ and 0 elsewhere, and $||p_n - p||_1 = 2\mathbb{P}(|X| > n)$. Since p_n has a bounded support, we can then take some f_n close enough to p_n such that $||f_n - p_n||_1 \leq 1/n$. As a result, $||f_n - p||_1 \to 0$ as $n \to \infty$ by triangle inequality.

□

Corollary 8 (Universality of finite-sum monotone flows). *If $q \in C^2 \cap \mathcal{P}$, then for any cumulative distribution function g, there exists a sequence of finite-sum monotone flows f_n with activation function σ_q that converges to g pointwise almost everywhere as $n \to \infty$.*

Proof. Assume g is an absolutely continuous CDF. Let f'_n be a sequence of mixture density functions converging in L_1 to g'. Then for any x, setting $f_n(x) := \int_{-\infty}^{x} f'_n(y)dy$ (which is a finite-sum monotone flow), we have

$$|f_n(x) - g(x)| \leq \int_{-\infty}^{x} |f'_n(y) - g'(y)|dy \leq \int_{\mathbb{R}} |f'_n(y) - g'(y)|dy = ||f'_n - g'||_1 \to 0.$$

Note that as $||f'_n - g'||$ is independent of x, the convergence is uniform.

For any CDF g, let X be a random variable distributed by g. Let $X_m = X + \epsilon/m$ which has an absolutely continuous distribution function g_m if the law of ϵ is absolutely continuous [Billingsley, 2008, p.266]. Since $X_m \to X$ almost surely, $X_m \to X$ in distribution and $g_m(x) \to g(x)$ for all the continuity points $x \in \mathbb{R}$ of g. Since g is continuous on a dense set (by Darboux-Froda's theorem), the above convergence holds almost everywhere. Now fix a continuity point x of g, and some $\delta > 0$. For each m, we can choose a sequence f_{mn} that converges uniformly to g_m as $n \to \infty$. For each m, choose n_m such that $|f_{mn_m} - g_m| \leq 1/m$. Then $|f_{mn_m}(x) - g(x)| \leq 1/m + |g_m(x) - g(x)| \leq \delta$ for a sufficiently large m (depending on x). □

D Appendix of Chapter 4

D.1 Proofs

Theorem 10 (Universality of NAF-DSF). *NAF-DSF of the form (4.5) are $\mathfrak{A} - \mathfrak{P}$ distributionally universal.*

Proof. This proof has two steps. We first prove that univariate DSFs itself can approximate arbitrary monotone functions. Unlike Corollary 8, we prove this by explicitly constructing an approximating function whose parameters would change continuously in a conditional setting, so that we can approximate these parameters using a universal hyper-conditioner network, which is the second part of the proof.

For simplicity, we assume μ and ν are both compactly supported with a strictly positive, continuous and bounded density function, and we want to transform μ into ν. This can be guaranteed since we can perturb them with a small Gaussian noise and then truncate the distributions and uniformly redistribute the mass outside of a compact support inside of the support. We deal with these extensions at the end of the proof.

Universality of univariate DSF To prove that DSF can approximate arbitrary monotone functions, we notice that the sigmoid activation function can approximate a step function via $x \mapsto \sigma(ax)$ by taking $a \to \infty$. This means we can first approximate the monotone function using a staircase function, and then approximate the staircase function using DSF. Concretely, fix an arbitrary total uniform error $\epsilon > 0$, and let $\epsilon_1 = \frac{1}{3}\epsilon$

and $\epsilon_2 = \frac{2}{3}\epsilon$. We will set the parameters of a staircase function and a pre-sigmoid DSF such that their approximation errors are bounded by ϵ_1 and ϵ_2, respectively, so that the DSF approximates the target pre-sigmoid monotone function with at most an ϵ uniform error.

Let $g : [0, 1] \to \mathbb{R}$ be a target continuous, non-decreasing function that we want to approximate (*i.e.* the unconditional univariate case of the KR map). We first show that we can approximate $\sigma \circ g$ uniformly using a staircase function $S(x)$, defined by

$$S(x) = \sum_{j=1}^{n} w_j \cdot h\left(x + b_j\right), \tag{D.1}$$

where h is the heaviside step function.

We choose $n = \max\{\lceil \frac{1}{\epsilon_1} \rceil, 1\}$, and divide the range into $n + 1$ evenly spaced intervals:

$$(0, y_1),\ (y_1, y_2),\ \cdots,\ (y_n, 1).$$

Let $-b_j$ be the inverse point of y_j; that is $b_j = -\inf\{x : \sigma(g(x)) \geq y_j\}$. This means $h(x + b_j) = 1$ whenever $x \geq -b_j = \inf\{x : \sigma(g(x)) \geq y_j\}$, which gives S a jump at the first n $n + 1$-quantiles of $\sigma \circ g$.

We can determine the jump size w_j coefficients so that $S(x) = y_j$ at $x = \inf\{x : \sigma(g(x)) \geq y_j\}$ for $1 \leq j \leq n - 1$, and for the last step we set $S(x) = 1$ at $x = \inf\{x : \sigma(g(x)) \geq y_n\}$, which would double the jump size. The last jump size is to ensure w_j sums to one. This amounts to solving a system of linear equations (by evaluating S at the first n $n + 1$-quantiles):

$$\begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ 1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 1 & \cdots & 1 & 0 \\ 1 & 1 & \cdots & 1 & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_{n-1} \\ w_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \\ 1 \end{bmatrix}, \tag{D.2}$$

where the $n \times n$ lower-triangular matrix of ones comes from the n evaluations of n hidden units with the heaviside activation. Inverting it gives

$$\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_{n-1} \\ w_n \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ 1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 1 & \cdots & 1 & 0 \\ 1 & 1 & \cdots & 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ -1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & -1 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{n+1} \\ \frac{2}{n+1} \\ \vdots \\ \frac{n-1}{n+1} \\ 1 \end{bmatrix}, \quad \text{(D.3)}$$

which means

$$\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_{n-1} \\ w_n \end{bmatrix} = \begin{bmatrix} \frac{1}{n+1} \\ \frac{1}{n+1} \\ \vdots \\ \frac{1}{n+1} \\ \frac{2}{n+1} \end{bmatrix}. \quad \text{(D.4)}$$

Since S intersects with $\sigma \circ g$ at the first n $n+1$-quantiles and S is monotone, the approximation error is uniformly bounded by $\frac{1}{n+1} \leq \epsilon_1$.

Now we set the parameters of a pre-logit DSF defined as

$$f(x) = \sum_{j=1}^{n} w_j \cdot \sigma\left(ax + ab_j\right), \quad \text{(D.5)}$$

which will approximate the staircase function with large a. Specifically, we can choose a sufficiently large so that f intersects with S at all of the jumps. Since f is also non-decreasing, this will imply f is $2\epsilon_1$-close to S, since $2\epsilon_1$ is the biggest jump size (the last jump). This means $|f - g| \leq |f - S| + |S - g| \leq 3\epsilon_1 = \epsilon$ uniformly.

To choose this a, we first let $\kappa = \min_{j \neq j'} |b_j - b_{j'}|$ be the smallest distance between the quantiles. Let $\lambda = \max\{\sigma^{-1}(\epsilon_0), \sigma^{-1}(1 - \epsilon_0)\}$ where $\epsilon_0 = \frac{1}{2} \cdot \frac{1}{n+1}$ is the smallest half-step size. Then we let $a = \frac{\lambda}{\kappa}$. Then the following calculation shows f intercepts

with S at all jump points $-b_j$, since f is ϵ_0-close to the half steps at all the jumps:

$$\begin{aligned}\left|f(-b_j) - \sum_{j'=1}^{j-1} w_{j'} - 0.5 \cdot w_j\right| &= \left|\sum_{j'=1}^{n} w_{j'} \cdot \sigma\left(a(b_{j'} - b_j)\right) - \sum_{j'=1}^{j-1} w_{j'} - 0.5 \cdot w_j\right| \\ &\leq \sum_{j'<j-1} w_{j'} \left|\sigma\left(a(b_{j'} - b_j)\right) - 1\right| + 0.5 \cdot w_j - 0.5 \cdot w_j + \\ &\quad \sum_{j'>j+1} w_{j'} \left|\sigma\left(a(b_{j'} - b_j)\right)\right| \\ &\leq \sum_{j'} w_{j'} \epsilon_0 = \frac{1}{2} \cdot \frac{1}{n+1}.\end{aligned}$$

Universality of conditional DSF Now, we want to show that there is a neural conditioner outputting $w_{tj}(x_{1:t-1}), a_{tj}(x_{1:t-1}), b_{tj}(x_{1:t-1})$ that can approximate the t's conditional transform of the KR map. Note that by assumption, μ has a compact support. This means for fixed t and $x_{1:t-1}$, the above approximation holds, and the approximation error can be made small by controlling the number of hidden units n alone. Fixing $\epsilon > 0$, we choose n big enough such that the above approximation scheme holds with a uniform error of $\frac{\epsilon}{2}$ (uniform over x_t), and denote by

$$\tilde{c}(x_{1:t-1}) := (\tilde{w}_{tj}(x_{1:t-1}), \tilde{a}_t(x_{1:t-1}), \tilde{b}_{tj}(x_{1:t-1}))$$

the parameters of (D.5), now depending on t and $x_{1:t-1}$. That is,

$$|f(x_t; \tilde{c}(x_{1:t-1})) - \sigma(KR_{\mu\to\nu}(x_{1:t})_t)| \leq \frac{\epsilon}{2}, \tag{D.6}$$

for any $x_{1:t}$ being the first t elements of $x_{1:d}$ in the support of μ, where

$$f(x_t; \tilde{c}(x_{1:t-1})) := \sum_{j=1}^{n} \tilde{w}_{tj}(x_{1:t-1}) \cdot \sigma\left(\tilde{a}_t(x_{1:t-1})x + \tilde{a}_t(x_{1:t-1})\tilde{b}_{tj}(x_{1:t-1})\right). \tag{D.7}$$

We have abused the notation a bit when writing $KR_{\mu\to\nu}(x_{1:t})_t$, since the KR map is a

function of a d-dimensional vector. This can be understood as padding $x_{1:t}$ with $d-t$ zeros, since the t'th output does not depend on those entries anyways.

