

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Institut National Polytechnique de Toulouse (Toulouse INP)

Discipline ou spécialité :

Informatique et Télécommunication

Présentée et soutenue par :

M. FLORENTIN COEURDOUX

le vendredi 10 novembre 2023

Titre :

Echantillonnage Monte Carlo et modèles génératifs profonds pour inférence Bayésienne

---

Ecole doctorale :

Mathématiques, Informatique, Télécommunications de Toulouse (MITT)

Unité de recherche :

Institut de Recherche en Informatique de Toulouse (IRIT)

Directeur(s) de Thèse :

M. NICOLAS DOBIGEON

M. PIERRE CHAINAIS

Rapporteurs :

MME JULIE DELON, UNIVERSITE DE PARIS

M. NICOLAS COURTY, UNIVERSITE DE BRETAGNE SUD

Membre(s) du jury :

M. BRUNO GALERNE, UNIVERSITE D'ORLEANS, Président

MME EMILIE CHOUZENOUX, INRIA, Membre

M. NICOLAS DOBIGEON, TOULOUSE INP, Membre

M. PIERRE CHAINAIS, ECOLE CENTRALE DE LILLE, Membre

Abstract

The generation of realistic data with complex patterns often relies on the use of expressive probabilistic models. The goal of these models is to characterize and approximate data distributions in high-dimensional spaces. However, even with the current power of deep neural networks, constructing robust probabilistic models remains a significant challenge. While obtaining quality samples is a tough task, many scientific applications further require the ability to explicitly estimate the probability density learned by the model. This added requirement necessitates adopting constrained model architectures, a limitation that becomes quickly restrictive in a complex and high-dimensional space. This thesis proposes to address these challenges by adapting training protocols or sampling strategies of generative model architectures. Moreover, it explores several applications of deep generative models to solve practical problems.

In the first part, this work proposes a hybrid training strategy that strikes a balance between generating realistic samples and the ability to estimate the learned probability density of the model. This strategy involves enriching the usual training function with an approximate transport distance. This distance is also utilized in a second contribution aiming to solve the optimal transport problem between empirical distributions through the specific training of invertible generative models. Next, a sampling technique exploring latent space is introduced. Based on Langevin diffusion, it overcomes the limitations observed when using deep generative models to generate data with a multimodal distribution defined in disconnected spaces. Finally, a significant portion of this thesis work focuses on using these deep generative models to learn prior laws. These can then be easily employed within a Bayesian inversion framework in computational imaging, ensuring efficient sampling of the underlying posterior laws.

Overall, deep generative models offer a flexible, powerful, and versatile solution for data generation and density estimation. They encompass not only the field of machine learning but also find applications in various scientific disciplines and engineering techniques.

Résumé

La génération de données réalistes aux motifs complexes repose fréquemment sur l'utilisation de modèles probabilistes expressifs. Ces modèles ont pour objectif de caractériser et d'approcher les distributions de données dans des espaces en grandes dimensions. Cependant, même avec la puissance actuelle des réseaux de neurones profonds, la construction de modèles probabilistes robustes demeure un défi de taille. Alors que l'obtention d'échantillons de qualité est donc une tâche ardue, de nombreuses applications scientifiques exigent de surcroit la capacité à estimer explicitement la densité de probabilité apprise par le modèle. Cette thèse propose de relever ces défis en adaptant les protocoles d'entraînement ou les stratégies d'échantillonnage des architectures de modèles génératifs. Par ailleurs, elle explore plusieurs applications des modèles génératifs profonds dans le but de résoudre des problèmes pratiques.

Tout d'abord, ce travail propose une stratégie d'entraînement hybride qui établit un compromis entre la génération d'échantillons réalistes et la capacité à estimer la densité de probabilité apprise par le modèle. Cette stratégie consiste à enrichir la fonction d'entraînement usuelle par une distance de transport. Cette distance est également mise à profit dans une deuxième contribution qui vise à résoudre le problème de transport optimal entre distributions empiriques grâce à l'entraînement spécifique de modèles génératifs inversibles. Ensuite, une technique d'échantillonnage explorant l'espace latent est proposée. S'appuyant sur une diffusion de Langevin, elle permet de s'affranchir des limitations des modèles génératifs profonds pour générer des données dont la distribution est multimodale à support disjoint. Enfin, une dernière partie est consacrée à l'utilisation des modèles génératifs profonds pour l'inférence Bayésienne, assurant un échantillonnage efficace des lois a posteriori.

Dans leur ensemble, les modèles génératifs profonds offrent une solution souple et puissante pour la génération de données et l'estimation de densité, englobant non seulement le domaine de l'apprentissage automatique, mais trouvant également des applications dans diverses disciplines scientifiques.

Remerciements

En regardant en arrière sur le parcours de cette thèse, je suis profondément reconnaissant pour l'accompagnement, les conseils et le soutien de nombreuses personnes exceptionnelles. J'espère par ces quelques lignes parvenir à exprimer amitié et gratitude à ceux qui ont partagé ces trois années avec moi et contribué à la réussite de cette thèse.

Je tiens à exprimer ma plus profonde gratitude à mes directeurs de thèse, Pierre et Nicolas, pour leur confiance, leur encadrement, et leur soutien indéfectible tout au long de ce parcours. L'ampleur des connaissances techniques et des leçons de vie que j'ai acquises sous votre tutelle est inestimable. Les moments difficiles ont été nombreux, mais votre présence et vos conseils avisés m'ont toujours permis de les surmonter. C'est avec une grande émotion et une reconnaissance infinie que je quitte votre équipe.

Je souhaite exprimer ma profonde gratitude envers les membres de mon jury de thèse pour leur investissement précieux et leurs critiques constructives. Un grand merci à Julie Delon et Nicolas Courty pour avoir accepté le rôle de rapporteurs de cette thèse, et mes sincères remerciements à Émilie Chouzenoux et Bruno Galerne pour avoir été examinateurs. Votre volonté d'évaluer mes travaux est un honneur pour moi, et je suis très reconnaissant pour le temps et l'attention que vous y avez consacrés.

J'aimerais exprimer ma profonde gratitude à l'équipe SC, à laquelle j'ai eu l'honneur d'appartenir. Un merci tout particulier à Elsa, Henrique, Marie, Emmanuel et Cédric pour nos conversations du midi qui ont eu le pouvoir de rendre mes brocolis bien plus agréables. Je suis également reconnaissant envers les doctorants et postdoctorants qui ont marqué mon parcours : Camille, Pierre-Hugo, Baha, Arthur, Claire et Paul, ainsi qu'à l'équipe des stagiaires, Jean et Marien. Mes remerciements s'étendent aussi à mes contemporains doctorants et ATER : Léo, Hugues, Paul et M'hamed. J'aimerais également grandement remercier Vanessa pour son aide précieuse.

Durant cette thèse, j'ai eu l'opportunité enrichissante de passer quelques mois au Department of Statistics de l'Université d'Oxford. Je tiens à adresser un grand merci à Arnaud Doucet pour son accueil chaleureux et son soutien. Mes remerciements s'étendent également aux doctorants, Freddie et Branson, qui ont grandement contribué à rendre mon expérience à Oxford à la fois mémorable et formatrice.

Merci également aux collègues lillois pour leur gentillesse et leur énergie : Pierre, Markus, Rony, Maxime, Etienne, Clemence, Barbara, Pierre-Antoine. Un grand merci aussi à toute l'équipe de l'école d'été de Peyresq 2022 pour cette semaine incroyable.

À toute la Guilde Bretilienne, ma deuxième famille : Josselin, Brice, Jean-Noël, Lucas, Germain, Joé, Pierre-Eliott, Émilie, Tanguy et Youn. Un grand merci à mon frère poète, Aymeric.

Avant de conclure, j'aimerais exprimer ma gratitude spéciale à mes parents, qui m'ont toujours soutenu.

Contents

1 Introduction **13**
- 1.1 Statistical inference . . . 14
  - 1.1.1 Bayesian estimation . . . 14
  - 1.1.2 Uncertainty . . . 14
  - 1.1.3 Toward learning models . . . 15
- 1.2 Sampling methods . . . 16
  - 1.2.1 Monte Carlo integration . . . 16
  - 1.2.2 Independent sampling . . . 18
  - 1.2.3 Markov chain Monte Carlo methods . . . 20
- 1.3 Generative models for high dimensional distribution . . . 25
  - 1.3.1 Learning distributions . . . 25
  - 1.3.2 Statistical foundations of deep generative models . . . 26
  - 1.3.3 Training deep generative models . . . 28
- 1.4 Deep generative model architectures . . . 31
  - 1.4.1 Normalizing Flows . . . 32
  - 1.4.2 Denoising diffusion probabilistic modeling . . . 35
- 1.5 Remaining challenges . . . 37
- 1.6 Contributions of the manuscript . . . 38

2 Sliced-Wasserstein normalizing flows: beyond maximum likelihood training **41**
- 2.1 Conventional training of normalizing flows . . . 42
  - 2.1.1 Learning a change of variable . . . 42
  - 2.1.2 Empirical loss function . . . 43
- 2.2 Problem statement . . . 44
  - 2.2.1 Limitations of MLE training . . . 45
  - 2.2.2 Limitations of GANs . . . 45
  - 2.2.3 Towards a hybrid loss function . . . 46
- 2.3 Sliced-Wasserstein distance . . . 46
  - 2.3.1 Vector space Sliced-Wasserstein . . . 46
  - 2.3.2 Convolution Sliced-Wasserstein . . . 48

2.4 Hybrid objective function 49
2.4.1 Logit space for images 50
2.5 Numerical experiments 52
2.5.1 Implementation details 52
2.5.2 Goodness-of-fit 54
2.5.3 Out-of-distribution detection 55
2.6 Conclusion 56

3 Normalizing flows to learn optimal transport between empirical distributions 57
3.1 Relaxation of the optimal transport problem 58
3.1.1 Background on optimal transport 59
3.1.2 Proposed relaxation of OT 60
3.1.3 Discrete formulation 61
3.2 Normalizing flows to approximate OT 62
3.2.1 Loss function 62
3.2.2 Intermediate transports 63
3.3 Numerical experiments 64
3.3.1 Toy examples 64
3.3.2 Unsupervised word translation 68
3.4 Discussion 72
3.5 Conclusion 73

4 Normalizing flow sampling with Langevin dynamics in the latent space 75
4.1 Problem statement 77
4.1.1 Learning of change of variable 78
4.1.2 A Gaussian latent space? 79
4.1.3 Beyond conventional NF sampling 79
4.2 Related works 80
4.3 Implications of a topological mismatch 81
4.3.1 Topology preservation 81
4.3.2 Partition of the latent space 83
4.4 NF sampling in the latent space 84
4.4.1 Difficulties of sampling the target space 84
4.4.2 Latent diffusion equation 85
4.5 Efficient implementation 89
4.5.1 Fast approximation of the proposal move 90
4.5.2 Fast computation of the latent score 90
4.6 NF-SAILS 91
4.6.1 Local exploration with MALA 91

4.6.2 Global exploration with I-MH 92
4.6.3 Local/Global exploration 92
4.7 Experiments 93
4.7.1 Figures-of-merit 94
4.7.2 Results obtained on synthetic dataset 94
4.7.3 Results obtained on real image datasets 96
4.8 Conclusion 97

5 Plug-and-Play split Gibbs sampler 99
5.1 Solving inverse problems: state of the art 101
5.1.1 Problem statement 101
5.1.2 A brief history 101
5.2 SGS and generative models for PnP 104
5.2.1 Split Gibbs sampling (SGS) 104
5.2.2 Denoising diffusion probabilistic models (DDPM) . . . 105
5.2.3 Proposed DDPM-based PnP-SGS algorithm 107
5.2.4 Some insights into the number t^* of steps 108
5.3 Application to Bayesian inverse problems 109
5.3.1 Inverse problems 109
5.3.2 Image deblurring 111
5.3.3 Image inpainting 112
5.3.4 Image super-resolution 112
5.4 Experiments 113
5.4.1 Experimental setup 113
5.4.2 Compared methods & figures-of-merit 114
5.4.3 Technical details of PnP-SGS 117
5.4.4 Experimental results 118
5.5 Conclusion 122

6 Conclusion 125

Appendices

A Langevin dynamics in the latent space 151

B From Normalizing Flow to DDPM, the continuous bridge 155

Chapter 1

Introduction

Contents

1.1 Statistical inference 14
1.1.1 Bayesian estimation 14
1.1.2 Uncertainty 14
1.1.3 Toward learning models 15
1.2 Sampling methods 16
1.2.1 Monte Carlo integration 16
1.2.2 Independent sampling 18
1.2.3 Markov chain Monte Carlo methods 20
1.3 Generative models for high dimensional distribution 25
1.3.1 Learning distributions 25
1.3.2 Statistical foundations of deep generative models . 26
1.3.3 Training deep generative models 28
1.4 Deep generative model architectures 31
1.4.1 Normalizing Flows 32
1.4.2 Denoising diffusion probabilistic modeling 35
1.5 Remaining challenges 37
1.6 Contributions of the manuscript 38

In the fields of machine learning and signal/image processing, a multitude of challenges deal with solving complex statistical inference problems. These challenges resonate with various scientific domains ranging from medical imaging [1] to astronomy [2]. Advances in the field hold the promise

of significantly enhancing the efficiency of solutions for critical inverse problems. In the realm of geophysics, such advances have the potential to tangibly reduce the expenses associated with valuable mineral prospecting [3]. In medical imaging, they pave the way for more cost-effective and expeditious diagnostics, steering us towards a preventive approach to addressing illnesses [4, 5]. Furthermore, these developments contribute to a deeper comprehension of climate and ocean dynamics through the refinement of remote sensing and radar signal reconstruction [6, 7]. They also play a pivotal role in advancing our understanding of the universe laws by leveraging the latest advancements in telescopes [8, 9]. Within this framework, the first section of this chapter recalls that addressing these scientific challenges can be formulated as a statistical inference task.

1.1 Statistical inference

1.1.1 Bayesian estimation

For a large part of the encountered scientific problems mentioned above, the object $\mathbf{x}$ to be inferred typically represents the solution of a variational or stochastic optimization problem. Within a Bayesian framework, this solution $\hat{\mathbf{x}}$ minimizes a cost function known as the posterior loss, defined as:

$$\hat{\mathbf{x}} \in \underset{\mathbf{x}}{\operatorname{argmin}}\ \mathbb{E}[L(\delta, \mathbf{x}) \mid \mathbf{y}] \quad \text{with} \quad \mathbb{E}[L(\delta, \mathbf{x}) \mid \mathbf{y}] = \int L(\delta, \mathbf{x}) p(\mathbf{x} \mid \mathbf{y}) d\mathbf{x}$$

Here, $\mathbf{y}$ represents the available data, modeled as the realization of a random variable characterized by the likelihood function $p(\mathbf{y} \mid \mathbf{x})$. The posterior distribution $p(\mathbf{x} \mid \mathbf{y})$ relates to the likelihood function $p(\mathbf{y} \mid \mathbf{x})$ and prior distribution $p(\mathbf{y})$ through Bayes' formula. Additionally, $L(\cdot, \cdot)$ is a specified loss function. When this loss function takes the form of a quadratic function, i.e., $L(\mathbf{y}, \mathbf{z}) \triangleq \|\mathbf{y} - \mathbf{z}\|_2^2$, the Bayesian estimator $\hat{\mathbf{x}}$ is recognized as the posterior mean $\hat{\mathbf{x}}_{\text{MMSE}} = \mathbb{E}[\mathbf{x} \mid \mathbf{y}]$, minimizing the mean square error (MSE) [10].

1.1.2 Uncertainty

In addition to point-wise Bayesian estimators, the Bayesian framework also enables the derivation of valuable credibility intervals. These credibility intervals serve as vital tools for assessing the uncertainty surrounding the estimation of unknown parameters. Such credibility information proves especially crucial when there is no available ground truth for the parameters under inference. Similarly to the MMSE estimator, these intervals are expressed as

integrals and write $\int_{\mathcal{C}_\alpha} p(\mathbf{x} \mid \mathbf{y})\mathrm{d}\mathbf{x}$ where $\mathcal{C}_\alpha$ is an $(1-\alpha)$ credibility region such that $\mathbb{P}_p(\mathbf{x} \in \mathcal{C}_\alpha) = 1-\alpha$, with $\alpha \in (0,1)$.

In practice, computing such integrals is often complex. A practical alternative is to employ Monte Carlo integration, which approximates any expectations of the form:

$$\mathbb{E}[h(Z) \mid Z \sim p(\mathbf{z})] = \int f(\mathbf{z})p(\mathbf{z})d\mathbf{z}$$

by empirical averaging:

$$\bar{f} = \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{z}_i).$$

Here, $f(\cdot)$ is an arbitrary function, $\{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$ is a sample drawn from the distribution $p(\mathbf{z})$ [11]. In Bayesian inference, this distribution $p(\mathbf{z})$ corresponds to the target posterior distribution $p(\mathbf{x} \mid \mathbf{y})$. Monte Carlo integration necessitates efficient algorithms for generating samples from the desired distribution. A substantial body of literature dedicated to random variable generation has proposed various Monte Carlo algorithms [11]. For instance, Markov chain Monte Carlo (MCMC) methods, including well-known techniques like the Gibbs sampler and Metropolis-Hastings algorithm, define a broad class of algorithms capable of generating a Markov chain $\{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$ with a stationary distribution of interest [12]. However, despite their apparent simplicity and generality, MCMC algorithms may prove computationally inefficient for large-scale or highly structured problems. Particularly when dealing with high-dimensional spaces or highly nonlinear and non-standard distributions. Popular Monte Carlo methods will be discussed further in Section 1.2.

1.1.3 Toward learning models

Fundamental challenges in statistical inference lie in the potential misalignment or underspecification of the probabilistic model describing the dataset and its relationship to the parameters of interest. This misalignment can significantly impact the performance of various statistical inference techniques. Indeed, both the likelihood and prior distributions play pivotal roles in the foundation of these methods, such as the Bayesian framework discussed above. However, it is worth noting that, at the same time, the machine learning community has made substantial progress in recent years in the quest of learning complex high dimensional distributions. This progress offers a potential solution to the intricate task of describing the underlying probabilistic model.

Consequently, this thesis embarks on an innovative journey, proposing a synergy between machine learning and statistical inference. In particular, it explores the integration of deep generative models to hybridize existing inference methods. Achieving this fusion necessitates the training of precise models tailored to the desired probability distributions and the development of efficient sampling strategies to enable the seamless integration of these models with existing methodologies.

This chapter aims at providing the reader key elements of the tools mobilized by this thesis. Section 1.2 discussed existing Monte Carlo method to sample from a targeted distribution. Section 1.3.2 will describes the underlying mechanism behind statistical modelling and learning a distribution from data. Finally, Section 1.4 will present in detail two deep generative architectures that will be later used in the manuscript.

1.2 Sampling methods

The achievement of statistical inference objectives often relies on the utilization of various essential techniques. They involve the drawing of samples from probability distributions, which are then employed to generate Monte Carlo estimates for desired quantities. One key advantage of sampling is its ability to provide cost-effective approximations for a wide range of sums and integrals especially expectations. Moreover, sampling can be the primary objective itself, particularly when training a model to generate samples from the target distribution, e.g, a posterior distribution. Samples also unlocks to capacity to approximate uncertainty quantification.

1.2.1 Monte Carlo integration

In cases where the exact computation of a sum or integral is not feasible, an alternative approach is to approximate it using Monte Carlo integration. The fundamental concept behind this method is to treat the sum or integral as an expectation under a specific distribution, and then approximate this expectation by computing the corresponding empirical average. Let

$$\mathbf{I} = \int f(\mathbf{x})p(\mathbf{x})d\mathbf{x} = \mathbb{E}_p[f(\mathbf{x})] \tag{1.1}$$

be the integral to estimate, rewritten as an expectation of $f(.)$ over $p(x)$.

To approximate $\mathbf{I}$, one can use a method that involves selecting n samples, labeled as $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}$, from the distribution p. Computing the empirical

average as the sum of function evaluations over the sampled values, gives

$$\hat{\mathbf{I}}_n = \frac{1}{n} \sum_{i=1}^{n} f\left(\mathbf{x}^{(i)}\right). \tag{1.2}$$

This approximation is justified by a few different properties. The first trivial observation is that the estimator $\hat{\mathbf{I}}$ is unbiased, since

$$\mathbb{E}\left[\hat{\mathbf{I}}_n\right] = \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}\left[f\left(\mathbf{x}^{(i)}\right)\right] = \frac{1}{n} \sum_{i=1}^{n} \mathbf{I} = \mathbf{I} \tag{1.3}$$

Moreover, according to the law of large numbers, if the samples $\mathbf{x}^{(i)}$ are independent and identically distributed (i.i.d.), the empirical average converges almost surely to the expected value:

$$\lim_{n \to \infty} \hat{\mathbf{I}}_n = \mathbf{I} \tag{1.4}$$

Assuming that the variance of the individual terms, denoted as $\mathrm{Var}\left[f\left(\mathbf{x}^{(i)}\right)\right]$, remains bounded, an analysis of the variance of $\hat{\mathbf{I}}_n$ as the value of n increases reveals that, with growing n, the variance $\mathrm{Var}\left[\hat{\mathbf{I}}_n\right]$ steadily decreases and converges to 0, provided that $\mathrm{Var}\left[f\left(\mathbf{x}^{(i)}\right)\right] < \infty$

$$\begin{aligned} \mathrm{Var}\left[\hat{\mathbf{I}}_n\right] &= \frac{1}{n^2} \sum_{i=1}^{n} \mathrm{Var}[f(\mathbf{x})] \\ &= \frac{\mathrm{Var}[f(\mathbf{x})]}{n}. \end{aligned} \tag{1.5}$$

This valuable outcome also provides insights into estimating the uncertainty associated with a Monte Carlo average, or equivalently, quantifying the expected error of the Monte Carlo approximation. To achieve this, one computes both the empirical average of $f\left(\mathbf{x}^{(i)}\right)$ and its empirical variance. Subsequently, the estimated variance is divided by the number of samples n to obtain an estimator for $\mathrm{Var}\left[\hat{\mathbf{I}}_n\right]$. According to the central limit theorem, the distribution of the average $\hat{\mathbf{I}}_n$ converges to a normal distribution with mean $\mathbf{I}$ and variance $\frac{\mathrm{Var}[f(\mathbf{x})]}{n}$. Consequently, confidence intervals around the estimate $\hat{\mathbf{I}}_n$ can be estimated using the cumulative distribution of the normal density.

However, the aforementioned approaches heavily rely on the feasibility of easily sampling from the base distribution $p(\mathbf{x})$. Unfortunately, sampling from p may not always be possible.

1.2.2 Independent sampling

This section 1.2.2 considers some simple strategies to generate random samples from a given distribution. The process of generating such numbers involves several nuances, which are thoroughly addressed and examined in details in [11]. Because the samples will be generated by a computer algorithm they will in fact be pseudo-random numbers, that is, they will be deterministically calculated, but must nevertheless pass appropriate tests for randomness. Here one shall assume that an algorithm has been provided that generates pseudo-random numbers distributed uniformly over $(0, 1)$, and indeed most software environments have such a facility built in.

Inverse transform sampling

As a starting point, one may consider a basic Monte Carlo sampler employing a uniform distribution and a change of variable. Let $F(x)$ be a cumulative density function (CDF) and U be a uniform random variable on $[0, 1]$, then then the random variable $F^{-1}(U)$ is distributed according to F:

$$\begin{aligned} \Pr\left(F^{-1}(U) \leq x\right) &= \Pr(U \leq F(x)) \\ &= F(x). \end{aligned} \tag{1.6}$$

An important practical consequence of this observation is a process for sampling from an arbitrary CDF: first draw a variable U uniformly distributed over $(0, 1)$ and then apply F^{-1} to get a sample x distributed according to a distribution which F as a CDF.

Rejection sampling

Rejection sampling consists in sampling x_0 from a proposal distribution $q(x)$ such that $kq(x) \geq \tilde{p}(x)$ where k is some constant and $\tilde{p}(x)$ is the unnormalized version of $p(x)$. Subsequently, u_0 is sampled from the uniform distribution $U(0, kq(x_0))$. In this way, (x_0, u_0) is drawn uniformly under the graph of $kq(x)$. The sample x_0 is accepted if $u_0 \leq \tilde{p}(x_0)$.

Importance sampling

Unlike other methods discussed above, importance sampling is not a method to sample from p. Rather, it is a method to approximate the integral (1.1). As with rejection sampling, importance sampling takes an easy-to-sample distribution $q(x)$. The expectation of any function can be expressed as an

expectation over $q(x)$ of a ratio multiplied by the function of interest. The idea of importance sampling is illustrated by the following approximation

$$\int f(x)p(x)dx = \int f(x)\frac{p(x)}{q(x)}q(x)dx \approx \frac{1}{n}\sum_{i=1}^{n}\frac{p\left(x^{(i)}\right)}{q\left(x^{(i)}\right)}f\left(x^{(i)}\right), \quad x^{(i)} \sim q(x). \tag{1.7}$$

Note that this technique does not requires the evaluation of the normalizing constant of neither $p(x)$ and $q(x)$.

Ancestral sampling

When sampling from a joint distribution defined by several related variables, a common strategy consists in benefiting from the conditioning relations and performing ancestral sampling. This is the case, for instance, when handling a Bayesian model (or Bayesian network) described by a directed acyclic graph, as represented in Fig.1.1. It commences with the sampling of top-level variables from their marginal distributions, followed by the sampling of other nodes conditioned on their parent nodes' samples. For instance, consider sampling from the following distribution:

$$p(A, B, C, D) = P(A)P(B)P(C \mid A, B)P(D \mid B, C). \tag{1.8}$$

The following procedure can be implemented

$$A \sim P(A), \quad B \sim P(B), \quad C \sim P(C \mid A, B), \quad D \sim P(D \mid B, C). \tag{1.9}$$

Nonetheless, ancestral sampling may encounter limitations when the distribution cannot be decomposed into straightforward conditional distributions. This complexity arises, for instance, in undirected graphical models where computing the partition function is challenging. Another scenario is evident in a conditional distribution like $P(A, B, C \mid D)$ from the above Bayesian network, where the marginal distribution $P(D)$ might remain unknown, or if one of the intermediate step is hard to sample from.

Limitations of simple Monte Carlo Methods

Rejection sampling and importance sampling may not work well in high dimensions. Here is an example. Let $p(x)$ and $q(x)$ be the probability density function of $N(0, I)$ and $N\left(0, \sigma^2 I\right)$, respectively with $x \in \mathcal{R}^D$. Then the acceptance rate of rejection sampling is

$$\int \frac{p(x)}{kq(x)}q(x)dx = \frac{1}{k} \tag{1.10}$$

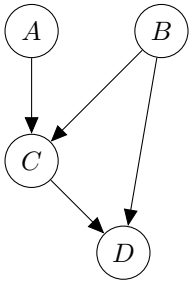


Figure 1.1: DAGs associated with the proposed hierarchical model.

where $k = \sigma^D$ (recall the density at origin for normal distributions with mean zero). This means that the acceptance rate will be very low when the dimensionality D becomes large. For importance sampling, one can show that the variance of the importance weight is $\left(\frac{\sigma^2}{2-1/\sigma^2}\right)^{D/2} - 1$, which also becomes large as D increases. In general, for $kq(x) \geq p(x)$ to hold, the ratio of the volume of $p(x)$ to the volume outside $p(x)$ tends to zero as D increases, so it is very inefficient to use the proposal distribution to sample from $p(x)$.

1.2.3 Markov chain Monte Carlo methods

When independent samples from the distribution $p(x)$ cannot be drawn, Monte Carlo techniques can still be employed using Markov chain Monte Carlo (MCMC) methods. Markov chain Monte Carlo methods draw these samples by running a cleverly constructed Markov chain for a long time [11]. The objective is to employ Markov chains for sampling from a specified distribution. This can be achieved by establishing a Markov chain where the desired distribution remains invariant.

Principle

MCMC methods mainly consist in performing stochastic updates iteratively until the state of the chain begins to generate samples from the equilibrium distribution. The specification of the Markov chain involves providing the probability distribution for the initial variable $p(x^{(0)})$ and specifying the conditional probabilities for subsequent variables through transition probabilities

$$T(x^{(i)}, x^{(i+1)}) = p(x^{(i+1)} \mid x^{(i)}). \tag{1.11}$$

A Markov chain is called homogeneous if the transition probabilities are the same for all i. Additionally, it is essential to ensure that as $i \to \infty$, the distribution $p(x^{(i)})$ converges to the desired invariant distribution $p(x)$, regardless

of the initial distribution choice $p(x^{(0)})$. This property is called ergodicity, and the invariant distribution is then called the equilibrium distribution and is unique. A sufficient condition to ensure that the required distribution $p(x)$ is invariant is to choose the transition probabilities to satisfy the property of detailed balance, defined by

$$p(x)T\left(x, x'\right) = p\left(x'\right)T\left(x', x\right). \tag{1.12}$$

Intuitively, the reversibility constraint means that the proposed kernel is direction and time-invariant. This property ensures that no matter the starting point, asymptotically the samples will be drawn by $p(x)$. It can be shown that a homogeneous Markov chain will be ergodic, subject only to weak restrictions on the invariant distribution and the transition probabilities [13].

In the subsequent subsections, three types of MCMC algorithms will be discussed: the Metropolis-Hastings algorithm, along with the Langevin and Gibbs sampling algorithms. These three algorithms are among the most popular and will be later used in Chapters 4 and 5.

Metropolis-Hastings algorithm

The most famous MCMC algorithm is by far the Metropolis-Hastings algorithm [14]. Broadly, the Metropolis-Hastings algorithm formulates a Markov process by deriving transition probabilities from the proposal density $q\left(x' \mid x\right)$. and will choose to accept or reject the proposed candidates according to the probability :

$$\alpha(x', x) = \min\left(1, \frac{p\left(x'\right)q\left(x \mid x'\right)}{p(x)p\left(x' \mid x\right)}\right). \tag{1.13}$$

The algorithm is outlined in Algo.1. Two conditions are needed for the Metropolis-Hastings algorithm to work: (i) the resulting forms a Markov Chain characterized by a unique stationary distribution, (ii) the stationary distribution needs to be equal to the targeted distribution. Condition (i) holds if the Markov chain is irreducible, aperiodic and not transient. The latter two conditions (aperiodic and not transient) hold for a random walk on any proper distribution, and irreducibility holds if the random walk has positive probability of eventually reaching any state from any other state. This holds for all proposal densities that could be utilized. Next, the rationale for equation (1.13) is elucidated through the proof of the detailed balance

Algorithm 1: Metropolis-Hastings

Input: Initial state $x^{(0)}$, size of the chain N_{MC}, a transition kernel $q(\cdot \mid \cdot)$ and a target density $p(\cdot)$

for $i \leftarrow 1$ **to** N_{MC} **do**
  Draw $x' \sim q\left(x' \mid x^{(i)}\right)$
  Draw $u \sim \mathcal{U}(0,1)$
  if $u < \alpha(x', x^{(i)})$ **then**
    Set $x^{(i+1)} = x'$
  else
    Set $x^{(i+1)} = x^{(i)}$
  end
end

Output: Collection of samples $\left\{x^{(1)} \cdots x^{(N_{\mathrm{MC}})}\right\}$ drawn approximately from $p(\cdot)$.

condition as outlined below.

$$\begin{aligned} p(x)T(x',x) &= p(x)q(x' \mid x)\min\left(1, \frac{p(x')\,q(x \mid x')}{p(x)q(x' \mid x)}\right) \\ &= \min\left(p(x)q(x' \mid x), p(x')\,q(x \mid x')\right) \\ &= p(x')\,q(x \mid x')\min\left(1, \frac{p(x)q(x' \mid x)}{p(x')\,q(x \mid x')}\right) \\ &= p(x')\,T(x,x'). \end{aligned} \tag{1.14}$$

The choice of $q(x \mid x')$ should satisfy additional technical criteria. Typical proposals employ $q(x' \mid x) = \mathcal{N}(x, \sigma^2)$ and require a suitable variance (σ^2) : large σ^2 leads to an excessive number of rejections while a low σ^2 results in slow diffusion (limited exploration). In practical terms, an acceptance rate ranging from 40% to 70% can serve as an indicator of an appropriate step size, as noted in [11]. This aspect represents a limitation of the Metropolis-Hastings (MH) algorithm. On the one hand, with a simple proposal such as the Gaussian one considered above, it may encounter major difficulties to sample from multi-modal distributions. On the other hand, the benefit of MH is that it is simple to implement and it is reasonable for sampling from correlated high dimensional distributions. An example of implemented results is illustrated in Fig.1.2. Visual inspection of the random walk algorithm aimed at approximating a mixture of Gaussians, employing a Normal proposal distribution $\mathcal{N}(0, \delta^2)$. The figures from left to right correspond to different values of δ: $\delta = 0.01$, and $\delta = 1$. When δ is exceedingly small,

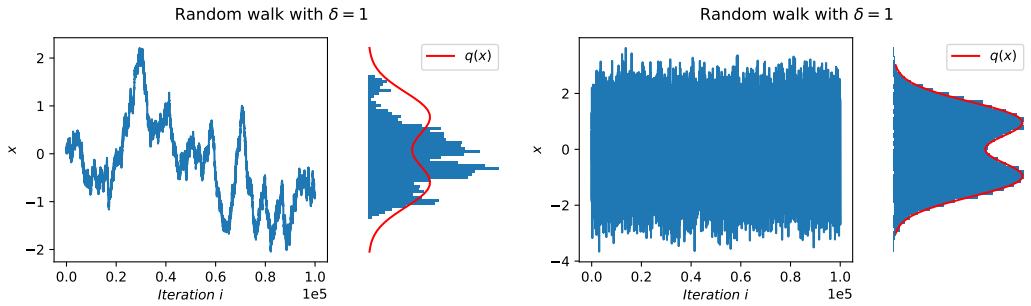


Figure 1.2: Example of Metropolis-Hastings algorithm. From left to right correspond to different values of δ: $\delta = 0.01$, and $\delta = 1$. Where the target density is $q(x) = \frac{1}{2}(\mathcal{N}(-1, 0.7) + \mathcal{N}(1, 0.7))$.

as in the case of $\delta = 0.01$, the random-walk algorithm generates samples with high correlation and faces challenges in efficiently exploring the parameter space. Conversely, setting $\delta = 1$ results in improved convergence and enhanced mixing properties of the Markov chain.

Langevin algorithms

Alternatives to the Metropolis-Hastings algorithm can be derived from the diffusion theory as proposed in [15]. The fundamental concept is to derive a diffusion equation (or a stochastic differential equation) that yields a continuous-time diffusion process with stationary distribution $p(x)$. Subsequently, the process is discretized to implement the method. More specifically, the Langevin diffusion X_t is defined by the stochastic differential equation

$$dX_t = \frac{1}{2}\nabla \log p\left(X_t\right) dt + dB_t \tag{1.15}$$

where B_t is the standard Brownian motion. Under appropriate assumptions on $p(x)$, it can be shown that the dynamic generated by (1.15) is ergodic with unique invariant distribution $p(x)$ [16]. This is a key property of (1.15) and taking advantage of it permits to sample from the invariant distribution $p(x)$. In particular, if one could solve (1.15) analytically and take time t to infinity then it would be possible to generate samples from $p(x)$. Nonetheless, the presence of an analytical formula exists in only a limited number of instances, as indicated in [17]. The actual implementation of the diffusion algorithm involves an Euler-Maruyama discretization step where (1.15) is replaced by a random walk like transition

$$x^{(t+1)} = x^{(t)} + \frac{\sigma^2}{2}\nabla \log p\left(x^{(t)}\right) + \sigma\varepsilon_t \tag{1.16}$$

where $\varepsilon_t \sim \mathcal{N}_p(0, I_p)$ and σ^2 corresponds to the discretization step. The drawback of this approach is that it introduces a bias, because in general $p(x)$ is not invariant with respect to the Markov chain defined by the discretization [18, 19, 20]. Alternatively, the Metropolis-adjusted Langevin algorithm (MALA) uses a combination of two mechanisms to generate the states of a random walk that has the target probability distribution as an invariant measure. First, new states are proposed using Langevin dynamics (1.16). Second, the proposals are accepted or rejected using the Metropolis–Hastings algorithm Sec.1.2.3. This two step MALA procedure will be later used in Chapter 4.

Gibbs sampling algorithm

Gibbs sampling is used to draw samples from multivariate distributions. It can be seen as a variant of the Metropolis-Hastings algorithm in which a sequence of proposal distributions, denoted as $q(x \mid x')$, is defined based on the conditional distributions of the joint distribution $p(x)$. Interestingly, the Gibbs algorithm systematically accept samples without ever need to compute the acceptance rate α in (1.13). This approach relies on the assumption that while direct sampling from the complex distribution $p(x)$ may be challenging, its conditional distributions $p(x_i \mid x_{j\neq i})$ are tractable and easy to sample from.

For many graphical models, these conditional distributions are straightforward to sample. The procedure is reported as Algo.2. It is not hard to

Algorithm 2: Gibbs

Input: Initialize $x_1^{(0)}, x_2^{(0)}, \ldots, x_K^{(0)}$, size of the chain N_{MC},

for $i \leftarrow 1$ **to** N_{MC} **do**

  Draw $x_1^{(i+1)} \sim p\left(x_1 \mid x_2^{(i)}, x_3^{(i)}, \ldots, x_K^{(i)}\right)$

  Draw $x_2^{(i+1)} \sim p\left(x_2 \mid x_1^{(i+1)}, x_3^{(i)}, \ldots, x_K^{(i)}\right)$

  $\cdots$

  Draw $x_K^{(i+1)} \sim p\left(x_K \mid x_1^{(i+1)}, x_2^{(i+1)}, \ldots, x_{K-1}^{(i+1)}\right)$

end

Output: Collection of samples $\left\{x^{(1)} \cdots x^{(N_{\text{MC}})}\right\}$ approximately drawn from $p(\cdot)$.

see that the original joint distribution is the stationary distribution of the Markov chain defined by these transition kernels as discussed in [11].