Now we want to invoke the UFA theorem (Theorem 1) to show that we can approximate $\tilde{c}$ using a neural conditioner c, with a pre-specified error. First of all, note that the support of $\tilde{c}$ is compact. We just need to make sure $\tilde{c}$ is continuous. Note that $\tilde{w}_{tj}$ and $\tilde{a}_t$ are just constants, so we can approximate them (exactly) by setting the weights to be zero and returning the output biases only. This means we just need to approximate $\tilde{a}_t\tilde{b}_{tj}$ using b_{tj}. Note that $\tilde{a}_t$ is just a constant, which means the only thing changing value as $x_{1:t-1}$ varies is $\tilde{b}_{tj}$. To check its continuity, recall that by definition

$$\begin{aligned}
\tilde{b}_{tj}(x_{1:t-1}) &= -\inf\left\{x : \sigma\left(\Phi^{-1}_{\nu_t(\cdot\,|\,KR_{\mu\to\nu}(x_{1:t-1})_{1:t-1})}\left(\Phi_{\mu_t(\cdot\,|\,x_{1:t-1})}(x)\right)\right) \geq y_j\right\} \\
&= -\left(\sigma \circ \Phi^{-1}_{\nu_t(\cdot\,|\,KR_{\mu\to\nu}(x_{1:t-1})_{1:t-1})} \circ \Phi_{\mu_t(\cdot\,|\,x_{1:t-1})}\right)^{-1}(y_j) \\
&= -\Phi^{-1}_{\mu_t(\cdot\,|\,x_{1:t-1})}\left(\Phi_{\nu_t(\cdot\,|\,KR_{\mu\to\nu}(x_{1:t-1})_{1:t-1})}\left(\sigma^{-1}(y_j)\right)\right).
\end{aligned}$$

The inverse function is well defined since both μ and ν have strictly positive density.

Now we prove continuity. First, note that

$$\Phi_{\nu_t(\cdot\,|\,KR_{\mu\to\nu}(x_{1:t-1})_{1:t-1})}(x) = \int_{-\infty}^{x} \frac{q(x_1,\cdots,x_{t-1},x')}{q(x_1,\cdots,x_{t-1})}\,\mathrm{d}x',$$

where q is the joint density associated with ν. Since the integrand is continous wrt $x_{1:t-1}$ and bounded by assumption, the CDF transform for a fixed input is continuous wrt $x_{1:t-1}$.

Second, for $u \in [0,1]$,

$$\begin{aligned}
&\left|\Phi^{-1}_{\mu_t(\cdot\,|\,x_{1:t-1})}(u) - \Phi^{-1}_{\mu_t(\cdot\,|\,x'_{1:t-1})}(u)\right| \\
&= \left|\Phi^{-1}_{\mu_t(\cdot\,|\,x'_{1:t-1})}\left(\Phi_{\mu_t(\cdot\,|\,x'_{1:t-1})}\left(\Phi^{-1}_{\mu_t(\cdot\,|\,x_{1:t-1})}(u)\right)\right) - \Phi^{-1}_{\mu_t(\cdot\,|\,x'_{1:t-1})}(u)\right| \\
&=: \left|\Phi^{-1}_{\mu_t(\cdot\,|\,x'_{1:t-1})}(u') - \Phi^{-1}_{\mu_t(\cdot\,|\,x'_{1:t-1})}(u)\right| \\
&\leq \sup_{u\in[0,1]}\left|\left(\Phi^{-1}_{\mu_t(\cdot\,|\,x'_{1:t-1})}\right)'(\tilde{u})\right| \cdot |u'-u|.
\end{aligned}$$

Similarly to the previous step, $u' \to u$ as $x'_{1:t-1} \to x_{1:t-1}$. The Lipschitz constant (the first term) is also bounded, since

$$\left(\Phi^{-1}_{\mu_t(\cdot\,|\,x'_{1:t-1})}\right)'(\tilde{u}) = \frac{1}{\left(\Phi_{\mu_t(\cdot\,|\,x'_{1:t-1})}\right)'(\tilde{x})} = \frac{p(x'_1,\cdots,x'_{t-1})}{p(x'_1,\cdots,x'_{t-1},\tilde{x})},$$

which is bounded, by the compactness, continuity and strict positivity of the density p, where $\tilde{x} = \Phi^{-1}_{\mu_t(\cdot\,|\,x'_{1:t-1})}(\tilde{u})$. Therefore, $\Phi^{-1}_{\mu_t(\cdot\,|\,x_{1:t-1})}(u)$ is continuous in $x_{1:t-1}$.

Furthermore, we can actually show that it is continuous wrt $x_{1:t-1}$ and u jointly. This means $\tilde{b}_{tj}$ is continuous wrt $x_{1:t-1}$, and that we can find a neural conditioner network c such that

$$|c(x_{1:t-1}) - \tilde{c}(x_{1:t-1})| \leq \delta$$

for an arbitrary pre-specified $\delta > 0$. Now since $f(x_t;\cdot)$ is uniformly continuous on some compact neighorhood of $\tilde{c}$, there exists some $\delta > 0$ such that if $|c-\tilde{c}| \leq \delta$, $|f(x_t;c) - f(x_t;\tilde{c})| \leq \frac{\epsilon}{2}$. We let this δ be the pre-specified error of the neural conditioner. Combining (D.7), we have

$$|f(x_t; c(x_{1:t-1})) - \sigma(KR_{\mu\to\nu}(x_{1:t})_t)| \leq \epsilon.$$

That is, pre-logit NAF-DSF can approximate $\sigma \circ KR_{\mu\to\nu}$ uniformly, which means NAF-DSF can approximate the KR map pointwise. Lemma 5 then implies convergence in distribution.

Relaxing assumptions The above approximation holds for μ and ν having a strictly positive density on a compact support.

More generally, we assume $\mu \in \mathfrak{A}$ and $\nu \in \mathfrak{P}$. That is, μ has a Lebesgue density p, and ν is an arbitrary probability measure. Let $X \sim \mu$ and $Y \sim \nu$. We now construct an approximation to μ with a compactly supported probability measure that has a strictly positive, continuous and bounded probability density function. Denote by B_k a ball of radius $k > 0$ centered at the origin, *i.e.* $B_k := \{x : ||x|| \leq k\}$. Let ϵ denote an independent, standard normal random vector, and U_k denote an independent, random vector uniformly distributed on B_k. For $x, u \in \mathbb{R}^d$ and $B \subseteq \mathbb{R}^d$, denote by $T(x, u, B) = x1_{x \in B} + u1_{x \notin B}$. Now let $X_k = T(X + \sigma_k \epsilon, U, B_k)$ where $\sigma_k > 0$ decreases to 0. The addition $X + \sigma_k \epsilon$ ensures a strictly positive, continuous and bounded density [Billingsley, 2008]. The truncation T redistributes the mass outside of B_k uniformly in B_k, making the density compactly supported. Let μ_k denote the law of X_k, and let Y_k and ν_k be similarly defined.