The benefits of Gibbs sampling are as follows: (*i*) assessing the conditional distributions can be straightforward, (*ii*) conditionals may be conjugate and permit exact sampling, (*iii*) conditionals typically have lower dimensions compared to the joint distribution, which facilitates the use of techniques like rejection sampling or importance sampling. Nonetheless, a significant limitation arises when variables exhibit strong interdependencies, making it challenging to navigate through the sampling process.

Limitations

Our exploration of Monte Carlo sampling has equipped us with powerful tools to sample from complex probability distributions. However, a crucial caveat arises: these methods presuppose explicit knowledge of the targeted density distribution, at least up to the normalization constant. In the domain of probabilistic modeling, scenarios frequently arise in which obtaining such an explicit density proves challenging. Having explored various techniques for effective distribution sampling, the forthcoming section will discuss modern approaches for learning these elusive probability distributions.

1.3 Generative models for high dimensional distribution

1.3.1 Learning distributions

Modeling and generating high-dimensional data are two essential components of machine learning and statistics. These tasks form the foundation of probabilistic modeling and the process of making informed decisions in the presence of uncertainty. One initial strategy for probabilistic modeling involves relying on expert knowledge to deduce an explicit law for a given object of interest. However, this approach has its limitations, primarily constrained by the extent of existing knowledge and the capability to translate it into mathematical equations.

Alternatively, another approach entails employing sensors to observe the target object and infer its underlying probability law based on the collected data. In that sense, probabilistic modeling addresses one of the most fundamental problems in machine learning i.e., the discovering of structure from data in an unsupervised manner. Interestingly density estimation can be seen as the reverse paradigm of the sampling task. For density estimation, the task entails working with provided samples to recover the underlying

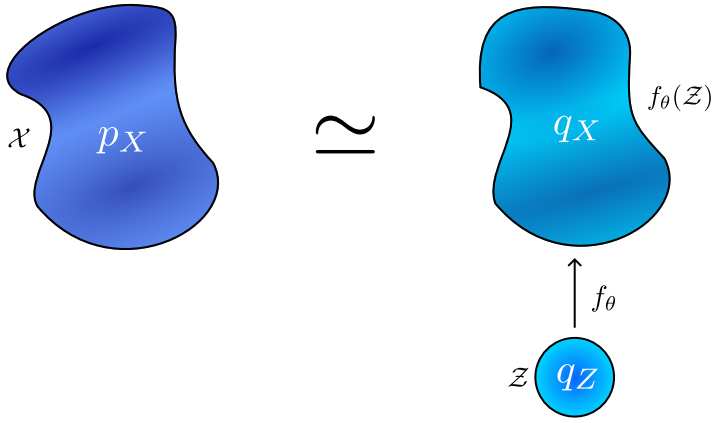


Figure 1.3: The deep generative model $f_{\boldsymbol{\theta}}$ aims at mapping samples from a simple distribution q_Z (located at the bottom right) to a more complex distribution q_X (located at the top right), with the objective of achieving similarity to the true distribution p_X (located at the top left).

density function that generated those samples. In contrast, for sampling, the objective involves working with a given density function to generate samples from it.

Utilizing expressive deep neural networks, generative models can parameterize complex probability distributions to estimate the underlying data distribution of a given dataset. On can then compare the quality of various data points by computing the corresponding likelihood function, or create new data points by sampling from the estimated distribution.

1.3.2 Statistical foundations of deep generative models

Generative models

The main objective of generative modeling is to acquire a representation of an intricate and high-dimensional probability distribution p_X defined over $\mathcal{X}$. This distribution is often characterized by its complexity, such as being multimodal with possibly a disjoint support. To achieve this, one uses a potentially large, but finite, set of independent and identically distributed (i.i.d) samples from p_X known as the training data. Unlike conventional statistical inference approaches that aim to derive a mathematical expression for the underlying probability, the goal in generative modeling is to obtain a generator function parametrized by θ

$$f_\theta : \mathcal{Z} \to \mathcal{X} \tag{1.17}$$

that maps samples from a tractable distribution q_Z supported in $\mathcal{Z}$ to points in $\mathcal{X}$ resembling the given data. In other words, the assumption is that for each sample $\mathbf{x}$ drawn from the distribution p_X, there exists at least one point $\mathbf{z}$ drawn from the distribution q_Z such that $f_\theta(\mathbf{z})$ provides an approximation of $\mathbf{x}$. The transformation of the latent distribution q_Z by $f_\theta(\cdot)$ is denoted as q_X. The presence of a generator capable of mapping points from the simpler distribution q_Z to the more complex distribution q_X enables the generation of samples within the complex space $\mathcal{X}$, which is desirable in various applications.

The latent space

The latent variable $\mathbf{z}$, which corresponds to a given vector $\mathbf{x}$, is often unknown, leading to its characterization as a latent variable and referring to $\mathcal{Z}$ as the latent space. For convenience, Gaussian is a common choice for q_Z. This assumption is made without loss of generality, as q_Z can, in principle, represent any tractable distribution. It is essential to have the capability to sample from q_Z and, in certain cases, compute or evaluate the probability density function $q_Z(\mathbf{z})$. Figure 1.3 provides an illustration of our notations.

It is worth noting that the dimensionality of the latent space $\mathcal{Z}$, may differ from the dimensionality of the data space $\mathcal{X}$. For instance, in the case of high-resolution images with millions of pixels, the actual representation does not truly reside in such a high-dimensional space. Instead, their content is typically preserved even when the dimensionality is reduced, indicating the existence of a hidden manifold with an unknown dimension in which the images are embedded. This is usually referred as the manifold hypothesis [21].

Deep Generative models

Once the generator function f_θ is known, it is possible to generate new data points by sampling from the distribution $\mathbf{z} \sim q_Z$ and computing $f_\theta(\mathbf{z})$. This capability of generating new samples is the primary objective in various applications, including deep fakes and Bayesian statistics. However, this direct sampling approach called ancestral sampling (see paragraph 1.2.2), may lead to undesirable consequence as discussed in Chapter 4.

For most datasets of interest, it is not feasible or practical to derive the function f_θ from fundamental principles. For instance, modeling the transformation process that converts a sample from a Gaussian distribution to an image of a celebrity can be highly challenging. As a result, there has been a growing trend in recent years to use generic function approximators, such as

neural networks with multiple hidden layers. This approach forms the foundation of deep generative models (DGMs), where the generator function f_θ is represented by a deep neural network (DNN). One of the key advantages of DNNs is their ability to effectively approximate functions in high-dimensional spaces. The DGM f_θ, with $\theta \in \mathbb{R}^{N_\theta}$ represents the parameters of the neural network.

Determining the architecture of the deep neural network f_θ is a complex topic that requires careful considerations. It involves making decisions about the number of layers and the specific operations performed within each layer. For a more comprehensive understanding and a wider range of options. Interested readers are directed to the excellent textbook [22].

It is important to emphasize that the choice of an architecture plays a crucial role in both modeling the generative process and effectively training the parameters of the generator. However, the lack of clear theoretical guidelines makes this task challenging. The quality of the architecture directly impacts our ability to accurately represent the generative process and successfully address the learning problem. Hence, careful considerations and experimentations are necessary to determine an effective architecture.

1.3.3 Training deep generative models

As discussed in section 1.3.2 the common goal of deep generative models is to learn a parameter θ such that new samples, $f_\theta(\mathbf{z})$ where $\mathbf{z} \sim q_Z$, are statistically indistinguishable from samples from the training data. In other words, once θ is learned, f_θ will transforms the latent probability distribution, q_Z, close to the probability distribution of the data $q_X \approx p_X$. Determining the distance between two distributions is a two-sample hypothesis testing problem. This problem is very difficult, especially for complicated distributions in high dimensions. The quality of the generator can be also assessed visually when handling image datasets.

Loss functions

Generative models can be trained using many methodologies for various applications. In tasks related to density estimation and similar objectives, the log-likelihood (or equivalently Kullback-Leibler divergence) has traditionally served as the standard loss function for both training and evaluating generative models. However, computing the likelihood for interesting models can be computationally infeasible. For instance, the normalization constant of energy-based models is typically a challenging task [22]. These models may still be trained using alternative objectives related to log-likelihood. These

include techniques like contrastive divergence [23], score matching [24], lower bounds on the log-likelihood [25], noise-contrastive estimation [26], probability flow [27], maximum mean discrepancy (MMD) [28, 29], or approximations of the Jensen-Shannon divergence (JSD) [30].

Maximum likelihood training

Instead of guessing that some loss function might make a good candidate to learn a data distribution, the aim is to establish guiding principles that facilitate the derivation of effective loss functions for various models.

The most common principle is the maximum likelihood principle. Consider a set of m examples $\mathbb{X} = \left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}\right\}$ drawn independently from the true but unknown data generating distribution $p_X(\mathbf{x})$. Let $q_\theta(\mathbf{x})$ be a parametric family of probability distributions over the same space indexed by θ. In other words, $q_\theta(\mathbf{x})$ maps any configuration $\mathbf{x}$ to a real number estimating the true probability $p_X(\mathbf{x})$. The maximum log-likelihood estimator for θ is then defined as

$$\theta_{\mathrm{ML}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{i=1}^{m} \log q_\theta \left(\mathbf{x}^{(i)}\right). \tag{1.18}$$

As the arg max remains unaffected by rescaling the cost function, it is possible to divide by m in order to derive a form of the criterion expressed as an expectation with respect to the empirical distribution $\hat{p}_{\mathrm{data}}$ defined by the training data:

$$\theta_{\mathrm{ML}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, \mathbb{E}_{\mathbf{x}\sim\hat{p}_{\mathrm{data}}}\, \log q_\theta(\mathbf{x}) \tag{1.19}$$

An alternative interpretation involves seeing it as minimizing the dissimilarity between the model distribution q_θ and the empirical distribution $\hat{p}_{\mathrm{data}}$ defined by the training set. This dissimilarity between the two distributions is defined by the KL divergence:

$$D_{\mathrm{KL}}\left(p_X \| q_\theta\right) = \mathbb{E}_{p_X}\left[\log p_X(\mathbf{x}) - \log q_\theta(\mathbf{x})\right]. \tag{1.20}$$

Where p_X is approximated using the empirical distribution $\hat{p}_{\mathrm{data}}$. Since the quantity $\hat{p}_{\mathrm{data}}(x)$ does not depend on the network parameters, minimizing the KL divergence is clearly equivalent as maximizing the likelihood function (1.18).

Choosing the learning objectives

The main motivation for introducing new training methods revolves around the challenge of adapting probabilistic models with likelihoods that are com-

putationally challenging to handle. Most training procedures exhibit consistency, meaning that if the data is sampled from a model distribution, then this particular model distribution will approach optimality according to the training objective as the number of training examples tends towards infinity. Nevertheless, in scenarios where a disparity exists between the data distribution and the model, distinct objective functions can lead to very different results.

Usual losses

One well-known objective function among the available options is the Maximum Mean Discrepancy (MMD) [28] is defined as,

$$D_{\mathrm{MMD}}(p, q) = \left(\mathbb{E}_{p,q}\left[k\left(\mathbf{x}, \mathbf{x}'\right) - 2k(\mathbf{x}, \mathbf{y}) + k\left(\mathbf{y}, \mathbf{y}'\right)\right]\right)^{\frac{1}{2}} \tag{1.21}$$

where k is any kernel function, $\mathbf{x}, \mathbf{x}'$ are independent and distributed according to the data distribution p, and $\mathbf{y}, \mathbf{y}'$ are independently distributed according to the model distribution q. Popularized by GAN, the Jensen-Shannon divergence (JSD) is defined as

$$D_{\mathrm{JSD}}(p, q) = \frac{1}{2} D_{\mathrm{KL}}(p \| m) + \frac{1}{2} D_{\mathrm{KL}}(q \| m), \tag{1.22}$$

where $m = (p+q)/2$ is an equal mixture of distributions p and q. The JSD is directly optimized using the data density, which is generally not possible in practice when its distribution can only be accessed from samples. In this case, one can employ generative adversarial networks (GANs) to approximate the optimization of the Jensen-Shannon divergence (JSD). However, it's worth noting that in practical applications, the objective function optimized by GANs can substantially deviate from JSD.

Trade-offs

Figure 1.4 taken from [31] illustrates this on a toy example where an isotropic Gaussian distribution has been fit to a mixture of Gaussians by minimizing various cost functions. Maximizing average log-likelihood or equivalently minimizing Kullback-Leibler divergence (KLD) avoids assigning extremely small probability to any data point but assigns a lot of probability mass to non-data regions. In contrast, MMD has been used with generative moment matching networks [29, 32] and JSD is usually used in generative adversarial networks [30]. Minimizing MMD or JSD results in a Gaussian distribution that fits one mode effectively but disregards other data regions.

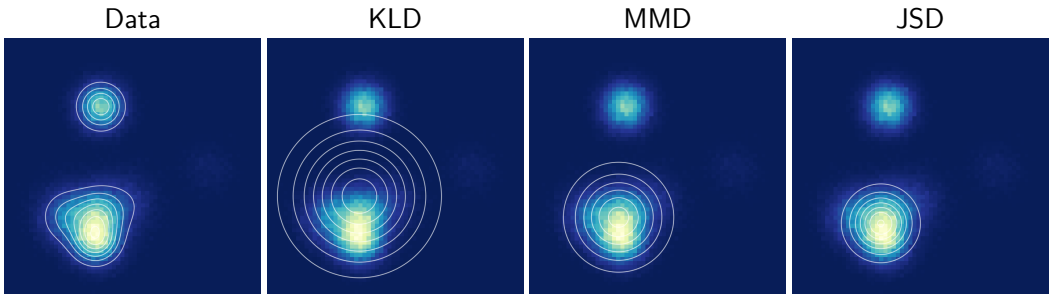


Figure 1.4: An isotropic Gaussian distribution was fit to data drawn from a mixture of Gaussians. Training procedure from left to right: minimizing Kullback-Leibler divergence (KLD), MMD, JSD. The different fits demonstrate different tradeoffs made by the three divergences. Results reported in [31].

Understanding the trade-offs between different metrics is crucial. Different applications demand varying trade-offs, necessitating the selection of appropriate metrics. For instance, while compression benefits from assigning probability to all plausible images, image reconstruction may only require generating one plausible example [33].

Additionally, grasping these trade-offs better enhances the interpretation of empirical results. Generative image models [30, 28, 34, 29], are often evaluated based on visual fidelity. Figure 1.4 suggests that optimizing a model with Kullback-Leibler divergence (KLD) may yield atypical samples, in contrast to models optimized with other measures. In essence, generating plausible samples, with high target distribution density, doesn't necessarily imply a good density model when measured by KLD, but it might be expected when optimizing Jensen-Shannon divergence (JSD).

The discussion of the theoretical foundations of generative models and their training methodologies now converges toward a practical juncture. In the upcoming section, the transition from theory to practice is marked by an examination of two deep generative model architectures. These models will serve as central focal points for the subsequent discussions and applications.

1.4 Deep generative model architectures

This section explores two key deep generative models: Normalizing Flows (NF) and Denoising Diffusion Probabilistic Models (DDPM). These architectures are pivotal for addressing the challenges of learning complex and poten-

tially high dimensional distributions. In particular, this section presents the modelization and the sampling of the associated distributions, as discussed in the previous paragraphs

Our examination of NF and DDPM encompasses their definitions, training procedures, and inference methodologies. These models are specifically chosen for their remarkable properties, including NF tractable likelihood and DDPM extensive capacity. Chapters 2, 3, and 4 will expand upon and apply NF in diverse applications, while Chapter 5 will harness DDPM to efficiently tackle inverse problems.

Appendix B further explores the connection between the discrete transformations of NF and DDPM and their continuous counterparts, Continuous Normalizing Flow and Stochastic Differential Equation (SDE) Diffusion Models. This exploration offers a comprehensive perspective on the interplay between these models.

1.4.1 Normalizing Flows

Principle

The fundamental concept behind NF entails representing the generator, denoted as f_θ, as a diffeomorphic function with orientation-preserving properties. In pursuit of this objective, NF models are deliberately designed to maintain equivalence between the latent space dimension and the dimension of the data space, i.e., $\mathcal{X}, \mathcal{Z} \in \mathbb{R}^n$. While this is a significant restriction in practice, NF can be used as an add-on in other approaches that overcome this restriction. Under these assumptions, the change of variables formula used to approximate the likelihood of a given data point $\mathbf{x}$ is given by

$$p_X(\mathbf{x}) \approx q_X(\mathbf{x}) = q_Z\left(f_\theta^{-1}(\mathbf{x})\right)\left|J_{f_\theta^{-1}}(\mathbf{x})\right| \tag{1.23}$$

where $J_{f^{-1}}(\mathbf{x})$ is the Jacobian of the function f_θ^{-1} evaluated at $\mathbf{x}$, $|\cdot|$ denotes the determinant of a matrix and $q_X(\cdot)$ is the likelihood prescribed by the learned model f_θ (see 1.3.2).

Using (1.23), one can evaluate q_X exactly when q_Z has a sufficiently smooth density, and efficient computation of both f_θ^{-1} and its Jacobian determinant are achievable. To sample from q_X, an additional prerequisite is the development of an efficient method for evaluating $f_\theta(\mathbf{z})$ to transform samples from the latent distribution q_Z. These requirements drive the choice about the generator modeling.

A finite NF [35, 36] is constructed by concatenating diffeomorphic transformations with tractable Jacobian determinants, which leads to the gener-

ator

$$f_\theta(\mathbf{z}) = f_K \circ f_{K-1} \circ \cdots \circ f_1(\mathbf{z}). \tag{1.24}$$

In the context of deep learning, f_j commonly represent the layers of the network and K the depth of the network. Assuming that efficient expressions for the inverses of the layer functions f_j are available, the maximum likelihood loss (1.23) can be computed using

$$f_\theta^{-1}(\mathbf{x}) = f_1^{-1} \circ f_2^{-1} \circ \cdots \circ f_K^{-1}(\mathbf{x}) \quad \text{and} \quad \log\left|J_{f_\theta^{-1}}(\mathbf{x})\right| = \sum_{j=1}^{K} \log\left|J_{f_j^{-1}}(\mathbf{h}_j)\right| \tag{1.25}$$

Here, $\mathbf{h}_K, \mathbf{h}_{K-1}, \ldots, \mathbf{h}_1$ are the hidden features, $\mathbf{h}_0 = \mathbf{z} = f_\theta^{-1}(\mathbf{x})$, and

$$\mathbf{h}_{j-1} = f_j^{-1}(\mathbf{h}_j), \quad \text{for } j = K, \ldots, 1, \quad \text{with} \quad \mathbf{h}_K = \mathbf{x} \tag{1.26}$$

Note that maximum likelihood training is feasible as long as it is possible to compute the inverse of the generator and the logarithm of the determinant of its Jacobian.

Examples

The key trade-off in NF involves designing the layers f_j to be expressive while also ensuring that the determinants of their Jacobians remain computationally tractable. Ideally, this entails achieving equal computational costs for evaluating both f_j and its inverse. These considerations enable the categorization of existing approaches based on their capacity to compute either f_θ, f_θ^{-1}, or both:

- Examples of NF that efficiently compute both the generator and its inverse efficiently are non-linear independent components estimation (NICE) [37], real non-volume preserving (real NVP) flow [38], Generative Flow with Invertible 1x1 Convolutions (Glow) [39] and [40]. A central concept in these approaches is that their layers partition the variables into two blocks and utilize components known as coupling layers, which are easily invertible. These approaches fall within the broader category of invertible neural networks. Further details and applications can be found in the comprehensive literature review, including its applications to inverse problems, as outlined in [41].

- Instances of NF that can efficiently compute f_θ but not its inverse include the planar and radial flows [35], Sylvester flows [42] and inverse autoregressive flows (IAF) [43]. These models lack a closed-form expression for the inverse, which is essential to train the generator using the maximum likelihood objective function.

- An example of NF capable of forward pass but slow reverse is the masked autoregressive flow (MAF) [44]. Although these models can be trained straightforwardly using maximum likelihood, their capability remains constrained when learning complex distributions.
- Examples of normalizing flow that cannot perform explicit density estimation are Residual Flows [45] and i-ResNets [46]. These residual networks architectures are composed of simple transformations $f(x) = x + g(x)$, and are invertible when g is contractive. These architectures allow great capacity but lack of explicit formulation of the determinant of the Jacobian. Applying the change of variable formula reveals a trace of power series which can be only approximated numerically.

Coupling layers

Some examples discussed in this manuscript are special instances of coupling layers. These layers guarantee an invertible transformation and provide an explicit expression of the Jacobian, as necessitated in the change of variables (1.23). The relationship between the input and output of the jth layer can be expressed as

$$\left(\mathbf{h}_{j+1}^{\mathrm{id}}, \mathbf{h}_{j+1}^{\mathrm{ch}}\right) = f_j\left(\mathbf{h}_j^{\mathrm{id}}, \mathbf{h}_j^{\mathrm{ch}}\right) \tag{1.27}$$

with

$$\begin{cases} \mathbf{h}_{j+1}^{\mathrm{id}} = \mathbf{h}_j^{\mathrm{id}} \\ \mathbf{h}_{j+1}^{\mathrm{ch}} = \left(\mathbf{h}_j^{\mathrm{ch}} + \mathsf{D}_j\left(\mathbf{h}_j^{\mathrm{id}}\right)\right) \odot \exp\left(\mathsf{E}_j\left(\mathbf{h}_j^{\mathrm{id}}\right)\right) \end{cases} \tag{1.28}$$

where $\mathbf{h}_j^{\mathrm{id}}$ and $\mathbf{h}_j^{\mathrm{ch}}$ (resp. $\mathbf{h}_{j+1}^{\mathrm{id}}$ and $\mathbf{h}_{j+1}^{\mathrm{ch}}$) are disjoint subsets of components of the input vector $\mathbf{h}_j$ (resp. the output vector $\mathbf{h}_{j+1}$). The partitioning of the input, $\mathbf{h}$, into $\mathbf{h}^{\mathrm{id}}$ and $\mathbf{h}^{\mathrm{ch}}$ is achieved through a masking process. This process transforms $\mathbf{h}^{\mathrm{ch}}$, into a function of the unchanged part, $\mathbf{h}^{\mathrm{id}}$. The scale function $\mathsf{E}_j(\cdot)$ and the offset function $\mathsf{D}_j(\cdot)$ are described by neural networks whose parameters $\boldsymbol{\theta}_j$ are adjusted during the training. It is worth noting that imposing the flow architecture detailed in (1.24) will lead to an explicit discretization scheme of the mapping $f_\theta(\cdot)$ into a sequence of elementary functions $f_j(\cdot)$. This property will be used in the Chapter 3.

Our prior examination has centered on normalizing flows, which involve a deterministic mapping from a Gaussian latent space to the data space. This deterministic approach imposes specific constraints on the model architecture. In the upcoming section, the discussion will explore a more flexible model type characterized by a stochastic mapping between the latent space and the data distribution.

1.4.2 Denoising diffusion probabilistic modeling

Principle

A denoising diffusion probabilistic model (DDPM) [47, 48] employs two Markov chains: a forward chain that introduces noise to the data, and a reverse chain that converts noise back to data. Formally, starting from a data distribution $\mathbf{x}_0 \sim p_X(\mathbf{x})$, the forward Markov process generates a sequence of random variables $\mathbf{x}_1, \mathbf{x}_2 \dots \mathbf{x}_T$ using a transition kernel $q(\mathbf{x}_t \mid \mathbf{x}_{t-1})$. By applying the chain rule of probability and leveraging the Markov property, one can factorize the joint distribution into

$$q(\mathbf{x}_1, \dots, \mathbf{x}_T \mid \mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t \mid \mathbf{x}_{t-1}). \tag{1.29}$$

One typical design for the transition kernel for DDPM is a Gaussian perturbation, and the most common choice for the transition kernel is

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}\left(\mathbf{x}_t; \sqrt{1-\beta(t)}\mathbf{x}_{t-1}, \beta(t)\mathbf{I}\right), \tag{1.30}$$

where $\beta(t) \in (0,1)$ is a predefined function of significant importance. It governs the level of noise introduced during the process, with higher values leading to noisier samples. Typically, it is chosen as a linearly increasing function [47]. However, recent techniques have proposed to use cosine-based functions [49].

As observed by *Sohl-Dickstein et al.* [48], this Gaussian transition kernel allows us to marginalize the joint distribution in Eq.(1.29) to obtain the analytical form of $q(\mathbf{x}_t \mid \mathbf{x}_0)$ for all $t \in \{0, 1, \cdots, T\}$. Specifically, with $\alpha(t) := 1 - \beta(t)$ and $\bar{\alpha}(t) := \prod_{s=0}^{t} \alpha(s)$, this leads to

$$q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_t; \sqrt{\bar{\alpha}(t)}\mathbf{x}_0, (1-\bar{\alpha}(t))\,\mathbf{I}\right). \tag{1.31}$$

Given $\mathbf{x}_0$, one can easily obtain a sample of $\mathbf{x}_t$ by sampling a Gaussian vector $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathrm{I})$ and applying the transformation

$$\mathbf{x}_t = \sqrt{\bar{\alpha}(t)}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}(t)}\boldsymbol{\epsilon}. \tag{1.32}$$

When $\bar{\alpha}(T) \approx 0, \mathbf{x}_T$ is almost Gaussian in distribution, one can write $q(\mathbf{x}_T) := \int q(\mathbf{x}_T \mid \mathbf{x}_0)\, q(\mathbf{x}_0)\, \mathrm{d}\mathbf{x}_0 \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$.

To generate new data samples, DDPMs initiate the process by creating an unstructured noise vector sampled from the prior distribution. Then, it gradually removes noise from this vector by running a trainable Markov

chain in the reverse time direction. Specifically, the reverse Markov chain is parameterized by a prior distribution $p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathrm{I})$ and a trainable transition kernel $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$. The trainable transition kernel $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ is given by

$$p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t)) \tag{1.33}$$

where the mean $\mu_\theta(\mathbf{x}_t, t)$ and variance $\Sigma_\theta(\mathbf{x}_t, t)$ are deep neural networks with parameters θ. With this reverse Markov chain in hand, one can generate a data sample $\mathbf{x}_0$ by first sampling a noise vector $\mathbf{x}_T \sim \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathrm{I})$, then iteratively sampling from the learnable transition kernel $\mathbf{x}_{t-1} \sim p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ until $t = 1$.

Training

The key to the success of this sampling procedure lies in training the reverse Markov chain to mirror the actual time reversal of the forward Markov chain. To accomplish this, it is essential to adjust the parameter θ such that the joint distribution of the reverse Markov chain $p_\theta(\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_T) := p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ closely approximates that of the forward process $q(\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_T) := q(\mathbf{x}_0) \prod_{t=1}^{T} q(\mathbf{x}_t \mid \mathbf{x}_{t-1})$ (Eq.(1.29)). This is achieved by minimizing the KL divergence:

$$\mathrm{KL}(q(\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_T) \| p_\theta(\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_T)) \tag{1.34a}$$

$$\overset{(i)}{=} -\mathbb{E}_{q(\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_T)}[\log p_\theta(\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_T)] + \text{ const} \tag{1.34b}$$

$$\overset{(ii)}{=} \underbrace{q(\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_T)\left[-\log p(\mathbf{x}_T) - \sum_{t=1}^{T} \log \frac{p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)}{q(\mathbf{x}_t \mid \mathbf{x}_{t-1})}\right]}_{:=-L_{\mathrm{VLB}}(\mathbf{x}_0)} + \text{const} \tag{1.34c}$$

$$\overset{(iii)}{\geq} \mathbb{E}[-\log p_\theta(\mathbf{x}_0)] + \text{ const}, \tag{1.34d}$$

where (i) is the definition of KL divergence, (ii) arises from the fact that both $q(\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_T)$ and $p_\theta(\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_T)$ are products of distributions, and (iii) is a consequence of Jensen's inequality. The first term in Eq.(1.34c) corresponds to the Variational Lower Bound (VLB) of the log-likelihood of the data $\mathbf{x}_0$, a common objective for training probabilistic generative models. The training objective of DDPM is to minimize the negative VLB, which is particularly straightforward to optimize. It consists of a summation of independent terms [11] and optimized by stochastic optimization techniques [50].

Ho et al. [47] propose to reweight some terms in L_{VLB} for better sample

quality. The loss in [47] takes the form of

$$\mathbb{E}_{t\sim\mathcal{U}[1,T],\mathrm{x}_0\sim q(\mathrm{x}_0),\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathrm{I})}\left[\lambda(t)\left\|\boldsymbol{\epsilon}_\theta\left(\mathbf{x}_t,t\right)-\boldsymbol{\epsilon}\right\|^2\right] \tag{1.35}$$

where $\lambda(t)$ is a positive weighting function, $\mathbf{x}_t$ is computed from $\mathbf{x}_0$ and $\boldsymbol{\epsilon}$ by Eq.(1.32), $\mathcal{U}[1,T]$ is a uniform distribution over the set $\{1,2,\cdots,T\}$, and $\boldsymbol{\epsilon}_\theta$ is a deep neural network with parameter θ that predicts the noise vector $\boldsymbol{\epsilon}$ given $\mathbf{x}_t$ and t.

1.5 Remaining challenges

This concise overview has highlighted the multitude of strategies aimed at enhancing the efficiency of sampling methods through the reciprocal integration of simulation and machine learning. However, despite these advancements, certain challenging statistical problems persist. The subsequent sections will explore two of these problems, which will be discussed in this manuscript.

- **Probability distributions -** The growing volume and diversity of data, coupled with recent advances in specialized research domains such as signal and image processing, have rendered statistical inference problems increasingly intricate. This complexity is particularly notable within the Bayesian framework, where numerous scenarios present great challenges. The accumulation of extensive observations, some of which may include outliers, leads to intricate likelihood functions. Additionally, the requirement to incorporate supplementary prior information further complicates posterior inference.

- **Scalable MCMC sampling -** Despite recent advances in Monte Carlo sampling that have reduced the number of iterations needed for convergence and the associated computational time, MCMC algorithms generally remain computationally expensive. Unlike machine learning, MCMC approaches do not leverage many of the sophisticated tools that enhance their attractiveness for large-scale inference tasks. Consequently, addressing one of the many ongoing challenges is essential: bridging the gap between the fields of machine learning and stochastic simulation in terms of computational cost and scalability.

- **Imperfect deep generative models -** Although machine learning methods have the ability to approximate high-dimensional probability distributions, they encounter challenges when it comes to representing complex densities. Several push-forward generative models (e.g VAE,

GAN, NF) have shown a propensity to generate unrealistic data, such as images, and have difficulties in detecting out-of-distribution data. The fusion of deep generative models with existing statistical inference methods necessitates a deeper understanding of these challenges and the development of methodologies to circumvent these limitations.

1.6 Contributions of the manuscript

The content within this manuscript is an attempt to address the sampling challenges mentioned earlier. In alignment with the works discussed earlier, the solutions presented in this manuscript are closely intertwined with machine learning, fostering the establishment of new connections between this domain and Monte Carlo sampling. The subsequent chapters of this work delineate the primary contributions in detail.

Chapter 2 introduces a new NF training paradigm based on a hybrid objective function combining the maximum likelihood (ML) principle and a sliced-Wasserstein distance. Vanilla ML-based training generally suffers from several shortcomings including out-of-distribution sampling. One reason for these deficiencies lies in the training strategy which traditionally only exploits a ML principle. Our proposed hybrid training shows better generative abilities in terms of both likelihood and visual aspects of the generated samples. Reciprocally, the proposed approach leads to a lower likelihood of out-of-distribution data, demonstrating a greater data fidelity of the resulting flows.

Chapter 3 proposes to leverage the flexibility of neural networks to learn an approximate optimal transport plan. More precisely, SWOT-Flow, a new and original method is presented to address the problem of transporting a finite set of samples associated with a first underlying unknown distribution towards another finite set of samples drawn from another unknown distribution. To this aim, the approach involves relaxing the Monge formulation of optimal transport (OT) by replacing the equality constraint on the push-forward measure with the minimization of the corresponding Wasserstein distance. The push-forward operator to be retrieved is then restricted to be a NF which is trained by optimizing the resulting cost function. This approach allows the transport plan to be discretized as a composition of functions. Each of these functions is associated to one sub-flow of the network, whose output provides intermediate steps of the transport between the original and target measures. This discretization yields also a set of interme-

diate barycenters between the two measures of interest.

Chapter 4 studies the pathological behaviours of push-forward generative model when targeting complex distributions. For instance, such problems may appear for distributions on multi-component topologies or characterized by multiple modes with high probability regions separated by very unlikely areas. A typical symptom is the explosion of the Jacobian norm of the transformation in very low probability areas. The proposed approach, NF-SAILS, aims to overcome this issue through the utilization of a new Markov chain Monte Carlo algorithm to sample from the target distribution in the latent domain before transporting it back to the target domain. The approach relies on a Metropolis adjusted Langevin algorithm (MALA) whose dynamics explicitly exploits the Jacobian of the transformation. Contrary to alternative approaches, the proposed strategy preserves the tractability of the likelihood and it does not require a specific training. Notably, it can be straightforwardly used with any pre-trained NF network, regardless of the architecture.

Chapter 5 is dedicated to solving inverse imaging problems using deep generative model as prior. It introduces PnP-SGS, a stochastic plug-and-play (PnP) sampling algorithm that leverages variable splitting to efficiently sample from a posterior distribution. The algorithm based on split Gibbs sampling (SGS) draws inspiration from the alternating direction method of multipliers (ADMM). It divides the challenging task of posterior sampling into two simpler sampling problems. The first problem depends on the likelihood function, while the second is interpreted as a Bayesian denoising problem that can be readily carried out by diffusion-based generative models. Akin to its deterministic PnP-based counterparts, the proposed method exhibits the great advantage of not requiring an explicit choice of the prior distribution, which is rather encoded into a pre-trained generative model. However, unlike optimization methods (e.g., PnP-ADMM) which generally provide only point estimates, the proposed approach allows conventional Bayesian estimators to be accompanied by confidence intervals at a reasonable additional computational cost.

The concluding chapter summarizes the results presented in this thesis. It then discusses remaining open questions and draws some perspectives for future work.

For sake of reproducible research, the codes associated to the numerical results presented in our research works are available online at

https://github.com/FlorentinCDX

List of publications

Most of the works presented in this manuscript has been published or is currently under review for publication.

Submitted journal articles

- F. Coeurdoux, N. Dobigeon, P. Chainais. *"Normalizing flow sampling with Langevin dynamics in the latent space"*, Submitted to Machine Learning, arXiv:2305.12149
- F. Coeurdoux, N. Dobigeon, P. Chainais. *"Plug-and-Play split Gibbs sampler: embedding deep generative priors in Bayesian inference"*, Submitted to IEEE Transactions on Image Processing, arXiv:2304.11134

International conferences

- F. Coeurdoux, N. Dobigeon, P. Chainais. *"Sliced-Wasserstein normalizing flows: beyond maximum likelihood training"*. ESANN 2022, Bruges, arXiv:2207.05468
- F. Coeurdoux, N. Dobigeon, P. Chainais. *"Learning Optimal Transport Between two Empirical Distributions with Normalizing Flows"*. ECLM-PKDD 2022, Grenoble, arXiv:2207.01246

National conferences

- F. Coeurdoux, N. Dobigeon, P. Chainais. *"Méthode MCMC plug-and-play avec a priori génératif profond"*. GRETSI August 2023 Grenoble, hal-04154783
- F. Coeurdoux, N. Dobigeon, P. Chainais. *"Approximation du transport optimal entre distributions empiriques par flux de normalisation"*. GRETSI Septembre 2022 Nancy, hal-03704666

Chapter 2

Sliced-Wasserstein normalizing flows: beyond maximum likelihood training

Contents

2.1 Conventional training of normalizing flows . . . 42
2.1.1 Learning a change of variable . . . 42
2.1.2 Empirical loss function . . . 43
2.2 Problem statement . . . 44
2.2.1 Limitations of MLE training . . . 45
2.2.2 Limitations of GANs . . . 45
2.2.3 Towards a hybrid loss function . . . 46
2.3 Sliced-Wasserstein distance . . . 46
2.3.1 Vector space Sliced-Wasserstein . . . 46
2.3.2 Convolution Sliced-Wasserstein . . . 48
2.4 Hybrid objective function . . . 49
2.4.1 Logit space for images . . . 50
2.5 Numerical experiments . . . 52
2.5.1 Implementation details . . . 52
2.5.2 Goodness-of-fit . . . 54
2.5.3 Out-of-distribution detection . . . 55
2.6 Conclusion . . . 56

Approximating probability distributions thanks to NF has proven to be a powerful approach to accurately represent the underlying processes at the origin of collected data [51]. Despite being a commonly used method for NF models, maximum likelihood training has inherent limitations. Specifically, it exhibits sensitivity to the selection of the reference latent distribution q_Z, resulting in suboptimal performance and limited expressiveness [52]. Additionally, the constraint of relying on a parametric family of distributions can give rise to significant challenges and concerns [53, 54]. In this chapter, we delve into the known imperfections of Normalizing Flows (NF) models and propose an alternative approach to their training that overcomes the limitations of maximum likelihood training.

To overcome these challenges and move beyond maximum likelihood estimation (MLE), we introduce a novel hybrid loss function which incorporates a term to measure the discrepancy between the generated and the targeted distribution. This term derives from the Sliced-Wasserstein distance (SW) [55] between the true data distribution and the generated samples. Experimental results show that augmenting the MLE objective with this term consistently achieves higher likelihood as well as better quality of the generated samples. It also demonstrates better out of distribution (OOD) detection capabilities compared to classical training of flow-based models.

Section 2.1 will review the typical NF training procedure. Section 2.2 will state the problem and discuss the known flaws of NF. Section 2.3 will then introduce the Sliced-Wasserstein statistical distance. Section 2.4 presents the proposed method referred to as SW-NF and its hybrid learning objective. Section 2.5 illustrates its performances on numerical experiments. Conclusions and prospects are reported in Section 2.6.

The major part of the material of this chapter has been presented at an international conference:

- F. Coeurdoux, N. Dobigeon, P. Chainais. *"Sliced-Wasserstein normalizing flows: beyond maximum likelihood training"*. ESANN 2022, Bruges, arXiv:2207.05468

2.1 Conventional training of normalizing flows

2.1.1 Learning a change of variable

As discussed in the introduction, NFs define a flexible class of deep generative models that seek to learn a change of variable between a reference Gaussian

measure q_Z and a target measure p_X through an invertible transformation $f : \mathcal{Z} \to \mathcal{X}$ with $f \in \mathcal{F}$ where $\mathcal{F}$ defines the class of NFs. NF training aims to minimize a discrepancy measure between the target measure p_X and the push-forwarded measure q_X defined as

$$q_X = f_\sharp q_Z \tag{2.1}$$

where $f_\sharp$ stands for the associated push-forward operator. This discrepancy measure is generally chosen as the Kullback-Leibler (KL) divergence $D_{\mathrm{KL}}(p_X \| q_X)$. Explicitly writing the change of variables

$$q_X(x) = q_Z(f^{-1}(x))\,|J_{f^{-1}}(x)| \tag{2.2}$$

where $J_{f^{-1}}$ is the Jacobian matrix of f^{-1}, the training is thus formulated as the minimization problem

$$\begin{aligned} D_{\mathrm{KL}}(p_X \| q_X) &= \mathbb{E}_{p_X}\left[\log\left(\frac{p_X(x)}{q_X(x)}\right)\right] \\ &= \mathbb{E}_{p_X}[\log p_X(x) - \log q_Z(f^{-1}(x)) + \log|J_f^{-1}(x)|]. \end{aligned} \tag{2.3}$$

We can then find transport maps by solving the following optimization problem:

$$\min_{f\in\mathcal{F}} \quad \mathbb{E}_{p_X}[-\log q_Z(f^{-1}(x)) + \log|J_{f^{-1}}(x)|]. \tag{2.4}$$

Note that the term $\log p_X(x)$ does not appear in the objective function since this latter does not depend on f. In the present work, the class $\mathcal{F}$ of admissible transformations is chosen as the structures composed of coupling layers ([51, 36, 39]) ensuring the Jacobian matrix of f to be lower triangular with positive diagonal entries. Because of this triangular structure, the Jacobian J_f and the inverse of the map f^{-1} are available explicitly. In particular the Jacobian determinant $|J_f(z)|$ evaluated at $z \in \mathcal{Z}$ measures the dilation, the change of volume of a small neighborhood around z induced by f, i.e., the ratio between the volumes of the corresponding neighborhoods of x and z.

2.1.2 Empirical loss function

In practice, the target measure p_X is available only though observed samples $x = \{x_n\}_{n=1}^N$. Adopting a sample-average approximation, the objective function in (2.4) is replaced by its Monte Carlo estimate. For this fixed set of samples per data batch, the NF loss function is formulated as

$$\mathcal{L}_{\mathrm{MLE}}(f) = \frac{1}{N}\sum_{n=1}^{N}\left[-\log q_Z(f^{-1}(x_n)) + \log|J_{f^{-1}}(x_n)|\right]. \tag{2.5}$$

Note that this loss function optimizes over f^{-1}, however since the mapping f is bijective the network is the same and can be used in both direction. Moreover, training NF explicitly uses a Gaussian likelihood $q_Z(.)$, i.e, a function of the two first moments. We can write the complete explicit loss function as follows:

$$\mathcal{L}_{\text{MLE}}(f) = \frac{1}{N}\sum_{n=1}^{N}\left[\|f^{-1}(x_n)\|_2^2 + \log|J_{f^{-1}}(x_n)|\right] \tag{2.6}$$

It appears that optimizing a Gaussian likelihood function forces the latent space to be centered around zero but leaves any higher order moments completely free of the prescribed latent distribution. We will see in chapter 4 that in practice the latent space is not really Gaussian. A more refined way of fully characterizing the targeted distribution would be to match all the other higher order moments as well. Clearly, using a statistical distance between distributions during the training process would shift the optimization task from a nonlinear regression problem w.r.t. the likelihood parameters to a more relevant problem of looking for the best matching between the generated distribution and the targeted one.

2.2 Problem statement

The choice of the architecture of a deep generative model usualy boils down to a trade off between perceptually good-looking samples at the expense of tractable likelihoods. This compromise generally hides a training method decision. First, MLE training is asymptotically statistically efficient and serves as natural objective for learning generative models, but is also widely known to output bad quality samples [51]. Second, adversarial training strategies have demonstrated state of the art sample quality, but lacking characterization of an explicit density. Recent work [53] showed that exact likelihood generative models fail to distinguish training data from OOD data according to the model likelihood. This phenomenon occurs not only when the data sets are similar but also when they have dramatically different underlying semantics. For instance, Glow [39], a state-of-the-art normalizing flow, trained on CIFAR-10 will assign a higher likelihood to SVHN than to its CIFAR-10 training data [53]. This result is surprising since CIFAR-10 contains images of frogs, horses, ships, trucks, etc. and SVHN contains house numbers. A human would be very unlikely to confuse the two sets.

2.2.1 Limitations of MLE training

This well known issue can be traced back to training procedure of NF and more specificaly MLE. To further examine this curious phenomenon, Nalisnick *et al.* [53] inspect the change-of-variables objective itself, investigating if one or both terms give the OOD data a higher value. They report the constituent $q_Z(z)$ and $\log J_{f^{-1}}(z)$ terms in (2.2) for in distribution data as well as OOD data. While $q_Z(z)$ behaves mostly as expected, they noticed that the volume element $|J_{f^{-1}}(z)|$ seems to cause higher likelihood for OOD data. This behaviour can be explained by the change-of-variables objective which rewards the maximization of the Jacobian determinant and encourages the model to increase its sensitivity to perturbations in $\mathcal{X}$.

For the particular form of f, most works to date have constructed the bijection from affine coupling layers [51] which transform x by way of translation and scaling operations. Each coupling layer updates the masked part x_{change} of the input x to be $x_{\text{change}} \leftarrow (x_{\text{change}} + t(x_{\text{id}})) \cdot \exp(s(x_{\text{id}}))$, where x_{id} is the non-masked part of x, and s and t are the outputs of the st-network given x_{id} . The flow is encouraged to predict high values for s since for a given coupling layer the Jacobian term in the likelihood is given by $\sum_j s(x_{\text{id}})_j$ (see Introduction). Intuitively, to afford large values for scale s without making the latent representations large in norm and hence decreasing the density term $q_Z(z)$, the shift $t(.)$ has to be an accurate approximation of the masked input x_{change} . The likelihood for a given image will be high whenever the coupling layers can accurately predict masked pixels. These mechanisms has been studied by [56] which demonstrates that MLE trained NF do not represent images based on their semantic contents, but rather directly encode their visual appearance.

2.2.2 Limitations of GANs

In contrast, an alternate training procedure consists of generating data indistinguishable from the training data. Adversarially learned models such as GAN [57], Wasserstein GAN [58] and Sliced-Wasserstein generative model [59] can sidestep specifying an explicit density for any data point and belong to the class of implicit generative models. Those architectures are known to partition the latent space according to the semantic character of the training dataset [60]. The lack of characterization of an explicit density in GANs is however problematic. Several application areas of deep generative models rely on density estimates; for instance, count based exploration strategies based on density estimation or inverse problem resolution. To sidestep the above issues, Grover *et al.* [61] proposed a hybrid objective that bridges im-

plicit and prescribed learning by combining MLE and adversarial training using a GAN. The hybrid objective has a balancing effect between perceptually good-looking samples and an accurate density estimation of the inputs. The authors also demonstrate that this hybrid objective has a regularizing effect, which permits the model to outperform MLE as well as adversarial learning. However the choice of using an adversarial architecture is accompanied by the well-documented drawbacks of GANs. An adversarial architecture requires the training of an additional discriminator network which is notoriously unstable, can lead to mode collapse [62] and can produce overconfident predictions from OOD inputs [63].

2.2.3 Towards a hybrid loss function

In order to overcome the aforementioned challenges, this Chapter presents a novel hybrid loss function for training NF models. In addition to the conventional maximum likelihood estimation (MLE)-based term, our proposed approach incorporates a term that measures the discrepancy between the generated samples and the desired distribution. This discrepancy is quantified using the Sliced-Wasserstein distance (SW), as introduced by Rabin *et al.* [55], which compares the true data distribution with the generated samples.

2.3 Sliced-Wasserstein distance

In recent years, Wasserstein distance, which is intimately related to the theory of optimal transport (OT), has received a considerable attention from the machine learning (ML) community because of its theoretical properties when comparing distributions. However, it suffers from strong computational and statistical limitations, which have severely hindered its effective use in problems in high dimensions. Several workarounds have been proposed to alleviate these issues and to enable the use of OT in ML applications. In particular, the Sliced-Wasserstein (SW) distance is an alternative OT metric [55]. It has been increasingly popular since it benefits from a significantly reduced computational cost over the Wasserstein distance, especially on large-scale problems.

2.3.1 Vector space Sliced-Wasserstein

Formally, for any $p \geq 1$ and dimension $d \geq 1$, we first define the Wasserstein-p distance [64, 65] between two probability measures p_X and p_Y, which is given

by:

$$\mathrm{W}_p(p_X, p_Y) := \left(\inf_{\pi \in \Pi(p_X, p_Y)} \int_{\mathbb{R}^d \times \mathbb{R}^d} \|x - y\|_p^p d\pi(x, y) \right)^{\frac{1}{p}} \tag{2.7}$$

where $\Pi(p_X, p_Y) := \left\{ \pi \in \left(\mathbb{R}^d \times \mathbb{R}^d \right) \mid \int_{\mathbb{R}^d} \pi(x, y) dx = p_Y, \int_{\mathbb{R}^d} \pi(x, y) dy = p_X \right\}$ is the set of transportation plans between p_X and p_Y.

When $d = 1$, the Wasserstein distance has a closed form which is $W_p(p_X, p_Y) = \left(\int_0^1 \mid F_X^{-1}(z) - \ F_Z^{-1}(z) \right|^p dz \Big)^{1/p}$ where F_X and F_Y are the cumulative distribution function (CDF) of p_X and p_Y respectively. Given this closed-form property of Wasserstein distance in one dimension, the sliced Wasserstein distance [55] between p_X and p_Y had been introduced and admits the following formulation:

$$SW_p\left(p_X, p_Y\right) = \left(\int_{\mathbb{S}^{d-1}} W_p^p \left(S_{u\sharp} p_X, S_{u\sharp} p_Y \right)^p du \right)^{\frac{1}{p}} \tag{2.8}$$

where the projection operator $S_u : \mathbb{R}^d \to \mathbb{R}$ is defined as $S_u(x) \triangleq \langle u, x \rangle$ for any vector on the unit sphere $u \in \mathbb{S}^{d-1}$ and $S_{u\sharp} p_X$ is the push-forward operator of $S_u(.)$ on the measure p_X. However, the integration over the unit sphere in the sliced Wasserstein distance is intractable. Therefore, a Monte Carlo scheme is employed to approximate the integration, namely, $u_1, \ldots, u_L$ are drawn uniformly on the sphere $\mathbb{S}^{d-1}$ and the approximation of the vector sliced Wasserstein distance is given by:

$$\mathcal{L}_{\mathrm{VSW}}(x, y) = \frac{1}{L} \sum_{\ell=1}^{L} W_p^p \left(\frac{1}{N} \sum_{n=1}^{N} S_{u_\ell}(x_n), \frac{1}{N} \sum_{n=1}^{N} S_{u_\ell}(y_n) \right) \tag{2.9}$$

where x_n and z_n are realisations drawn from p_X and p_Y respectively. In practice, the number of projections L should be chosen to be sufficiently large compared to the dimension d, but not too large to avoid computational burden.

The usual SW is defined between two probability measures that have realizations as vectors. To be able to compute SW over images we have to first vectorize the image tensor. For better understanding, we visualize an example of projecting a probability measure $p_X \in \mathbb{R}^{c \times d \times d}$ in Figure 2.1. Input images are represented in RBG (Red Blue Green), then vectorized to form vectors. Moreover, vectors in orange, yellow and purple representing $u_1, \ldots, u_L$ are drawn uniformly on the sphere and multiplied to the vectorized images.

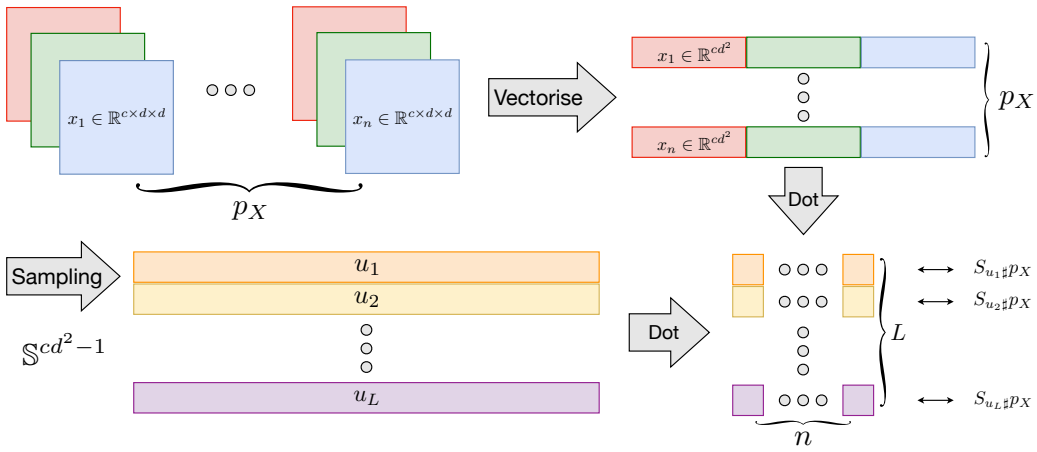


Figure 2.1: The conventional slicing process of sliced Wasserstein distance.

2.3.2 Convolution Sliced-Wasserstein

Images inherently contain spatial relations across channels and local information. Therefore, when images are transformed into vectors, it becomes challenging to preserve and access that information. Furthermore, vectorization necessitates the use of projecting directions from the unit hyper-sphere, and of these directions may lack effective discriminative power. Additionally, working within the unit sphere in high-dimensional spaces can be both time-consuming and memory-intensive. As a result, avoiding the vectorization step can enhance the overall efficiency of the process. To address these challenges, Nguyen et *al.* [66], proposed the so-called convolution SW (CSW) [1] which generalizes SW to images using a series of convolutions in the spirit of a multiresolution approach.

For $M \geq 1$, given a sequence of matrix kernels $\kappa_1 \in \mathbb{R}^{c_1\times d_1\times d_1}, \ldots, \kappa_M \in \mathbb{R}^{c_M\times d_M\times d_M}$, a convolution slicer $\mathcal{S}_\kappa(x)$ is a composition of M convolution functions with kernels :

$$\kappa = \kappa_1 \circledast \kappa_2 \circledast \ldots \circledast \kappa_M \tag{2.10}$$

$$= \overset{M}{\underset{m=1}{\circledast}} \kappa_m \tag{2.11}$$

(with stride or dilation if needed) such that $\mathcal{S}_\kappa(x) \triangleq x \circledast \kappa$ is a scalar for all $x \in \mathbb{R}^{c\times d\times d}$, with $\circledast$ the convolution operator and where each κ_m is drawn from a uniform distribution and normalized such that the sum of all the entries equal one.

[1] First ever implementation of CSW https://github.com/FlorentinCDX/Convolution-Sliced-Wasserstein

The concept behind the convolution slicer is to transform a given data point x into a one-dimensional subspace using a series of convolution kernels. These kernels are designed to capture both the spatial relationships across channels and the local information within the data. Note that this sequence of convolution imposes a dimensional constraint on the kernels, several dimensioning methods are discussed in [66]. For our experiments we used the so-called *convolution-base slicer* where $\kappa_1 \in \mathbb{R}^{c\times\left(2^{-1}d+1\right)\times\left(2^{-1}d+1\right)}$ and $\kappa_h \in \mathbb{R}^{1\times\left(2^{-h}d+1\right)\times\left(2^{-h}d+1\right)}$ for $h = 2, \ldots, M-1$, and $\kappa_M \in \mathbb{R}^{1\times a\times a}$ where $a = \frac{d}{2^{N-1}}$.

Formally, for any $p \geq 1$, the convolution sliced Wasserstein (CSW) of order $p > 0$ between two given probability measures $p_X, p_Y \in \mathbb{R}^{c\times d\times d}$ is given by:

$$\mathrm{CSW}_p(p_X, p_Y) := \left(\mathbb{E}\left[W_p^p\left(\mathcal{S}_{\kappa\sharp}p_X, \mathcal{S}_{\kappa\sharp}p_Y\right)\right]\right)^{\frac{1}{p}},$$

where the expectation is taken with respect to $\kappa_m \sim \mathcal{U}\left(\mathcal{K}_m\right)$. Here, $\mathcal{U}\left(\mathcal{K}_m\right)$ is the uniform distribution with the realizations being in the set $\mathcal{K}_m$ which is defined as $\mathcal{K}_m := \left\{\kappa_m \in \mathbb{R}^{c_m\times k_m\times k_m} \mid \|\kappa_l\|_2^2 = 1\right\}$.

Similar to the conventional sliced Wasserstein, the expectation with respect to kernels $\kappa_1, \ldots, \kappa_M$ uniformly drawn from the sets $\mathcal{K}_1, \ldots, \mathcal{K}_M$ in the convolution sliced Wasserstein is intractable to compute. Therefore, we also make use of Monte Carlo method to approximate the expectation, which leads to the following approximation of the convolution sliced Wasserstein:

$$\mathcal{L}_{\mathrm{CSW}}(x, y) = \frac{1}{L}\sum_{\ell=1}^{L} W_p^p\left(\frac{1}{N}\sum_{n=1}^{N}\mathcal{S}_{\kappa^{(\ell)}}(x_n), \frac{1}{N}\sum_{n=1}^{N}\mathcal{S}_{\kappa^{(\ell)}}(y_n)\right) \tag{2.12}$$

where $\kappa_m^{(\ell)}$ are uniform samples from the sets $\mathcal{K}_m$ for any integer $m \in [1, M]$. An illustration of the convolution slicer mecanism is given in Figure 2.2. Since each of the convolution slicer $\mathcal{S}_{\kappa^{(\ell)}}(\cdot)$ is in one dimension, we can utilize the closed-form expression of Wasserstein metric in one dimension to compute the Wasserstein distance. More theoretically, the convolution sliced Wasserstein $\mathrm{CSW}_p(.,.)$ is a pseudo-metric on the space of probability measures on $\mathbb{R}^{c\times d\times d}$, namely, it is symmetric, and satisfies the triangle inequality. Nguyen *et al.* [66] establishes that $\mathrm{CSW}_p(.,.) < \mathrm{W}_p(.,.)$ and shows that the convolution sliced Wasserstein does not suffer from the curse of dimensionality.

2.4 Hybrid objective function

The proposed SW-NF method builds on a NF neural architecture f to target a normal latent distribution q_Z so that the likelihood of the observed data p_X

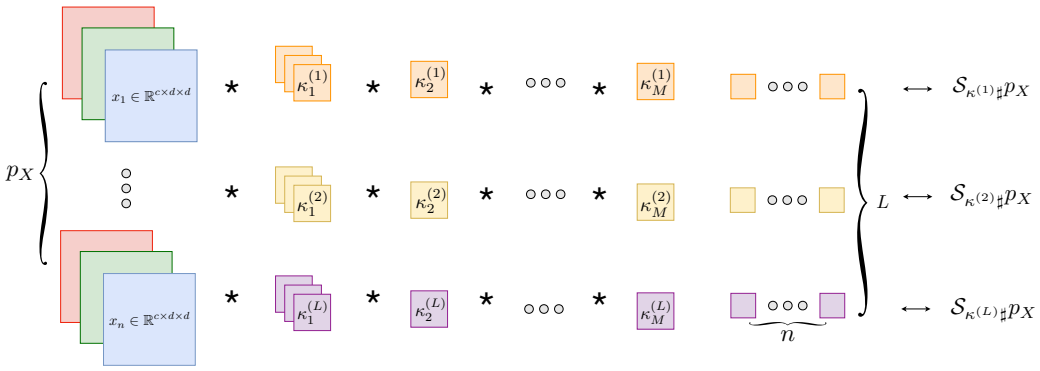


Figure 2.2: The convolution slicing process. The images are directly mapped to a scalar by a sequence of convolution functions which have kernels as random tensors

is well-defined and tractable for exact evaluation and MLE training. Departing from conventional strategies deployed to train NFs, this work proposes to derive a hybrid objective function that binds the likelihood of a prescribed model to higher order moment matching. To this aim, the conventional MLE-based objective is augmented with an additional term measuring the discrepancy between the respective distributions of the original data $x \sim p_X$ and the generated data $\tilde{x} = f(z)$ with $z \sim q_Z$. Note that the likelihood loss is prescribed on the latent space while the SW-based distance between the generated and target distributions can be prescribed over the data space. Thus the proposed hybrid objective is a combination of reconstruction and feature losses defining the new NF training as

$$\hat{f} \in \min_{f\in\mathcal{F}} \mathcal{L}_{\star\text{SW}}(x, f(z)) + \alpha\mathcal{L}_{\text{MLE}}(f) \tag{2.13}$$

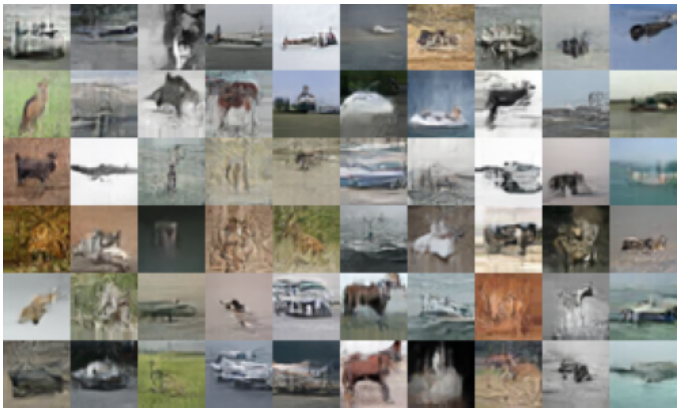
where α is a hyperparameter balancing the two terms and $\mathcal{L}_{\star\text{SW}}$ refers to either $\mathcal{L}_{\text{VSW}}$ for vector data sets or to $\mathcal{L}_{\text{CSW}}$ for image inputs, respectively. It is worth noting that the new objective function can be interpreted as a regularized counterpart of the change of variable on the data space. Moreover, it has the great advantage of not depending on an auxiliary network as in [61]. Note that the SW-based discrepancy measure between the generative model and the data distributions can also be prescribed over the latent space by replacing the SW-based term in (2.13) by $L_{\star\text{SW}}(z, f^{-1}(x))$.

2.4.1 Logit space for images

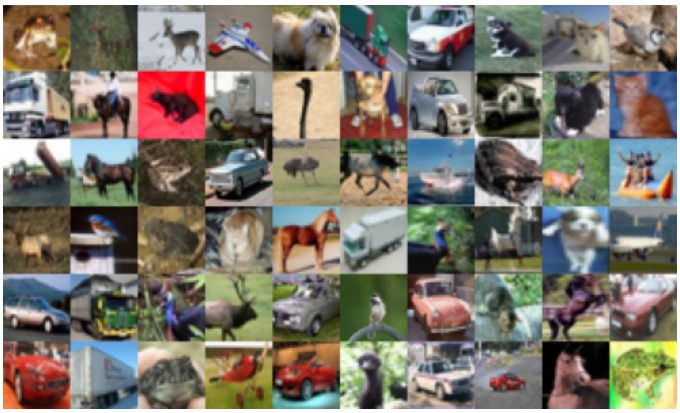
The hybrid loss proposed in (2.13) shows how to learn distributions on unbounded space. In general, the data of interest have bounded magnitude.



(a) Hybrid training



(b) MLE training



(c) True CIFAR-10

Figure 2.3: Samples from an unconditional Glow model with affine coupling layers trained on the CIFAR-10 dataset with temperature 1.0

For instance, the pixel values of an image typically lies in $[0, 256]$. Training images samples are thus drawn from a discrete distribution, while the NF model is a continuous distribution with infinite support. In order to reduce the impact of boundary effects, we instead model the density over the logit space [44]. Let x be an image of D pixels in logit space and x' be the corresponding image in $[0, 256]$ image space. The transformation from x to x' is :

$$x = \text{logit}\left(\lambda + (1 - 2\lambda)\frac{x'}{256}\right) \tag{2.14}$$

where λ is picked here at 0.05. We take into account this transformation when computing the final log likelihood in the original pixel space. Given that $p(x)$ is the density in the logit space as returned by the model, using the above transformation the density of x' can be calculated from the change of variable formula as

$$p(x') = p(x)\left(\frac{1 - 2\lambda}{256}\right)^D \left(\prod_i \sigma(x_i)\left(1 - \sigma(x_i)\right)\right)^{-1} \tag{2.15}$$

where $\sigma(\cdot)$ is the logistic sigmoid function. From that, we can calculate the neg bits per dimensions of image x as follows:

$$-\frac{\log p(x)}{D \log 2} - \log_2(1 - 2\lambda) + 8 + \frac{1}{D}\sum_i \left(\log_2 \sigma(x_i) + \log_2\left(1 - \sigma(x_i)\right)\right) \tag{2.16}$$

The above equation was used to convert between the average log likelihoods reported in the figure 2.5.

2.5 Numerical experiments

This section assesses the versatility and the accuracy of proposed SW-NF method through numerical experiments. First, experiments conducted on the Circle data set from scikit-learn are presented to provide some insights about key ingredients of the proposed approach. Then the performance of SW-NF is illustrated through more realistic experimental settings exploiting the CIFAR-10 and SVHN image data sets. It is compared to the alternative training strategies which consist in relying on the sole MLE or SW terms in (3.9) and to the Flow-GAN method which hybridizes MLE and GAN losses [61].

2.5.1 Implementation details

For all results reported below, the stochastic gradient descent is implemented in Pytorch, with the Adam optimizer, a learning rate of 10^{-4} and a batch

Objective	NLL	SW	$\|c_3\|_2^2$	$\|c_4\|_2^2$
MLE	0.52	0.0033	0.2233	5.6124
SW	1.78	**0.0007**	**0.0026**	**0.1822**
SW-Flow	**0.41**	0.0008	0.0501	0.2259
Flow-GAN [61]	0.51	1.23	0.4756	7.7725

Table 2.1: Circle data set: assessment of goodness-of-fit (for all metrics, the lower the better).

Objective	Inception	NLL (bits/dim)	CSW	$\|c_3\|_2^2$	$\|c_4\|_2^2$
MLE	2.42	3.54	1514.26	64.94	2462.37
SW	1.28	9.81	1190.11	13.13	**598.54**
SW-Flow	3.04	**3.19**	**1014.26**	**6.24**	656.37
Flow-GAN	**3.21**	4.21	1621.78	72.3	3079.12

Table 2.2: CIFAR-10 data set: assessment of goodness-of-fit (for inception score, the higher the better; for all other metrics, the lower the better).

size of 4096 or 8192 samples. When dealing with toy examples, the NF implementing the unknown mapping f is chosen as a RealNVP [36] and the conventional SW distance is chosen for hybridization. When dealing with the image-driven experiments, the network architecture is Glow [39] and the CSW is considered as a statistical distance. The NF models are composed of $M = 4$ flows, each composed of two four-layer neural networks corresponding to $\mathsf{D}_m(\cdot)$ and $\mathsf{E}_m(\cdot)$ ($d \to 8 \to 8 \to d$) (see Sec.1.4.1) using hyperbolic tangent activation function. During training, the number J of slices drawn to approximate the SW distance has been progressively increased, starting from $J = 500$ to $J = 2000$ by step of 50 slices. At each epoch, new slices are uniformly drawn over the unit sphere and 100 epochs are carried out for each number of slices. The proposed learning strategy is compared with conventional methods from two task-driven perspectives, namely goodness-of-fit and OOD detection.

Figure 2.4 shows the evolution of the negative log likelihood prescribred by the model on a test dataset during training. Not only our hybrid loss function performs better (i.e attain lower optimum) than the other approches but also stabilizes the evolution of the training. Indeed, MLE training shows multiple training instabilities when our approches demonstrate a smoother learning curve. However, the hybrid approach seems to converge slower than MLE in the first iterations.

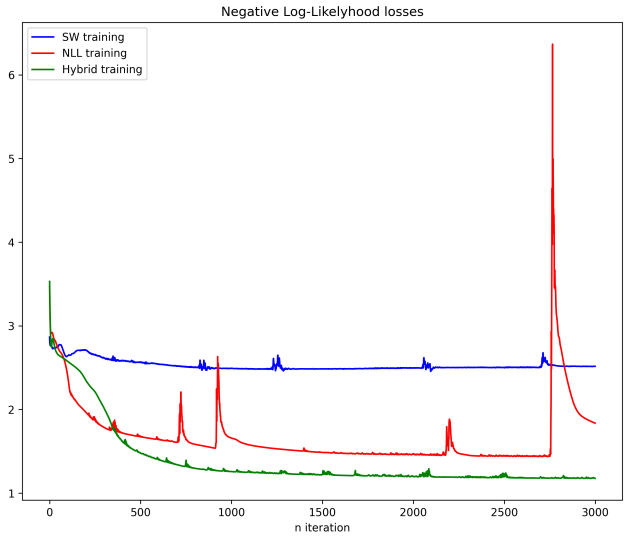


Figure 2.4: Evolution of the negative log likelihood prescribed by the model during training. In blue SW training, in red MLE trainig, in green hybrid training.

2.5.2 Goodness-of-fit

We first study the goodness-of-fit of the targeted latent space through the learned inverse transform. This evaluation is conducted by evaluating not only the negative log likelihood (NLL) but also the (C)SW distance and the 3rd- and 4th-order cumulants c_3 and c_4 defined as

$$c_3(X) = \mathrm{E}\left((X - \mathrm{E}(X))^3\right) \tag{2.17}$$

$$c_4(X) = \mathrm{E}\left((X - \mathrm{E}(X))^4\right) - 3\left(\mathrm{E}\left((X - \mathrm{E}(X))^2\right)\right)^2 \tag{2.18}$$

which are expected to be equal to zero for the prescribed normal distribution q_Z.

Table 2.1 and Table 2.2 report the results reached by the proposed SW-NF approach for the circle-shaped and the CIFAR-10 data sets, respectively. Table 2.2 also gives the inception score for visual inspection of the images. The Inception Score is a metric for automatically evaluating the quality of image generative models [62]. This metric was shown to correlate well with human scoring of the realism of generated images from the CIFAR-10 dataset. The IS uses an Inception v3 Network pre-trained on ImageNet and calculates a statistic of the network's outputs when applied to generated images.

Interestingly, the proposed SW-NF method provides significantly better NLL scores than the sole MLE-based learning strategy. For the two data sets,

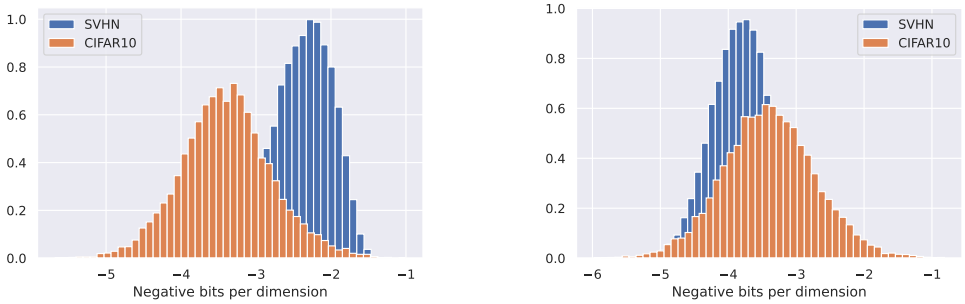
it also leads to competitive results in terms of (C)SW and normality features, reaching scores close to SW-based learning. In addition, when considering the CIFAR-10 images, it leads to higher inception score, thus suggesting higher visual quality of the generated samples as demonstrated on the samples from both training methods in Fig.2.3.

Method	Moons	Blobs
MLE	0.37	0.54
SW	**0.61**	0.70
SW-NF	0.53	**0.73**

Table 2.3: Circle data set: performance of OOD in term of AUROC (the closer to 1 the better).

2.5.3 Out-of-distribution detection

The second set of experiments assesses the ability of detecting OOD data. The classification decision is made from the model log-likelihood, more precisely when the likelihood is below 0.4 the data points is consider OOD and in distribution otherwise. Table 2.3 reports the area under the receiving operator characteristics (AUROC) for moon-shaped and blob-shaped data sets given a model trained on circles. In the absence of SW term in the training loss, the model constantly shows lower ability to discriminate OOD data from the training distribution data. In the context of CIFAR-10 data sets, the experimental setup of [56] has been considered. Glow-based NFs have been trained on CIFAR-10 and we monitor the prescribed likelihood for both CIFAR-10 and SVHN images. Note that this likelihood is in bits per dimension as disced in section 2.4.1. Fig. 2.5 (a) shows the results obtained by a sole MLE-based training: this model predicts a higher likelihood of OOD data. We see that for the MLE training the prescribed likelihood for OOD data corresponding to the blue histogram is shifted on the right (higher value) of the in distribution data likelihood in orange. Fig. 2.5 (b) depicts similar plots when considering the proposed hybrid loss: it leads to lower likelihood for OOD data coming from the SVHN data set. Indeed the OOD likelihood in blue is shifted in the left of the in-distribution data in orange.



(a) MLE training (b) Hybrid training

Figure 2.5: Likelihood histogram for 1000 CIFAR-10 images (orange) and 1000 SVHN images (blue) prescribed by a CSW-NF trained on CIFAR-10 images.

2.6 Conclusion

This chapter presented a novel hybrid training strategy for normalizing flow models. The resulting hybrid loss function thus combines a Gaussian likelihood with a (convolutional) sliced-Wasserstein distance between distributions. Numerical experiments show the better performance of the proposed hybrid training procedure in terms of perceptual as well as statistical quantitative metrics. On top of that, one observes a better robustness of the out-of-distribution behavior. This works consists of a step towards the design of more powerful NF implemented as true generative models, beyond their simple use as nonlinear regressors structurally imposed by a conventional MLE training.

In the next chapter we will use NF models and the Sliced-Wasserstein statistical distance to approximate the optimal transport between two empirical distributions.

Chapter 3

Normalizing flows to learn optimal transport between empirical distributions

Contents

3.1 Relaxation of the optimal transport problem . . 58

3.1.1 Background on optimal transport 59

3.1.2 Proposed relaxation of OT 60

3.1.3 Discrete formulation 61

3.2 Normalizing flows to approximate OT 62

3.2.1 Loss function 62

3.2.2 Intermediate transports 63

3.3 Numerical experiments 64

3.3.1 Toy examples 64

3.3.2 Unsupervised word translation 68

3.4 Discussion 72

3.5 Conclusion 73

The optimal transport (OT) problem was initially formulated by the French mathematician Gaspard Monge. In his seminal paper published in 1781 [67], he raised the following question: how to move a pile of sand to a target location with the least possible effort or cost? The objective was to find the best way to minimize this cost by a transport plan, without having to list all the possible matches between the starting and ending points. More

recently, thanks to recent advances related to computational issues [65], OT has founded notable successes with respect to applications ranging from image processing and computer vision [68] to machine learning [69] and domain adaptation [70].

Normalizing flows (NFs) have also attracted a lot of interest in the machine learning community, motivated in particular by their ability to model high dimensional data [51, 71]. Motivated by the similarities between the problem of OT and the training of NF, this chapter discused a neural architecture and a corresponding training strategy that permits to learn an approximate transport plan between any two empirical distributions. The proposed framework is based on a relaxation of the Monge formulation of OT. To adapt the training loss to the flow-based structure of the network, this loss function is supplemented with a Sobolev regularisation to promote minimal efforts achieved by each flow. Numerical simulations show that this regularisation results in a smoother and more efficient trajectory. Interestingly, the discretization inherent to the flow-based structure of the network implicitly provides intermediate transports and, at the same time, Wasserstein barycenters [72].

Section 3.1 will recall the Monge formulation of OT and proposes a relaxation in the case of a transport between two empirical distributions. Section 3.2 presents the generic framework based on NFs and describes a particular instance to solve the OT problem. Section 3.3 presents some experimental results illustrating the performance of the proposed method. Conclusions and prospects are reported in Section 5.5.

The major part of this chapter has been published and presented at two conferences:

- F. Coeurdoux, N. Dobigeon, P. Chainais. *"Learning Optimal Transport Between two Empirical Distributions with Normalizing Flows"*. ECML-PKDD 2022 arXiv:2207.01246
- F. Coeurdoux, N. Dobigeon, P. Chainais. *"Approximation du transport optimal entre distributions empiriques par flux de normalisation"*. GRETSI Septembre 2022 Nancy, hal-03704666

3.1 Relaxation of the optimal transport problem

Let μ and ν be two probability measures with finite second order moments. More general measures, for example on $\mathcal{X} = \mathbb{R}^d$ (where $d \in \mathbb{N}^*$ is the di-

mension), can have a density $\mathrm{d}\mu(x) = p_X(x)dx$ and $\mathrm{d}\nu(y) = p_Y(y)dy$ with respect to the Lebesgue measure, often noted $p_X = \frac{\mathrm{d}\mu}{\mathrm{d}x}$ and $p_Y = \frac{\mathrm{d}\nu}{\mathrm{d}y}$, which means that

$$\forall h \in \mathcal{C}\left(\mathbb{R}^d\right), \quad \int_{\mathbb{R}^d} h(x)\mathrm{d}\mu(x) = \int_{\mathbb{R}^d} h(x)p_X(x)\mathrm{d}x \tag{3.1}$$

where $\mathcal{C}(\cdot)$ is the class of continuous functions. In the remainder of this Chapter, $\mathrm{d}\mu(x)$ and $p_X(x)\mathrm{d}x$ will be used interchangeably as well as $\mathrm{d}\nu(y)$ and $p_Y(y)\mathrm{d}y$.