From above, since X_k and Y_k both have a continuous, strictly positive and bounded density function, we know fixing k, we can find a sequence of $f_{k,n}$ such that $f_{k,n}(X_k) \to Y_k$ in distribution as $n \to \infty$. Now since weak convergence is metrizable, choose n_k to be large enough such that the distance between the distribution of $f_{k,n_k}(X_k)$ and ν_k is at most $1/k$. An application of the triangle inequality of the weak metric implies $f_{k,n_k}(X_k) \to Y$ in distribution as $k \to \infty$. Finally, note that since $\{||f_{k,n_k}(X_k) - f_{k,n_k}(X + \sigma_k \epsilon)|| > \epsilon\} \subseteq \{||X + \sigma_k \epsilon|| > k\}$, by monotonicity, we have

$$\mathbb{P}\left(\limsup_k ||f_{k,n_k}(X_k) - f_{k,n_k}(X + \sigma_k \epsilon)|| > \epsilon\right) \leq \mathbb{P}\left(\limsup_k ||X + \sigma_k \epsilon|| > k\right) = 0 \tag{D.8}$$

since σ_k is decreasing. Thus $||f_{k,n_k}(X_k) - f_{k,n_k}(X + \sigma_k \epsilon)|| \to 0$ almost surely (where $\mathbb{P}$ is the underlying probability measure of the measure space). By Lemma 32, $f_{k,n_k}(X + \sigma_k \epsilon)$ converges in distribution to Y. Now, to get rid of the additive regularization $\sigma_k \epsilon$, we note that the the convolved density $\mathcal{N}(0, \sigma_k^2 I) \star p$ satisfies [Brezis and Brézis, 2011,

Theorem 4.22]

$$\left\|\mathcal{N}(0, \sigma_k^2 I) \star p - p\right\|_1 \to 0. \tag{D.9}$$

That is, $X + \sigma_k \epsilon$ converges in total variation to X. Furthermore, since f_{k,n_k} is a bijection, and the total variation distance is invariant to reparameterization (Proposition 36), the total variation distance between the distribution of $f_{k,n_k}(X + \sigma_k \epsilon)$ and the distribution of $f_{k,n_k}(X)$ converges to 0. By Proposition 35, since the former converges to the distribution of Y, so does the latter.

□

E Appendix of Chapter 5

E.1 SOFTPLUS-TYPE ACTIVATION

In this section, we let $r(x) = \max(0, x)$ be the ReLU activation function.

Definition 37 (Softplus-type activation)**.** *We say a function s is of the softplus type if the following holds*

(a) $s \geq r$.

(b) s *is convex.*

(c) $|s(x) - r(x)| \to 0$ *as* $|x| \to \infty$.

Note that a softplus-type activation function is necessarily continuous, non-decreasing, and uniformly approximating ReLU in the following sense:

$$|s(xa)/a - r(x)| \to 0$$

uniformly for all $x \in \mathbb{R}$ as $a \to \infty$.

The following proposition characterizes a big family of softplus-type functions, and establishes a close connection between softplus type functions and probability distribution functions.

Proposition 38 (CDFs and softplus-type activations)**.** *Let p be a probability density function of a random variable with mean zero. Then the convolution $s := p * r$ is a*

softplus-type function. Moreover, $s(x) = \int_{-\infty}^{x} F_p(y)\,\mathrm{d}y$*, where* F_p *is the distribution function of* p*, and* s *is at least twice differentiable.*

Proof. We first prove a claim (i): $xF_p(x) \to 0$ as $x \to -\infty$. First, for $x \leq 0$,

$$0 \geq xF_p(x) = \int_{-\infty}^{x} xp(y)\,\mathrm{d}y \geq \int_{-\infty}^{x} yp(y)\,\mathrm{d}y.$$

Since $1_{y\leq x}yp(y) \to 0$ as $x \to -\infty$ and $|1_{y\leq x}yp(y)| \leq |yp(y)|$, which is integrable by assumption, the integral on the RHS of the above goes to 0 by the dominated convergence theorem.

We now show the identity. By definition, since $\int yp(y)dy = 0$

$$\begin{aligned} s(x) &= \int \max(x-y,0)p(y)\,\mathrm{d}y = \int \max(x,y)p(y)\,\mathrm{d}y \\ &= \int_{-\infty}^{x} xp(y)\,\mathrm{d}y + \int_{x}^{\infty} yp(y)\,\mathrm{d}y = xF_p(x) + \int_{x}^{\infty} yp(y)\,\mathrm{d}y, \end{aligned} \tag{E.1}$$

where we've used claim (i) to evaluate $xF_p(x)$ as $x \to -\infty$. On the other hand, integration by part implies

$$\int_{-\infty}^{x} F_p(y)\,\mathrm{d}y = yF_p(y)\big|_{-\infty}^{x} - \int_{-\infty}^{x} yp(y)\,\mathrm{d}y = xF_p(x) + \int_{x}^{\infty} yp(y)\,\mathrm{d}y. \tag{E.2}$$

Twice differentiability follows from the differentiability of F_p.

(a) Now since $r(x)$ is convex, Jensen's inequality gives

$$s(x) = \int r(y)p(x-y)\,\mathrm{d}y = \mathbb{E}[r(y)] \geq r(\mathbb{E}[y]) = r(x).$$

(b) s is convex because $s' = F_p$ is non-decreasing.

(c) To show that s ane r are asymptotically the same, we notice the integral on the

RHS of (E.1) goes to 0 as $|x| \to \infty$ (by the dominated or monotone convergence theorem). It suffices to show $xF_p(x)$ goes to 0 as $x \to -\infty$, which is just claim (i), and $x - xF_p(x) > 0$ goes to 0 as $x \to \infty$. To show the latter, we can rewrite $x - xF_p(x) = x(1 - F_p(x)) = \int_x^\infty xp(y)\,\mathrm{d}y$, and the same argument as the claim holds with a vanishing upper bound.

□

When p is taken to be the standard logistic density, the corresponding $s = p * r$ is simply the regular softplus activation function. We list a few other softplus-type functions in E.1 and visualize them in E.1.

	$p(y)$	$s := p * r$
Logistic	$\frac{\exp(-x)}{(1+\exp(-x))^2}$	$\log(1+\exp(x))$
Laplace	$\frac{e^{-\lvert x\rvert}}{2}$	$r(x) + \frac{e^{-\lvert x\rvert}}{2}$
Gaussian	$\frac{e^{-\frac{x^2}{2}}}{\sqrt{2\pi}}$	$\frac{\sqrt{\frac{\pi}{2}}x\,\mathrm{erf}\left(\frac{x}{\sqrt{2}}\right)+e^{-\frac{x^2}{2}}+\sqrt{\frac{\pi}{2}}x}{\sqrt{2\pi}}$

Table E.1: Formula of some softplus-type functions.

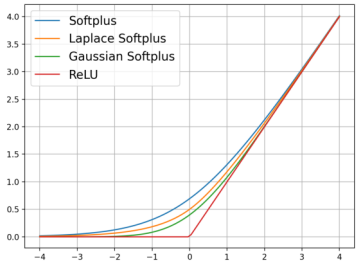


Figure E.1: Softplus-type functions.

E.2 PROOFS

Theorem 11 (Invertibility of convex potential flow)**.** *Let $F : \mathbb{R}^d \to \mathbb{R}$ be a C^2 strictly convex mapping. Then $f(x) := \nabla F(x)$ is injective. If F is furthermore strongly convex, then $x \mapsto F(x) - y^\top x$ has a unique minimizer, which implies $f(x)$ is surjective.*

Proof. We provide a formal proof of the invertibility of CP-Flow, and then establish the connection to convex conjugacy (the Legendre-Fenchel transform). We first check that $f := \nabla F$ is injective if F is *strictly* convex. This is because if F is twice differentiable and strictly convex, the Hessian matrix $H := \nabla^2 F$ is symmetric positive definite, and thus $z^\top H z > 0$ for any non-zero vector z. We then have, for any $x \neq y$,

$$f(x) - f(y) = \int_\gamma H(\gamma)\,\mathrm{d}\gamma = \int_0^1 H(y + t(x-y))(x-y)\,\mathrm{d}t,$$

where we used the *gradient theorem* for the line integral on a path γ connecting x and y, and substituted $t \mapsto y + t(x-y)$ for t going from 0 to 1. Positive-definiteness (along with monotonicity of the integral) implies $(x-y)^\top(f(x) - f(y)) > 0$, and since $x \neq y$, $f(x) \neq f(y)$.

Now we further assume F is *strongly* convex. Then for any y, $F_y(x) := F(x) - x^\top y$ is also strongly convex, which, by Taylor's theorem, implies that we can place a quadratic lower bound on F_y and thus $F_y(x) \to \infty$ whenever $||x|| \to \infty$. This means for a sufficiently large constant R, the sub-level set $S_R := \{x : F_y(x) \leq R\}$ is non-empty and compact. By the *Weierstrass extreme value theorem*, F_y (restricted on S_R) has a minimizer x^*, and it is also the global minimizer over $\mathbb{R}^d$. Now lets differentiate F_y at x^*, which gives $\nabla F(x^*) - y$. The gradient must be equal to 0 by the first order condition, meaning x^* is the inverse point of y under f. Since this holds for any $y \in \mathbb{R}^d$, f is surjective.

Now recall the definition of the convex conjugate:

$$F^*(y) := \sup_x x^\top y - F(x) = {x^*}^\top y - F(x^*),$$

where x^* found by the above procedure depends on y. Note that x^* is differentiable by the inverse function theorem. Thus, differentiating F^* yields

$$\nabla_y F^*(y) = (\nabla_y x^*)^\top y + x^* - (\nabla_y x^*)^\top \nabla F(x^*) = x^*,$$

since $\nabla F(x^*) = y$. This means if $y = f(x) = \nabla_x F(x)$, then $x = \nabla_y F^*(y)$; *i.e.* $\nabla F^* = (\nabla F)^{-1}$. □

We first prepare a few functional universality results for proving Theorem 13.

Notation: Given a convex set $\Omega \subseteq \mathbb{R}^d$, we let $\mathcal{C}(\Omega)$ denote the set of continuous functions on Ω, and $\mathcal{C}_\times(\Omega) := \{f \in \mathcal{C}(\Omega) : f \text{ is convex}\}$ denote the set of convex, continuous functions.