3.1.1 Background on optimal transport

Let consider $\mathcal{X}$ and $\mathcal{Y}$ two separable metric spaces. Any measurable application $f : \mathcal{X} \to \mathcal{Y}$ can be extended to the so-called push-forward operator $f_\sharp$ which moves a probability measure on $\mathcal{X}$ to a new probability measure on $\mathcal{Y}$. For any measure μ on $\mathcal{X}$, one defines the image measure $\nu = f_\sharp\mu$ on $\mathcal{Y}$ such that

$$\forall h \in \mathcal{C}(\mathcal{Y}), \quad \int_{\mathcal{Y}} h(y)\mathrm{d}\nu(y) = \int_{\mathcal{X}} h(f(x))\mathrm{d}\mu(x). \tag{3.2}$$

Intuitively, the application $f : \mathcal{X} \to \mathcal{Y}$ can be interpreted as a function moving a single point from one measurable space to another [65]. The operator $f_\sharp$ pushes each elementary mass of a measure μ on $\mathcal{X}$ by applying the function f to obtain an elementary mass in $\mathcal{Y}$. The problem of OT as formulated by Monge is now stated in a general framework. For a given cost function $c : \mathcal{X} \times \mathcal{Y} \to [0, +\infty]$, the measurable application $f : \mathcal{X} \to \mathcal{Y}$ is called the OT plan from a measure μ to the image measure $\nu = f_\#\mu$ if it reaches the infimum

$$\inf_f \left\{ \int_{\mathcal{X}} c(x, f(x))\mathrm{d}\mu(x) : f_\sharp\mu = \nu \right\}. \tag{3.3}$$

Alternatively the Kantorovitch formulation of OT results from a convex relaxation of the Monge problem (3.3). By defining Π as the set of all probabilistic couplings with marginals μ and ν, it yields the optimal π that reaches

$$\min_{\pi \in \Pi} \int_{\mathcal{X}\times\mathcal{Y}} c\left(\mathbf{x}, \mathbf{y}\right) d\pi\left(\mathbf{x}, \mathbf{y}\right) \tag{3.4}$$

Under this formulation, the optimal π, which is a joint probability measure with marginals μ and ν, can be interpreted as the optimal transportation plan. It allows the Wasserstein distance of order p between μ and ν to be defined as

$$W_p\left(\mu, \nu\right) \stackrel{\text{def}}{=} \inf_{\pi \in \Pi} \left\{ \left(\mathop{\mathbb{E}}_{(x,y)\sim\pi} d\left(\mathbf{x}, \mathbf{y}\right)^p \right)^{\frac{1}{p}} \right\} \tag{3.5}$$

where $d(\cdot,\cdot)$ is a distance defining the cost function $c(\mathbf{x},\mathbf{y}) = d(\mathbf{x},\mathbf{y})^p$. The Wasserstein distance is also known as the Earth mover's distance in the computer vision community. It defines a metric over the space of square integrable probability measures.

3.1.2 Proposed relaxation of OT

OT boils down to a variational problem, i.e., it requires the minimization of an integral criterion in a class of admissible functions. Given two probability measures μ and ν, the existence and uniqueness of an operator f that belongs to the class of bijective, continuous and differentiable functions such that $f_\sharp\mu = \nu$ is not guaranteed. The difficulty lies in the class defining these admissible functions. Indeed, even when μ and ν are regular densities on regular subsets of $\mathbb{R}^d$, the search for a transport plan such that $f_\sharp\mu = \nu$ makes the problem (3.3) difficult in a general case. To overcome the difficulty of solving this equation on $f_\sharp$, we propose to reformulate the Monge's OT statement by relaxing the equality on the operator defining the image measure.

More precisely, the equality between the image measure $f_\sharp\mu$ and the target measure ν is replaced by the minimization of their statistical distance $d(f_\sharp\mu,\nu)$. The choice of the distance $d(\cdot,\cdot)$ is crucial because it determines the quality of the approximation of the image measure by the transport plan f. In this work, we propose to choose $d(\cdot,\cdot)$ as the Wasserstein distance $W_p(\cdot,\cdot)$. This choice will be motivated by the fact that this distance can be easily approximated numerically without explicit knowledge of the probability distributions μ and ν, in particular when they are empirically described by samples only. The relaxation of the Monge problem (3.3) can then be written as

$$\inf_f \left\{ W_p(f_\sharp\mu,\nu) + \lambda \int_{\mathcal{X}} c(x,f(x))\mathrm{d}\mu(x) \right\} \tag{3.6}$$

where the cost function defined in (3.3) is interpreted here as a regularisation term adjusted by the hyperparameter λ.

remark *The relaxed formulation* (3.6) *relies on the Wasserstein distance between the target measure ν and the image measure $f_\sharp\mu$. This term should not be confused with the Wasserstein distance $W_p(\mu,\nu)$ which is the infimum reached by the solution of the Kantorovitch's formulation of OT* (3.4).

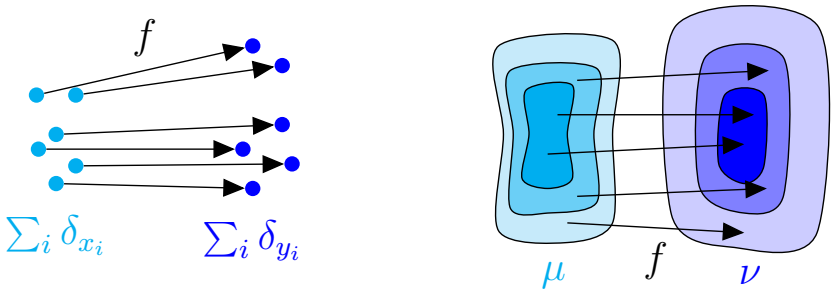


Figure 3.1: Comparison of the push-forward operator $f_\sharp$, which can take as an input any measure

3.1.3 Discrete formulation

In a machine learning context, the underlying continuous measures are conventionally approximated by empirical point measures thanks to available data samples. Therefore, we are interested in discrete measures and the empirical formulation of the OT problem. Figure 3.1 illustrates the distinction between these discrete and continuous formulation. Within this framework, we will consider μ and ν two discrete measures described by the respective samples $\mathbf{x} = \{x_n\}_{n=1}^N$ and $\mathbf{y} = \{y_n\}_{n=1}^N$ such that $\mu = \frac{1}{N}\sum_{n=1}^N \delta_{x_n}$ and $\nu = \frac{1}{N}\sum_{n=1}^N \delta_{y_n}$. In the following, an empirical version of the criterion (3.6) is proposed in the case of discrete measures.

The formulation (3.6) requires the evaluation of a Wasserstein distance whose computation is not trivial in its original form, especially in high dimension. An alternative consists in considering its rewriting in the form of the *Sliced-Wasserstein* (SW) distance discussed in Section 2.3. Depending on the application any implementation of the SW might be used, we will denote $\widehat{SW}_p(\cdot,\cdot)$ any Monte Carlo SW between two mathematical objects of same dimension, i.e, Eq (2.9) or (2.12). The empirical form of the relaxation of the Monge problem (3.6) is then written as

$$\min_f \left\{ \widehat{SW}_p(f(\mathbf{x}), \mathbf{y}) + \lambda \sum_{n=1}^N c\left(x_n, f\left(x_n\right)\right) \right\} \tag{3.7}$$

where, with a slight abuse of notations, $f(\mathbf{x}) \triangleq \{f(x_n)\}_{n=1}^N$.

3.2 Normalizing flows to approximate OT

This section proposes to solve the problem (3.7) by restricting the class of the operator f to a class of invertible deep networks referred to as normalisation flows. The strategy proposed to train these networks to solve the problem (3.7) is then detailed in paragraph 3.2.1.

3.2.1 Loss function

As mentioned before, the objective of this work is to learn a bijective operator relating any two distributions p_X and p_Y described by samples $\mathbf{x}$ and $\mathbf{y}$. The search for this operator is restricted to the class of invertible deep networks f described in Section 1.4.1. The conventional strategy to train the network would be to maximize the likelihood defined by (2.5). However this approach cannot be implemented in the context of interest here since the base distribution p_Y is no longer explicitly given: it is only available through the knowledge of the set of samples $\mathbf{y}$. As a consequence, to adjust the weights of the network, the proposed alternative interprets the underlying learning task as the search for a transport plan. Then a first idea would be to adjust these weights by directly solving the problem (3.7). However, to take advantage of the flow-based architecture of the operator $f(\cdot)$, it seems legitimate to equally distribute the transport efforts provided by each flow. Thus, the regularization in (3.7) will be instantiated for each elementary transformation $f_m(\cdot)$ associated to each flow of the network.

Moreover, when fitting deep learning-based models a major challenge arises from the stochastic nature of the optimization procedure, which imposes to use partial information (i.e., mini-batches) to infer the whole structure of the optimization landscape. On top of that, the cost function to be optimized is not numerically constant since the approximation $\widehat{SW}$ of the SW distance in (3.7) depends on the precise set of random vectors $\{u_j\}_{j=1}^{J}$ drawn over the unit sphere. To alleviate these optimization difficulties, we propose to further regularize the objective function by penalizing the energy $|J_{f_m}(\cdot)|^2$ of the Jacobians associated with the transformations $f_m(\cdot)$, $m = 1, \ldots, M$. These Sobolev-like penalties promote regular operators $f_m(\cdot)$, promoting an overall operator $f(\cdot)$ regular itself. In the context of optimal transport, this regularization has already been studied in depth in [73]. In that work, the author focused on the penalization of the Monge's formulation of OT by the ℓ_2-norm of the Jacobian. It stated the existence of an optimal transport plan f solving the minimization problem

$$\inf_f \left\{ \int_{\mathcal{X}} \left(|f(x) - x|^2 + \gamma |J_f|^2 \right) f(x) \mathrm{d}x : f_{\#}\mu = \nu \right\}. \tag{3.8}$$

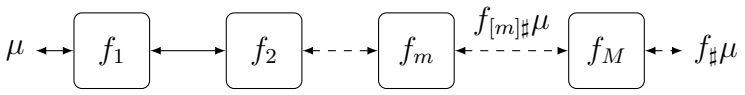


Figure 3.2: Architecture of the proposed SWOT-Flow.

This formulation of OT imposes the transport plan f to be regular rather than deducing its regularity from its optimal properties.

Finally, the training of the NF is carried out by minimizing the loss function

$$\underbrace{\widehat{SW}_p(f(\mathbf{x}),\mathbf{y})}_{\text{SW}} + \underbrace{\sum_{n=1}^{N}\sum_{m=1}^{M}\left[\lambda c(f_{m-1}(x_n), f_m(x_n)) + \gamma \left|J_{f_m}(x_n)\right|^2\right]}_{\text{Reg}} \tag{3.9}$$

with $f_0(x_n) = x_n$. The proposed networt, whose general architecture is depicted in Fig. 3.2, will be referred to as SWOT-Flow in what follows.

3.2.2 Intermediate transports

As a consequence of the multiple-flow architecture of the NF, the transport plan operated by the proposed SWOT-Flow is a composition of the M individual flows $f_m(\cdot)$ $(m = 1, \ldots, M)$. Thus each flow implements an elementary transport and the composition of the first m flows defined as

$$f_{[m]}(\cdot) \triangleq f_m \circ \ldots \circ f_1(\cdot) \tag{3.10}$$

can be interpreted as an intermediate step of the transport plan from the input measure μ towards the target measure ν, with $f_{[M]}(\cdot) \triangleq f(\cdot)$. Interestingly, these intermediate transports can be related to Wasserstein barycenters between μ and ν defined by [72]

$$\inf_{\beta} \left\{\alpha W_p(\mu,\beta) + (1-\alpha) W_p(\beta,\nu)\right\}. \tag{3.11}$$

Indeed, the next section dedicated to numerical experiments will empirically show that $f_{[m]\sharp}\mu$ approaches the solution of the problem (3.11) for the specific choice of the weight $\alpha = \frac{m}{M}$. In other words, the image measures provided by each intermediate transport operated by SWOT-Flow, i.e., as the outputs of each of the M flows, can legitimately be interpreted as Wasserstein barycenters.

3.3 Numerical experiments

This section assesses the versatility and the accuracy of SWOT-Flow through two sets of numerical experiments. First, several toy experiments are presented to provide some insights about key ingredients of the proposed approach. Then the performance of SWOT-Flow is illustrated through the more realistic and challenging task of unsupervised alignment of word embeddings in natural language processing. The source code is publicly available on GitHub [1].

3.3.1 Toy examples

In these experiments, the proposed framework SWOT-Flow is implemented and tested with synthetic data. In all experiments, the input distributions are described by the respective samples $\mathbf{x} = \{x_n\}_{n=1}^N$ and $\mathbf{y} = \{y_n\}_{n=1}^N$ such that $\mu = \frac{1}{N}\sum_{n=1}^N \delta_{x_n}$ and $\nu = \frac{1}{N}\sum_{n=1}^N \delta_{y_n}$ with $N = 20000$. The cost function $c(\cdot,\cdot)$ is chosen as the squared Euclidean distance, i.e., $c(x,y) = \|x-y\|_2^2$. However, it is worth noting that the proposed method is not limited to this Euclidean distance and can handle other costs.

Implementation details.

The stochastic gradient descent used to solve (3.9) is implemented in Pytorch. We use Adam optimizer with learning rate 10^{-4} and a batch size of 4096 or 8192 samples. The NF implementing $f(\cdot)$ is a RealNVP [38] for the example of Fig. 3.3 and an ActNorm type architecture network [39] for Fig. 3.4 and Fig. 3.5. It is composed of $M = 4$ flows, each composed of two four-layer neural networks corresponding to $\mathsf{D}_m(\cdot)$ and $\mathsf{E}_m(\cdot)$ ($d \to 8 \to 8 \to d$) using hyperbolic tangent activation function. During training, the number J of slices drawn to approximate the SW distance in (2.9) has been progressively increased, starting from $J = 500$ to $J = 2000$ by step of 50 slices. At each epoch, new slices are uniformly drawn over the unit sphere and 100 epochs are carried out for each number of slices. The training procedure consists in 1) defining the loss function as the sole SW term in (3.9) from $J = 500$ to 1500 slices and then 2) incorporating the regularization term denoted as Reg in (3.9) where hyperparameters λ and γ are increased by a factor of 5% every step of 100 slices.

[1] FlorentinCDX/SWOT-Flow

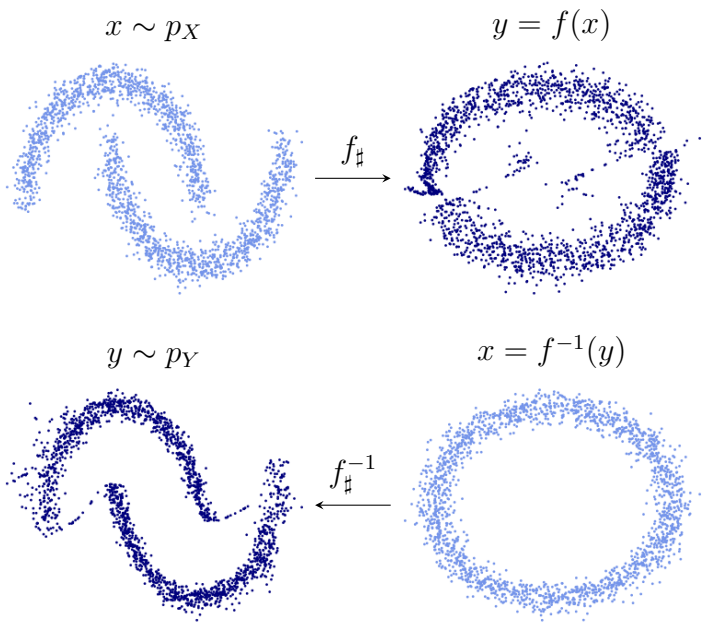


Figure 3.3: Operator f learnt by SWOT-Flow when the base distribution p_X is a double-moon (top left) and the target distribution p_Y is a circle (bottom right).

Qualitative results.

As a first illustration of the flexibility of the proposed approach, Fig. 3.3 shows the results obtained after learning an operator f that transports a double moon-shaped distribution p_X (top left) to a circle-shaped distribution (bottom right). The empirical image measures $f_\sharp p_X$ (top right) and $f_\sharp^{-1} p_Y$ (bottom left) are obtained by applying the estimated $f(\cdot)$ operator or its inverse $f^{-1}(\cdot)$. It is worth noting that the difficulty inherent to this experiment lies in the respective disjoint and non-disjoint supports of the two distributions. The difficulty of NF models to learn bijective map between topologically different probability measure is known and will be discussed in the next chapter. Despite the regularity of the trained NF, a very good approximation of the OT is learnt, even in presence of this topological change.

Fig. 3.4 aims at illustrating the relevance of the Sobolev-like regularization (i.e., the ℓ_2-norm of the Jacobian) included into the loss function (3.9) defined to train the NF. The first simulation protocol considers circle-shaped distributions while the second case considers rectangle-shaped distributions. In what follows, these two cases will be referred to as $\mathcal{P}_1$ and $\mathcal{P}_2$, respectively. In this experiment, the objective is to learn the transport plan from an initial distribution p_X (light blue) to a target distribution p_Y (dark blue) which is

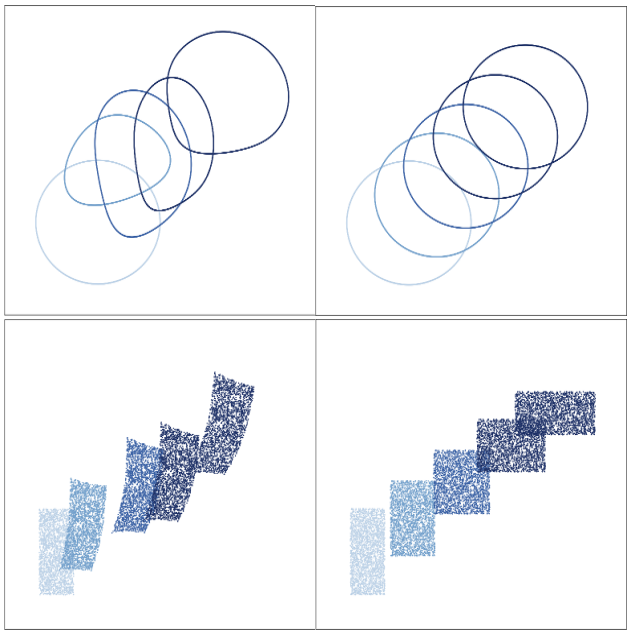


Figure 3.4: Elementary transports achieved by the proposed NF when trained without (left) or with (right) the regularization for protocols $\mathcal{P}_1$ (top) and $\mathcal{P}_2$ (bottom).

translated for $\mathcal{P}_1$ and both translated and stretched for $\mathcal{P}_2$. The color gradient shows the outputs of the M successive flows of the network, i.e. the image measures $f_{[m]\sharp}p_X$ for $m = 1, \ldots, M$. In the absence of regularization (left), the successive elementary transports clearly suffer from multiple unexpected deformations (superfluous translations and dilations). In contrast, when the loss is complemented with the proposed Sobolev-type penalty (right), the learnt operator f is decomposed as a sequence of much more regular elementary transports. The resulting transport appears to be very close to optimal. In case $\mathcal{P}_1$, the expected translation is recovered, as well as the combined translation and stretching in case $\mathcal{P}_2$.

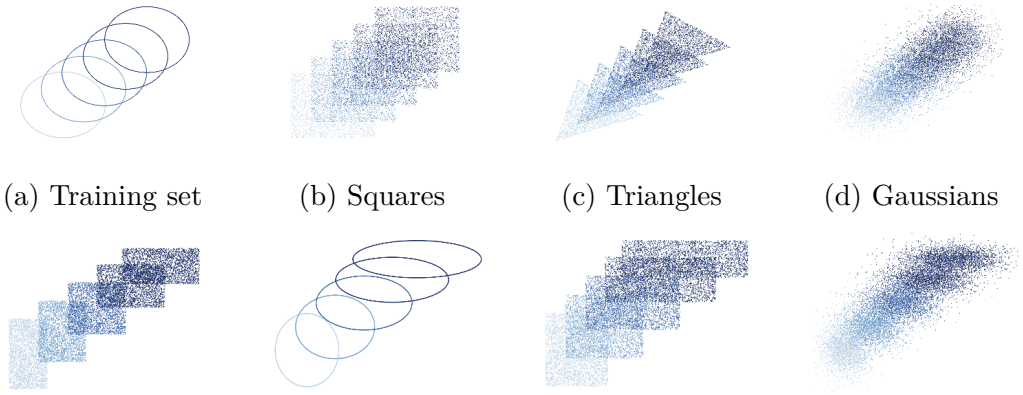
To be more precise quantitatively, Table 3.1 compares some metrics obtained when the NF has been trained using the regularization-free or regularized loss function, as defined in (3.9). For the two aforementioned simulation protocols, it reports the elementary costs

$$\bar{c}_m = \frac{1}{N} \sum_{n=1}^{N} \|f_{m-1}(x_n) - f_m(x_n)\|_2^2 \tag{3.12}$$

spent by each of the M flows $f_1(\cdot), \ldots, f_M(\cdot)$ to achieve the transport plans retrieved by SWOT-Flow. This table (last column) also reports the overall

Table 3.1: Overall cost $\bar{C}$ and elementary costs $\bar{c}_m$ required by each flow $f_m(\cdot)$ of the NF trained with or without (w/o) regularization for protocols $\mathcal{P}_1$ (circle-shaped distributions) and $\mathcal{P}_2$ (rectangle-shaped distributions).

		$\bar{c}_1$	$\bar{c}_2$	$\bar{c}_3$	$\bar{c}_4$	$\bar{C}$
$\mathcal{P}_1$	w/o regularization	150.13	110.94	108.41	151.65	521.12
	with regularization	90.20	90.70	90.71	90.22	361.22
$\mathcal{P}_2$	w/o regularization	154.99	98.67	52.49	101.21	407.38
	with regularization	88.77	89.42	89.43	89.38	357.0



(a) Training set (b) Squares (c) Triangles (d) Gaussians

(e) Training set (f) Circles (g) Squares (h) Gaussians

Figure 3.5: Examples of transported data sets for protocols $\mathcal{P}_1$ (top) and $\mathcal{P}_2$ (bottom).

cost $\bar{C} = \sum_{m=1}^{M} \bar{c}_m$. For the two simulation protocols $\mathcal{P}_1$ and $\mathcal{P}_2$, these results clearly show cheaper transports when using the proposed regularization. For instance, for the simulation protocol $\mathcal{P}_1$, the overall cost is $\bar{C} = 360$ with the regularization, compared to $\bar{C} = 520$ when it is omitted. Moreover, when using the regularized loss function, this cost is distributed homogeneously over the successive flows, with a variation of at most $\pm 1\%$ from one flow to another, against $\pm 20\%$ otherwise.

Fig. 3.5 aims at illustrating the capacity of generalization of the transport plan learnt by SWOT-Flow. In this experiment, SWOT-Flow has been trained following the simulation protocols $\mathcal{P}_1$ (Fig. 3.5a) or $\mathcal{P}_2$ (Fig. 3.5e). Once trained on the data set associated with each protocol, the NFs are fed with differently shaped data and the elementary transports are monitored

as above. Fig 3.5b-3.5d and 3.5g-3.5h show the results when using square-, triangle-, Gaussian-shaped data sets for both protocols, respectively. As expected, all initial distributions are either simply translated in case $\mathcal{P}_1$ or translated and stretched in case $\mathcal{P}_2$. The intermediate distributions correspond to the expected barycenters as well. Fig. 3.5 clearly demonstrates the generalization capacity of the proposed approach.

Multivariate Gaussians with varying dimensions.

When the source and target distributions μ and ν of a transportation problem are multivariate Gaussians, the Wasserstein barycenters defined by (3.11) are also multivariate Gaussian distributions. In this case, an efficient fixed-point algorithm can be used to estimate its mean vector $\mathbf{a}$ and covariance matrix $\mathbf{\Sigma}$ [74]. This experiment capitalizes on this finding to assess the ability of SWOT-Flow to approximate Wasserstein barycenters, as stated in Section 3.2.2. To this end, the algorithm designed in [74] is implemented to estimate the actual barycenter associated with two prescribed multivariate Gaussian distributions for $\alpha = 1 - \alpha = \frac{1}{2}$. This barycenter is compared to the image measure $f_{[m]\sharp}\mu$ estimated by SWOT-Flow with $m = \frac{M}{2}$. More precisely, the mean vector and the covariance matrix of the barycenter are compared to their maximum likelihood estimates $\hat{\mathbf{a}}$ and $\hat{\mathbf{\Sigma}}$ computed from the samples $\{f_{[m]}(x_n)\}_{n=1}^N$ transported by the first m flows. The resulting mean square errors (MSEs)

$$\mathrm{MSE}(\mathbf{a}) = \|\mathbf{a} - \hat{\mathbf{a}}\|_2^2 \quad \text{and} \quad \mathrm{MSE}(\mathbf{\Sigma}) = \|\mathbf{\Sigma} - \hat{\mathbf{\Sigma}}\|_{\mathrm{F}}^2 \tag{3.13}$$

are reported in Table 3.2 for varying dimensions ranging from 2 to 8. This table also reports the MSEs reached by other state-of-the-art free-support methods [75, 76, 77]. For the methods [75] and [76], $n = 5000$ and $n = 100$ support points have been used, respectively, since these are the maximum numbers allowed for the algorithms to terminate in reasonable computational times. SWOT-Flow compares favorably to state-of-the-art methods since reported MSEs in Table 3.2 appear to be most often the smallest. These observation may call for a more general study, but remains noticeable since SWOT-Flow has not been specifically designed to compute the Wasserstein barycenters, contrary to alternate methods.

3.3.2 Unsupervised word translation

In a second set of experiments, the performance of SWOT-Flow has been assessed on the task of unsupervised word translation. Given word embed-

Table 3.2: Performance of the estimation of the median barycenters. Reported scores result from the average over 5 Monte Carlo runs.

			[75]	[76]	[77]	SWOT-Flow
Dimension	2	MSE($\mathbf{a}$)	$9.99 \cdot 10^{-5}$	$3.14 \cdot 10^{-4}$	$1.17 \cdot 10^{-4}$	$\mathbf{8.09 \cdot 10^{-5}}$
		MSE($\mathbf{\Sigma}$)	$7.28 \cdot 10^{-4}$	$2.39 \cdot 10^{-3}$	$1.98 \cdot 10^{-3}$	$\mathbf{1.44 \cdot 10^{-4}}$
	4	MSE($\mathbf{a}$)	$1.73 \cdot 10^{-3}$	$1.68 \cdot 10^{-3}$	$1.44 \cdot 10^{-3}$	$\mathbf{1.44 \cdot 10^{-4}}$
		MSE($\mathbf{\Sigma}$)	$1.35 \cdot 10^{-2}$	$2.50 \cdot 10^{-2}$	$1.22 \cdot 10^{-2}$	$\mathbf{3.61 \cdot 10^{-4}}$
	6	MSE($\mathbf{a}$)	$2.04 \cdot 10^{-3}$	$\mathbf{2.58 \cdot 10^{-3}}$	$3.24 \cdot 10^{-3}$	$1.23 \cdot 10^{-2}$
		MSE($\mathbf{\Sigma}$)	$4.38 \cdot 10^{-2}$	$8.86 \cdot 10^{-2}$	$2.37 \cdot 10^{-2}$	$\mathbf{5.29 \cdot 10^{-4}}$
	8	MSE($\mathbf{a}$)	$\mathbf{1.23 \cdot 10^{-3}}$	$1.48 \cdot 10^{-3}$	$3.14 \cdot 10^{-3}$	$1.29 \cdot 10^{-2}$
		MSE($\mathbf{\Sigma}$)	$8.31 \cdot 10^{-2}$	$1.64 \cdot 10^{-1}$	$4.23 \cdot 10^{-2}$	$\mathbf{2.22 \cdot 10^{-3}}$

dings trained on two monolingual corpora, the goal is to infer a bilingual dictionary by aligning the corresponding word vectors.

Experiment description.

This experiment considers the task of aligning two sets of points in high dimension. More precisely, it aims at inferring a bilingual lexicon, without supervision, by aligning word embeddings trained on monolingual data. FastText [78] has been implemented to learn the word vectors used for representation. It provides monolingual embeddings of dimension 300 trained on Wikipedia corpora. Words are lower-cased, and those that appear less than 5 times are discarded for training. As a post-processing step, only the first 50k most frequent words are selected in the reported experiments.

Architecture.

The proposed SWOT-Flow method has been implemented using a RealNVP architecture. The scale function $\mathsf{E}_m(\cdot)$ and the offset function $\mathsf{D}_m(\cdot)$ (See Sec.1.4.1) are multilayer neural networks with two hidden layers of size 512 and hyperbolic tangent activation function. Adam has been used as an optimizer with a learning rate of $1 \cdot 10^{-3}$. The number of slices involved in the Monte Carlo approximation of the SW distance in (2.9) has been progressively increased from $J = 500$ slices to $J = 3000$ by steps of 50. For each number of slices, 100 epochs have been performed. The hyperparameters λ and γ adjusting the weights of the composite regularization have been increased by a factor of 5% every steps of 500 slices.

Table 3.3: Comparison of accuracies obtained by SWOT-Flow and adv-net [79] for unsupervised word translation ('en' is English, 'fr' is French, 'de' is German, 'ru' is Russian).

Method		en-es	es-en	en-fr	fr-en	en-de	de-en	en-ru	ru-en
SWOT-Flow	20-NN	37.4	24.2	46.6	34.1	44.4	27.6	14.4	3.8
	10-NN	**33.5**	**22.5**	**42.5**	32.5	39.5	26.8	**10.2**	2.1
adv-net [79]	10-NN	31.4	21.2	39.6	**35.1**	**40.1**	**27.1**	7.1	**2.3**

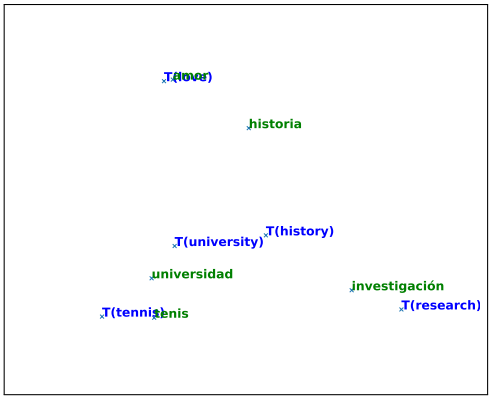
Main results.

To quantitatively measure the quality of SWOT-Flow, the problem of bilingual lexicon induction is addressed, with the same setting as in [79]. The same evaluation data sets and codes, as well as the same word vectors have been used. Given an input word embedding ($n = 1, \ldots, N$ with $N_{\text{test}} = 1000$) in a given language, the objective is to assess if its counterpart $f(x_n)$ transported by SWOT-Flow belongs to the close neighborhood of the output word embedding y_n in the target language. The neighborhood $\mathcal{V}(y_n)$ is defined as the set of K-nearest neighbors computed in a cosine similarity sense with $K = 10$ or 20 in dimension 300. The overall accuracy is computed as the percentage of correctly transported input samples. Denoting by $\mathbf{1}_{\mathsf{A}}$ the indicator function, i.e., $\mathbf{1}_{\mathsf{A}} = 1$ if the assertion A is true and $\mathbf{1}_{\mathsf{A}} = 0$ otherwise,

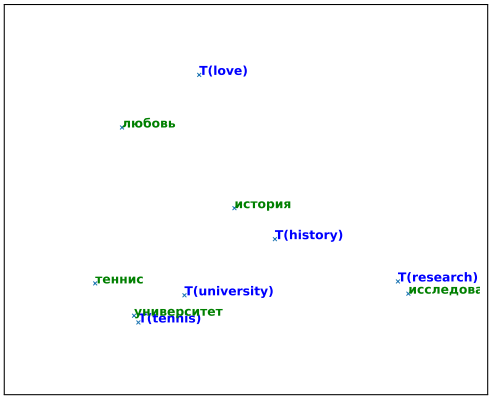
$$\text{accuracy} = \frac{1}{N_{\text{test}}} \sum_{n=1}^{N_{\text{test}}} \mathbf{1}_{\{f(x_n) \in \mathcal{V}(y_n)\}} \times 100 \ (\%) \tag{3.14}$$

Table 3.3 reports the accuracy scores for several pairs of languages. Although SWOT-Flow has not been specifically designed to perform word translation, these results show that its overall performance is on par with the adversarial network (adv-net) proposed specifically for this task in [79]. In particular, SWOT-Flow seems to perform well for translation between languages with close origins.

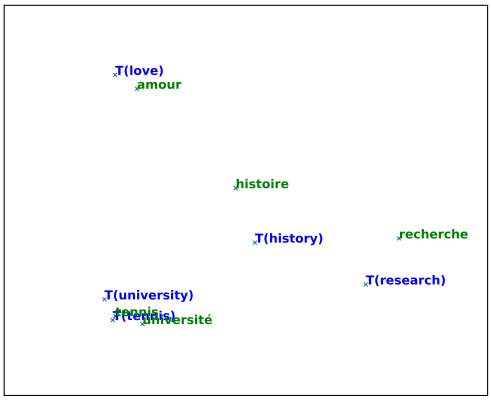
Fig. 3.6 qualitatively illustrates this good performance by showing how close a set of translated words $f(x_n)$ are to their true translation y_n. This representation is obtained by a classical projection on the 2 first PCA components of the target embedded space. The translation of 5 specific words from English to French or German fall in the close vicinity of their true counterparts.



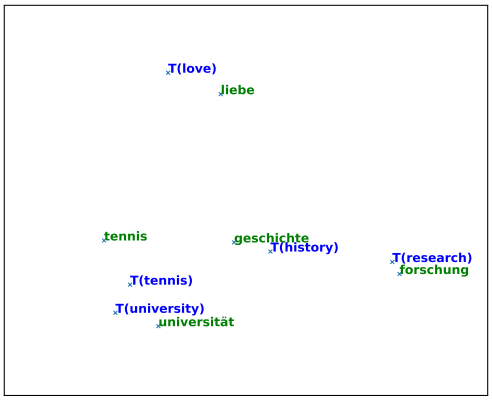
(a) english → spanish



(b) english → russian



(c) english → french



(d) english → german

Figure 3.6: 2D PCA representation of the target word embedding space: the targeted translated (in green) and the transported source (in blue) embedded words.

3.4 Discussion

Cycle consistency.

Cycle consistency, as proposed in CycleGAN [80], aims at learning meaningful cross-domain mappings such that the data translated from the domain $\mathcal{X}$ to the domain $\mathcal{Y}$ via $f_{\mathcal{X}\to\mathcal{Y}}$ can be mapped back to the original data points in $\mathcal{X}$ via $f_{\mathcal{Y}\to\mathcal{X}}$. That is, $f_{\mathcal{Y}\to\mathcal{X}} \circ f_{\mathcal{X}\to\mathcal{Y}}(x) \approx x$ for all $x \in \mathcal{X}$. For CycleGan, and many other domain transfer models such as [69], this key property should be enforced by including a cycle consistency term into the loss function. Conversely, since NF-based generative models learn bijective mappings, the proposed SWOT-Flow inherits the cycle consistency property by construction.

Semi-discrete formulation.

The proposed SWOT-Flow framework has been explicitly derived to approximate OT between two discrete empirical distributions. It can be instanciated to perform semi-discrete OT, i.e., to handle the case where one of distribution is not described by data points but rather given as an explicit continuous probability measure. Instead of relaxing the Monge formulation (3.3) as in (3.6), it would consist in replacing the SW distance with a log-likelihood term $\log p_Y(\cdot)$ associated with the target continuous measure ν. The loss function in (3.9) would be replaced by

$$-\sum_{n=1}^{N} \log p_Y(f(x_n)) + \sum_{n=1}^{N}\sum_{m=1}^{M} \left[\lambda c(f_{m-1}(x_n), f_m(x_n)) + \gamma \left|J_{f_m}(x_n)\right|^2\right] \quad (3.15)$$

where the log-likelihood term is evaluated at the data points $\{f(x_n)\}_{n=1}^N$ transported by the NF.

NF to approximate barycenters.

As discussed in Section 3.2.2 and experimentally illustrated in Section 3.3.1, the flow-based architecture of the SWOT-Flow network leads to intermediate transports, that can be related to Wasserstein barycenters. On the toy Gaussian example considered in Section 3.3.1, SWOT-Flow provides good approximation of the barycenters, although it has not been specifically designed to perform this task. If one is interested in devising a NF approximating these barycenters, the definition (3.11) would lead to the optimization problem

$$\inf_f \left\{ \sum_{m=1}^{M} \alpha_m W_p(\mu, f_{[m]\sharp}\mu) + (1-\alpha_m) W_p(f_{[m]\sharp}\mu, \nu) \right\} \quad (3.16)$$

with $\alpha_m = \frac{m}{M}$. When handling empirical measures described by samples, the subsequent discretization would require to replace both terms with Monte Carlo approximations (2.9) of the SW distances. However, this would lead to a computationally demanding training procedure.

3.5 Conclusion

We propose a new method to learn the optimal transport plan between two empirical distributions from sets of available samples. To this aim, we write a relaxed and penalized formulation of the Monge problem. This formulation is used to build a loss function that balances between the cost of the transport and the proximity in Wasserstein distance between the transported base distribution and the target one. The proposed approach relies on normalizing flows, a family of invertible neural networks. As a side benefit, the multiple flow architecture of the proposed network interestingly yields intermediate transports and Wasserstein barycenters. The proposed method is illustrated by numerical experiments on toy examples as well as an unsupervised word translation task. Up to our knowledge, this is the first method that is able to learn such a generalizable transport operator.

Chapter 4

Normalizing flow sampling with Langevin dynamics in the latent space

Contents

4.1 Problem statement 77
4.1.1 Learning of change of variable 78
4.1.2 A Gaussian latent space? 79
4.1.3 Beyond conventional NF sampling 79
4.2 Related works 80
4.3 Implications of a topological mismatch 81
4.3.1 Topology preservation 81
4.3.2 Partition of the latent space 83
4.4 NF sampling in the latent space 84
4.4.1 Difficulties of sampling the target space 84
4.4.2 Latent diffusion equation 85
4.5 Efficient implementation 89
4.5.1 Fast approximation of the proposal move 90
4.5.2 Fast computation of the latent score 90
4.6 NF-SAILS 91
4.6.1 Local exploration with MALA 91
4.6.2 Global exploration with I-MH 92
4.6.3 Local/Global exploration 92

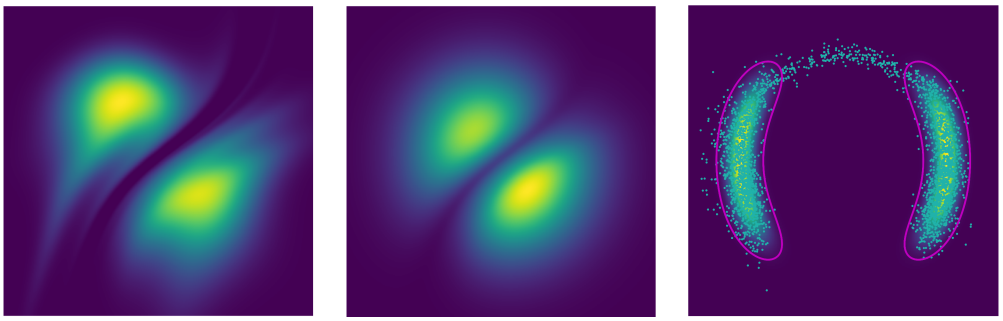
4.7 Experiments 93
4.7.1 Figures-of-merit 94
4.7.2 Results obtained on synthetic dataset 94
4.7.3 Results obtained on real image datasets 96
4.8 Conclusion 97

We saw in Chapter 2 and 3 that NF use a continuous generator to map a simple latent (e.g. Gaussian) distribution, towards an empirical target distribution associated with a training data set. Once trained by minimizing a variational objective, the learnt map provides an approximate generative model of the target distribution. Since standard NF implement differentiable maps, they may suffer from pathological behaviors when targeting complex distributions. For instance, such problems may appear for distributions on multi-component topologies or characterized by multiple modes with high probability regions separated by very unlikely areas as previously encountered in figure 3.3. A typical symptom is the explosion of the Jacobian norm of the transformation in very low probability areas. This chapter proposes to overcome this issue thanks to a new Markov chain Monte Carlo algorithm to sample from the target distribution in the latent domain before transporting it back to the target domain. The approach relies on a Metropolis adjusted Langevin algorithm (MALA) whose dynamics explicitly exploits the Jacobian of the transformation. The proposed strategy preserves the tractability of the likelihood and it does not require a specific training. Notably, it can be straightforwardly used with any pre-trained NF network, regardless of the architecture. Interestingly the resulting Langevin diffusion is defined on the Riemannian manifold whose geometry is driven by the Jacobian of the NF.

Section 4.1 will state the problem and its causes. Section 4.2 reports on related works. Section 4.3 studies the theoretical implications of a topological mismatch between the latent distribution and the target distribution. Section 4.4 introduces the proposed sampling method based on a Langevin dynamics in the latent space. In Section 4.7, numerical experiments illustrate the advantages of the proposed approach by reporting performance results both for 2D toy distributions and in high dimensions on the usual Cifar-10, CelebA and LSUN data sets.

This work is based on a long paper which has been submitted to an international journal:

F. Coeurdoux, N. Dobigeon, P. Chainais. *"Normalizing flow sampling with Langevin dynamics in the latent space"*, Submitted to Machine Learning Journal, arXiv:2305.12149



(a) (b) (c)

Figure 4.1: (a) Target distribution p_Z in the latent space; (b) Learnt latent distribution in the latent space $\tilde{q}_Z$; (c) Output of naive sampling, i.e., $\left\{x^{(n)}\right\}_{n=1}^{N}$ with $x^{(n)} = f(z^{(n)})$ and $z \sim q_Z$.

4.1 Problem statement

Normalizing flows are known to be a very efficient generative model to approximate probability distributions in an unsupervised setting [39, 51]. Despite some early theoretical results about their stability [53] or their approximation and asymptotic properties [81], their training remains challenging in the most general cases. Their capacity is limited by intrinsic architectural constraints, resulting in a variational mismatch between the targeted distribution and the actually learnt distribution. In particular Cornish *et al.* [82] pointed out the capital issue of target distributions with disconnected support featuring several components. Since NFs provide a continuous differentiable change of variable, they are not able to deal with such distributions when using a monomodal (e.g., Gaussian) latent distribution. Even targeting multimodal distributions featuring high probability regions separated by very unlikely areas remains problematic. The trained NF is a continuous differentiable transformation so that the transport of latent samples to the target space may overcharge low probability areas with (undesired) samples. These out-of-distribution samples will correspond to smooth transitions between different modes, which leads to out-of-distribution samples, as discussed by Cornish *et al.* [82].

Fig. 4.1 illustrates this behavior on a archetypal bimodal 2D two-moon distribution using a latent Gaussian distribution. The NF is first trained on a large set of samples from the true target distribution. Fig. 4.1(a) shows the likelihood of the inverse transformation of the target distribution back to the latent space. It appears that the NF splits the Gaussian latent space into two sub-regions separated by an area of minimal likelihood. In the target

domain, these minima correspond to the low probability area between the two modes of the target distribution, which is somewhat expected.

Fig. 4.1(c) shows the result of a naive sampling from the Gaussian model latent distribution to explore the target distribution thanks to the NF change of variable. Latent Gaussian independent samples are transformed by the NF to the target domain. The purple line represents the 97.5% level set. It appears that many samples through this naive sampling procedure are out-of-distribution in the low probability area between the two moons, see the top of the plot. They correspond to samples of the latent distribution around the low-likelihood area highlighted on Fig. 4.1(a). Note that this illustrative example and in particular Fig. 4.1(b) will be more deeply discussed in the above contributive sections.

The observations made above illustrate a behavior that is structural. NFs are diffeomorphisms that preserve the topological structure of the support of the latent distribution. If the information about the structure of the target distribution is ignored, many out-of-distribution samples will be generated. This effect is reinforced by the fact that the NF is trained on a finite dataset so that in practice there exist close to empty areas in low probability regions. In other words, since the latent distribution is usually a simple Gaussian unimodal distribution, there is a topological mismatch with the often much more complex target distribution [82], in particular when it is multimodal. This chapter will present a so called method NF SAmpling In Latent Space (NF-SAILS) to circumvent this unwanted behavior.

4.1.1 Learning of change of variable

As discussed in section 2.1, NFs seek to learn a change of variable between a reference Gaussian measure q_Z and a target measure p_X through an invertible transformation $f : \mathcal{Z} \to \mathcal{X}$ with $f \in \mathcal{F}$ where $\mathcal{F}$ defines the class of NFs. We summarized the relation between the target measure p_X, the pushed-forwarded measure q_X, the Gaussian measure q_Z and the true latent measure p_Z in figure 4.2.

In the following, it will be worth keeping in mind that the obtained solution $\hat{f} \in \min_{f\in\mathcal{F}} \mathcal{L}_{MLE}(x)$ is only an approximation of the exact transport map due to two reasons. First, the feasible set $\mathcal{F}$ (i.e., the class of admissible NFs) is reduced to the set of continuous, differentiable and bijective functions. The existence of a transformation belonging to this class such that $D_{\mathrm{KL}}(p_X \| q_X) = 0$ is not guaranteed. This remark holds even in the case of a parametric form of the target distribution p_X, i.e., when minimizing the real (non-empirical) objective function in (2.4). Second, even when such a transformation exists in $\mathcal{F}$, the solution $\hat{f}$ recovered by (2.5) coincides with the

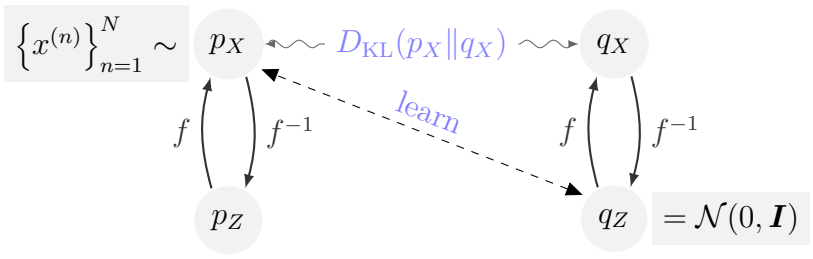


Figure 4.2: NF learns a mapping f from data points $\left\{x^{(n)}\right\}_{n=1}^{N}$ assumed to be drawn from p_X towards the latent Gaussian measure q_Z. The training consists in minimizing the KL divergence between p_X and $q_X = f_\sharp q_Z$. Once trained, the learnt map permits to go from q_Z to q_X, which is an approximation of the true target distribution p_X.

minimizer of (2.4) in an asymptotic sense only, i.e., when $N \to \infty$. On top of that, only a proxy of $\hat{f}$ can be computed because of the use of a stochastic optimization procedure (e.g., stochastic gradient descent), which may suffer from issues not discussed anyfurther here.

4.1.2 A Gaussian latent space?

As noticed by Marzouk *et al.* [83], learning the transformation f by variational inference can be reformulated with respect to (w.r.t.) the corresponding inverse map f^{-1}. Since the KL divergence is invariant to changes of variables, minimizing $D_{\mathrm{KL}}(p_X \| q_X)$ is equivalent to minimizing $D_{\mathrm{KL}}(p_Z \| q_Z)$ with $p_Z = f_\sharp^{-1} p_X$. The training procedure is thus formulated in the latent space instead of the target space. In other words, the NF aims at fitting the target measure p_Z expressed in the latent space to the latent Gaussian measure q_Z. However, due to inescapable shortcomings similar to those highlighted above, the target measure p_Z in the latent space is only an approximation of the latent Gaussian measure q_Z. This mismatch can be easily observed in Fig. 4.1(a) where the depicted actual measure p_Z is clearly not Gaussian. This issue may be particularly critical when there is a topological mismatch between the respective supports of the target and latent distributions. This will be discussed in more details in Section 4.3.

4.1.3 Beyond conventional NF sampling

Once the NF has been trained, the standard method to sample from the learnt target distribution is straightforward. It consists in drawing a sample z_k from the latent Gaussian distribution q_Z and then applying the learnt

transformation f to obtain a sample $x^{(n)} = f\left(z^{(n)}\right)$. This method will be referred to as "naive sampling" in the sequel of this paper.

Unfortunately, as discussed in Section 4.1.2 (see also Fig. 4.1), the latent distribution q_Z is expected to be different from the actual target distribution p_Z expressed in the latent space. As suggested in the next section, this mismatch will be even more critical when it results from topological differences between the latent and target spaces. As a consequence the naive NF sampling is doomed to be suboptimal and to produce out-of-distribution samples, as illustrated in Fig. 4.1(c). In contrast, the approach proposed in Section 4.4 aims at devising an alternative sampling strategy that explicitly overcomes these shortcomings.

4.2 Related works

Geometry in neural networks – Geometry in neural networks as a tool to understand local generalization was first discussed by Bengio *et al.* [84]. As a key feature, the Jacobian matrix controls how smoothly a function interpolates a surface from some input data. As an extension, Rifai *et al.* [85] showed that the norm of the Jacobian acts as a regularizer of the deterministic autoencoder. Later Arvanitis *et al.* [86] were the first to establish the link between push forward generative models and surface modeling. In particular, they showed that the latent space could reveal a distorted view of the input space that can be characterized by a stochastic Riemannian metric governed by the local Jacobian.

Distribution with disconnected support – As highlighted by Cornish *et al.* [82], when using ancestral sampling, the structure of the latent distribution should fit the unknown structure of the target distribution. To tackle this issue, several solutions have been proposed. These strategies include augmenting the space on which the model operates [87], continuously indexing the flow layers [82], and including stochastic [88] or surjective layers [89]. However, these approaches sacrifice the bijectivity of the flow transformation. In most cases, this sacrifice has dramatic impacts: the model is no longer tractable, memory savings during training are no longer possible [90], and the model is no longer a perfect encoder-decoder pair. Other works have promoted the use of multimodal latent distributions [91, 92, 93]. Nevertheless, rather than capturing the inherent multimodal nature of the target distribution, their primary motivation is to perform a classification task or to solve inverse problems with flow-based models. Pamakarios *et al.* [44] has shown that choosing a mixture of Gaussians as a latent distribution could

lead to an improvement of the fidelity to multimodal distributions. Alternatively, Pires *et al.* [94] has studied the learning of a mixture of generators. Using a mutual information term, they encourage each generator to focus on a different submanifold so that the mixture covers the whole support. More recently, Stimper *et al.* [95] predicted latent importance weights and proposed a sub-sampling method to avoid the generation of the most irrelevant samples. However, all these methods require to implement elaborated learning strategies which handle several sensitive hyperparameters or impose specific neural architectures. On the contrary, as emphasized earlier, the present approach does not require a specific training strategy, is computationally efficient, and can be implemented to any pre-trained NF.

Sampling with normalizing flows – Recently NFs have been used to facilitate the sampling from distributions with non-trivial geometries by transforming them into distributions that are easier to handle. To solve the problem, samplers that combine Monte Carlo methods with NF have been proposed. On the one hand, flows have been used as reparametrization maps that improve the geometry of the target distribution before running local conventional samplers such as Hamiltonian Monte Carlo (HMC) [96, 97]. On the other hand, the push-forward of the NF base distribution through the map has also been used as an independent proposal in importance sampling [98] and Metropolis-Hastings steps [99, 100]. In this context, NFs are trained using the reverse Kullback-Leiber divergence so that the push-forward distribution approximates the target distribution. These approaches are particularly appealing when a closed-form expression of the target distribution is available explicitly. In contrast, this paper does not assume an explicit knowledge of the target distribution. The proposed approach aims at improving the sampling from a distribution learnt by a given NF trained beforehand.

4.3 Implications of a topological mismatch

4.3.1 Topology preservation

The push-forward operator $f_\sharp$ learnt by a NF transports the mass allocated by q_Z in $\mathcal{Z}$ to $\mathcal{X}$, thereby defining q_X by specifying where each elementary mass is transported. This imposes a global constraint on the operator f if the model distribution q_X is expected to match a given target measure p_X perfectly. Let $\text{supp}(q_Z) = \{z \in \mathcal{Z} : q_Z(z) > 0\}$ denote the support of q_Z. Then the push-forward operator $f_\sharp$ can yield $q_X = p_X$ only if

$$\text{supp}(p_X) = \overline{f\left(\text{supp}(q_Z)\right)} \tag{4.1}$$

where $\overline{B}$ is the closure of set B. The constraint (4.1) is especially onerous for NFs because of their bijectivity. The operators f and f^{-1} are continuous, and f is a homeomorphism. Consequently, for these models, q_Z and p_X are isomorphic, i.e., homeomorphic as topological spaces [101, Def. 3.3.10]. This means that $\text{supp}(q_Z)$ and $\text{supp}(p_X)$ must share exactly the same topological properties, in particular the number of connected components. This constraint may be unlikely satisfied when learning complex real-world distributions, leading to an insurmountable topological mismatch. In such cases, this finding has serious consequences on the operator f learnt and implemented by a NF. Indeed, the following proposition states that if the respective supports of the latent distribution q_Z and the target distribution p_X are not homeomorphic, then the norm of the Jacobian $|J_f|$ of f may become arbitrary large. Here $\xrightarrow{\mathcal{D}}$ denotes weak convergence.

Proposition 1. *Let q_Z and p_X denote distributions defined on $\mathbb{R}^d$. Assume that $supp(q_Z) \ncong supp(p_X)$. For any sequence of measurable, differentiable Lipschitz functions $f_t : \mathbb{R}^{d_Z} \to \mathbb{R}^{d_X}$, if $f_{t\sharp}q_Z \xrightarrow{\mathcal{D}} p_X$ when $t \to +\infty$, then*

$$\lim_{t\to\infty} \sup_{z\in\mathcal{Z}}(\|J_{f_t}(z)\|) = +\infty.$$

The proof is reported in Appendix A.1.

It is worth noting that training a generative model is generally conducted by minimizing a statistical divergence. For most used divergence measures, (e.g., KL and Jensen-Shannon divergences, Wasserstein distance), this minimization implies a weak convergence of the approximated distribution q_X towards the target distribution p_X [58]. As a consequence, Proposition 1 states that, when training a NF to approximate p_X by $q_X = \lim_{t\to\infty} f_{t\sharp}q_Z$, the supremum of the Jacobian of the learnt mapping may become arbitrarily large in some regions. This result is in line with the experimental findings early discussed and visually illustrated by Fig. 4.1. Indeed, Fig. 4.1(b) depicts the heatmap of the log-likelihood

$$\log q_X(f(z)) = \log q_Z(z) - \log|J_f(z)| \tag{4.2}$$

given by (2.2) after training an NF. The impact of the term governed by the determinant of the Jacobian is clear. It highlights a boundary separating two distinct areas, each associated with a mode in the target distribution p_X. This result still holds when q_Z and q_X are defined on $\mathbb{R}^{d_Z}$ and $\mathbb{R}^{d_X}$, respectively, with $d_Z \neq d_X$. This shortcoming is thus also unavoidable when learning injective flow models [102] and other push-forward models such as GANs [57].

In practice, models are trained on a data set of finite size. In other words, the underlying target measure p_X is available only through the empirical measure $\frac{1}{N}\sum_{n=1}^{N}\delta_{x^{(n)}}$. During the training defined by (2.5), areas of low probability possibly characterizing a multi-modal target measure are likely interpreted as areas of null probability observed in the empirical measure. This directly results in the topological mismatch discussed above. Thus, even when targeting a distribution p_X defined over a connected support with regions of infinitesimal support, the learnt mapping is expected to be characterized by a Jacobian with exploding norm in these regions, see Fig 4.1. The following section suggests that these regions correspond to the frontiers between cells defining a partition of the latent space.

4.3.2 Partition of the latent space

A deterministic generative model can be interpreted as a surface model or a Gauss map, if the generator f is sufficiently smooth [86]. When targeting a multi-modal distribution, the learnt model implicitly partitions the Gaussian latent space into a set of disjoint subsets, each associated with a given mode. The Gaussian multi-bubble conjecture was formulated when looking for a way to partition the Gaussian space with the least-weighted perimeter. This conjecture was proven recently by Milman *al.* [103]. The result states that partitioning a Gaussian space of $\mathbb{R}^d$ into m clusters of equal measures ($2 \leqslant m \leqslant d+1$) consists in recovering "simplicial clusters" defined as Voronoi cells of m equidistant points in $\mathbb{R}^d$. According to convex geometry principles, the boundary is the union of m convex cones, each of them contained in a distinct hyperplane that goes through the origin of $\mathbb{R}^d$. Recently, Issenbuth *al.* [104] leveraged on this finding to assess the optimality of the precision of GANs. They show that the precision of the generator vanishes when the number of components of the target distribution tends towards infinity.

Again, in practice, NFs models are trained on a data set with a finite number of samples. This results in a partitioning of the Gaussian latent space into cells separated by frontiers of arbitrarily large widths. Figure 4.1(b) allows a connection to be drawn between the statement by Milman *al.* [103] and the Proposition 1. Indeed, in this figure, the frontiers defining this partitioning are clearly identified as the areas with exploding Jacobian norm. As a consequence, exploring naively the Gaussian latent space to sample from the target distribution seems to be inappropriate. Because of these wide frontiers, large areas of the latent space are expected to be associated with potentially numerous out-of-distribution samples.

4.4 NF sampling in the latent space

4.4.1 Difficulties of sampling the target space

As explained in Section 4.1.3, naive NF sampling boils down to drawing a Gaussian variable before transformation by the learnt mapping f. This strategy is expected to produce out-of-distribution samples, due to the topological mismatch between q_X and p_X discussed in Section 4.3. The proposed alternative elaborates directly on the learnt target distribution q_X.

The starting point of our rational consists in expressing a Langevin diffusion in the target space. This Markov chain Monte Carlo (MCMC) algorithm would target the distribution q_X using only the derivative of its likelihood $\nabla_x \log q_X(x)$. After initializing the chain by drawing from an arbitrary distribution $x_0 \sim \pi_0(x)$, the updating rule writes

$$x_{k+1} \leftarrow x_k + \frac{\epsilon^2}{2}\nabla_x \log q_X(x_k) + \epsilon\xi \tag{4.3}$$

where $\xi \sim \mathcal{N}(0, I)$. When $\epsilon \to 0$ and the number of samples $K \to \infty$, the distribution of the samples generated by the iterative procedure (4.3) converges to q_X under some regularity conditions. In practice, the error is negligible when ϵ is sufficiently small and K is sufficiently large. This algorithm referred to as the unadjusted Langevin Algorithm (ULA) always accepts the generated sample proposed by (4.3), neglecting the errors induced by the discretization scheme of the continuous diffusion. To correct this bias, Metropolis-adjusted Langevin Algorithm (MALA) applies a Metropolis-Hastings step to accept or reject a sample proposed by ULA [15].

Again, sampling according to q_X thanks to the diffusion (4.3) is likely to be inefficient due to the expected complexity of the target distribution possibly defined over a subspace of $\mathbb{R}^d$. In particular, this strategy suffers from the lack of prior knowledge about the location of the mass. Conversely, the proposed approach explores the latent space by leveraging on the closed-form change of variable (2.2) operated by the trained NF.

After technical derivations reported in the next section 4.4.2, the counterpart of the diffusion (4.3) expressed in the latent space writes

$$z' = z_k + \frac{\epsilon^2}{2}G^{-1}(z_k)\nabla_z \log \tilde{q}_Z(z_k) + \epsilon\sqrt{G^{-1}(z_k)}\xi \tag{4.4}$$

where

$$\tilde{q}_Z(z) = q_Z(z)\left|J_f(z)\right|^{-1} \tag{4.5}$$

and

$$G^{-1}(z) = \left[J_f^{-1}(z)\right]^2. \tag{4.6}$$

Note that the distribution $\tilde{q}_Z$ in (4.5) originates from the change of variable that defines q_X in (2.2) and has been already implicitly introduced by (4.2) in Section 4.3. Interestingly, the matrix $G(\cdot)$ is a definite positive matrix (see Appendix A.2). Thus the diffusion (4.4) characterizes a Riemannian manifold Langevin dynamics where $G(\cdot)$ is the Riemannian metric associated with the latent space [105, 106]. More precisely, it defines the conventional proposal move of the Riemannian manifold ajusted Langevin algorithm (RMMALA) which targets the distribution $\tilde{q}_Z$ defined by (4.5). This distribution is explicitly defined through the Jacobian $J_f(\cdot)$ of the transformation whose behavior has been discussed in depth in Section 4.3. It can be interpreted as the Gaussian latent distribution q_Z tempered by the (determinant of the) Jacobian of the transformation. It has also been evidenced by depicting the heatmap of (4.2) in Fig. 4.1(b), which shows that it appears as a better approximation of p_Z than q_Z. Since it governs the drift of the diffusion through the gradient of its logarithm, the diffusion is expected to escape from the areas where the determinant of the Jacobian explodes, see Section 4.3.

4.4.2 Latent diffusion equation

This technical section will derive the latent diffusion equation (4.4). First we recall some preliminary results from stochastic calculus and multivariate calculus. We then propose a step by step derivation of diffusion equation and its induced proposal distribution.

Preliminaries

The Langevin diffusion is a particular instance of the Itô process defined in the following Lemma of which a proof is given in [107].

Lemma 1 (Itô's lemma)**.** *Let X_t denote an Itô drift-diffusion process defined by the stochastic differential equation*

$$dX_t = \mu_t dt + \sigma_t dB_t. \tag{4.7}$$

If $f : \mathbb{R}^2 \to \mathbb{R}$ is a differentiable scalar function, then

$$df\left(t, X_t\right) = \left(\frac{\partial f}{\partial t} + \mu_t \frac{\partial f}{\partial x} + \frac{\sigma_t^2}{2}\frac{\partial^2 f}{\partial x^2}\right) dt + \sigma_t \frac{\partial f}{\partial x} dB_t. \tag{4.8}$$

It yields that $f\left(t, X_t\right)$ is an Itô drift-diffusion process itself.

The following property shows that for any bijective transformation, the Jacobian of the inverse transformation is equal to the inverse of the Jacobian of the transformation. This result will be useful later.

Property 1. *Let $f : \mathcal{Z} \to \mathcal{X}$ denote a bijective transformation and $J_f(\cdot)$ its Jacobian, then*

$$J_{f^{-1}}(f(z)) = J_f^{-1}(z).$$

Proof. Let h and g denote two multivariate functions. The chain rule writes

$$J_{h\circ g}(\cdot) = J_h(g(\cdot))J_g(\cdot) \tag{4.9}$$

thus

$$J_h(g(\cdot)) = J_{h\circ g}(\cdot)J_g^{-1}(\cdot). \tag{4.10}$$

Moreover, for any multivariate bijective function f, we have

$$J_{f\circ f^{-1}}(\cdot) = J_{f^{-1}\circ f}(\cdot) = I_d. \tag{4.11}$$

Combining (4.10) and (4.11) with $h = f^{-1}$ and $g = f$ yields

$$J_{f^{-1}}(f(z)) = J_{f^{-1}\circ f}(z)J_f^{-1}(z) = I_d J_f^{-1}(z) = J_f^{-1}(z). \tag{4.12}$$

□

The following property demonstrates that the gradient of the score of q_X can be expressed over the latent space $\mathcal{Z}$ using $\tilde{q}_Z$ defined in (4.5).

Property 2. *Let $f : \mathcal{Z} \to \mathcal{X}$ be a bijective transformation which maps a latent measure q_Z towards a target measure q_X. Then the score of $q_X(x)$ is given by*

$$\nabla_x \log q_X(x) = J_f^{-1}(z) \cdot \nabla_z \log \tilde{q}_Z(z)$$

where $\tilde{q}_Z(z) = q_Z(z)\left|J_f(z)\right|^{-1}$.

Proof. From the definition of $q_X(x)$ in equation (2.2), the score of $q_X(x)$ writes

$$\nabla_x \log q_X(x) = \nabla_x \left[\log q_Z(f^{-1}(x)) + \log\left|J_{f^{-1}}(x)\right|\right] \tag{4.13}$$

and, from Property 1,

$$\begin{aligned}\nabla_x \log q_X(x) &= \nabla_{f(z)} \left[\log q_Z(z) + \log\left|J_f(z)\right|^{-1}\right] &(4.14)\\ &= \nabla_{f(z)} \log \tilde{q}_Z(z) &(4.15)\end{aligned}$$

The chain rule now leads to

$$\begin{aligned}\nabla_x \log q_X(z) &= \nabla_x f^{-1}(x) \cdot \nabla_z \log \tilde{q}_Z(z) &(4.16)\\ &= J_{f^{-1}}(f(z)) \cdot \nabla_z \log \tilde{q}_Z(z) &(4.17)\end{aligned}$$

which, using Property 1, can be finally rewritten as

$$\nabla_x \log q_X(x) = J_f^{-1}(z) \cdot \nabla_z \log \tilde{q}_Z(z). \tag{4.18}$$

□

Derivation of the proposal distribution

The following property shows that the Lanvegin diffusion which targets the distribution q_X can be rewritten as a diffusion over the latent space $\mathcal{Z}$.

Property 3. *We consider the overdamped Langevin Itô diffusion*

$$dX_t = \nabla_x \log q_X(X_t)dt + \sigma_t dB_t \tag{4.19}$$

driven by the time derivative of a standard Brownian motion B_t. In the limit $t \to \infty$, this probability distribution X_t approaches a stationary distribution q_X. Let $f : \mathcal{Z} \to \mathcal{X}$ be a bijective transformation which maps a latent measure q_Z towards the target measure q_X. A counterpart Langevin diffusion expressed over the latent space $\mathcal{Z}$ writes

$$dZ_t = G^{-1}(Z_t)\nabla_z \log \tilde{q}_Z(Z_t)dt + \sigma_t\sqrt{G^{-1}(Z_t)}dB_t \tag{4.20}$$

Proof. The Langevin diffusion is a particular instance of the Itô process where the drift μ_t in (4.7) is given by the gradient of the log-density $\nabla_x \log q_X(X_t)$, i.e.,

$$dX_t = \nabla_x \log q_X(X_t)dt + \sigma_t dB_t. \tag{4.21}$$

We are interested in the diffusion process of $f^{-1}(X_t)$ when $f(\cdot)$ is a NF which is continuous, differentiable and bijective such that $f(Z_t) = X_t$ and $f^{-1}(X_t) = Z_t$. The Îto's Lemma 1 states

$$df^{-1}(X_t) = \left(J_{f^{-1}}(X_t)\nabla_x \log q_X(X_t) + \frac{\sigma_t^2}{2}\mathrm{tr}(H_{f^{-1}}(X_t))\right)dt + \sigma_t J_{f^{-1}}(X_t)dB_t. \tag{4.22}$$

Neglecting the second-order terms yields

$$df^{-1}(X_t) = J_{f^{-1}}(X_t)\nabla_x \log q_X(X_t)dt + \sigma_t J_{f^{-1}}(X_t)dB_t. \tag{4.23}$$

Using Property 1, Eq. (4.23) can be rewritten as

$$dZ_t = J_f^{-1}(Z_t)\nabla_x \log q_X(X_t)dt + \sigma_t J_f^{-1}(Z_t)dB_t. \tag{4.24}$$

Finally, by denoting $G^{-1}(z) = \left[J_f^{-1}(z)\right]^2$ and using Property 2, the diffusion in the latent space writes

$$dZ_t = G^{-1}(Z_t)\nabla_z \log \tilde{q}_Z(Z_t)dt + \sigma_t\sqrt{G^{-1}(Z_t)}dB_t. \tag{4.25}$$

□

The discretization of the stochastic differential equation (4.25) using the Euler–Maruyama scheme can be written as in (4.4). This discretized counterpart of the diffusion corresponds to the proposal move of a Riemann manifold Metropolis-Adjusted Langevin algorithm which targets $\tilde{q}_Z$. The following property shows that the associated proposal kernel can be rewritten as (4.35).

Property 4. *The discrete Langevin diffusion given by*

$$z' = z + \frac{\epsilon^2}{2} \cdot G^{-1}(z) \nabla_z \log \tilde{q}_Z(z) + \epsilon \cdot \sqrt{G^{-1}(z)} \xi \tag{4.26}$$

with $\xi \sim \mathcal{N}(0, I)$ *is defined by the transition kernel*

$$g\left(z' \mid z\right) \propto |J_f(z)| \exp\left[-\frac{1}{2\epsilon^2} \left\| J_f(z)(z' - z) + \frac{\epsilon^2}{2} J_f^{-1}(z) \nabla_z \log \tilde{q}(z) \right\|^2 \right]. \tag{4.27}$$

Proof. From the Gaussian nature of ξ, the conditional distribution of z' is a Gaussian distribution whose mean is governed by the drift and covariance matrix is parametrized by the inverse of the Jacobian, namely

$$z' \mid z \sim \mathcal{N}(\mu, \Sigma) \tag{4.28}$$

with

$$\mu = z + \frac{\epsilon^2}{2} \cdot G^{-1}(z) \nabla_z \log \tilde{q}_Z(z) \tag{4.29}$$

$$\Sigma = \epsilon^2 J_f^{-1}(z) J_f^{-\top}(z). \tag{4.30}$$

The corresponding probability density function writes

$$g\left(z' \mid z\right) = \left(\frac{1}{2\pi}\right)^{\frac{d}{2}} \frac{1}{|\Sigma|^{1/2}} \exp\left(-\frac{1}{2} (z' - \mu)^\top \Sigma^{-1} (z' - \mu) \right). \tag{4.31}$$

First, let notice that $\Sigma^{-1} = \epsilon^{-2} J_f^\top(z) J_f(z)$. Then we have

$$(z' - \mu)^\top \Sigma^{-1} (z' - \mu) = \epsilon^{-2} \left[z' - z - \frac{\epsilon^2}{2} \cdot \left[J_f^{-1}(z) \right]^2 \nabla_z \log \tilde{q}_Z(z) \right]^\top J_f^\top(z) \tag{4.32}$$

$$\times J_f(z) \left[z' - z - \frac{\epsilon^2}{2} \cdot \left[J_f^{-1}(z) \right]^2 \nabla_z \log \tilde{q}_Z(z) \right]$$

$$= \epsilon^{-2} \left\| J_f(z)(z' - z) - \frac{\epsilon^2}{2} J_f^{-1}(z) \nabla_z \log \tilde{q}(z) \right\|^2 \tag{4.33}$$

Finally, using $|\Sigma|^{1/2} = \epsilon|J_f^{-1}(z)| = \epsilon|J_f(z)|^{-1}$ yields

$$g\left(z' \mid z\right) \propto |J_f(z)| \exp\left[-\frac{1}{2\epsilon^2}\left\|J_f(z)(z'-z) - \frac{\epsilon^2}{2}J_f^{-1}(z)\nabla_z \log \tilde{q}(z)\right\|^2\right]. \tag{4.34}$$

□

It is worth noting that the pdf of this transition kernel should be computed when evaluating the acceptance ratio (4.37). When using the canonical writing (4.31), evaluating this pdf would require to compute $G^{-1}(z)$ in (4.29) and $J_f^{-1}(z)J_f^{-\top}(z)$ in (4.30) which can be very costly.

The next section 4.5 will discus some efficient implementation techniques to avoid expensive computation.

4.5 Efficient implementation

The proposal distribution

The above Property 4 derived the proposal kernel $g(z'|z)$ associated with the diffusion (4.4). We demonstrate in (4.34) that $g(z'|z)$ is a Gaussian distribution whose probability density function can be conveniently rewritten as:

$$g\left(z' \mid z_k\right) \propto |J_{f(z_k)}| \exp\left[-\frac{1}{2\epsilon^2}\left\|J_f(z_k)(z'-z_k) - \frac{\epsilon^2}{2}\tilde{s}_Z(z_k)\right\|^2\right]. \tag{4.35}$$

where $\tilde{s}_Z(\cdot)$ denotes the so-called latent score

$$\tilde{s}_Z(z) = J_f^{-1}(z)\nabla_z \log \tilde{q}_Z(z). \tag{4.36}$$

The sample proposed according to (4.4) is then accepted with probability

$$\alpha_{\text{RMMALA}}(z_k, z') = \min\left(1, \frac{\tilde{q}_Z\left(z'\right) g\left(z_k \mid z'\right)}{\tilde{q}_Z\left(z_k\right) g\left(z' \mid z_k\right)}\right). \tag{4.37}$$

It is worth noting that the formulation (4.35) of the proposal kernel leads to a significantly faster implementation than its canonical formulation. Indeed, it does not require to compute the metric $G^{-1}(\cdot)$ defined by (4.6), which depends on the inverse of the Jacobian matrix twice. Moreover, the evaluation of the latent score (4.36) can be achieved in an efficient manner, bypassing the need for evaluating the inverse of the Jacobian matrix, as elaborated bellow in 4.5.2. Finally, only the Jacobian associated with the forward transformation is required to compute (4.35). This approach enables a streamlined calculation of the acceptance ratio (4.37), ensuring an overall computational efficiency.

4.5.1 Fast approximation of the proposal move

As discussed above, the proposal scheme (4.4) requires to generate high dimensional Gaussian variables with covariance matrix $\epsilon^2 G^{-1}(\cdot)$ [108]. Generating such an high dimension Gaussian variables according to (4.4) is expected to be very costly, even if the corresponding Cholesky factor $\epsilon J_f^{-1}(\cdot)$ is lower triangular and explicit (see Appendix A.2.1). Alternatively, to lighten the computation, we take advantage of the 1st order expansion

$$f^{-1}\left(f(z_k)+\epsilon\xi\right) \simeq f^{-1}\circ f(z_k) + \epsilon J_{f^{-1}}(f(z_k))\xi. \tag{4.38}$$

Using Property 1, this amounts to approximate (4.4) by the diffusion

$$\begin{aligned} z' &= f^{-1}\left(f(z_k)+\epsilon\xi\right) + \tfrac{\epsilon^2}{2} G^{-1}(z_k)\nabla_z \log \tilde{q}(z_k) && (4.39)\\ &= f^{-1}\left(f(z_k)+\epsilon\xi\right) + \tfrac{\epsilon^2}{2} J_f^{-1}(z_k)\tilde{s}_Z(z_k). && (4.40)\end{aligned}$$

According to (4.40), this alternative proposal scheme only requires to generate high dimensional Gaussian variables whose covariance matrix is now identity, i.e., most cheaper. Moreover, it is worth noting that *i)* the latent score $\tilde{s}_Z(\cdot)$ can be evaluated efficiently (see below) and *ii)* using $J_f^{-1}(z) = J_{f^{-1}}(f(z))$ (see Property 1), sampling z' according to (4.4) only requires to evaluate the Jacobian associated with the backward transformation. The algorithmic procedure to sample according to this kernel denoted $\mathcal{K}_{\text{RMMALA}}(\cdot)$ is summarized in Algo. 3.