We first show that ICNNs with a suitable activation function are dense in $C_\times$. A similar result can be found in Chen et al. [2019b], where they use a different constructive proof: first show that piecewise maximum of affine functions, *i.e.* the maxout unit [Goodfellow et al., 2013], can approximate any convex function, and then represent maxout using ICNN. We emphasize our construction is simpler (see proof of Proposition 40).

The following proposition proves that functions that are pointwise maximum of affine functions, are a dense subset of $C_\times$.

Proposition 39 (Universality of maxout)**.** *Pointwise maximum of affine functions is dense in* $\mathcal{C}_\times([0,1]^d)$.

Proof. Fix some $\epsilon > 0$. Since $f \in \mathcal{C}_\times([0,1]^d)$ is uniformly continuous on $[0,1]^d$, there exists some $\delta > 0$ such that $|f(x) - f(y)| < \epsilon$ provided that $||x - y|| < \delta$. Let n be big enough such that $2^{-n} < \delta$, and let $\mathcal{X}$ be the set of points whose coordinates sit on $i2^{-n}$ for some $1 \leq i \leq 2^n - 1$ (i.e. there are $|\mathcal{X}| = (2^n - 1)^d$ points in $\mathcal{X}$). For each $y \in \mathcal{X}$, let $L_y(x) := \nabla f(y)^\top (x - y) + f(y)$ be a supporting hyperplane of the graph of f, where $\nabla f(y)$ is a subgradient of f evaluated at y. Then we have a convex approximation $f_\epsilon(x) := \max_{y \in \mathcal{X}} L_y(x)$ which bounds f from below. Moreover, letting $y_{|x} := \arg\min_{y \in \mathcal{X}} ||x - y||$, we have (for $x \notin \mathcal{X}$),

$$
\begin{aligned}
f(x) - f_\epsilon(x) &= f(x) - \max_{y \in \mathcal{X}} L_y(x) \\
&\leq f(x) - L_{y_{|x}}(x) \\
&= f(x) - f(y_{|x}) - \sum_{i=1}^{d} \nabla f(y_{|x})_i (x_i - y_{|x,i}) \\
&\leq f(x) - f(y_{|x}) + \sum_{i=1}^{d} |\nabla f(y_{|x})_i| \cdot |x_i - y_{|x,i}| \\
&\leq f(x) - f(y_{|x}) + \sum_{i=1}^{d} \frac{\epsilon}{|x_i - y_{|x,i}|} \cdot |x_i - y_{|x,i}| \\
&\leq (d+1)\epsilon.
\end{aligned}
$$

Since ϵ is arbitrary, this construction forms a sequence of approximations converging uniformly to f from below. □

The following proposition shows that maxout units can be equivalently represented by ICNN with the ReLU activation, and thus entails the density of the latter (as well as ICNN with softplus activation).

Proposition 40 (Universality of ICNN)**.** *ICNN with ReLU or softplus-type activation is dense in* $\mathcal{C}_\times([0,1]^d)$.

Proof. Let $r(x) = \max(0, x)$ be the ReLU activation funciton. Any convex piecewise linear function $f(x)$ can be represented by $f(x) = \max(L_1, ..., L_k)$ where $L_j = a_j^\top x + b_j$, which can then be reduced to

$$
\begin{aligned}
f(x) &= r(\max(L_1 - L_k, ..., L_{k-1} - L_k)) + L_k \\
&= r(r(\max(L_1 - L_{k-1}, ..., L_{k-2} - L_{k-1})) + L_{k-1} - L_k) + L_k \\
&= z_k,
\end{aligned}
$$

where $z_j := r(z_{j-1}) + L'_j$ for $2 \leq j \leq k$, $z_1 = L_1 - L_2$, $L'_j := L_j - L_{j+1}$ for $2 \leq j \leq k-1$, and $L'_k := L_k$.

Since by Proposition 39, pointwise maximum of affine functions is dense in $\mathcal{C}_\times$, so is ICNN with the ReLU activation function. The same holds for softplus since softplus can be used to uniformly approximate ReLU.

□

Theorem 12. *Let $F_n : \mathbb{R}^d \to \mathbb{R}$ be differentiable convex functions and $G : \mathbb{R}^d \to \mathbb{R}$ be a proper convex function. Assume $F_n \to G$. Then for almost every $x \in \mathbb{R}^d$, G is differentiable and $f_n(x) := \nabla F_n(x) \to \nabla G(x) =: g(x)$.*

Proof. We let x be a differentiable point of G. Since convergence of derivatives wrt each coordinate can be dealt with independently, we assume $d = 1$ without loss of generality. We can write f_n as

$$f_n(x) = \lim_m f_{nm}(x) \quad \text{where} \quad f_{nm}(x) = \frac{F_n(x - 1/m) - F_n(x)}{-1/m}.$$

The problem can be rephrased as proving [1]

$$\lim_n \lim_m f_{nm} = \lim_m \lim_n f_{nm}. \tag{E.3}$$

Note that f_{nm} is non-decreasing in m since F_n is convex, and thus $f_{nm} \leq f_n$. Since f_{nm} converges to f_n, for any $\epsilon > 0$, we can find an integer $\mu(\epsilon, n)$ such that for all $m \geq \mu(\epsilon, n)$, $|f_{nm} - f_n| \leq \epsilon$. Let m_k be a subsequence of $\{m \geq 1\}$ defined as $m_k = \mu(2^{-k}, n)$.

Then $|f_{nm_{k+1}} - f_{nm_k}| \leq 2^{-k}$, which is integrable wrt the counting measure on positive

[1] Note that there is an implicit dependency on x since the result is pointwise.

integers k, since

$$\int 2^{-k} = \lim_{K\to\infty} \sum_{k=1}^{K} 2^{-k} = 1.$$

Thus, letting $f_{nm_0} = 0$, by the Dominated Convergence Theorem, we have

$$\lim_n \lim_K f_{nm_K} = \lim_n \int f_{nm_k} - f_{nm_{k-1}} = \int \lim_n f_{nm_k} - f_{nm_{k-1}} = \lim_K \lim_n f_{nm_K}.$$

Although we are only looking at the limit of the subsequence m_k, this is sufficient for (E.3), since the LHS is equal to $\lim_n f_n$ (since each F_n is differentiable), and by linearity of the limit, the RHS is equal to $\lim_K \frac{G(x-1/m_K)-G(x)}{-1/m_K} = g(x)$ (since G is differentiable at x by assumption).

Since the set of points over which G is not differentiable is a set of measure zero [Rockafellar, 1970, Thm. 25.5], the convergence holds almost everywhere. □

Theorem 13 (**Universality of CP-Flow**). *Gradient maps of ICNN with softplus-type activation are* $\mathfrak{A} - \mathfrak{P}$ *distributionally universal.*

Proof. Assume μ and ν have finite second moments. Since μ is absolutely continuous, by **Brenier's theorem**, there exists a convex function $G : \mathbb{R}^d \to \mathbb{R}$ such that $\nabla G(X) \overset{d}{=} \nu$ (where the gradient is unique up to changes on a null set). By Proposition 40, there exists a sequence of ICNN F_n converging to G pointwise everywhere. Such a sequence can be found since we can let F_n approximate G with a uniform error of $1/n$ on a compact domain $[-n, n]^d$. Theorem 12 then implies the gradient map $f_n := \nabla F_n$ converges to ∇G pointwise almost everywhere. This implies the weak convergence of the pushforward measure of $f_n \circ X$.

Relaxing assumptions Now remove the finite second moment assumption and let X and Y be random variables distributed according to μ (with Lebesgue density p) and ν, respectively. We will now use an argument similar to (but simpler than) the relaxation argument used in the proof of Theorem 10. Denote by B_k a ball of radius $k > 0$ centered at the origin, i.e. $B_k := \{x : ||x|| \leq k\}$. Let $X_k = X1_{X \in B_k} + U_k 1_{X \notin B_k}$, where U_k is an independent random variable distributed uniformly on B_k, and let μ_k be the law of X_k. Then $X_k \to X$ almost surely as $k \to \infty$, and μ_k is still absolutely continuous wrt the Lebesgue measure, with its density being $p(x) + \frac{1}{\text{vol}(B_k)}\mu(||X|| > k)$ if $||x|| \leq k$ or 0 otherwise. Let Y_k and ν_k be defined similarly (while ν_k may not be absolutely continuous wrt Lebesgue). From above, since X_k and Y_k are bounded and admit a finite second moment, we know fixing k, we can find a sequence of $f_{k,n} = \nabla F_{k,n}$ such that $f_{k,n}(X_k) \to Y_k$ in distribution as $n \to \infty$. Now since weak convergence is metrizable, choose n_k to be large enough such that the distance between the distribution of $f_{k,n_k}(X_k)$ and ν_k is at most $1/k$. An application of the triangle inequality of the weak metric implies $f_{k,n_k}(X_k) \to Y$ in distribution as $k \to \infty$. Finally, note that since $\{||f_{k,n_k}(X_k) - f_{k,n_k}(X)|| > \epsilon\} \subseteq \{||X|| > k\}$, by monotonicity, we have

$$\mathbb{P}\left(\limsup_k ||f_{k,n_k}(X_k) - f_{k,n_k}(X)|| > \epsilon\right) \leq \mathbb{P}\left(\limsup_k ||X|| > k\right) = 0, \quad \text{(E.4)}$$

and thus $||f_{k,n_k}(X_k) - f_{k,n_k}(X)|| \to 0$ almost surely (where $\mathbb{P}$ is the underlying probability measure of the measure space). By Lemma 32, $f_{k,n_k}(X)$ converges in distribution to Y.

□

Theorem 15 (Optimality of CP-Flow). *Let G be the Brenier potential of $X \sim \mu$ and $Y \sim \nu$, and let F_n be a convergent sequence of differentiable, convex potentials, such that $\nabla F_n \circ X \to Y$ in distribution. Then ∇F_n converges almost surely to ∇G.*

The result can be deduced from the fact that optimality is "stable" under weak limit; see for example, Santambrogio [2015, Theorem 1.50]. We prove the special case of quadratic cost function.

Proof. We claim that if F is a convex potential such that $Z = \nabla F(X)$ has ν as its law, then $\nabla F \equiv \nabla G$ almost surely. The proof of the claim is originally due to **?**, but we present it here for completeness. Let Z' be another random variable distributed by ν. Then by the Fenchel-Young inequality (applied to the convex potential F),

$$\begin{aligned}\mathbb{E}[X^\top Z'] &\leq \mathbb{E}[F(X) + F^*(Z')] = \mathbb{E}[F(X) + F^*(Z)] \\ &= \mathbb{E}[F(X) + F^*(\nabla F(X))] = \mathbb{E}[X^\top \nabla F(X)].\end{aligned}$$

This concludes the proof since ∇G uniquely solves the transportation problem, which is equivalent to finding a transport map $\tilde{g}$ that maximizes the covariance:

$$\mathbb{E}[||X - \tilde{g}(X)||^2] = \mathbb{E}[||X||^2 + ||\tilde{g}(X)||^2] - 2\mathbb{E}[X^\top \tilde{g}(X)].$$

Let F_∞ to be the pointwise limit of F_n. Then for any x_1, x_2 and $t \in [0, 1]$,

$$\begin{aligned}F_\infty(tx_1 + (1-t)x_2) &= \lim_{n\to\infty} F_n(tx_1 + (1-t)x_2) \\ &\leq \lim_{n\to\infty} tF_n(x_1) + (1-t)F_n(x_2) \\ &= \lim_{n\to\infty} tF_n(x_1) + \lim_{n\to\infty}(1-t)F_n(x_2) \\ &= tF_\infty(x_1) + (1-t)F_\infty(x_2).\end{aligned}$$

That is, F_∞ is convex. Now since F_n is a convergent sequence of convex functions, its gradient ∇F_n also converges pointwise almost everywhere to ∇F_∞ by Theorem 12. Let ρ denote the Prokhorov metric, which metrizes the weak convergence, and by abuse of notation, we write $\rho(X, Y)$ to denote the distance between the law of X and Y. Then

$$\rho(\nabla F_\infty(X), Y) \leq \rho(\nabla F_\infty(X), \nabla F_n(X)) + \rho(\nabla F_n(X), Y),$$

which means $\nabla F_\infty(X)$ and Y have the same law, ν. Then by the claim, $\nabla F_\infty \equiv \nabla G$, and thus $\nabla F_n \to \nabla G$ a.s. as $n \to \infty$.

□

F Appendix of Chapter 7

F.1 Proofs

Theorem 17 (Universality of parametric flow map). *Let $\mathcal{V}$ be a family of (parametric) Lipschitz functions that is is dense (wrt the uniform metric) in $C(K \times [0,T]; \mathbb{R}^d)$ for any compact domain $K \subset \mathbb{R}^d$. Let $v(x,t) : \mathbb{R}^d \times [0,T] \to \mathbb{R}^d$ be a velocity field uniformly Lipschitz in x for all t, and continuous in t. Then for any compact set $K \in \mathbb{R}^d$, we can find a (parametric) velocity $v' \in \mathcal{V}$ such that $\sup_{x \in K, t \in [0,T]} ||\phi_{v'}(x,t) - \phi_v(x,t)|| \leq \epsilon$.*

Proof. Let us start by studying the flow maps induced by some velocity that is globally ϵ-close to v. Denote by $\mathcal{E}(C,T,\epsilon)$ the set of functions Lipschitz functions that are uniformly ϵ-small over $C \times [0,T]$ for any $C \subseteq \mathbb{R}^d$ and $T > 0$, *i.e.*

$$\mathcal{E}(C,T,\epsilon) := \left\{ u : \|u\|_L < \infty, \sup_{x,t \in C \times [0,T]} \|u(x,t)\| \leq \epsilon \right\}.$$

Let $L > 0$ be the uniform Lipschitz constant of v. We first bound the approximation

error of the flow map of $v' := v+u$ for any $u \in \mathcal{E}(\mathbb{R}^d, T, \epsilon)$. For any $(x,t) \in K \times [0,T]$,

$$\begin{aligned}
&\frac{\mathrm{d}}{\mathrm{d}t} \left\| \phi_{v'}(x,t) - \phi_v(x,t) \right\| \\
&\quad = \frac{1}{2} \cdot \frac{1}{\left\| \phi_{v'}(x,t) - \phi_v(x,t) \right\|} \cdot 2 \cdot (\phi_{v'}(x,t) - \phi_v(x,t))^\top (v'(\phi_{v'}(x,t),t) - v(\phi_v(x,t),t)) \\
&\quad \overset{(1)}{\leq} \left\| v'(\phi_{v'}(x,t),t) - v(\phi_v(x,t),t) \right\| \\
&\quad \overset{(2)}{\leq} \left\| v'(\phi_{v'}(x,t),t) - v(\phi_{v'}(x,t),t) \right\| + \left\| v(\phi_{v'}(x,t),t) - v(\phi_v(x,t),t) \right\| \\
&\quad \overset{(3)}{\leq} \epsilon + L \left\| \phi_{v'}(x,t) - \phi_v(x,t) \right\|,
\end{aligned}$$

where (1) is by Cauchy-Schwarz, (2) is by triangle inequality, and (3) is by definition of v' and Lipschitzness of v.

Gronwall's inequality then implies $\|\phi_{v'}(x,t) - \phi_v(x,t)\| \leq \epsilon \cdot te^{Lt} \leq \epsilon \cdot Te^{LT}$ (which does not depend on x or t).

This also allows us to shrink the approximating domain of v'. Let

$$K' = \mathrm{Cl}\left(\bigcup_{x,t \in K \times [0,T]} B(\phi_v(x,t), \epsilon \cdot te^{Lt}) \right).$$

where Cl denotes the closure of a set. Since $\phi_v(K,[0,T])$ is compact and $\epsilon \cdot te^{lT}$ is bounded, k' is also compact. The error bound tells us that ϕ'_v will be contained in the set K' for any $x, t \in K \times [0,T]$. The same error bound still holds for any $u \in \mathcal{E}(K', T, \epsilon)$, since the above derivation only makes use of the uniform approximation error within the set $K' \times [0,T]$. Finally, let $v' \in \mathcal{V}$ be ϵ-close to v uniformly on $K' \times [0,T]$. Since $v' - v \in \mathcal{E}(K', T, \epsilon)$, we have that $\|\phi_{v'}(x,t) - \phi_v(x,t)\| \leq \epsilon \cdot Te^{LT}$ uniformly on $K \times [0,T]$.

□

Theorem 18 (Universality of Neural ODE). *Flow maps of neural ODE are $\mathfrak{A} - \mathfrak{P}$ distributionally universal.*

Proof. Following the same regularization treatment as Theorem 13, we assume the input and output measures both have densities that are smooth and positively supported in the unit ball without loss of generality. Then the Brenier map g is a smooth diffeomorphism [Urbas, 1997]. We claim that the velocity field (7.17) is uniformly Lipschitz in x for all t. First, we bound the Jacobian norm of the inverse map $g^{-1}(x,t)$. Let $x' = g^{-1}(x,t)$. Then by the inverse function theorem,

$$\begin{aligned}\left\|\nabla g^{-1}(x,t)\right\| &= \left\|\nabla g(x',t)^{-1}\right\| \\ &= \left\|(t\nabla g(x') + (1-t)I)^{-1}\right\| \\ &= \frac{1}{t\lambda_{\min}(x') + 1 - t},\end{aligned}$$

where $\lambda_{\min}(x')$ is the smallest eigenvalue of $\nabla g(x')$. Since $g^{-1}(x)$ is smooth,

$$\infty > \sup_{\|x\|=1} \left\|\nabla g^{-1}(x)\right\| = \sup_{\|x\|=1} \left\|\nabla g(g^{-1}(x))^{-1}\right\| = \frac{1}{\inf_{\|x\|=1} \lambda_{\min}(g^{-1}(x))}.$$

Therefore,

$$\left\|\nabla g^{-1}(x,t)\right\| \leq \frac{1}{\inf_{t\leq[0,1]} t \inf_{\|x\|=1} \lambda_{\min}(x') + 1 - t} \leq \max\left\{1, \frac{1}{\inf_{\|x\|=1} \lambda_{\min}(x')}\right\}.$$

This implies $v(x,t)$ is uniformly Lipschitz. By Theorem 17 and Lemma 5, we conclude the proof.