4.5.2 Fast computation of the latent score

The latent score $\tilde{s}_Z(z)$ is a critical quantity in the proposed method, as it contributes to the drift term in the proposal move (4.4) and to the proposal kernel (4.35). Property 2 shows that the latent score is equal to the score of q_X expressed in the target domain, i.e.,

$$\nabla_x \log q_X(x) = J_f^{-1}(z)\cdot \nabla_z \log \tilde{q}_Z(z) \quad \text{with } x = f(z)$$

The adopted implementation bypasses the costly evaluation and storage of the inverse Jacobian by directly computing the latent score as $\nabla_x \log q_X(x)$. The evaluation of the score of q_X can be conveniently performed thanks to the auto-differentiation modules provided by numerous deep learning frameworks.

The next section will present our proposed model NF-SAILS which consists of combining MCMC steps using properties discussed above.

Algorithm 3: Sampling kernel $\mathcal{K}_{\text{RMMALA}}(\cdot)$.

Input: trained NF $f(\cdot)$, time step ϵ, current state z_k of the sampler.

`/* Draw the candidate */`

Draw $\xi \sim \mathcal{N}(0, 1)$

Set $z' = f^{-1}\left(f(z_k) + \epsilon \cdot \xi\right) + \frac{\epsilon^2}{2} J_f^{-1}(z_k)\tilde{s}_Z(z_k)$

`/* Accept/reject procedure */`

Draw $u \sim \mathcal{U}(0, 1)$

if $u < \alpha_{\text{RMMALA}}(z_k, z')$ **then**

  Set $z_{k+1} = z'$

else

  Set $z_{k+1} = z_k$

Output: New state $z_{k+1} = \mathcal{K}_{\text{RMMALA}}(z_k)$ of the sampler.

4.6 NF-SAILS

The adaptive MCMC we propose concurrently sample new data by combining a local sampler based on NF and a nonlocal one which leverage the Normal shape of the latent space. The so-called procedure SAmpling In Latent Space (NF-SAILS) is summarized in Algorithm 5 which combines local MALA from Algorithm 3 and a nonlocal 4.

4.6.1 Local exploration with MALA

Our proposed MALA algorithm combines two steps, one diffusion step from (4.4) accelerated by the fast approximation in 4.5.2 and one accept-reject step using the corresponding proposal distribution 4.35. Algorithm 3 shows the pseudo code of the proposed Riemannian MALA algorithm. This MALA kernel will leverage the knowledge of the curvature of the latent space and thus perform great exploration at a local space while rejecting zones corresponding to out of distribution samples.

Handling distributions that exhibit several modes or defined on a complex multi-component topology is a major issue raised by the problem addressed here. In practice, local information based sampling schemes such as those based on Langevin dynamics fail to explore the full distribution when modes are isolated since they may get stuck around one of these modes. Thus, the samples proposed according to (4.4) in areas with high values of $\|J_f(\cdot)\|$ are expected to be rejected. These areas have been identified in Section 4.3 as the low probability regions between modes when targeting a multimodal distribution. To alleviate this problem, one strategy consists in resorting to

another kernel to propose moves from one high probability region to another, without requiring to cross the low probability regions.

4.6.2 Global exploration with I-MH

Following this strategy, we propose to combine the diffusion (4.4) with an independent Metropolis-Hastings (I-MH) with the distribution q_Z as a proposal. The corresponding acceptance ratio writes

$$\begin{aligned} \alpha_{\text{I-MH}}(z_k, z') &= \min\left(1, \frac{\tilde{q}_Z(z')\, q_Z(z_k)}{\tilde{q}_Z(z_k)\, q_Z(z')}\right) \\ &= \min\left(1, \frac{|J_f(z_k)|}{|J_f(z')|}\right). \end{aligned} \tag{4.41}$$

It is worth noting that this probability of accepting the proposed move only depends on the ratio between the Jacobians evaluated at the current and the candidate states. In particular, candidates located in regions of the latent space characterized by exploding Jacobians in case of a topological mismatch (see Section 4.3) are expected to be rejected with high probability. Conversely, this kernel will favor moves towards other high probability regions not necessarily connected to the regions of the current state. The algorithmic procedure is sketched in Algo. 4.

Algorithm 4: Sampling kernel $\mathcal{K}_{\text{I-MH}}(\cdot)$.

Input: trained NF $f(\cdot)$, current state z_k of the sampler.

`/* Draw candidate */`

Draw $z' \sim \mathcal{N}(0, 1)$

`/* Accept/reject procedure */`

Draw $u \sim \mathcal{U}(0, 1)$

if $u < \alpha_{\text{I-MH}}(z_k, z')$ **then**

  Set $z_{k+1} = z'$

else

  Set $z_{k+1} = z_k$

Output: New state $z_{k+1} = \mathcal{K}_{\text{I-MH}}(z_k)$ of the sampler.

4.6.3 Local/Global exploration

Finally, the overall proposed sampler, referred to as NF-SAILS for NF SAmpling In the Latent Space and summarized in Algo. 5, combines the transition kernels $\mathcal{K}_{\text{RMMALA}}$ and $\mathcal{K}_{\text{I-MH}}$, which permits to efficiently explore the

latent space both locally and globally. At each iteration k of the sampler, the RMMALA kernel $\mathcal{K}_{\text{RMMALA}}$ associated with the acceptance ratio (4.37) is selected with probability p and the I-MH kernel $\mathcal{K}_{\text{I-MH}}$ associated with acceptance ratio (4.41) is selected with the probability $1-p$. Again, one would like to emphasize that the proposed strategy does not depend on the NF architecture and can be adopted to sample from any pretrained NF model.

Algorithm 5: NF-SAILS: NF SAmpling In the Latent Space.

Input: trained NF $f(\cdot)$, time step ϵ, probability p

`/* Initialization */`

Draw $z_0 \sim \pi_0(z)$

for $k = 0$ **to** K **do**

  `/* Choose the kernel */`

  Draw $u \sim \mathcal{U}(0,1)$

  if $u < p$ **then**

    `/* LOCAL EXPLORATION (see Algo. 3) */`

    $z_{k+1} = \mathcal{K}_{\text{RMMALA}}(z_k)$

  else

    `/* GLOBAL EXPLORATION (see Algo. 4) */`

    $z_{k+1} = \mathcal{K}_{\text{I-MH}}(z_k)$

end

Output: Collection of samples $\{z_k\}_{k=1}^{K}$.

4.7 Experiments

This section reports performance results to illustrate the efficiency of NF-SAILS thanks to experiments based on several models and synthetic data sets. It is compared to state-of-the-art generative models known for their abilities to handle multimodal distributions. These results will show that the proposed sampling strategy achieves good performance, without requiring to adapt the NF training procedure or resorting to non-Gaussian latent distributions. We will also confirm the relevance of the method when working on popular image data sets, namely Cifar-10 [109], CelebA [110] and LSUN [111].

To illustrate the versatility of proposed approach w.r.t. the NF architecture, two types of coupling layers are used to build the trained NFs. For the experiments conducted on the synthetic data sets, the NF architecture is RealNVP [36]. Conversely, a Glow model is used for experiments conducted on the image data sets [39]. However, it is worth noting that the

proposed method can apply on top of any generative model fitting multi-modal distributions. In all experiments we trained our models to maximize the log-likelihood using the ADAM optimiser with default hyperparameters and no weight decay. We used a held-out validation set and trained each model until its validation score stopped improving, except for the synthetic data experiments where we train for a fixed number of 1000 epochs.

4.7.1 Figures-of-merit

To evaluate the performance of the NFs, several figures-of-merit have been considered. When addressing bi- dimensional problems, we perform a Kolmogorov Smirnov test to assess the quality of the generated samples w.r.t. the underlying true target distribution [112]. The goodness-of-fit is also monitored by evaluating the mean log-likelihood of the generated samples and the entropy estimator between samples, which approximates the Kullback-Leibler divergence between empirical samples [113].

For applications to higher dimensional problems, such as image generation, the performances of the compared algorithms are evaluated using the Fréchet inception distance (FID) [114] using a classifier pre-trained specifically on each data set. Besides, for completeness, we report the bits per dimension (bpd) [44], i.e., the log-likelihoods in the logit space, since this is the objective optimized by the trained models.

The model architectures used in the experiments conducted on images are reported in Table 4.1.

Dataset	Minibatch Size	Levels	Depth per level	Coupling
CIFAR-10	512	3	32	Affine
LSUN, 64×64	128	4	48	Affine
LSUN, 96×96	320	5	64	Affine
LSUN, 128×128	160	5	64	Affine
CelebA, 96×96	320	5	64	Affine
CelebA, 128×128	160	6	32	Affine

Table 4.1: Architectures of the Glow model implemented for the experiments conducted on the image data sets.

4.7.2 Results obtained on synthetic dataset

As a first illustration of the performance of NF-SAILS, we consider to learn a mixture of k bidimensional Gaussian distributions, with $k \in \{2, 3, 4, 6, 9\}$.

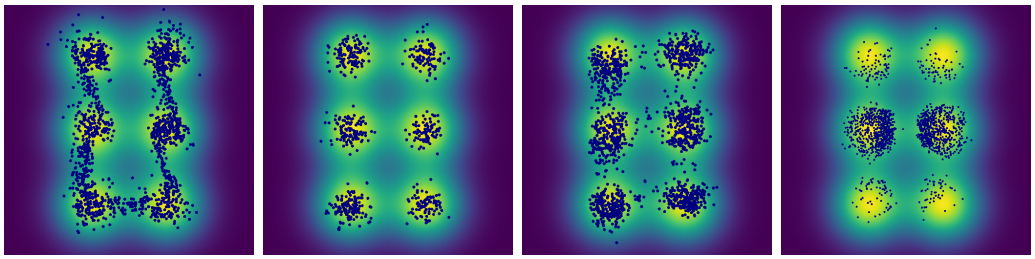
		$\uparrow \log p_X$	$\downarrow$KL	$\downarrow$KS
$k=2$	Naive sampling	-4.13	0.263	0.177
	NF-SAILS	$\mathbf{-1.41}$	**0.057**	**0.047**
	WGAN-GP	N/A	0.308	0.287
	DDPM	-2.98	0.121	0.066
$k=3$	Naive sampling	-3.52	0.914	0.248
	NF-SAILS	$\mathbf{-1.83}$	**0.056**	**0.034**
	WGAN-GP	N/A	0.973	0.237
	DDPM	-2.97	0.364	0.124
$k=4$	Naive sampling	-3.08	0.967	0.295
	NF-SAILS	$\mathbf{-1.07}$	**0.044**	**0.041**
	WGAN-GP	N/A	1.012	0.317
	DDPM	-1.81	0.427	0.127
$k=6$	Naive sampling	-2.06	1.219	0.205
	NF-SAILS	$\mathbf{-1.09}$	**0.039**	0.309
	WGAN-GP	N/A	1.392	0.212
	DDPM	-1.99	1.004	**0.179**
$k=9$	Naive sampling	-2.297	1.764	0.215
	NF-SAILS	$\mathbf{-0.801}$	**0.151**	**0.052**
	WGAN-GP	N/A	1.939	0.340
	DDPM	-1.258	0.906	0.205

Table 4.2: Goodness-of-fit of the generated samples w.r.t. the number k of Gaussians. Reported scores result from the average over 5 Monte Carlo runs.

The NF model $f(\cdot)$ is a RealNVP [36] composed of $M = 4$ flows, each composed of two three-layer neural networks ($d \to 16 \to 16 \to d$) using an hyperbolic tangent activation function. We use the Adam optimizer with learning rate 10^{-4} and a batch size of 500 samples.

Table 4.2 reports the considered metrics when comparing the proposed NF-SAILS sampling method to a naive sampling (see Section 4.1.3) or to state-of-the-art sampling techniques from the literature, namely Wasserstein GAN with gradient penalty (WGAN-GP) [115] and denoising diffusion probabilistic models (DDPM) [47]. These results show that NF-SAILS consistently competes favorably against the compared methods, in particular as the degree of multimodality of the distribution increases. Note that WGAN-GP exploits a GAN architecture. Thus, contrary to the proposed NF-based sampling method, it is unable to provide an explicit evaluation of the likelihood, which explains the N/A values in the table.

Figure 4.3 illustrates this result for $k = 6$ and shows that our method



(a) Naive sampling (b) NF-SAILS (c) WGAN-GP (d) DDPM

Figure 4.3: Mixture of $k = 6$ Gaussian distributions (green), and 1000 generated samples (blue). The proposed NF-SAILS method in Fig. 4.3b does not generate samples in-between modes.

considerably reduces the number of out-of-distribution generated samples. These qualitative results demonstrate that, unlike other models, our method allows each mode to be sampled evenly, without producing out-of-dribution samples.

Figure 4.4 shows the difference of sampling quality between naive sampling and the proposed NF-SAILS method for RealNVP model trained on k-mixtures of Gaussians for $k \in \{2, 3, 4, 9\}$. Our model seems to sample well the targeted distributions, we also notice that our method produces samples with less variance.

4.7.3 Results obtained on real image datasets

Moreover, we further study the performance of NF-SAILS on three different real image data sets, namely Cifar-10 [109], CelebA [110] and LSUN [111]. Following the same protocol as implemented by [39], we use a Glow architecture where each neural network are composed of three convolutional layers. The two hidden layers have ReLU activation functions and 512 channels. The first and last convolutions are 3×3, while the center convolution is 1×1, since its input and output have a large number of channels, in contrast with the first and last convolutions.

We compare the FID score as well as the average negative log-likelihood (bpd), keeping all training conditions constant and averaging the results over 10 Monte Carlo runs. The results are depicted in Fig. 4.5 reports the results when compared to those obtained by naive sampling or WGAN-GP [115]. As shown by the different panels of this figure, the proposed NF-SAILS method considerably improves the quality of the generated images, both quantitatively (in term of FID) and semantically. Our methodology

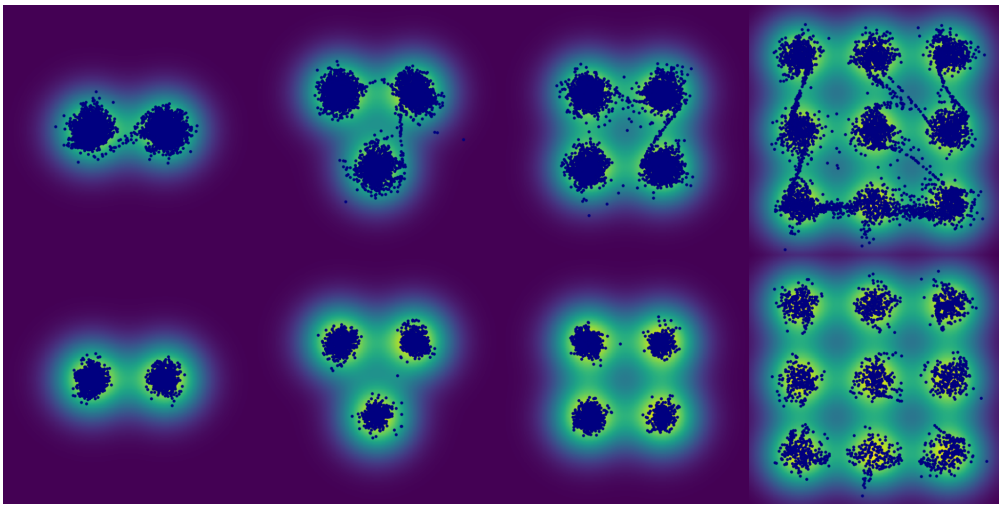


Figure 4.4: Mixture of k Gaussian distributions (green), and 1000 samples (blue) generated by the naive sampling (top) and the proposed NF-SAILS method (bottom) with, from left to right, $k = 2$, $k = 3$, $k = 4$ and $k = 9$.

compares favourably w.r.t. to WGAN-GP for the two data sets CelebA and LSUN.

4.8 Conclusion

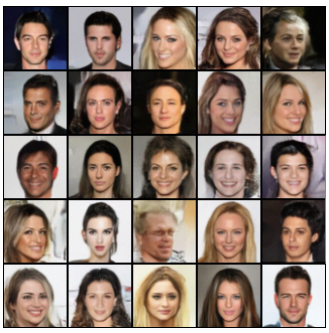
This chapter discusses the sampling from the target distribution learnt by a normalizing flow. Architectural constraints prevent normalizing flows to properly learn disconnect support measures due to the topological mismatch between the latent and target spaces. Moreover, we theoretically prove that the Jacobian norm of the transformation become arbitrarily large to closely represent such target measures. The conducted analysis exhibits the existence of pathological areas in the latent space corresponding to points with exploding Jacobian norms. Using a naive sampling strategy leads to out of distribution samples located in these areas. To overcome this issue, we propose a new sampling procedure based on a Langevin diffusion directly formulated in the latent space. This sampling is interpreted as a Riemanian manifold Metropolis adjusted Langevin algorithm, whose metrics is driven by the Jacobian of the learnt transformation. This local exploration of the latent space is complemented by an independent Metropolis-Hastings kernel which allows moves from one high probability region to another while avoiding crossing pathological areas. One particular advantage of the proposed method is that it can be applied to any pre-trained NF model. Indeed it

	↓ bpd	↓ FID
Naive sampling	3.35	44.6
NF-SAILS	**3.08**	43.11
WGAN-GP	N/A	**18.8**



(a) Cifar10

	↓ bpd	↓ FID
Naive sampling	1.03	15.82
NF-SAILS	**0.96**	14.12
WGAN-GP	N/A	**12.89**



(b) CelebA

	↓ bpd	↓ FID
Naive sampling	2.38	8.91
NF-SAILS	**2.11**	**7.86**
WGAN-GP	N/A	9.56



(c) LSUN (bedroom)

Figure 4.5: Tables report quantitative and perceptual metrics computed from the samples generated by the compared methods. The figures show some samples generated from Glow using the proposed NF-SAILS method.

does not require a particular training strategy of the NF or to adapt the distribution assumed in the latent space. The performances of the proposed sampling strategy compares favorably to state-of-the art, with very few out-of-distribution samples.

Chapter 5

Plug-and-Play split Gibbs sampler

Contents

5.1 Solving inverse problems: state of the art 101
5.1.1 Problem statement 101
5.1.2 A brief history 101
5.2 SGS and generative models for PnP 104
5.2.1 Split Gibbs sampling (SGS) 104
5.2.2 Denoising diffusion probabilistic models (DDPM) . 105
5.2.3 Proposed DDPM-based PnP-SGS algorithm 107
5.2.4 Some insights into the number t^* of steps 108
5.3 Application to Bayesian inverse problems 109
5.3.1 Inverse problems 109
5.3.2 Image deblurring 111
5.3.3 Image inpainting 112
5.3.4 Image super-resolution 112
5.4 Experiments 113
5.4.1 Experimental setup 113
5.4.2 Compared methods & figures-of-merit 114
5.4.3 Technical details of PnP-SGS 117
5.4.4 Experimental results 118
5.5 Conclusion 122

This chapter introduces a stochastic plug-and-play (PnP) sampling algorithm that leverages variable splitting to efficiently sample from a posterior distribution. The algorithm based on split Gibbs sampling (SGS) draws inspiration from the alternating direction method of multipliers (ADMM). It divides the challenging task of posterior sampling into two simpler sampling problems. The first problem depends on the likelihood function, while the second is interpreted as a Bayesian denoising problem that can be readily carried out by a deep generative model. Specifically, for an illustrative purpose, the proposed method is implemented using state-of-the-art diffusion-based generative models. Akin to its deterministic PnP-based counterparts, the proposed method exhibits the great advantage of not requiring an explicit choice of the prior distribution, which is rather encoded into a pre-trained generative model. However, unlike optimization methods (e.g., PnP-ADMM) which generally provide only point estimates, the proposed approach allows conventional Bayesian estimators to be accompanied by confidence intervals at a reasonable additional computational cost. Experiments on commonly studied image processing problems illustrate the efficiency of the proposed sampling strategy. Its performance advantageously compares to recent state-of-the-art optimization and sampling methods.

Section 5.1 outlines existing methods for solving inverse problems. Section 5.2 recalls necessary notions about the split Gibbs sampler (SGS) and Denoising Diffusion Probabilistic Models (DDPMs) that will be used as PnP-denoisers in the sequel. Section 5.3 describes how the proposed PnP-SGS adapts to several usual inverse problems frequently encountered in image processing. Section 5.4 describes numerical experiments and reports the performances in comparison with state-of-the-art methods. Section 5.5 finally enlightens the contributions.

This work is based on a long paper which has been submitted to an international journal:

F. Coeurdoux, N. Dobigeon, P. Chainais. *"Plug-and-Play split Gibbs sampler: embedding deep generative priors in Bayesian inference"*, Submitted to IEEE Transactions on Image Processing, arXiv:2304.11134

5.1 Solving inverse problems: state of the art

5.1.1 Problem statement

Many scientific problems raise the challenge of inferring an unknown object of interest $\mathbf{x} \in \mathbb{R}^N$ from partial and noisy measurements $\mathbf{y} \in \mathbb{R}^M$. These inverse problems frequently encountered in image processing are typically formulated as the minimization task

$$\min_{\mathbf{x}} f(\mathbf{x}, \mathbf{y}) + g(\mathbf{x}) \tag{5.1}$$

where $f(\cdot, \mathbf{y})$ denotes the data-fitting term. Due to the ill-posed or ill-conditioned nature of the inverse problem, it is often not possible to uniquely and stably recover $\mathbf{x}$ from the sole observations $\mathbf{y}$. Therefore, additional information about the unknown object $\mathbf{x}$ is incorporated in the form of the regularization $g(\cdot)$ to obtain a well-posed estimation problem, leading to meaningful solutions [116]. Due to the increasing volume, dimensionality, and variety of available data, solving such inference problems can be computationally demanding and may rely on methods such as variational optimization or stochastic sampling.

5.1.2 A brief history

Until recently, most of the optimization methods have relied on priors designed as explicit model-based regularizations such as the total variation, promoting piecewise constant behaviors, or the ℓ_1 norm, promoting sparsity. In this context, convex optimization algorithms have played an important role and their convergence properties have been well-established [117, 118, 119, 120]. However, for an always larger family of problems related to image processing, methods based on explicit convex priors are now significantly outperformed by deep learning based approaches.

End to End

There exist a number of deep neural network architectures that can directly learn a description of the solution space [121, 122, 123, 124, 34]. Such so-called end-to-end approaches that bypass the problem of explicitly defining the prior knowledge do not even need the knowledge of the forward operator itself. Instead, they are implicitly learnt from a large data set of degraded images (i.e., network input) along with their original versions (i.e., network output) when training the network. However, such end-to-end methods suffer from the lack of interpretability and generality of black-box deep neural

networks (DNN). Moreover, they do not take advantage of the generally well-established expertise of the end-users about the acquisition or damaging protocols, which makes the training process particularly energy and data intensive.

Plug-and-play

To overcome these limitations, more and more deep learning based methods propose to combine DNN with conventional optimization algorithms within the so-called plug-and-play (PnP) framework [125]. The main ingredient of PnP approaches is a variable splitting strategy as implemented by half-quadratic splitting (HQS) [126] or alternating direction method of multipliers (ADMM) [127]. The main idea of this splitting consists in introducing an auxiliary variable $\mathbf{z}$ such that the problem (5.1) rewrites

$$\min_{\mathbf{x},\mathbf{z}} f(\mathbf{x},\mathbf{y}) + g(\mathbf{z}) \quad \text{subject to} \quad \mathbf{x} = \mathbf{z}. \tag{5.2}$$

The equality constraint ensures that solving (5.2) is equivalent to solving the initial problem (5.1). Adopting an alternate minimization strategy, this tricks permits to separately deal with the data-fitting term and the regularization [128]. In particular, the subproblem with respect to $\mathbf{z}$ is solved by using the proximal operator of the regularization term, which can be interpreted as a denoising task. Recent PnP methods replace this proximal mapping by a DNN-based denoiser that implicitly encodes the regularization. They now stand as a reference that yield state-of-the-art performance in a variety of applications [129, 130].

Splitting for sampling

However, PnP-based optimization algorithms generally produce point estimates only. More generally, except in special cases [131], optimization methods do not give any information about the posterior distribution

$$\pi(\mathbf{x}) \triangleq p(\mathbf{x}|\mathbf{y}) \propto \exp[-f(\mathbf{x},\mathbf{y}) - g(\mathbf{x})] \tag{5.3}$$

associated with (5.1) and do not quantify uncertainties. Conversely, Bayesian approaches and Markov chain Monte Carlo (MCMC) methods have the great advantage of providing a comprehensive description of (5.3) in very general settings. In particular, this knowledge permits to derive credibility intervals on the parameter $\mathbf{x}$ of interest. This uncertainty quantification is often of crucial importance, for instance when only very few observations are available [132], when one is interested in extreme events [133] or when no ground

truth is available, like in astrophysics. There is still a price to pay: sampling methods and MCMC in particular suffer from their high computational cost which can be prohibitive in high-dimensional problems. Optimization-driven Monte Carlo methods [134, 135, 136] tentatively overcome this limitation.

More recently, the split Gibbs sampler (SGS) [137] proposes to sample from an augmented distribution defined as an asymptotically exact data augmentation model [138]. By introducing an auxiliary variable as in (5.2), it yields a divide-to-conquer strategy by splitting the initial sampling problem into individual simpler sampling tasks. Sampling according to the augmented distribution with Gibbs steps permits to deal separately with the distinct components of the problem, i.e., the likelihood on the one hand and the prior on the other hand. Per se, SGS can be seen as a stochastic counterpart of HQS or ADMM algorithms. It both makes the sampling more scalable to high dimensions and significantly improves the mixing properties of the Markov chain.

Proposed approach : PnP-SGS

The main contribution of the work reported in this chapter is to provide a straightforward and systematic instantiation of the PnP paradigm within a Monte Carlo sampling framework. This is made possible thanks to the splitting strategy implemented by the SGS scheme. Moreover the timeliness of devising such an approach can be easily justified by the recent advances in the design of powerful deep generative models. The proposed approach coined as PnP-SGS is based on three main rationales. First, as any PnP-based methods, PnP-SGS allows Bayesian inference problems to be solved without explicitly defining a prior distribution, which is rather implicitly encoded into a DNN trained beforehand. Second, we show that diffusion-based or score-based models [139, 140, 141] initially derived for generative purposes can be diverted to be employed as universal stochastic denoisers. Third, PnP-SGS generate samples that can be used to build confidence intervals, which is not possible with its deterministic counterpart, i.e., PnP-ADMM, that only provides point estimates [129, 130]. High dimensional image processing experiments will illustrate the strong potential of the proposed approach when using a diffusion model [142, 47, 143, 48, 144, 145, 146] as a denoiser. These extensive experiments include various inverse problems such as inpainting, super-resolution, and deblurring. The experimental results show that the proposed PnP-SGS is a general approach to solve ill-posed inverse problems in high dimension with superior quality and uncertainty quantification.

5.2 SGS and generative models for PnP

5.2.1 Split Gibbs sampling (SGS)

Starting from the target posterior distribution (5.3), the introduction of a splitting variable $\mathbf{z} \in \mathbb{R}^N$ leads to the augmented distribution

$$\begin{aligned} \pi_\rho(\mathbf{x}, \mathbf{z}) &\triangleq p\left(\mathbf{x}, \mathbf{z} | \mathbf{y}; \rho^2\right) \\ &\propto \exp\left[-f(\mathbf{x}, \mathbf{y}) - g(\mathbf{z}) - \frac{1}{2\rho^2}\|\mathbf{x} - \mathbf{z}\|_2^2\right] \end{aligned} \tag{5.4}$$

where ρ is a positive parameter that controls the coupling between $\mathbf{x}$ and $\mathbf{z}$. As shown in [138], for a large variety of coupling kernels including the quadratic one, the marginal distribution of $\mathbf{x}$ under π_ρ in (5.4) coincides with the target distribution π in (5.3) when ρ^2 tends to zero, i.e.,

$$\|\pi - \pi_\rho\|_{\mathrm{TV}} \underset{\rho^2 \to 0}{\to} 0 \tag{5.5}$$

which defines an asymptotically exact data augmentation scheme with solid theoretical foundations [138]. In other words, the original target distribution $\pi(\mathbf{x})$ in ((5.3)) is recovered from the marginal distribution $\pi_\rho(\mathbf{x})$ derived from (5.4) in the limiting case $\rho \to 0$. Instead of sampling directly according to $\pi(\mathbf{x})$, SGS proposes to sample according to the augmented distribution $\pi_\rho(\mathbf{x}, \mathbf{z})$ using Gibbs steps. More specifically, the associated conditional distributions to sample from $\pi_\rho(\mathbf{x}, \mathbf{z})$ are given by

$$p\left(\mathbf{x} \mid \mathbf{z}, \mathbf{y}; \rho^2\right) \propto \exp\left[-f(\mathbf{x}, \mathbf{y}) - \frac{1}{2\rho^2}\|\mathbf{x} - \mathbf{z}\|_2^2\right] \tag{5.6}$$

$$p\left(\mathbf{z} \mid \mathbf{x}; \rho^2\right) \propto \exp\left[-g(\mathbf{z}) - \frac{1}{2\rho^2}\|\mathbf{z} - \mathbf{x}\|_2^2\right] \tag{5.7}$$

It is now clear that sampling alternatively from ((5.6)) and ((5.7)) dissociates the potential functions $f(\cdot, \mathbf{y})$ and $g(\cdot)$ associated with the likelihood and the prior distribution, respectively. The MCMC algorithm associated with the split distributions (5.4) are special instances of Gibbs samplers where samples are alternatively drawn according to the conditional distributions of each variable (5.6) and (5.7). The directed acyclic graph (DAG) associated with the proposed splitting model is depicted in Fig.5.1 As a consequence, SGS inherits well-known advantages already exhibited by its deterministic counterparts (i.e., HQS and ADMM), e.g., easier implementations, faster convergences and possibly distributed computations. In particular, sampling according to ((5.6)) can be interpreted as solving the initial problem defined

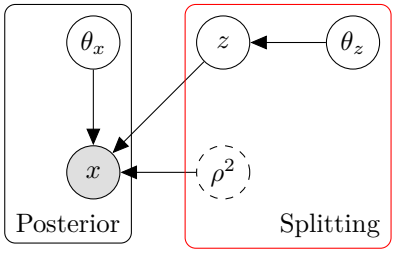


Figure 5.1: DAGs associated with the proposed hierarchical Bayesian model. In black: DAG associated to (5.6) and in red: DAG associated to (5.7). θ_z stand for that parameters of the deep learning based prior model and θ_x for possible additional parameters of the forward model.

by the same potential function $f(\cdot, \mathbf{y})$ but now granted with a Gaussian prior distribution of mean $\mathbf{z}$ and diagonal covariance matrix $\rho^2\mathbf{I}$. It is thus expected to be significantly simpler than sampling according to the initial posterior distribution $\pi(\mathbf{x})$ defined by ((5.3)).

Moreover, note that the conditional distribution ((5.7)) can be interpreted as the posterior distribution associated with a Bayesian denoising problem. Its goal is to recover an object $\mathbf{z}$ from a noisy observation $\mathbf{x}$ contaminated by an additive white Gaussian noise with variance ρ^2. Instead of sampling directly from ((5.7)), we propose to resort to deep generative models used as stochastic denoisers. Generative adversarial network (GAN), variational autoencoders (VAE) or more recently denoising diffusion probabilistic models (DDPM) are powerful candidates to tackle this task [139, 140, 141]. This work instantiates the PnP-SGS framework and reports experimental results based on DDPM-based denoisers. Note however that any pre-trained probabilistic denoising generative model can be plugged into the proposed approach.

5.2.2 Denoising diffusion probabilistic models (DDPM)

Denoising diffusion models [142, 47, 143, 48] and score based models [144, 145, 146] are trendy classes of generative models. They have recently drawn significant attention from the community due to their state-of-the-art performances. Although nourished by different inspirations, they share very similar aspects and can be presented as variants of each other [147, 143, 146]. They are often referred to under the generic name *diffusion models* 1.4.2.

DDPM as generative models

A denoising diffusion probabilistic model [142] makes use of two Markov chains: a forward chain that perturbs data to pure noise, and a backward chain that converts noise back to data. The former is typically model-based designed with the goal of transforming any data distribution into a simple prior distribution, i.e., a standard Gaussian. Conversely the latter Markov chain aims at reversing the noising process by learning transition kernels parametrized by a DNN. Once the DNN has been trained, new data points can be generated by first drawing from the prior distribution, and then sampling through the backward Markov chain.

Formally, given a data distribution $\mathbf{v}_0 \sim p(\mathbf{v}_0)$, the forward Markov process generates a sequence of random variables $\mathbf{v}_t \in \mathbb{R}^N$, $t \in \{0, \ldots, T\}$ according to the transition kernel $p(\mathbf{v}_t \mid \mathbf{v}_{t-1})$. Using the probability chain rule and the Markovian property, the joint distribution $p(\mathbf{v}_1, \ldots, \mathbf{v}_T \mid \mathbf{v}_0)$ can be factorized as

$$p(\mathbf{v}_1, \ldots, \mathbf{v}_T \mid \mathbf{v}_0) = \prod_{t=1}^{T} p(\mathbf{v}_t \mid \mathbf{v}_{t-1}). \tag{5.8}$$

In DDPMs, the transition kernel $p(\mathbf{v}_t \mid \mathbf{v}_{t-1})$ is arbitrarily chosen to incrementally transform the data distribution $p(\mathbf{v}_0)$ into a tractable prior distribution $p(\mathbf{v}_T) \approx \mathcal{N}(\mathbf{v}_T; \mathbf{0}, \mathbf{I})$. One typical design for the transition kernel exploits a Gaussian perturbation and the most common choice for the transition kernel is

$$p(\mathbf{v}_t \mid \mathbf{v}_{t-1}) = \mathcal{N}\left(\mathbf{v}_t; \sqrt{1-\beta(t)}\mathbf{v}_{t-1}, \beta(t)\mathbf{I}\right) \tag{5.9}$$

where $\beta(t) \in (0, 1)$ is a predefined function which plays a key role. It directly adjusts the amount of noise along the process such that larger values lead to noisier samples. Conventionally, it is chosen as a linearly increasing function [47]. More recent techniques have proposed to use cosine-based functions [49]. Intuitively speaking, this forward process slowly injects noise into data until all structures are lost and only noise prevails.

For generating new data samples, DDPMs start by first drawing a sample $\mathbf{v}_T$ from an instrumental prior distribution $q(\mathbf{v}_T) = \mathcal{N}(\mathbf{v}_T; \mathbf{0}, \mathbf{I})$. Then DDPMs gradually remove noise by running a Markov chain in the reverse time direction. This Markov chain is defined thanks to a kernel modeled by DNNs. The learnable transition kernel $q_{\boldsymbol{\theta}}(\mathbf{v}_{t-1}|\mathbf{v}_t)$ takes the form of

$$q_{\boldsymbol{\theta}}(\mathbf{v}_{t-1} \mid \mathbf{v}_t) = \mathcal{N}(\mathbf{v}_{t-1}; \boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{v}_t, t), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\mathbf{v}_t, t)) \tag{5.10}$$

where the mean $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{v}_t, t)$ and the covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\mathbf{v}_t, t)$ are DNNs parametrized by $\boldsymbol{\theta}$ and t with $\mathbf{v}_t$ as an input.

DDPM as stochastic denoisers

According to the above discussion, it is clear that the forward diffusion process (5.9) progressively adds noise to a noise-free image $\mathbf{v}_0$. Following a discretization scheme generally adopted by these deep generative models, each $\mathbf{v}_t$ corresponds to a scaled version of $\mathbf{v}_{t-1}$ corrupted by a Gaussian noise with covariance matrix $\beta(t)\mathbf{I}$. Thanks to the factorization induced by the direct Markov chain and the Gaussian nature of the transition kernel, the transition from the original image $\mathbf{v}_0$ to any intermediate noisy image $\mathbf{v}_t$ can be written as

$$p\left(\mathbf{v}_t \mid \mathbf{v}_0\right) = \mathcal{N}\left(\mathbf{v}_t; \sqrt{\bar{\alpha}(t)}\mathbf{v}_0, \alpha(t)\mathbf{I}\right) \tag{5.11}$$

$$\text{where} \quad \alpha(t) = \prod_{j=1}^{t}\left(1-\beta(j)\right) \tag{5.12}$$

and $\bar{\alpha}(t) = 1 - \alpha(t)$. In other words, at any arbitrary time instant $t^* < T$, the image $\mathbf{v}_{t^*}$ resulting from t^* steps of the forward process is a noisy version of the input image $\mathbf{v}_0$ corrupted by a Gaussian noise of variance $\alpha(t^*)$.

Therefore, it appears that a trained DDPM can be used as a stochastic Gaussian denoiser. Contrary to the normal use of a DDPM as a generator (see above), the key idea is rather to start the backward diffusion process from a noisy image $\mathbf{v}_{t^*}$ for some t^* and not as usual from a realization of noise $\mathbf{v}_T$. The noise-free image $\mathbf{v}_0$ can be recovered by applying the backward process defined by (5.10) from time instant t^*.

5.2.3 Proposed DDPM-based PnP-SGS algorithm

In a nutshell, the proposed PnP-SGS alternatively samples according to the conditional posterior distributions (5.6) and (5.7). Along this iterative process, SGS generates a set of N_{MC} samples $\left\{\mathbf{x}^{(n)}, \mathbf{z}^{(n)}\right\}_{n=1}^{N_{\text{MC}}}$ asymptotically distributed according to the augmented posterior $\pi_\rho(\mathbf{x}, \mathbf{z})$. From this set of samples, various Bayesian quantities can be approximated, such as Bayesian estimators and credibility intervals. In particular, the samples $\left\{\mathbf{x}^{(n)}\right\}_{n=1}^{N_{\text{MC}}}$ are marginally distributed according to $\pi_\rho(\mathbf{x})$. Thus the minimum mean square estimator (MMSE or posterior mean) $\hat{\mathbf{x}}_{\text{MMSE}} = \text{E}[\mathbf{x}|\mathbf{y}]$ associated with π_ρ can be easily approximated by the empirical average

$$\hat{\mathbf{x}}_{\text{MMSE}} \approx \frac{1}{N_{\text{MC}} - N_{\text{bi}}} \sum_{n=N_{\text{bi}}+1}^{N_{\text{MC}}} \mathbf{x}^{(n)} \tag{5.13}$$

where N_{bi} is the number of burn-in iterations.