□

G Appendix of Chapter 8

G.1 Notation

We use (Y_s, s) to denote the inference process (where Y_0 is the data), and (X_t, t) to denote the generative process (where X_0 is a random variable following an unstructured prior). We use s and t to distinguish the two directions, and always integrate the differential equations from 0 to $T > 0$ (different from the literature, where sometimes one might see integration from T to 0). $\hat{B}_s$ and B_t denote the Brownian motions associated with the inference and generative SDEs, respectively. B'_s is a reparameterization of $\hat{B}_s$ (see Section 8.4). $q(y, s)$ and $p(x, t)$ denote the probability density functions of Y_s and X_t, respectively. We let $\mathbf{s}_\theta$ denote a time-indexed parameterized function that will be used to approximate the score $\nabla \log q(y, s)$. ∇ is the gradient wrt the spatial variable (x or y, which we sometimes call position), ∂_t, ∂_s and ∂_{x_i} are partial derivatives, and H_* denotes Hessian.

G.2 1-D GIRSANOV AND VARIATIONAL INFERENCE

The Girsanov theorem (aka the Cameron–Martin-Girsanov theorem) describes how translation affects Wiener (Gaussian) measures. In Section 8.4, we deal with the infinite dimensional case, therefore demanding a formal measure-theoretic treatment. In this section, we use a one-dimensional case to illustrate how to interpret the Girsanov theorem and how we use it to derive the CT-ELBO, using the more familiar notion of probability density functions. Now imagine we do not have an infinite-dimensional latent variable (*i.e.* the Brownian motion B'). Instead, imagine we have a one-dimensional latent variable ϵ' following a standard normal distribution. One can think about it as a VAE. This way, instead of having a classical Wiener measure (*i.e.* the distribution of Brownian motion) we would only need to deal with the standard Gaussian distribution, then $\mathbb{P}$ in (8.15) has density $p = \mathcal{N}(0, 1)$. Suppose $\mathbb{Q}$ also has density q Then we can rewrite (8.15) using the more familiar density ratio

$$\mathbb{E}_q \left[\log \frac{p}{q} + \cdots \,\middle|\, \cdots \right].$$

Recall $p(\epsilon') = \mathcal{N}(\epsilon'; 0, 1) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\epsilon'^2}$. If we translate this density by a and let it be q, we have

$$q(\epsilon') = \frac{1}{2\pi} e^{-\frac{1}{2}(\epsilon' - a)^2}.$$

This definition of q gives us the density ratio

$$\frac{q(\epsilon')}{p(\epsilon')} = e^{a\epsilon' - \frac{1}{2}a^2}.$$

Also, under the density q,

$$\hat{\epsilon} := \epsilon' - a.$$

is again a standard normal random variable (which means ϵ' is a Gaussian random variable with mean a). Note the striking resemblance between the last two formulas and (8.16, 8.17).

Now if we want to use this q to perform inference as well as reparameterization, we simply just invert the standardization formula, by first sampling $\hat{\epsilon}$ from the standard normal distribution, and letting $\epsilon' = \hat{\epsilon} + a$. Under this reparameterization, the log-likelihood ratio $\log p/q$ in the ELBO becomes

$$-a\epsilon' + \frac{1}{2}a^2 = -a(\hat{\epsilon} + a) + \frac{1}{2}a^2 = -a\hat{\epsilon} - \frac{1}{2}a^2.$$

Note that since $\hat{\epsilon}$ is the standard normal (under q), the first term is equal to 0 in expectation. This derivation leads to the CT-ELBO in (8.18). See Section G.3 for the formal proof.

G.3 PROOFS

Theorem 23 (Continuous-time ELBO). *Let $\mathbb{Q}$ be defined via the density (8.17). Then the RHS of (8.15) can be rewritten as*

$$\mathbb{E}\left[-\frac{1}{2}\int_0^T \|a(\omega,s)\|_2^2 \,\mathrm{d}s + \log p_0(Y_T) - \int_0^T \nabla\cdot\mu \,\mathrm{d}s \,\middle|\, Y_0 = x\right] =: \mathcal{E}^\infty, \tag{8.18}$$

where the expectation is taken wrt the Brownian motion $\hat{B}_s$, and Y_s solves[1]

$$\mathrm{d}Y = (-\mu + \sigma a)\,\mathrm{d}s + \sigma \mathrm{d}\hat{B}_s. \tag{8.19}$$

Proof. By inverting the relationship (8.16), we have

$$\mathrm{d}B'_s = \mathrm{d}\hat{B}_s + a(\omega,s)\,\mathrm{d}s.$$

This allows us to reparameterize Y_s as

$$\mathrm{d}Y = -\mu\,\mathrm{d}s + \sigma\,\mathrm{d}B'_s = -\mu\,\mathrm{d}s + \sigma(\mathrm{d}\hat{B}_s + a\,\mathrm{d}s) = (-\mu+\sigma a)\,\mathrm{d}s + \sigma\mathrm{d}\hat{B}_s.$$

The log density can be written as

$$\begin{aligned}
\log\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}} &= -\int_0^T a\cdot \mathrm{d}B'_s + \frac{1}{2}\int_0^T \|a\|^2\,\mathrm{d}s \\
&= -\int_0^T a\cdot(\mathrm{d}\hat{B}_s + a\,\mathrm{d}s) + \frac{1}{2}\int_0^T \|a\|^2\,\mathrm{d}s \\
&= -\int_0^T a\cdot \mathrm{d}\hat{B}_s - \frac{1}{2}\int_0^T \|a\|^2\,\mathrm{d}s.
\end{aligned}$$

Finally, since the first term is in expectation equal to zero [Øksendal, 2003, Theorem 3.2.1], we conclude the proof. □

[1] Note that μ and σ run backward in time from T, whereas a runs forward.

Theorem 26 (Variational gap and optimal inference SDE). *The variational gap can be written as*

$$\log p(x,T) - \mathcal{E}^{\infty} = \int_0^T \mathbb{E}\left[\left\|a(\omega,s) - \sigma^{\top}\nabla \log p(Y_s, T-s)\right\|^2\right] \mathrm{d}s. \tag{8.20}$$

In particular, $\mathcal{E}^{\infty} = \log p(x,T)$ *if and only if* $a(\omega,s)$ *can be written as* $a(\omega,s) = a(Y_s(\omega),s)$ *for almost every* $s \in [0,T]$ *and* $\omega \in \Omega$*, and* $a(y,s) = \sigma^{\top}\nabla \log p(y, T-s)$ *almost everywhere.*

Proof. To characterize the variational gap, we directly subtract the lower bound from the marginal likelihood:

$$\log p(x,T) - \mathcal{E}^{\infty} = \mathbb{E}\left[\log p(Y_0,T) - \log p(Y_T,0) + \frac{1}{2}\int_0^T \|a(\omega,s)\|_2^2 \, ds + \int_0^T \nabla\cdot\mu \, ds \,\middle|\, Y_0 = x\right].$$

The first two terms can be written as an integral

$$\log p(Y_0,T) - \log p(Y_T,0) = -\int_0^T \mathrm{d}\log p(Y_s, T-s). \tag{G.1}$$

Using Itô's formula, we can rewrite the differential as

$$\mathrm{d}\log p(Y_s,T-s) = -\frac{\partial_s p(Y_s,T-s)}{p(Y_s,T-s)}\mathrm{d}s + \nabla\log p\cdot \mathrm{d}Y_s + \frac{1}{2}H_{\log p} : \mathrm{d}Y_s \mathrm{d}Y_s^{\top},$$

where $\mathrm{d}Y_s\mathrm{d}Y_s^{\top} = \sigma\sigma^{\top}\,\mathrm{d}s$.

After rearrangement, we have

$$\int_0^T \left[\frac{\partial_s p}{p} - \nabla\log p^{\top}(-\mu+\sigma a) - \frac{1}{2}H_{\log p} : \sigma\sigma^{\top} + \frac{1}{2}\|a\|^2 + \nabla\cdot\mu\right]\mathrm{d}s - \int_0^T \nabla\log p^{\top}\sigma\,\mathrm{d}\hat{B}_s, \tag{G.2}$$

where the second term is equal to 0 in expectation.

Now using the Fokker-Planck equation to expand $\partial_s p$, with further rearrangement and cancellation and by a final application of the conditional Fubini's theorem, we end us with the desired characterization of the gap.

□

Theorem 28 (Consistency). *Assume μ, σ, σ^{-2}, a, $\|a\|^2$ and their derivatives up to the fourth order are all bounded and continuous, and that σ is non-singular. Then $\mathcal{E}^L \to \mathcal{E}^\infty$ as $L \to \infty$.*

Proof. By definition of the log transitional distributions

$$\log p(x_{i+1}|x_i) = -\frac{d}{2}\log 2\pi - \log\det(\tilde{\sigma}_i) - \frac{1}{2}\left\|x_{i+1} - \tilde{\mu}_i(x_i)\right\|^2_{\tilde{\sigma}_i^{-2}}. \tag{G.3}$$

Using the definition of $\tilde{\mu}_i$, the quadratic term becomes

$$\left\|x_{i+1} - x_i - \Delta t\mu(x_i, i\Delta t)\right\|^2_{\tilde{\sigma}_i^{-2}}.$$

Due the the Gaussian reparameterization (under q), we can write

$$\begin{aligned} x_i &= \hat{\mu}_{i+1}(x_{i+1}) + \hat{\sigma}_{i+1}\epsilon \\ &= x_{i+1} + \Delta t\big(-\mu(x_{i+1}, (i+1)\Delta t) \\ &\qquad + \sigma((i+1)\Delta t)a(x_{i+1}, T - (i+1)\Delta t)\big) + \sqrt{\Delta t}\sigma((i+1)\Delta t)\epsilon. \end{aligned} \tag{G.4}$$

Plugging this into the quadratic term yields

$$\begin{aligned} \|\cdots\|^2 = \|&\Delta t(\mu(x_{i+1}, (i+1)\Delta t) - \mu(x_i, i\Delta t)) \\ &- \Delta t\sigma((i+1)\Delta t)a(x_{i+1}, T - (i+1)\Delta t) - \sqrt{\Delta t}\sigma((i+1)\Delta t)\epsilon\|^2. \end{aligned} \tag{G.5}$$

We take care of the deviation in μ first, by taking the Taylor expansion around $(x_i, i\Delta t)$:

$$\mu(x_{i+1},(i+1)\Delta t) = \mu(x_i, i\Delta t) + \nabla\mu(x_i, i\Delta t)^\top (x_{i+1} - x_i) + \mathcal{O}(\Delta t). \tag{G.6}$$

Note that the first order term wrt the time variable is also $\mathcal{O}(\Delta t)$, so it's absorbed into the remainder. Combining the last three identities, we have

$$\begin{aligned}
&\frac{1}{2}\|x_{i+1} - \tilde{\mu}_i(x_i)\|^2_{\tilde{\sigma}_i^{-2}} \qquad (G.7)\\
&\quad = \frac{1}{2}\epsilon^\top \sigma^\top (\sigma\sigma^\top)^{-1}\sigma\epsilon + \Delta t \epsilon^\top \sigma^\top \nabla\mu^\top (\sigma\sigma^\top)^{-1}\sigma\epsilon + \frac{1}{2}\Delta t a^\top \sigma^\top (\sigma\sigma^\top)^{-1}\sigma a \\
&\qquad + o(\Delta t) + (\Delta t)^{1/2}\epsilon^\top \sigma^\top (\sigma\sigma^\top)^{-1}\sigma a.
\end{aligned}$$

Note that we've dropped the arguments of the functions for notational convenience. All the σs in the denominator are $\sigma(i\Delta t)$. The $o(\Delta t)$ term can be neglected since it decays fast enough even though there are $L = 1/\Delta t$ of them. The last term is 0 in expectation since ϵ is Gaussian distributed. To take care of the first term (*), we turn to the log density of the inference model.

$$\begin{aligned}
\log q(x_i|x_{i+1}) &= -\frac{d}{2}\log 2\pi - \log\det(\hat{\sigma}_{i+1}) - \frac{1}{2}\|x_i - \hat{\mu}_{i+1}(x_{i+1})\|^2_{\hat{\sigma}_{i+1}^{-2}} &(G.8)\\
&= -\frac{d}{2}\log 2\pi - \log\det(\hat{\sigma}_{i+1}) - \frac{1}{2}\|\hat{\sigma}_{i+1}\epsilon\|^2_{\hat{\sigma}_{i+1}^{-2}}\,. &(G.9)
\end{aligned}$$

Comparing the third term with (*), we have

$$\frac{1}{2}\epsilon^\top \sigma_{i+1}^\top \left((\sigma_{i+1}\sigma_{i+1}^\top)^{-1} - (\sigma_i\sigma_i^\top)^{-1}\right)\sigma_{i+1}\epsilon, \tag{G.10}$$

where $\sigma_i := \sigma(i\Delta)$. Using the differential notation, in expectation, the above can be rewritten as

$$\mathbb{E}\left[\frac{1}{2}\epsilon^\top \sigma^\top \left(\partial_t(\sigma\sigma^\top)^{-1}\right)\sigma\epsilon\right]\, dt = -\mathrm{Tr}(\sigma^{-1}\partial_t\sigma)\, dt = -\partial_t \log\det(\sigma)\, dt, \tag{G.11}$$

where we used Hutchinson's trace identity and Jacobi's formula. Therefore, the summation of the differences will converge to $\log\det(\sigma(0)) - \log\det(\sigma(T))$. This quantity will be negated by summing up the differences between the normalizing constants for all L terms, which gives us $\log\det(\sigma(T)) - \log\det(\sigma(0))$, by the telescoping cancellation.

Now we only have two terms from the quadratic function, which will converge to

$$\epsilon^\top \sigma^\top \nabla\mu^\top \sigma^{-\top}\epsilon\,\mathrm{d}t + \frac{1}{2}\left\|a\right\|^2\,\mathrm{d}t.$$

Using the trace identity again, and the fact that trace is similarity-invariant, we see that the above quantity is equal to

$$\left(\nabla\cdot\mu + \frac{1}{2}\left\|a\right\|^2\right)\,\mathrm{d}t,$$

in expectation. Now summing up all the layers, we can decompose the approximate error as

$$\begin{aligned}\left|\mathbb{E}\left[\sum\log\frac{p}{q}\right] - \mathbb{E}\left[-\int\left(\nabla\cdot\mu + \frac{1}{2}\left\|a\right\|^2\right)\right]\right| \leq \\ \left|\mathbb{E}\left[\sum\log\frac{p}{q} + \sum\left(\nabla\cdot\mu + \frac{1}{2}\left\|a\right\|^2\right)\Delta t\right]\right| \\ +\mathbb{E}\left[\left|\sum\left(\nabla\cdot\mu + \frac{1}{2}\left\|a\right\|^2\right)\Delta t - \int\left(\nabla\cdot\mu + \frac{1}{2}\left\|a\right\|^2\right)\right|\right].\end{aligned}$$

As all the approximation errors are bounded and converge to 0 as $L\to\infty$, the first term goes to 0 by the *Dominated Convergence Theorem*. The assumption on the coefficients also guarantees the convergence in mean square error [Milshtein, 1975] of the Euler Maruyama scheme, which implies the second term goes to 0. The same applies to the last step for the prior term: $x_0 \to y(T)$ in L^2.

□

Theorem 29 (Convergence of the OU process)**.** *If* $\lim_{T\to\infty}\int_0^T\theta(r)\,\mathrm{d}r = \infty$*, then* $q(y,T)$ *following the standard OU process converges to* $\mathcal{N}(0,I)$ *in total variation as*

$T \to \infty$.

Proof. We use a technique commonly used in the Markov chain theory known as the coupling method. Recall that Y_s solves the following SDE

$$\mathrm{d}Y_s = -\theta(s)Y_s\,\mathrm{d}s + \sqrt{2\theta(s)}\,\mathrm{d}\hat{B}_s, \tag{G.12}$$

which means

$$Y_s = \exp\left(-\int_0^s \theta(r)\,\mathrm{d}r\right)\left[Y_0 + \int_0^s \exp\left(\int_0^r \theta(r')\,\mathrm{d}r'\right)\sqrt{2\theta(r)}\mathrm{d}\hat{B}_r\right] \tag{G.13}$$

$$= \exp\left(-\int_0^s \theta(r)\,\mathrm{d}r\right)\left[Y_0 + \int_0^s \sqrt{\frac{\mathrm{d}}{\mathrm{d}r}\exp\left(\int_0^r 2\theta(r')\,\mathrm{d}r'\right)}\mathrm{d}\hat{B}_r\right]. \tag{G.14}$$

Let $\tilde{Z}_s$ be another process defined similarly to Y_s as

$$\tilde{Z}_s := \exp\left(-\int_0^s \theta(r)\,\mathrm{d}r\right)\left[\tilde{Z}_0 + \int_0^s \sqrt{\frac{\mathrm{d}}{\mathrm{d}r}\exp\left(\int_0^r 2\theta(r')\,\mathrm{d}r'\right)}\mathrm{d}\tilde{B}_r\right], \tag{G.15}$$

where $\tilde{Z}_0$ is an independent standard Gaussian random vector and $\tilde{B}_s$ is another independent Brownian motion. From (8.32) and (8.37), we know $\mathbb{E}[\tilde{Z}_s] = 0$ and $\mathrm{Var}(\tilde{Z}_s) = I$, which means $\tilde{Z}_s$ has a standard Gaussian distribution for all s. We define another process Z_s as follows. For each $i \in [d]$,

$$Z_{i,s} = \tilde{Z}_{i,s}1_{s<\tau_i} + Y_{i,s}1_{s\geq\tau_i}, \tag{G.16}$$

where τ_i is the first crossing time between $\tilde{Z}_{i,s}$ and $Y_{i,s}$, *i.e.*

$$\tau_i := \inf\{s \geq 0 : \tilde{Z}_{i,s} = Y_{i,s}\}. \tag{G.17}$$

That is, $Z_{i,s}$ is set to be equal to $\tilde{Z}_{i,s}$ before the crossing, and to $Y_{i,s}$ after. Finally, let $\tau = \max_i \tau_i$, which means $Z_s = Y_s$ for $s \geq \tau$. Note that $Z_s \overset{d}{=} \tilde{Z}_s$ for each $s \geq 0$. For

any Borel subset B of $\mathbb{R}^d$,

$$\begin{aligned}\mathbb{P}(Y_T \in B) &= \mathbb{P}(Y_T \in B, \tau \leq T) + \mathbb{P}(Y_T \in B, \tau > T) && \text{(G.18)}\\ &= \mathbb{P}(Z_T \in B, \tau \leq T) + \mathbb{P}(Y_T \in B, \tau > T) && \text{(G.19)}\\ &\leq \mathbb{P}(Z_T \in B) + \mathbb{P}(\tau > T). && \text{(G.20)}\end{aligned}$$