Regarding the first step of SGS, sampling according to (5.6) is problem dependent and should be suitably adapted to the targeted task. For illustration purpose, it will be explicitly specified for various imaging problems in Section 5.3. As expected and already pointed out in Section 5.2.1, sampling according to (5.6) will appear significantly simpler than directly sampling according to the target posterior distribution $\pi(\mathbf{x})$ defined by (5.3).

Regarding the second step of SGS, at the nth iteration of the algorithm, sampling according to (5.7) is interpreted as a stochastic denoising of the current value $\mathbf{x}^{(n)}$. This sampling according to (5.7) is performed in a PnP manner thanks to a previously trained DDPM, following the strategy detailed in Section 5.2.2. With the notations adopted in the previous paragraph, it assigns the current sample $\mathbf{x}^{(n)}$ to the variable $\mathbf{v}_{t^*}$ for some t^* at iteration n and then iterates the backward diffusion (5.10). After t^* steps, the produced denoised image $\mathbf{v}_0$ is allocated to the new sample $\mathbf{z}^{(n)}$ according to (5.7) of the current SGS iteration.

Note that DDPMs used as generators are known to be generally computationally demanding due to the number T of overall steps involved in the backward process. The proposed approach obviates this impediment by initiating the process from a generally weakly noisy image, which significantly reduces the necessary number $t^* \ll T$ of denoising steps to be applied [148]. Next section provides some insights into this number t^* and proposes a systematic and reliable strategy to adjust it.

5.2.4 Some insights into the number t^* of steps

Role of t^*: control regularization

This section discusses the role and the tuning of the time instant t^* which defines the number of denoising steps to be applied at a given iteration of the SGS sampler. As already stated, Eq. (5.11) shows that the variance of the noise corrupting $\mathbf{v}_0$ after t^* transitions of the forward Markov chain is $\alpha(t^*)$. This variance is defined by the product (5.12) of continuous strictly monotone functions $\beta(\cdot)$, thus it is also continuous and strictly monotone. This has two consequences: *i)* a level of noise $\alpha(t^*)$ is associated to a unique instant t^* of the forward diffusion process (i.e., $\alpha(t)$ is an invertible function of t) and *ii)* the larger t^*, the noisier the image $\mathbf{v}_{t^*}$. Reciprocally, when applying the backward diffusion to a noisy image, the larger t^*, the higher the impact of the denoising, that is of the regularization. Note that the DDPM, that is used for regularization here, has no explicit hyperparameter. An important consequence is that, within the framework of PnP-SGS, the number t^* of denoising steps can be interpreted as the hyperparameter that adjusts the

amount of imposed regularization, the coupling parameter ρ being kept fixed.

Inferring t^*

The proposed approach capitalizes on the explicit and unequivocal mapping between the hyperparameter t^* and the variance $\alpha(t^*)$ of the noise contained in $\mathbf{v}_{t^*}$. This relationship permits a simple and efficient strategy to set the number t^* of required denoising steps (5.10) when sampling according to (5.7). Given a current sample $\mathbf{x}^{(n)}$ generated by SGS, the identification of the appropriate instant t^* to generate $\mathbf{z}^{(n)}$ according to (5.7) boils down to estimating the level $\alpha(t^*)$ of the noise corrupting the sample $\mathbf{x}^{(n)}$. This is possible using any good conventional estimator $\hat{\sigma} = \Phi(\mathbf{x}^{(n)})$ of the noise level in $\mathbf{x}^{(n)}$ [149, 150, 151], see Appendix 5.4.3 for implementation details. Since the function $t \to \alpha(t)$ is invertible, one can finally set $\widehat{t^*} = \alpha^{-1}(\hat{\sigma}^2)$ to start the backward diffusion (5.10). Appendix 5.4.3 discusses technical details of the inversion of $\alpha(\cdot)$.

In practice, during the experiments reported in Section 5.4, the number $\widehat{t^*}$ of achieved steps has been shown to stabilize at a fixed value after the burn-in period of PnP-SGS (Figure 5.5). Therefore the transition kernel associated with the denoising procedure becomes invariant, which ensures that SGS converges towards a stationary distribution π_ρ; recall that ρ is fixed, typically of order 1.

The resulting distribution π_ρ is eventually similar to (5.4) where the role of the explicit regularizing potential $g(\cdot)$ has been implicitly replaced by the DDPM.

Algorithm 6 describes the final sampling PnP-SGS algorithm using a DDPM for the denoising step, with the proposed strategy to set the hyperparameter t^*.

5.3 Application to Bayesian inverse problems

5.3.1 Inverse problems

The proposed PnP-SGS method is now instantiated for three different imaging problems, namely deblurring, inpainting and superresolution, following the protocols already considered in [137]. The considered linear Gaussian inverse problems define an archetypal class of problems that can efficiently tackled by the proposed method. More specifically, a degraded image $\mathbf{y}$ is observed and one wants to infer a restored image $\mathbf{x}$ under the linear model

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{n} \tag{5.14}$$

Algorithm 6: PnP-SGS using DDPM

Input : Parameter ρ^2, total number of iterations N_{MC}, number of burn-in iterations N_{bi}, pre-trainted DDPM $s_{\boldsymbol{\theta}}(\cdot, \cdot)$, scheduling variance function $\alpha(\cdot)$, initialization $\mathbf{z}^{(0)}$

for $n \leftarrow 1$ **to** N_{MC} **do**
- *# Sampling the variable of interest* $\mathbf{x}^{(n)}$
- Draw $\mathbf{x}^{(n)} \sim p(\mathbf{x} \mid \mathbf{z}, \mathbf{y}; \rho^2)$ according to (5.6)
- *# Estimating noise level in* $\mathbf{x}^{(n)}$
- Set $\hat{\sigma} = \Phi(\mathbf{x}^{(n)})$ using [149]
- *# Setting the number of diffusion steps to denoise* $\mathbf{x}^{(n)}$
- Set $\widehat{t^*} = \alpha^{-1}(\hat{\sigma}^2)$
- *# Sampling the splitting variable* $\mathbf{z}^{(n)}$ *according to* (5.7)
- Set $\mathbf{v}_{\widehat{t^*}} = \mathbf{x}^{(n)}$
- **for** $j \leftarrow \widehat{t^*}$ **downto** 1 **do**
  - Draw $\mathbf{v}_{j-1} \sim q_{\boldsymbol{\theta}}(\mathbf{v}_{j-1} \mid \mathbf{v}_j)$ according to (5.10)
- **end**
- Set $\mathbf{z}^{(n)} = \mathbf{v}_0$

end

Output: Collection of samples $\left\{\mathbf{x}^{(n)}, \mathbf{z}^{(n)}\right\}_{t=N_{\text{bi}}+1}^{N_{\text{MC}}}$ asymptotically distributed according to (5.4).

where $\mathbf{H}$ is a forward operator and $\mathbf{n}$ accounts for noise or error modeling. Assuming that $\mathbf{n}$ is a Gaussian random vector with covariance matrix $\mathbf{\Omega}^{-1}$, the likelihood function associated with the observation $\mathbf{y}$ writes

$$p(\mathbf{y} \mid \mathbf{x}) \propto \exp\left[-\frac{1}{2}(\mathbf{H}\mathbf{x}-\mathbf{y})^T\mathbf{\Omega}(\mathbf{H}\mathbf{x}-\mathbf{y})\right].$$

In most applicative contexts, inferring the unknown parameter vector $\mathbf{x}$ from the observation vector $\mathbf{y}$ under the linear model (5.14) is known to be an ill-posed or ill-conditioned inverse problem. A common approach to tackle such problems consists in using some regularization defined through the choice of a prior distribution $p(\mathbf{x}) \propto \exp[-g(\mathbf{x})]$, leading to the posterior distribution (5.3). Instead of explicitly specifying the potential function $g(\cdot)$ in (5.3), the proposed PnP-SGS algorithm targets an augmented posterior similar to (5.4) to capitalize on a pre-trained denoising diffusion model presented in Section 5.2.2.

The three considered tasks mainly differ by the nature of the linear operator $\mathbf{H}$. Following the SGS algorithmic scheme, a special care should be taken to ensure an efficient sampling according to the conditional posterior (5.6) which involves $\mathbf{H}$, see Algo. 6, line 3. Since the sampling according to (5.7) does not depend on the forward operator, it is achieved in a unique manner from a DDPM. Thus the sequel of this section is only devoted to the technical derivations associated with (5.6). Experimental results obtained by the proposed PnP-SGS will be reported in Section 5.4.

5.3.2 Image deblurring

In this setup, the operator $\mathbf{H}$ is assumed to be an $N \times N$ circulant convolution matrix associated to a blurring kernel. The noise covariance matrix is assumed to be diagonal, i.e., $\mathbf{\Omega}^{-1} = \text{diag}\,[\sigma_1^2, \ldots, \sigma_N^2]$ where distinct diagonal elements mimic a spatially-variant noise level. Even when choosing a simple model-based regularizing potential $g(\cdot)$, direct sampling according to the posterior distribution (5.3) may remain a challenging task, mainly due to the presence of the precision matrix $\mathbf{\Omega}$ which prevents a direct computation in the Fourier domain. Conversely, the proposed PnP-SGS algorithm yields the conditional distribution (5.6) defined here as

$$p(\mathbf{x} \mid \mathbf{z}, \mathbf{y}; \rho^2) = \mathcal{N}\left(\mathbf{x}; \boldsymbol{\mu}_{\mathbf{x}}, \mathbf{Q}_{\mathbf{x}}^{-1}\right) \tag{5.15}$$

with

$$\begin{cases} \mathbf{Q}_{\mathbf{x}} = \mathbf{H}^T\mathbf{\Omega}\mathbf{H} + \frac{1}{\rho^2}\mathbf{I}_N \\ \boldsymbol{\mu}_{\mathbf{x}} = \mathbf{Q}_{\mathbf{x}}^{-1}\left(\mathbf{H}^T\mathbf{\Omega}\mathbf{y} + \frac{1}{\rho^2}\mathbf{z}\right). \end{cases} \tag{5.16}$$

Thanks to the splitting trick inherent to the proposed PnP-SGS algorithm, this step does not depend on $g(\cdot)$ and boils down to a high-dimensional Gaussian sampling task. This task has been deeply investigated in [108] and can be efficiently achieved by using the auxiliary method of [152]. Finally, sampling from (5.7) is straightforward using the pre-trained network as discussed in Section 5.2.2.

5.3.3 Image inpainting

Image inpainting problems aim at recovering an original image $\mathbf{x} \in \mathbb{R}^N$ from the noisy and partial measurements $\mathbf{y} \in \mathbb{R}^M$ under the linear model (5.14). The operator $\mathbf{H} \in \{0,1\}^{N \times M}$ now stands for a binary matrix associated with a irregular subsampling with $M \ll N$. The noise is assumed to be white and Gaussian such that $\mathbf{\Omega}^{-1} = \sigma^2 \mathbf{I}_M$. As for the deblurring task, the conditional distribution (5.6) is (5.15) with

$$\begin{cases} \mathbf{Q}_{\mathbf{x}} = \frac{1}{\sigma^2}\mathbf{H}^T\mathbf{H} + \frac{1}{\rho^2}\mathbf{I}_N \\ \boldsymbol{\mu}_{\mathbf{x}} = \mathbf{Q}_{\mathbf{x}}^{-1}\left(\frac{1}{\sigma^2}\mathbf{H}^T\mathbf{y} + \frac{1}{\rho^2}\mathbf{z}\right). \end{cases} \tag{5.17}$$

The difficulty of sampling according to this Gaussian distribution comes from the operator $\mathbf{H}$ which is not diagonalizable in the Fourier domain. However, since it consists of a subset of rows of the identity matrix $\mathbf{I}_N$, one has $\mathbf{H}\mathbf{H}^T = \mathbf{I}_M$ and the Sherman-Morrison-Woodbury formula yields

$$\mathbf{Q}_{\mathbf{x}}^{-1} = \rho^2 \left(\mathbf{I}_N - \frac{\rho^2}{\sigma^2 + \rho^2}\mathbf{H}^T\mathbf{H}\right). \tag{5.18}$$

Since $\mathbf{H}^T\mathbf{H}$ is diagonal, the covariance matrix (5.18) is diagonal and sampling from (5.15) can be conducted efficiently with the exact perturbation-optimization (E-PO) algorithm [152].

5.3.4 Image super-resolution

Image super-resolution is characterized by a forward model composed of a blurring kernel followed by a subsampling step. The forward operator writes

$$\mathbf{H} = \mathbf{S}\mathbf{B} \tag{5.19}$$

where $\mathbf{B}$ is a $N \times N$ circulant convolution matrix, as in Section 5.3.2, and $\mathbf{S} \in \{0,1\}^{M \times N}$ is associated with a binary mask, as in Section 5.3.3. The noise $\mathbf{n}$ is assumed to be white and Gaussian. To fully benefit from the

advantages of the SGS, two auxiliary variables $\mathbf{z}_1$ and $\mathbf{z}_2$ are introduced to define the augmented posterior distribution

$$p(\mathbf{x}, \mathbf{z}_1, \mathbf{z}_2|\mathbf{y}; \rho_1^2, \rho_2^2) \propto \exp\left[-\frac{1}{2\sigma^2}\|\mathbf{y} - \mathbf{S}\mathbf{z}_1\|^2 - \frac{1}{2\rho_1^2}\|\mathbf{z}_1 - \mathbf{B}\mathbf{x}\|^2 - g(\mathbf{z}_2) - \frac{1}{2\rho_2^2}\|\mathbf{z}_2 - \mathbf{z}_1\|^2\right] \quad (5.20)$$

This double splitting leads to a SGS algorithm which samples alternatively according to the conditional distributions

$$p(\mathbf{z}_1 \mid \mathbf{x}, \mathbf{y}) \propto \exp\left[-\frac{1}{2\sigma^2}\|\mathbf{y} - \mathbf{S}\mathbf{z}_1\|^2 - \frac{1}{2\rho_1^2}\|\mathbf{z}_1 - \mathbf{B}\mathbf{x}\|^2\right] \quad (5.21)$$

$$p(\mathbf{x} \mid \mathbf{z}_1, \mathbf{z}_2) \propto \exp\left[-\frac{1}{2\rho_1^2}\|\mathbf{z}_1 - \mathbf{B}\mathbf{x}\|^2 - \frac{1}{2\rho_2^2}\|\mathbf{z}_1 - \mathbf{z}_2\|^2\right] \quad (5.22)$$

$$p(\mathbf{z}_2 \mid \mathbf{z}_1) \propto \exp\left[-g(\mathbf{z}_2) - \frac{1}{2\rho_2^2}\|\mathbf{z}_2 - \mathbf{z}_1\|^2\right] \quad (5.23)$$

The two distributions (5.21) and (5.22) correspond to the previously discussed tasks of inpainting and deblurring, respectively. Sampling according to the last one (5.23) is achieved thanks to a DDPM used as a stochastic PnP denoiser.

5.4 Experiments

5.4.1 Experimental setup

Experiments have been conducted on two data sets each composed of 1000 RGB images of size 256×256 with various characteristics, namely FFHQ 256×256 [153], and Imagenet 256×256 [154]. Pre-trained diffusion models have been directly taken from [142, 155] and used without any additional fine-tuning. The test images have never been seen by the model while training to avoid any bias due to potentially over-fitted pre-trained models. All images are normalized to the range $(0, 1)$. For the inversion tasks described in Section 5.3, the forward measurement operators have been designed as follows:

- *deblurring*: two blurring kernels are considered, namely a Gaussian blur with a kernel size of 61×61 with standard deviation of 3.0, and a randomly generated motion blur[1] with size 61×61 and intensity value 0.5,

[1]Following the code available at https://github.com/LeviBorodenko/motionblur.

		PnP-SGS	SPA [137]	TV-ADMM [156]	PnP-ADMM [157]	Score-SDE [146]	DDRM [158]	MCG [159]
Inpainting	PSNR ↑	**32.59**	26.09	22.03	8.41	13.52	9.19	<u>21.57</u>
	SSIM ↑	**0.913**	0.524	0.784	0.325	0.437	0.319	<u>0.751</u>
	FID ↓	<u>37.36</u>	71.12	181.56	123.61	76.54	<u>69.71</u>	**29.26**
	LPIPS ↓	**0.144**	0.785	0.463	0.692	0.612	0.587	<u>0.286</u>
Deblurring (Gaussian)	PSNR ↑	**27.96**	23.17	22.37	<u>24.93</u>	7.12	23.36	6.72
	SSIM ↑	**0.837**	0.499	<u>0.801</u>	<u>0.812</u>	0.109	0.767	0.051
	FID ↓	**59.667**	78.67	186.74	90.42	109.07	<u>74.92</u>	101.2
	LPIPS ↓	**0.331**	0.452	0.507	0.441	0.403	<u>0.332</u>	0.340
Deblurring (motion)	PSNR ↑	**28.46**	17.73	21.36	<u>24.65</u>	6.58	N/A	6.72
	SSIM ↑	**0.828**	0.211	0.751	<u>0.825</u>	0.102	N/A	0.055
	FID ↓	**60.01**	103.87	152.39	<u>89.08</u>	292.28	N/A	310.5
	LPIPS ↓	**0.294**	0.446	0.508	<u>0.405</u>	0.657	N/A	0.702
Superres. (×4)	PSNR ↑	**25.99**	N/A	23.86	<u>26.55</u>	17.62	25.36	19.97
	SSIM ↑	<u>0.812</u>	N/A	0.803	**0.865**	0.617	0.835	0.703
	FID ↓	**58.82**	N/A	110.64	66.52	96.72	<u>62.15</u>	87.64
	LPIPS ↓	**0.259**	N/A	0.428	0.353	0.563	<u>0.294</u>	0.520

Table 5.1: FFHQ 256×256 data set: image reconstruction (PSNR, SSIM) obtained by the compared methods. **Bold**: best, <u>underline</u>: second. PnP-SGS is evaluated on $\mathbf{x}_{\text{MMSE}}$.

- *inpainting*: 80% of the total pixels have been randomly masked accross all RGB channels,
- *superresolution*: the operator $\mathbf{S}$ corresponds to a downsampling factor $d = 4$ in both directions and the operator $\mathbf{B}$ stands for a Gaussian blur with a kernel size of 9×9 and a standard deviation of 1.5.

5.4.2 Compared methods & figures-of-merit

The proposed method has been compared to state-of-the-art methods related to the rationales motivating PnP-SGS:

- SPA [137]: split-and-augmented Gibbs sampler is an extension of SGS; in our experiments, it is used with a usual Tikhonov regularizer for deblurring and superresolution and with total-variation (TV) for inpainting;
- TV-ADMM [156]: ADMM using the isotropic regularization. The regularization parameter λ and some penalty parameter ρ linked to the

		PnP-SGS	SPA [137]	TV-ADMM [156]	PnP-ADMM [157]	Score-SDE [146]	DDRM [158]	MCG [159]
Inpainting	PSNR ↑	**25.22**	23.14	20.96	8.39	18.62	14.29	19.03
	SSIM ↑	**0.870**	0.802	0.676	0.300	0.517	0.403	0.546
	FID ↓	**34.28**	41.33	189.3	114.7	127.1	114.9	39.19
	LPIPS ↓	**0.297**	0.323	0.510	0.677	0.659	0.665	0.414
Deblurring (Gaussian)	PSNR ↑	21.76	21.08	19.99	**21.81**	15.97	22.73	16.32
	SSIM ↑	0.701	0.577	0.634	0.669	0.436	**0.705**	0.441
	FID ↓	64.12	98.78	155.7	100.6	120.3	**63.02**	95.04
	LPIPS ↓	**0.399**	0.537	0.588	0.519	0.667	0.427	0.550
Deblurring (motion)	PSNR ↑	21.47	20.49	20.79	**21.98**	7.21	N/A	5.89
	SSIM ↑	0.695	0.681	0.677	**0.702**	0.120	N/A	0.037
	FID ↓	**47.57**	91.51	138.8	89.76	98.25	N/A	186.9
	LPIPS ↓	**0.372**	0.538	0.525	0.483	0.591	N/A	0.758
Superres. (×4)	PSNR ↑	24.33	N/A	22.17	23.75	12.25	**24.96**	13.39
	SSIM ↑	0.772	N/A	0.679	0.761	0.256	**0.790**	0.227
	FID ↓	**59.09**	N/A	130.9	97.27	170.7	59.57	144.5
	LPIPS ↓	**0.418**	N/A	0.523	0.433	0.701	0.339	0.637

Table 5.2: Imagenet 256×256 data set: image reconstruction (PSNR, SSIM) obtained by the compared methods. **Bold**: best, underline: second. PnP-SGS is evaluated on $\mathbf{x}_{\text{MMSE}}$.

splitting have been adjusted by grid search to reach the best performance. Final values are $(\lambda, \rho) = (2.7 \times 10^{-2}, 1.4 \times 10^{1})$ for deblurring, $(\lambda, \rho) = (2.7 \times 10^{-2}, 1.0 \times 10^{-2})$ for inpainting and $(\lambda, \rho_1, \rho_2) = (2.7 \times 10^{-2}, 1.0 \times 10^{-2})$ for superresolution which requires a double splitting.

- PnP-ADMM [157]: ADMM with a PnP regularization chosen as DnCNN [122]; this can be interpreted as the deterministic counterpart of PnP-SGS. The implementation is from the `SCICO`[2]. library. The parameters are set to $\rho = 0.2$ (ADMM penalty parameter) and `maxiter` $= 12$. Proximal mappings use the pretrained DnCNN denoiser [122].

- DDRM [158]: the denoising diffusion restoration model is implemented using the same DDPM as PnP-SGS. All experiments have been performed with the default setting $\eta_B = 1.0$ and $\eta = 0.85$. For the Gaussian deblurring task, the forward model was implemented by separable 1D convolutions for efficient SVD.

- MCG [159]: manifold constrained gradients. The variance scheduling function $\alpha(\cdot)$ has been chosen as the one used by PnP-SGS. At each step, complementary data consistency steps are applied as Euclidean projections onto the measurement set $\mathcal{C} = \{\mathbf{x}_i \mid \mathbf{H}\mathbf{x}_i = \mathbf{y}_i, \mathbf{y}_i \sim p(\mathbf{y}_i \mid \mathbf{x}_0)\}$

- Score-SDE [146]: implemented using the same DDPM as the one used by PnP-SGS. Score-SDE solves the inverse problems by iteratively applying a denoising step followed by data consistency projections onto the measurement set $\mathcal{C}$, as in MCG.

Uncertainty quantification

Note that PnP-ADMM, TV-ADMM, DDRM and Score-SDE yield point estimates only. In contrast, PnP-SGS provides a comprehensive description of the targeted posterior distribution so that it permits to quantify uncertainties. It yields variances and credibility intervals and multiple statistics of the posterior for a variety of estimators such as MMSE and MAP.

Fig. 5.2 and 5.3 first qualitatively evaluated through visual inspection. Quantitative comparisons are conducted based on four widely-used metrics. The first two criteria are standard image reconstruction metrics, namely peak signal-to-noise-ratio (PSNR) and structural similarity index (SSIM). The two other criteria are perceptual metrics: Fréchet Inception Distance (FID), and

[2] `https://scico.readthedocs.io`

Learned Perceptual Image Patch Similarity (LPIPS) distance. Results are averaged over 1000 test images.

5.4.3 Technical details of PnP-SGS

For the experiments on the FFHQ data set, the pre-trained DDPM has been taken from [155] also available online[3] and the coupling parameter has been manually set to $\rho = 0.7$. For the Imagenet data set, we have used the pre-trained DDPM of [142] and available online[4] and the coupling parameter has been choosen as $\rho = 1.625$ after a gridsearch procedure.

Burn-in and early stopping

For all experiments, the number of iterations of the PnP-SGS has been fixed as $N_{\text{MC}} = 100$ including $N_{\text{bi}} = 20$ burn-in iterations. During the burn-in period, instead of applying the kernel (5.10) for $t = \widehat{t^*}, \ldots, 1$, we suspend the process in the middle of the diffusion, i.e., $t = \widehat{t^*}, \ldots, \frac{\widehat{t^*}}{2}$. This early-stopping trick not only provides empirically better results but also allows the computational burden to be lightened by reducing the number of DNN evaluations.

Estimating the noise level

The stochastic denoising task corresponding to the conditional distribution (5.7) requires an estimation of the level $\hat{\sigma} = \Phi(\mathbf{x}^{(n)})$ of a Gaussian noise assumed to affect the current state $\mathbf{x}^{(n)}$ at each iteration of PnP-SGS (see Algo. 6, line 5). The problem of estimating the level of the noise corrupting natural images has motivated plenty of research works, see [149, 150, 151]. In our implementations, this estimation has been carried out following the strategy proposed in [149]. This robust wavelet-based estimator is already implemented in the library `scikit-image` (aka `skimage`) as the function `estimate_sigma()`. When handling RGB natural images, this function has been used with the parameter `average_sigmas=True` to average the noise level estimates over the three channels.

Inverting the variance function $\alpha(\cdot)$

Given the current estimate of the noise level $\hat{\sigma}$, sampling according to (5.7) is achieved by performing the backward diffusion with kernel (5.10) from a

[3] `https://github.com/jychoi118/ilvr_adm`

[4] `https://github.com/openai/guided-diffusion`

time instant $\widehat{t^*}$ such that $\hat{\sigma}^2 = \alpha(\widehat{t^*})$ where $\alpha(t)$ is defined by (5.12). This diffusion scheduling function is controlled by the function $b(\cdot)$ that adjusts the variance of the forward transition kernel (5.9) from $t-1$ to t. Various choices of $b(\cdot)$ exist in the literature. We have tested two particular choices. For experiments with FFHQ, we chose a linearly increasing function

$$b(t) = b(0) + rt \tag{5.24}$$

where $b(0) = 10^{-4}$ and the slope r has been adjusted such that $b(T) = 2.0 \times 10^{-2}$ [47]. For experiments with ImageNet, we adopted the cosine-based variance schedule [49]

$$\alpha(t) = 1 - \frac{\gamma(t)}{\gamma(0)} \tag{5.25}$$

with $\gamma(t) = \cos\left(\frac{\pi}{2}\frac{t/T+s}{1+s}\right)^2$. In both cases, an explicit inverse function $\alpha^{-1}(\cdot)$ can be derived, which yields $\widehat{t^*} = \alpha^{-1}(\hat{\sigma}^2)$. For more complex scheduling functions, an alternative is to use a tabbing strategy, which saves computation cost as well. Given a pre-computed list of $T+1$ values $\boldsymbol{\alpha} = \{\alpha(0), \ldots, \alpha(T)\}$, the diffusion start time is set to

$$\widehat{t^*} = \underset{t \in \{0,\ldots,T\}}{\operatorname{argmin}} \, |\alpha(t) - \hat{\sigma}^2|.$$

In our experiments, the scheduling functions have been sampled on $T = 1000$ regularly spaced time instants.

5.4.4 Experimental results

Illustration of performances

Tables 5.1 and 5.2 report the quantitative results in terms of image reconstruction and perceptual metrics for the two data sets FFHQ and Imagenet, respectively. The proposed method outperforms all the other compared methods by significant margins for the SNR and for the visual perception metrics. Particularly, DDRM and Score-SDE rely on a DDPM where the pre-trained generative model is exactly the same as the one implemented in PnP-SGS. Results appearing as N/A correspond to tasks which are either not relevant for the model or not implemented by the original authors. It is worth noting that PnP-ADMM performs very poorly for inpainting tasks on both datasets. The implementation shows that PnP-ADMM has diverged for some of the tested images which explains the reported numbers, similar behaviors have been observed in the literature [160].

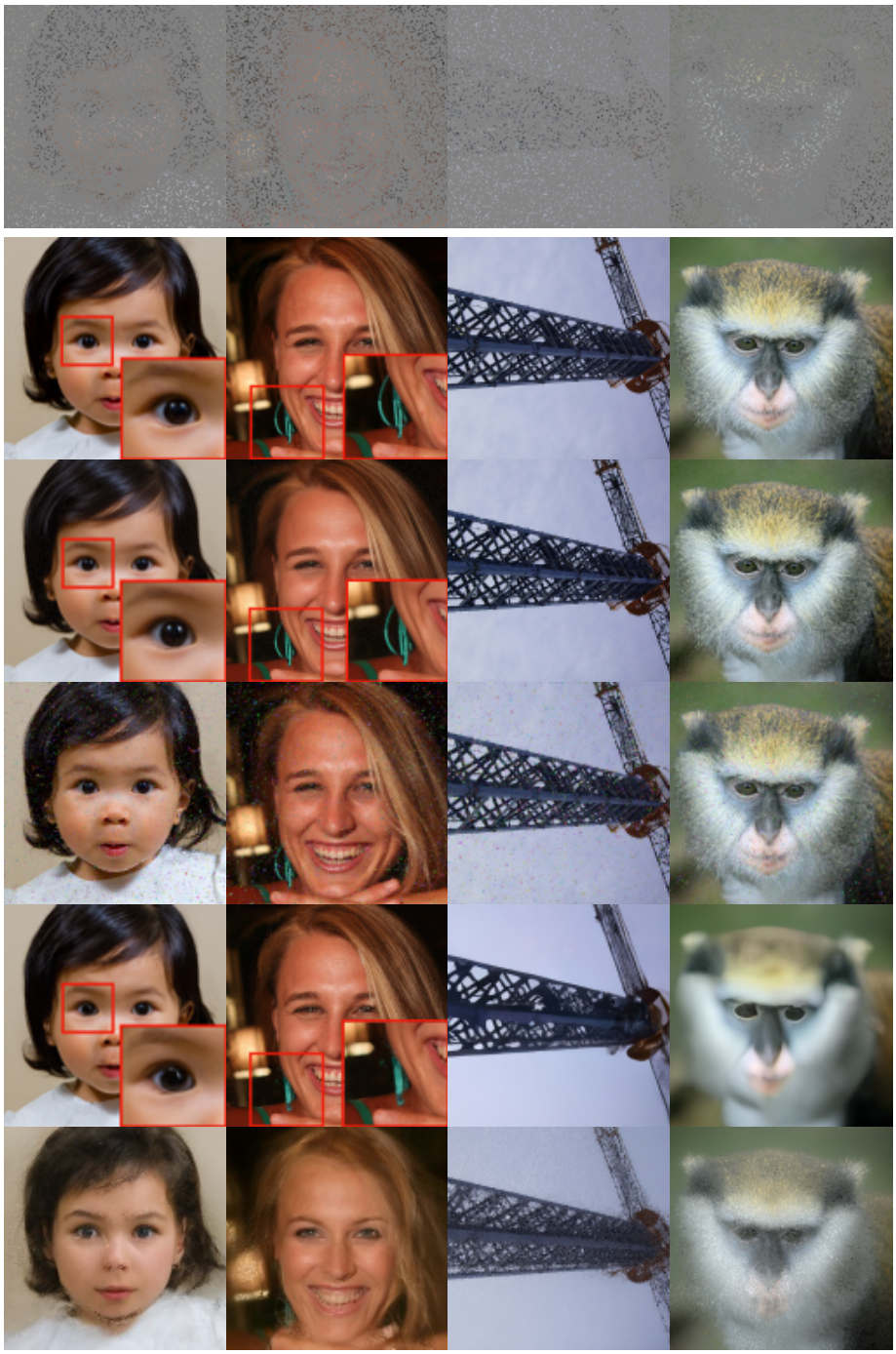


Figure 5.2: Inpainting task on the FFHQ (first two columns) and Imagenet data sets (last two columns), from up to bottom: measurement, true image, PnP-SGS ($\mathbf{x}_{\text{MMSE}}$), SPA [137], MCG [159], DDRM [158].

PnP-SGS	SPA	PnP-ADMM	Score-SDE	DDRM	MCG
13.81	218.90	3.63	36.71	2.03	80.10

Table 5.3: Inpainting: computational times (in s.) of the compared methods.

Fig. 5.2 permits to assess the performances by visual inspection when inpainting 4 test images taken from the FFHQ and Imagenet data sets. In particular, PnP-SGS is compared to state-of-the-art methods which are known to be robust to measurement noise. PnP-SGS is able to provide high-quality reconstructions that are crisp and realistic. In particular it is able to recover more granular details.

Uncertainty quantification

As already stated, the proposed PnP-SGS generate samples asymptotically distributed according to the posterior distribution. These samples can be used to approximate various Bayesian estimators but also to derive credibility intervals. Fig. 5.3 illustrates this advantage by depicting various restored images (in term of MMSE estimates) as well as 90% credibility intervals for different tasks. This added value cannot be provided by optimization-based methods, e.g., TV-ADMM and PnP-ADMM, which provide point estimates only. Besides, stochastic samplers such as DDRM, MCG and Score-SDE are not able to provide this information either. Indeed, they do not generate multiple samples drawn from a stationary posterior. Several runs of these methods produce outputs that may be individually relevant but that are not consistent between them in their details, in particular because they originate from different noise realizations. This is also why averaging multiple outputs of these methods does not yield reliable MMSE estimators but rather tends to recover blurred images, as illustrated in Fig. 5.4 (6th right panel for MCG).

When targeting (5.4), PnP-SGS generates two sets of samples $\left\{\mathbf{x}^{(n)}\right\}_n$ and $\left\{\mathbf{z}^{(n)}\right\}_n$ that are marginally distributed according to the marginals of $\pi_\rho(\mathbf{x}, \mathbf{z})$. It follows a splitting strategy where the variables $\mathbf{x}$ and $\mathbf{z}$ are coupled thanks to a quadratic kernel that is controlled by the parameter ρ. Thus the posterior means $\hat{\mathbf{x}}_{\text{MMSE}} = \text{E}[\mathbf{x}|\mathbf{y}]$ and $\hat{\mathbf{z}}_{\text{MMSE}} = \text{E}[\mathbf{z}|\mathbf{y}]$ should be very similar up to some variations adjusted by the coupling parameter ρ. Fig. 5.4 depicts the two estimates as well as pixel-wise 90% credibility intervals. As expected, slight differences are observed. In particular, the point estimate $\hat{\mathbf{x}}_{\text{MMSE}}$ seems to be characterized by sharper details (better viewed by zooming on screen). Recall that this estimate is closer to the observation, while $\hat{\mathbf{z}}_{\text{MMSE}}$ is closer to the prior, therefore smoother.



Figure 5.3: From left to right: measurement, true image, MMSE estimate, pixel-wise 90% credibility intervals. From top to bottom: Gaussian blur, superresolution and two motion blurs. The first three and the last images are from the FFHQ and Imagenet data sets, respectively.



Figure 5.4: Image inpainting, from left to right: measurement, true image, MMSE estimate $\hat{\mathbf{x}}_{\text{MMSE}}$, MMSE estimate $\hat{\mathbf{z}}_{\text{MMSE}}$, pixel-wise 90% credibility intervals, averaged samples generated by MCG [159]

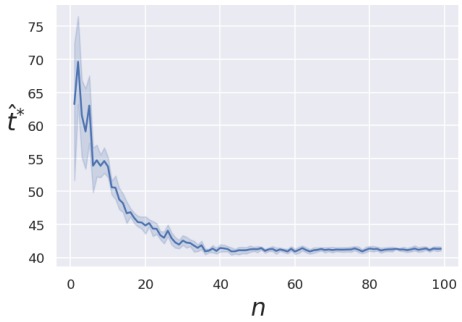


Figure 5.5: Inpainting: evolution of $\widehat{t^*}$ along the PnP-SGS iterations ($T = 1000$). Results have been averaged over 100 runs conducted on the same image. Shaded areas stand for the corresponding standard deviation.

Execution time

Table 5.3 reports the execution times for the task of inpainting of the various methods implemented on a single GTX 2080Ti GPU. Noticeably, the computational time of PnP-SGS is similar to its competitors. In particular, this stochastic MCMC method (13.81s) is more than twice faster than Score-SDE (36.71s). It remains within a factor less than 4 with respect to PnP-ADMM (3.63s), its deterministic counterpart. The price to pay to get quantified uncertainties sounds very reasonable. It is worth noting that using a DDPM-based PnP with SGS significantly reduces the number of iterations required by the sampler to reach the steady regime, which explains the reduced computational cost with respect to SPA.

Convergence of $t^\star$

In Section 5.4, the estimated number $\widehat{t^*}$ of denoising steps is automatically adjusted by the procedure described in Section 5.2.4. At the first iteration of the PnP-SGS, $\widehat{t^*}$ is usually a fraction of T. Along the iterations of the PnP-SGS, this number reduces and then stabilizes around a small fraction of T, as illustrated in Fig. 5.5 where the initial value of $\widehat{t^*}$ is around $0.07T$ for the inpainting task.

5.5 Conclusion

This work proposes the plug-and-play split Gibbs sampler (PnP-SGS) as a stochastic counterpart of the well-known PnP-ADMM. Thanks to the SGS divide-to-conquer strategy, the PnP-SGS algorithm permits to target a posterior distribution that involves an implicit PnP prior where the regularization

is ensured by some efficient stochastic denoiser. The proposed methodology can make use of any well-suited PnP prior, depending on the final application. For instance, it can be based on a denoising diffusion probabilistic model (DDPM), as proposed here, since it appears that a DDPM can be turned into a Bayesian sampler of a denoising problem. With the same versatility as PnP-ADMM, sampling from the posterior distribution noticeably permits to build credibility intervals on top of point estimates. Extensive numerical experiments show that the proposed approach competes favourably with existing state-of-the-art models on typical imaging problems, namely deblurring, inpainting and superresolution. The quantitative performances are at least comparable when not better, while the computational times remain very moderate as well. PnP-SGS appears as a scalable MCMC sampling method that can benefit from the most recent progress in machine (deep) learning at the price of a reasonable computational cost.