Likewise, $\mathbb{P}(Z_T \in B) \leq \mathbb{P}(Y_T \in B) + \mathbb{P}(\tau > T)$, which means

$$|\mathbb{P}(Z_T \in B) - \mathbb{P}(Y_T \in B)| \leq \mathbb{P}(\tau > T), \tag{G.21}$$

for any Borel $B \subseteq \mathbb{R}^d$. To show the total variation converges to 0, recall (B.1), it suffices to show the RHS converges to 0 as $T \to \infty$. By monotonicity of the probability measure, we have $\lim_{T\to\infty} \mathbb{P}(\tau > T) = \mathbb{P}(\tau = \infty)$. Let $D(T) := \int_0^T \theta(r)\,\mathrm{d}r$. Now, by definition of τ,

$$\begin{aligned}\{\tau = \infty\} &= \{\exists\, i \in [d] : \tau_i = \infty\} && \text{(G.22)}\\ &= \bigcup_{i\in[d]} \{\tau_i = \infty\} && \text{(G.23)}\\ &= \bigcup_{i\in[d]} \{\tilde{Z}_{i,s} \neq Y_{i,s}\ \forall\, s \geq 0\} && \text{(G.24)}\\ &= \bigcup_{i\in[d]} \left\{ e^{-D(s)} \left[\tilde{Z}_{i,0} - Y_{i,0} + \int_0^s \sqrt{\frac{\mathrm{d}}{\mathrm{d}r} e^{2D(r)}}\mathrm{d}\left(\tilde{B}_{i,r} - \hat{B}_{i,r}\right)\right] \neq 0\ \forall\, s \geq 0 \right\} && \text{(G.25)}\\ &= \bigcup_{i\in[d]} \left\{ \int_0^s \sqrt{\frac{\mathrm{d}}{\mathrm{d}r} e^{2D(r)}}\mathrm{d}\left(\tilde{B}_{i,r} - \hat{B}_{i,r}\right) \neq Y_{i,0} - \tilde{Z}_{i,0}\ \forall\, s \geq 0 \right\}. && \text{(G.26)}\end{aligned}$$

Since $\frac{1}{\sqrt{2}}(\tilde{B}_{i,r} - \hat{B}_{i,r})$ is also a Brownian motion, the stochastic integral in the last line is a Martingale with unbounded quadratic variation [Protter, 2005, Theorem 29]

$$\lim_{s\to\infty} 2\int_0^s \frac{\mathrm{d}}{\mathrm{d}r} e^{2D(r)} = \lim_{s\to\infty} 2e^{2D(s)} - 2 = \infty$$

by assumption. As continuous Martingales with unbounded quadratic variation can be represented as a time-changed process of some Brownian motion $\bar{B}$ [Karatzas and Shreve, 2014, Theorem 4.6, Chapter 3], we can write

$$\{\tau = \infty\} = \bigcup_{i\in[d]} \left\{ \bar{B}_{i,2e^{2D(s)}-2} \neq Y_{i,0} - \tilde{Z}_{i,0} \ \forall\, s \geq 0 \right\}, \tag{G.27}$$

which by continuity of Brownian motion and the law of the iterated logarithm [Durrett, 2019, Theorem 8.5.1] has measure zero.

□

Theorem 31 (Plug-in reverse SDE ELBO). *Assume the generative and inference SDEs follow (8.47). For $\lambda < 1$, then the CT-ELBO (denoted by $\mathcal{E}_\lambda^\infty$) can be written as*

$$\begin{aligned}\mathcal{E}_\lambda^\infty = \mathbb{E}_{Y_T}[\log p_0(Y_T) \,|\, Y_0 = x] - \int_0^T \left(1 - \frac{\lambda}{2}\right) \mathbb{E}_{Y_s}\left[\frac{1}{2}\left\|\boldsymbol{s}_\theta\right\|_{g^2}^2 + \nabla \cdot \left(g^2 \boldsymbol{s}_\theta - \left(\frac{2}{2-\lambda}\right) f\right) \,\middle|\, Y_0 = x\right] \\ + \frac{\lambda}{2}\,\mathbb{E}_{Y_s}\left[\frac{1}{2}\left\|\boldsymbol{s}_\theta\right\|_{g^2}^2 - g^2 \boldsymbol{s}_\theta^\top \nabla \log q(Y_s, s) \,\middle|\, Y_0 = x\right] \\ + \frac{\lambda^2}{4(1-\lambda)}\,\mathbb{E}_{Y_s}\left[\frac{1}{2}\left\|\boldsymbol{s}_\theta - \nabla \log q(Y_s, s)\right\|_{g^2}^2 \,\middle|\, Y_0 = x\right] \mathrm{d}s.\end{aligned}$$

Furthermore, averaging the ELBO over the data distribution yields

$$\begin{aligned}\mathbb{E}_{Y_0}[\mathcal{E}_\lambda^\infty] &= \mathbb{E}_{Y_T}[\log p_0(Y_T)] - \int_0^T \left(1 + \frac{\lambda^2}{4(1-\lambda)}\right) \mathbb{E}_{Y_s}\left[\frac{1}{2}\left\|\boldsymbol{s}_\theta\right\|_{g^2}^2 + \nabla \cdot (g^2 \boldsymbol{s}_\theta)\right] \mathrm{d}s \\ &\quad + Const. \qquad (8.48) \\ &= \mathbb{E}_{Y_0}[\mathcal{E}_0^\infty] - \left(\frac{\lambda^2}{4(1-\lambda)}\right) \int_0^T \mathbb{E}_{Y_s}\left[\frac{1}{2}\left\|\boldsymbol{s}_\theta(Y_s, s) - \nabla \log q(Y_s, s)\right\|_{g^2}^2\right] \mathrm{d}s. \qquad (8.49)\end{aligned}$$

Proof. Plugging (8.47) in (8.1) and (8.19), we get

$$\begin{aligned}\mu &= \left(1 - \frac{\lambda}{2}\right) g^2 \boldsymbol{s}_\theta - f \\ \sigma &= \sqrt{1-\lambda}\, g \\ a &= \frac{1}{\sqrt{1-\lambda}} \left[(1-\lambda)\, g \boldsymbol{s}_\theta + \frac{\lambda}{2} g \left(\boldsymbol{s}_\theta - \nabla \log q\right)\right].\end{aligned}$$

Then we have

$$\begin{aligned}\frac{1}{2}\left\|a\right\|_2^2 &= \frac{1}{2(1-\lambda)} \left[(1-\lambda)^2 \left\|\boldsymbol{s}_\theta\right\|_{g^2}^2 + (1-\lambda)\,\lambda g^2 \boldsymbol{s}_\theta^\top (\boldsymbol{s}_\theta - \nabla \log q) + \frac{\lambda^2}{4} \left\|\boldsymbol{s}_\theta - \nabla \log q\right\|_{g^2}^2\right] \\ &= \left(1 - \frac{\lambda}{2}\right) \frac{1}{2} \left\|\boldsymbol{s}_\theta\right\|_{g^2}^2 + \frac{\lambda}{2} \left(\frac{1}{2} \left\|\boldsymbol{s}_\theta\right\|_{g^2}^2 - g^2 \boldsymbol{s}_\theta^\top \nabla \log q\right) + \frac{\lambda^2}{4(1-\lambda)} \frac{1}{2} \left\|\boldsymbol{s}_\theta - \nabla \log q\right\|_{g^2}^2\end{aligned}$$

$$\nabla \cdot \mu = \left(1 - \frac{\lambda}{2}\right) \nabla \cdot \left(g^2 \boldsymbol{s}_\theta - \left(\frac{2}{2-\lambda}\right) f\right).$$

Summing up these two parts gives us $\mathcal{E}_\lambda^\infty$. Under the expectation, we can rewrite $\mathbb{E}_{\boldsymbol{y}_s}[g^2 \boldsymbol{s}_\theta^\top \nabla \log q] = -\mathbb{E}_{\boldsymbol{y}_s}[\nabla \cdot (g^2 \boldsymbol{s}_\theta)]$ using the score matching loss identity (see Section A.1), to obtain the second part of the statement. □