Chapter 6

Conclusion

Closing discussion

Expressive probabilistic models are often employed to generate realistic data exhibiting intricate patterns across various domains, including images, audio, and molecular arrangements. These models serve the purpose of characterizing and approximating data distributions in high dimensional spaces. However, even with the potential of deep neural networks, building robust probabilistic models remains a challenging endeavor. While providing high-quality samples is itself a difficult task, numerous scientific applications also requires the capability to undertake density estimation. This tractability requirement mandates the adoption of restricted model architectures, a decision that quickly becomes impractical in high-dimensional disconnected space. This manuscript addresses these challenges through some improvements of either the training protocol or the sampling methodology.

- Chapter 2 proposes NF-SW by hybridizing the vanilla NF loss function. The approach complements the conventional maximum likelihood training with a sliced Wasserstein statistical divergence. Experimental results show that augmenting training strikes a balance between the generation of realistic samples and improved density estimation capacity. The resulting model demonstrates better out of distribution (OOD) detection capabilities compared to classical training of flow-based models.

- Chapter 3 proposes SWOT-Flow, a training method which enables discrete normalizing flow architectures to tackle optimal transport problem between two arbitrary empirical distributions. This approach also allows easy sampling from Wasserstein barycenters distributions.

- Chapter 4 focuses on the sampling of generative models in the case of complicated pathological multimodal hight dimensional distributions with disjoint supports. We propose a new MCMC algorithm to sample from the NF learned distribution in the latent domain before transporting it back to the target domain. Notably, it can be straightforwardly used with any pre-trained NF network, regardless of the architecture. Interestingly the resulting Langevin diffusion is defined on the Riemann manifold whose geometry is driven by the Jacobian of the NF in the latent domain.

- Chapter 5 focuses on the resolution of inverse problems in the context of degraded image reconstruction. We propose PnP-SGS a new sampling algorithm that leverage deep generative model as prior in a Gibbs sampling framework. The method showcases state-of-the-art results in Bayesian estimation and credibility intervals while maintaining rapid computation times. It introduces a novel avenue of stochastic plug-and-play methodology, harnessing the potential of deep generative models in Bayesian inference tasks.

Perspectives for future work

The generality of deep generative models associated with MCMC sampling algorithm, along with their relations with optimization, allow to consider various prospective works. They can be divided into methodological generalizations, theoretical contributions and new applications.

Methodological generalizations

- **Deep generative models as likelihood surrogate -** In the context of inverse problem solving, the misalignment or lack of precision in the probabilistic model characterizing the dataset and its relationship with the parameters of interest poses challenges. Typically taking the form of a likelihood function, this description plays a pivotal role in fundamental statistical methods, including maximum likelihood and Bayesian estimations. For many inverse problems deriving a likelihood function is impossible. To address these challenges, Approximate Bayesian Computation (ABC) [161] presents a promising alternative. ABC aimed at estimating the posterior distribution without directly evaluating the likelihood function but with a direct model simulator. Notably, ABC methods offer robust and well-documented properties, facilitating the accurate quantification of the resulting approximations.

Deep architectures can be employed to implicitly represent the likelihood function through extensive training datasets within ABC or splitting-based Monte Carlo methods [137]. Given some pairs of data and observation $\{x, y\}$, one could train a conditional generative model to approximate the likelihood. Once trained the generative model could be used to sample from the posterior distribution. One could also force the latent distribution of a generative model to be the one of the prior distribution and perform semi-supervised training to map from the data to the observation.

- **Integrate reverse ODE to estimate posterior distribution -** A significant portion of the existing literature that applies diffusion-based models to address inverse problems relies on approximations [141, 158, 159]. Specifically, these approaches involve denoising from the learned prior distribution during each diffusion step, followed by a projection onto the observational space described by the likelihood function. Conversely, when the diffusion-based model described in paragraph 1.4.2 is cast as an Ordinary Differential Equation (ODE) (see Appendix B), the sampling process thus becomes deterministic and rely on ODE solver to map the latent space to the data space. The advances of efficient GPU implementations of ODE solvers have unlocked the potential to compute the true log posterior of a data point. For a conditional score model $\mathbf{s}_\theta(\mathbf{x}, y, t) \approx \nabla_{\mathbf{x}} \log p_t(\mathbf{x} \mid y)$, the learned probability flow ODE is given by

$$\frac{d\mathbf{x}_t}{dt} = f\left(\mathbf{x}_t, y, t\right) - \frac{1}{2} g(t)^2 \mathbf{s}_\theta\left(\mathbf{x}_t, y, t\right) =: \tilde{\mathbf{f}}_\theta\left(\mathbf{x}_t, y, t\right). \tag{6.1}$$

By the continuous-time change-of-variables formula [162], the log probability of an image $\mathbf{x} = \mathbf{x}_0$ under the p_0 distribution is given by the log probability of $\mathbf{x}_T$ under the p_T Gaussian, plus a normalization factor accounting for the change in probability density from $\mathbf{x}_0$ to $\mathbf{x}_T$. We compute the log-probability under the learned ODE (Eq. 6.1) by solving an initial-value problem:

$$\log p_0\left(\mathbf{x}_0 \mid y\right) = \log p_T\left(\mathbf{x}_T \mid y\right) + \int_0^T \nabla \cdot \tilde{\mathbf{f}}_\theta\left(\mathbf{x}_t, y, t\right) \mathrm{d}t, \tag{6.2}$$

where $\mathbf{x}_0 = \mathbf{x}$. The divergence $\nabla \cdot \tilde{\mathbf{f}}_\theta\left(\mathbf{x}_t, y, t\right)$ quantifies the instantaneous change in log-probability of $\mathbf{x}_t$ caused by applying $\tilde{\mathbf{f}}_\theta\left(\mathbf{x}_t, y, t\right)$ in either time direction. It can be estimated with Hutchinson-Skilling estimation of the trace of $\frac{\partial}{\partial \mathbf{x}_t} \tilde{\mathbf{f}}_\theta\left(\mathbf{x}_t, t\right)$ [163]. Consequently, this would allows to perform density estimation over the posterior density function and thus uncertainty quantification without the need for sampling.

Theoretical contributions

- **Theoretical convergence of Stochastic PnP methods -** While we have presented compelling experimental results for our PnP-SGS algorithm, methods involving classical sampling approaches and deep generative modelling lack a well-defined theoretical framework. Following a similar approach as that taken for PnP-ULA [164] or [165], it would be valuable to delve into the theoretical properties and convergence of PnP-SGS. Moreover, as contemporary models increasingly gravitate toward the plug-and-play methodology, it becomes even more crucial to investigate the convergence guarantees of Monte Carlo sampling with pre-trained deep priors. In particular, it would sound important to derive some bounds with respect to uncertainty quantification performance.

- **Adaptive PnP-SGS sampling -** So far, the presentation and the application of the PnP-SGS framework has considered a fixed tolerance parameter ρ. Even if the implemented method proposed to estimate the step $t^\star$ corresponding to the level of noise of the image, the sampler and the target augmented distribution are governed by a fixed ρ prescribed by the dissimilarity term. Nevertheless, it is still unknown at the moment whether an adaptive PnP-SGS associated to a sequence $\rho_{k\in\mathcal{N}}$ will perform better than its standard version with a fixed tolerance parameter ρ. Hence, an interesting methodological extension of the proposed work is to derive an adaptive PnP-SGS sampling strategy which will permit to bypass the empirical tuning of ρ. Note that such a method has been proposed for an optimization counterpart PnP method in [129].

Applications to other challenging problems

- **Non linear inverse problems -** The proposed PnP-SGS methodology has not been applied so far on Bayesian inference problems characterized by non-log-concave and potentially multimodal posteriors. In such scenarios, the efficiency of PnP-SGS remains unverified and may be compromised due to the presence of multiple modes in the target distribution and the sequential nature of the Gibbs sampling procedure. An example of such a problem is blind source separation and its constrained formulations, such as nonnegative matrix factorization and linear unmixing, which find applications in various domains like astrophysics and hyperspectral imaging [166, 167]. These problems entail the simultaneous estimation of the mixing matrix and the sources,

resulting in a high-dimensional posterior distribution that lacks log-concavity. Therefore, a potential avenue for future research involves investigating whether the proposed MCMC sampling algorithm can effectively address such challenges and scale with the dimension of the state space. More precisely, given an explicit likelihood corresponding to a non linear direct model, one could use a proximal MCMC [168] to sample from the first conditional distribution of the Gibbs kernel.

- **Posterior guided latent diffusion normalizing flow -** The NF-SAILS sampling method proposes to sample from any pretrained NF using a Langevin algorithm on the latent space. More specifically it leverages the information of the Jacobian to avoid out of distribution area in the case of disconnected support of the data. Now one can use this pretrained NF as a prior in the context of Bayesian inference. Using the Bayes formula, $\nabla_x \log\ p(x \mid y) \propto \nabla_x \log\ p(y \mid x) \cdot \nabla_x \log\ p(x)$ and one can deduce the posterior latent diffusion to sample from the posterior using the Jacobian of the transformation. Note that a similar idea has been recently introduced in [169] but in the data space.

Progress in deep generative modeling not only benefits immediate applications of data generation, like lage-scale datasets and generating high quality artwork, but also deepens our comprehension of the structure underlying the data. Leveraging this powerful capacity using well understand statistical method may ignite a transformative process that reshapes the very fabric of scientific discovery.

Bibliography

[1] M. Bertero, P. Boccacci, and C. De Mol, *Introduction to inverse problems in imaging*. CRC press, 2021.

[2] I. Craig and J. Brown, *Inverse problems in astronomy*. Adam Hilger Ltd, 1986.

[3] C. Ruíz García, "Development of tomographic systems for mining, mineral exploration and environmental purposes," *Transactions-Institution of Mining and Metallurgy. Section B. Applied Earth Science*, vol. 108, pp. 105–118, 1999.

[4] M. T. McCann, K. H. Jin, and M. Unser, "Convolutional neural networks for inverse problems in imaging: A review," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 85–95, 2017.

[5] S. Subramaniam, A. Bartesaghi, J. Liu, A. E. Bennett, and R. Sougrat, "Electron tomography of viruses," *Current opinion in structural biology*, vol. 17, no. 5, pp. 596–602, 2007.

[6] X. X. Zhu, D. Tuia, L. Mou, G.-S. Xia, L. Zhang, F. Xu, and F. Fraundorfer, "Deep learning in remote sensing: A comprehensive review and list of resources," *IEEE Geoscience and Remote Sensing Magazine*, vol. 5, no. 4, pp. 8–36, 2017.

[7] S. Cheng, C. Quilodrán-Casas, S. Ouala, A. Farchi, C. Liu, P. Tandeo, R. Fablet, D. Lucor, B. Iooss, J. Brajard, *et al.*, "Machine learning with data assimilation and uncertainty quantification for dynamical systems: a review," *Journal of Automatica Sinica*, vol. 10, no. 6, pp. 1361–1387, 2023.

[8] C. Guilloteau, T. Oberlin, O. Berné, É. Habart, and N. Dobigeon, "Simulated JWST data sets for multispectral and hyperspectral image fusion," *The Astronomical Journal*, vol. 160, no. 1, p. 28, 2020.

[9] K. Akiyama, A. Alberdi, W. Alef, K. Asada, R. Azulay, A.-K. Baczko, D. Ball, M. Baloković, J. Barrett, D. Bintley, *et al.*, "First M87 event horizon telescope results. IV. imaging the central supermassive black hole," *The Astrophysical Journal Letters*, vol. 875, no. 1, p. L4, 2019.

[10] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin, *Bayesian data analysis*. Chapman and Hall/CRC, 1995.

[11] C. P. Robert, G. Casella, and G. Casella, *Monte Carlo statistical methods*, vol. 2. Springer, 1999.

[12] W. R. Gilks, S. Richardson, and D. Spiegelhalter, *Markov chain Monte Carlo in practice*. CRC press, 1995.

[13] R. M. Neal, "Probabilistic inference using Markov chain Monte Carlo methods," 1993.

[14] W. K. Hastings, "Monte Carlo sampling methods using Markov chains and their applications," *Biometrika*, vol. 57, no. 1, pp. 97–109, 1970.

[15] U. Grenander and M. I. Miller, "Representations of knowledge in complex systems," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 56, no. 4, pp. 549–581, 1994.

[16] R. Khasminskii, *Stochastic stability of differential equations*, vol. 66. Springer Science & Business Media, 2011.

[17] E. Platen and N. Bruti-Liberati, *Numerical solution of stochastic differential equations with jumps in finance*, vol. 64. Springer Science & Business Media, 2010.

[18] A. Abdulle, G. Vilmart, and K. C. Zygalakis, "High order numerical approximation of the invariant measure of ergodic sdes," *Journal on Numerical Analysis*, vol. 52, no. 4, pp. 1600–1622, 2014.

[19] J. C. Mattingly, A. M. Stuart, and M. V. Tretyakov, "Convergence of numerical time-averaging and stationary measures via poisson equations," *Journal on Numerical Analysis*, vol. 48, no. 2, pp. 552–577, 2010.

[20] D. Talay and L. Tubaro, "Expansion of the global error for numerical schemes solving stochastic differential equations," *Stochastic analysis and applications*, vol. 8, no. 4, pp. 483–509, 1990.

[21] S. Rifai, Y. N. Dauphin, P. Vincent, Y. Bengio, and X. Muller, "The manifold tangent classifier," 2011.

[22] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[23] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural computation*, vol. 14, no. 8, pp. 1771–1800, 2002.

[24] A. Hyvärinen, J. Hurri, P. O. Hoyer, A. Hyvärinen, J. Hurri, and P. O. Hoyer, "Estimation of non-normalized statistical models," *Natural Image Statistics: A Probabilistic Approach to Early Computational Vision*, pp. 419–426, 2009.

[25] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*, vol. 4. Springer, 2006.

[26] M. Gutmann and A. Hyvärinen, "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010.

[27] J. Sohl-Dickstein, P. Battaglino, and M. R. DeWeese, "Minimum probability flow learning," in *International Conference on Machine Learning (ICML)*, 2011.

[28] A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf, and A. Smola, "A kernel method for the two-sample-problem," in *Advances in neural information processing systems (NeurIPS)*, 2006.

[29] Y. Li, K. Swersky, and R. Zemel, "Generative moment matching networks," in *International conference on machine learning (ICML)*, 2015.

[30] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems (NeurIPS)*, 2014.

[31] L. Theis, A. v. d. Oord, and M. Bethge, "A note on the evaluation of generative models," in *International Conference on Learning Representations (ICLR)*, 2015.

[32] G. K. Dziugaite, D. M. Roy, and Z. Ghahramani, "Training generative neural networks via maximum mean discrepancy optimization," in *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2015.

[33] J. Hays and A. A. Efros, "Scene completion using millions of photographs," *ACM Transactions on Graphics (ToG)*, vol. 26, no. 3, pp. 4–es, 2007.

[34] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *International Conference on Machine Learning (ICML)*, 2010.

[35] D. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *International Conference on Machine Learning (ICML)*, PMLR, 2015.

[36] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real NVP," in *International Conference on Learning Representations (ICLR)*, 2017.

[37] L. Dinh, D. Krueger, and Y. Bengio, "NICE: non-linear independent components estimation," in *International Conference on Learning Representations (ICLR)*, 2015.

[38] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real NVP," in *International Conference on Learning Representations (ICLR)*, 2017.

[39] D. P. Kingma and P. Dhariwal, "Glow: Generative flow with invertible 1x1 convolutions," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[40] J. Ho, X. Chen, A. Srinivas, Y. Duan, and P. Abbeel, "Flow++: Improving flow-based generative models with variational dequantization and architecture design," in *International Conference on Machine Learning (ICML)*, 2019.

[41] L. Ardizzone, J. Kruse, C. Rother, and U. Köthe, "Analyzing inverse problems with invertible neural networks," in *International Conference on Learning Representations (ICLR)*, 2019.

[42] L. Hasenclever, J. M. Tomczak, R. Van Den Berg, and M. Welling, "Variational inference with orthogonal normalizing flows," in *Workshop on Bayesian Deep Learning at NeurIPS*, 2017.

[43] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling, "Improved variational inference with inverse autoregressive flow," in *Advances in Neural Information Processing Systems (NeurIPS)*, Curran Associates, Inc., 2016.

[44] G. Papamakarios, T. Pavlakou, and I. Murray, "Masked autoregressive flow for density estimation," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[45] R. T. Chen, J. Behrmann, D. K. Duvenaud, and J.-H. Jacobsen, "Residual flows for invertible generative modeling," 2019.

[46] J. Behrmann, W. Grathwohl, R. T. Chen, D. Duvenaud, and J.-H. Jacobsen, "Invertible residual networks," in *International Conference on Machine Learning (ICML)*, 2019.

[47] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Advances on Neural Information Processing Systems (NeurIPS)*, 2020.

[48] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International Conference on Machine Learning (ICML)*, 2015.

[49] A. Q. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," in *International Conference on Machine Learning (ICML)*, 2021.

[50] J. C. Spall, "Stochastic optimization," *Handbook of computational statistics: Concepts and methods*, pp. 173–201, 2012.

[51] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, "Normalizing flows for probabilistic modeling and inference," *Journal of Machine Learning Research*, vol. 22, no. 57, pp. 1–64, 2021.

[52] P. Jaini, I. Kobyzev, Y. Yu, and M. Brubaker, "Tails of lipschitz triangular flows," in *International Conference on Machine Learning (ICML)*, 2020.

[53] E. Nalisnick, A. Matsukawa, Y. W. Teh, D. Gorur, and B. Lakshminarayanan, "Do deep generative models know what they don't know?," in *International Conference on Learning Representations (ICLR)*, 2019.

[54] P. Kirichenko, P. Izmailov, and A. G. Wilson, "Why normalizing flows fail to detect out-of-distribution data," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[55] J. Rabin, G. Peyré, J. Delon, and M. Bernot, "Wasserstein barycenter and its application to texture mixing," in *Scale Space and Variational Methods in Computer Vision (SSVM)*, pp. 435–446, Springer Berlin Heidelberg, 2012.

[56] E. Nalisnick, A. Matsukawa, Y. W. Teh, D. Gorur, and B. Lakshminarayanan, "Do deep generative models know what they don't know?," in *International Conference on Learning Representations (ICLR)*, 2019.

[57] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[58] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International Conference on Machine Learning (ICML)*, 2017.

[59] I. Deshpande, Z. Zhang, and A. G. Schwing, "Generative modeling using the sliced wasserstein distance," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[60] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2016.

[61] A. Grover, M. Dhar, and S. Ermon, "Flow-gan: Combining maximum likelihood and adversarial learning in generative models," in *AAAI Conference on Artificial Intelligence*, 2018.

[62] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 29, 2016.

[63] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[64] C. Villani *et al.*, *Optimal transport: old and new*, vol. 338. Springer, 2009.

[65] G. Peyré, M. Cuturi, *et al.*, "Computational optimal transport: With applications to data science," *Foundations and Trends® in Machine Learning*, vol. 11, no. 5-6, pp. 355–607, 2019.

[66] K. Nguyen and N. Ho, "Revisiting sliced wasserstein on images: From vectorization to convolution," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[67] G. Monge, "Mémoire sur la théorie des déblais et des remblais," *Histoire de l'Académie Royale des Sciences de Paris*, 1781.

[68] L. Paulin, N. Bonneel, D. Coeurjolly, J.-C. Iehl, A. Webanck, M. Desbrun, and V. Ostromoukhov, "Sliced optimal transport sampling," *Transactions on Graphics (TOG)*, vol. 39, no. 4, p. 99, 2020.

[69] E. de Bézenac, I. Ayed, and P. Gallinari, "CycleGAN through the lens of (dynamical) optimal transport," in *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECLM-PKDD)*, 2021.

[70] N. Courty, R. Flamary, and D. Tuia, "Domain adaptation with regularized optimal transport," in *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECLM-PKDD)*, 2014.

[71] I. Kobyzev, S. J. Prince, and M. A. Brubaker, "Normalizing flows: An introduction and review of current methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 11, pp. 3964–3979, 2020.

[72] M. Agueh and G. Carlier, "Barycenters in the Wasserstein space," *Journal on Mathematical Analysis (SIMA)*, vol. 43, no. 2, pp. 904–924, 2011.

[73] J. Louet, *Problèmes de transport optimal avec pénalisation en gradient*. PhD thesis, Université Paris-Sud, France, 2014.

[74] P. C. Álvarez-Esteban, E. Del Barrio, J. Cuesta-Albertos, and C. Matrán, "A fixed-point approach to barycenters in wasserstein space," *Journal of Mathematical Analysis and Applications*, vol. 441, no. 2, pp. 744–762, 2016.

[75] M. Cuturi and A. Doucet, "Fast computation of wasserstein barycenters," in *International Conference on Machine Learning (ICML)*, 2014.

[76] S. Claici, E. Chien, and J. M. Solomon, "Stochastic Wasserstein barycenters," in *International Conference on Machine Learning (ICML)*, 2018.

[77] A. Korotin, L. Li, J. Solomon, and E. Burnaev, "Continuous wasserstein-2 barycenter estimation without minimax optimization," in *International Conference on Learning Representations (ICLR)*, 2021.

[78] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the association for computational linguistics*, vol. 5, pp. 135–146, 2017.

[79] A. Conneau, G. Lample, M. Ranzato, L. Denoyer, and H. Jégou, "Word translation without parallel data.," in *International Conference on Learning Representations (ICLR)*, 2018.

[80] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *IEEE international conference on computer vision (ICCV)*, 2017.

[81] J. Behrmann, P. Vicol, K.-C. Wang, R. B. Grosse, and J.-H. Jacobsen, "On the invertibility of invertible neural networks," 2019.

[82] R. Cornish, A. Caterini, G. Deligiannidis, and A. Doucet, "Relaxing bijectivity constraints with continuously indexed normalising flows," in *International Conference on Machine Learning (ICML)*, PMLR, 2020.

[83] Y. Marzouk, T. Moselhy, M. Parno, and A. Spantini, "Sampling via measure transport: An introduction," *Handbook of uncertainty quantification*, vol. 1, p. 2, 2016.

[84] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[85] S. Rifai, G. Mesnil, P. Vincent, X. Muller, Y. Bengio, Y. Dauphin, and X. Glorot, "Higher order contractive auto-encoder," in *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, Springer, 2011.

[86] G. Arvanitidis, L. K. Hansen, and S. Hauberg, "Latent space oddity: on the curvature of deep generative models," in *International Conference on Learning Representations (ICLR)*, 2018.

[87] C.-W. Huang, L. Dinh, and A. Courville, "Augmented normalizing flows: Bridging the gap between generative flows and latent variable models," 2020.

[88] H. Wu, J. Köhler, and F. Noé, "Stochastic normalizing flows," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[89] D. Nielsen, P. Jaini, E. Hoogeboom, O. Winther, and M. Welling, "SurVAE flows: Surjections to bridge the gap between VAEs and flows," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[90] A. N. Gomez, M. Ren, R. Urtasun, and R. B. Grosse, "The reversible residual network: Backpropagation without storing activations," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[91] P. Izmailov, P. Kirichenko, M. Finzi, and A. G. Wilson, "Semi-supervised learning with normalizing flows," in *International Conference on Machine Learning (ICML)*, 2020.

[92] L. Ardizzone, R. Mackowiak, C. Rother, and U. Köthe, "Training normalizing flows with the information bottleneck for competitive generative classification," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[93] P. Hagemann and S. Neumayer, "Stabilizing invertible neural networks using mixture models," *Inverse Problems*, 2021.

[94] G. G. Pires and M. A. Figueiredo, "Variational mixture of normalizing flows," in *European Symposium on Artificial Neural Networks (ESANN)*, 2020.

[95] V. Stimper, B. Schölkopf, and J. M. Hernández-Lobato, "Resampling base distributions of normalizing flows," 2022.

[96] M. Hoffman, P. Sountsov, J. V. Dillon, I. Langmore, D. Tran, and S. Vasudevan, "Neutra-lizing bad geometry in Hamiltonian Monte Carlo using neural transport," in *Symposium on Advances in Approximate Bayesian Inference, Co-located with ICML*, 2019.

[97] F. Noé, S. Olsson, J. Köhler, and H. Wu, "Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning," *Science*, vol. 365, no. 6457, p. eaaw1147, 2019.

[98] T. Müller, B. McWilliams, F. Rousselle, M. Gross, and J. Novák, "Neural importance sampling," *Transactions on Graphics (ToG)*, vol. 38, no. 5, pp. 1–19, 2019.

[99] M. Gabrié, G. M. Rotskoff, and E. Vanden-Eijnden, "Adaptive Monte Carlo augmented with normalizing flows," *Proceedings of the National Academy of Sciences*, vol. 119, no. 10, p. e2109420119, 2022.

[100] S. Samsonov, E. Lagutin, M. Gabrié, A. Durmus, A. Naumov, and E. Moulines, "Local-global MCMC kernels: the best of both worlds," in *Advances on Neural Information Processing Systems (NeurIPS)*, 2022.

[101] V. Runde, K. Ribet, and S. Axler, *A taste of topology*. Springer, 2005.

[102] A. Kumar, P. Sattigeri, and T. Fletcher, "Semi-supervised learning with GANs: Manifold invariance with improved inference," in *Advances on Neural Information Processing Systems (NeurIPS)*, 2017.

[103] E. Milman and J. Neeman, "The gaussian double-bubble and multi-bubble conjectures," *Annals of Mathematics*, vol. 195, no. 1, pp. 89–206, 2022.

[104] T. Issenhuth, U. Tanielian, J. Mary, and D. Picard, "Unveiling the latent space geometry of push-forward generative models," 2023.

[105] M. Girolami and B. Calderhead, "Riemann manifold langevin and Hamiltonian Monte Carlo methods," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 73, no. 2, pp. 123–214, 2011.

[106] T. Xifara, C. Sherlock, S. Livingstone, S. Byrne, and M. Girolami, "Langevin diffusions and the Metropolis-adjusted Langevin algorithm," *Statistics & Probability Letters*, vol. 91, pp. 14–19, 2014.

[107] B. Øksendal, *Stochastic differential equations*. Springer Science & Business Media, 2003.

[108] M. Vono, N. Dobigeon, and P. Chainais, "High-dimensional Gaussian sampling: a review and a unifying approach based on a stochastic proximal point algorithm," *SIAM Review*, vol. 64, no. 1, pp. 3–56, 2022.

[109] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10 (Canadian Institute for Advanced Research)," 2010.

[110] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[111] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao, "LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop," 2016.

[112] A. Justel, D. Peña, and R. Zamar, "A multivariate Kolmogorov-Smirnov test of goodness of fit," *Statistics & probability letters*, vol. 35, no. 3, pp. 251–259, 1997.

[113] A. Kraskov, H. Stögbauer, and P. Grassberger, "Estimating mutual information," *Physical review E*, vol. 69, no. 6, p. 066138, 2004.

[114] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local Nash equilibrium," in *Adv. in Neural Information Process. Systems (NIPS)*, 2017.

[115] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of Wasserstein GANs," in *Advances on Neural Information Processing Systems (NeurIPS)*, 2017.

[116] J. Kaipio and E. Somersalo, *Statistical and computational inverse problems*, vol. 160. Springer Science & Business Media, 2006.

[117] V. Chandrasekaran, B. Recht, P. A. Parrilo, and A. S. Willsky, "The convex geometry of linear inverse problems," *Foundations of Computational mathematics*, vol. 12, pp. 805–849, 2012.

[118] A. Repetti, M. Pereyra, and Y. Wiaux, "Scalable Bayesian uncertainty quantification in imaging inverse problems via convex optimization," *Journal on Imaging Sciences*, vol. 12, no. 1, pp. 87–118, 2019.

[119] A. Chambolle, "An algorithm for total variation minimization and applications," *Journal of Mathematical imaging and vision*, vol. 20, pp. 89–97, 2004.

[120] C. Louchet and L. Moisan, "Posterior expectation of the total variation model: properties and experiments," *Journal on Imaging Sciences*, vol. 6, no. 4, pp. 2640–2684, 2013.

[121] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *European Conference on Computer Vision (ECCV)*, 2014.

[122] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3142–3155, 2017.

[123] K. Zhang, W. Zuo, and L. Zhang, "FFDNet: Toward a fast and flexible solution for CNN-based image denoising," *IEEE Transactions on Image Processing*, vol. 27, no. 9, pp. 4608–4622, 2018.

[124] E. Schwartz, R. Giryes, and A. M. Bronstein, "Deepisp: Toward learning an end-to-end image processing pipeline," *IEEE Transactions on Image Processing*, vol. 28, no. 2, pp. 912–923, 2018.

[125] S. V. Venkatakrishnan, C. A. Bouman, and B. Wohlberg, "Plug-and-play priors for model based reconstruction," in *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2013.

[126] D. Geman and C. Yang, "Nonlinear image recovery with half-quadratic regularization," *IEEE Transactions on Image Processing*, vol. 4, no. 7, pp. 932–946, 1995.

[127] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[128] N. Parikh and S. Boyd, "Proximal algorithms," *Foundations and Trends in Optimization*, vol. 1, no. 3, p. 127–239, 2014.

[129] K. Zhang, Y. Li, W. Zuo, L. Zhang, L. Van Gool, and R. Timofte, "Plug-and-play image restoration with deep denoiser prior," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 10, pp. 6360–6376, 2022.

[130] R. Ahmad, C. A. Bouman, G. T. Buzzard, S. Chan, S. Liu, E. T. Reehorst, and P. Schniter, "Plug-and-play methods for magnetic resonance imaging: Using denoisers for image recovery," *IEEE Signal Processing Magazine*, vol. 37, no. 1, pp. 105–116, 2020.

[131] M. G. Kendall *et al.*, *The advanced theory of statistics.* No. 2nd Ed, Charles Griffin and Co., Ltd., London, 1946.

[132] X. Cai, M. Pereyra, and J. D. McEwen, "Uncertainty quantification for radio interferometric imaging: I. Proximal MCMC methods," *Monthly Notices of the Royal Astronomical Society*, vol. 480, no. 3, pp. 4154–4169, 2018.

[133] E. Gaume, L. Gaál, A. Viglione, J. Szolgay, S. Kohnová, and G. Blöschl, "Bayesian MCMC approach to regional flood frequency analyses involving extraordinary flood events at ungauged sites," *Journal of hydrology*, vol. 394, no. 1-2, pp. 101–117, 2010.

[134] M. Pereyra, P. Schniter, E. Chouzenoux, J.-C. Pesquet, J.-Y. Tourneret, A. O. Hero, and S. McLaughlin, "A survey of stochastic simulation and optimization methods in signal processing," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 2, pp. 224–241, 2015.

[135] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth, "Hybrid Monte Carlo," *Physics letters B*, vol. 195, no. 2, pp. 216–222, 1987.

[136] M. Pereyra, "Maximum-a-posteriori estimation with Bayesian confidence regions," *Journal on Imaging Sciences*, vol. 10, no. 1, pp. 285–302, 2017.

[137] M. Vono, N. Dobigeon, and P. Chainais, "Split-and-augmented Gibbs sampler – Application to large-scale inference problems," *IEEE Transactions on Signal Processing*, vol. 67, no. 6, pp. 1648–1661, 2019.

[138] M. Vono, N. Dobigeon, and P. Chainais, "Asymptotically exact data augmentation: Models, properties, and algorithms," *Journal of Computational and Graphical Statistics*, vol. 30, no. 2, pp. 335–348, 2020.

[139] D. Im Im, S. Ahn, R. Memisevic, and Y. Bengio, "Denoising criterion for variational auto-encoding framework," in *Association for the Advancement of Artificial Intelligence (AAAI)*, 2017.

[140] L. D. Tran, S. M. Nguyen, and M. Arai, "Gan-based noise model for denoising real images," in *Asian Conference on Computer Vision (ACCV)*, 2020.

[141] Y. Song, L. Shen, L. Xing, and S. Ermon, "Solving inverse problems in medical imaging with score-based generative models," in *International Conference on Learning Representations (ICLR)*, 2022.

[142] P. Dhariwal and A. Nichol, "Diffusion models beat GANs on image synthesis," in *Advances on Neural Information Processing Systems (NeurIPS)*, 2021.

[143] D. Kingma, T. Salimans, B. Poole, and J. Ho, "Variational diffusion models," in *Advances on Neural Information Processing Systems (NeurIPS)*, 2021.

[144] Y. Song and S. Ermon, "Generative modeling by estimating gradients of the data distribution," in *Advances on Neural Information Processing Systems (NeurIPS)*, 2019.

[145] Y. Song and S. Ermon, "Improved techniques for training score-based generative models," in *Advances on Neural Information Processing Systems (NeurIPS)*, 2020.

[146] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," in *International Conference on Learning Representations (ICLR)*, 2021.

[147] C.-W. Huang, J. H. Lim, and A. C. Courville, "A variational perspective on diffusion-based generative models and score matching," in *Advances on Neural Information Processing Systems (NeurIPS)*, 2021.

[148] H. Chung, B. Sim, and J. C. Ye, "Come-closer-diffuse-faster: Accelerating conditional diffusion models for inverse problems through stochastic contraction," in *Computer Vision and Pattern Recognition Conference (CVPR)*, 2022.

[149] D. L. Donoho and J. M. Johnstone, "Ideal spatial adaptation by wavelet shrinkage," *Biometrika*, vol. 81, no. 3, pp. 425–455, 1994.

[150] X. Guo, F. Liu, and X. Tian, "Gaussian noise level estimation for color image denoising," *Journal of the Optical Society of America A*, vol. 38, no. 8, pp. 1150–1159, 2021.

[151] Y. Li, C. Liu, X. You, and J. Liu, "A single-image noise estimation algorithm based on pixel-level low-rank low-texture patch and principal component analysis," *Sensors*, vol. 22, no. 22, p. 8899, 2022.

[152] Y. Marnissi, E. Chouzenoux, A. Benazza-Benyahia, and J.-C. Pesquet, "An auxiliary variable method for Markov chain Monte Carlo algorithms in high dimension," *Entropy*, vol. 20, no. 2, p. 110, 2018.

[153] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Computer Vision and Pattern Recognition Conference (CVPR)*, 2019.

[154] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition Conference (CVPR)*, 2009.

[155] J. Choi, S. Kim, Y. Jeong, Y. Gwon, and S. Yoon, "ILVR: Conditioning method for denoising diffusion probabilistic models," *International Conference on Computer Vision (ICCV)*, 2021.

[156] M. K. Ng, P. Weiss, and X. Yuan, "Solving constrained total-variation image restoration and reconstruction problems via alternating direction methods," *Journal on Scientific Computing*, vol. 32, no. 5, pp. 2710–2736, 2010.

[157] S. H. Chan, X. Wang, and O. A. Elgendy, "Plug-and-play ADMM for image restoration: Fixed-point convergence and applications," *IEEE Transactions on Computational Imaging*, vol. 3, no. 1, pp. 84–98, 2016.

[158] B. Kawar, M. Elad, S. Ermon, and J. Song, "Denoising diffusion restoration models," in *Advances on Neural Information Processing Systems (NeurIPS)*, 2022.

[159] H. Chung, B. Sim, D. Ryu, and J. C. Ye, "Improving diffusion models for inverse problems using manifold constraints," in *Advances on Neural Information Processing Systems (NeurIPS)*, 2022.

[160] H. Chung, J. Kim, M. T. Mccann, M. L. Klasky, and J. C. Ye, "Diffusion posterior sampling for general noisy inverse problems," in *International Conference on Learning Representations (ICLR)*, 2023.

[161] S. A. Sisson, Y. Fan, and M. Beaumont, *Handbook of Approximate Bayesian Computation*. CRC Press, 2018.

[162] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," 2018.

[163] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," in *International Conference on Learning Representations*, 2021.

[164] R. Laumont, V. D. Bortoli, A. Almansa, J. Delon, A. Durmus, and M. Pereyra, "Bayesian imaging using plug & play priors: when langevin meets tweedie," *Journal on Imaging Sciences*, vol. 15, no. 2, pp. 701–737, 2022.

[165] F. Altekrüger, P. Hagemann, and G. Steidl, "Conditional generative models are provably robust: Pointwise guarantees for Bayesian inverse problems," *Transactions on Machine Learning Research*, 2023.

[166] J. M. Bioucas-Dias, A. Plaza, N. Dobigeon, M. Parente, Q. Du, P. Gader, and J. Chanussot, "Hyperspectral unmixing overview: Geometrical, statistical, and sparse regression-based approaches," *IEEE Journal of selected topics in applied earth observations and remote sensing*, vol. 5, no. 2, pp. 354–379, 2012.

[167] C. Guilloteau, T. Oberlin, O. Berné, and N. Dobigeon, "Hyperspectral and multispectral image fusion under spectrally varying spatial blurs–application to high dimensional infrared astronomical imaging," *IEEE Transactions on Computational Imaging*, vol. 6, pp. 1362–1374, 2020.

[168] A. Durmus, E. Moulines, and M. Pereyra, "Efficient bayesian computation by proximal Markov chain Monte Carlo: when Langevin meets Moreau," *Journal on Imaging Sciences*, vol. 11, no. 1, pp. 473–506, 2018.

[169] Z. Cai, J. Tang, S. Mukherjee, J. Li, C. B. Schönlieb, and X. Zhang, "NF-ULA: Langevin Monte Carlo with normalizing flow prior for imaging inverse problems," 2023.

[170] R. M. Dudley, *Real analysis and probability*. Cambridge University Press, 2002.

[171] L. S. Pontryagin, *Mathematical theory of optimal processes*. Routledge, 2018.

[172] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[173] A. Griewank and A. Walther, *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM, 2008.

[174] W. Grathwohl, R. T. Q. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud, "Ffjord: Free-form continuous dynamics for scalable reversible generative models," *International Conference on Learning Representations (ICLR)*, 2019.

[175] M. F. Hutchinson, "A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines," *Communications in Statistics-Simulation and Computation*, vol. 18, no. 3, pp. 1059–1076, 1989.

[176] B. D. Anderson, "Reverse-time diffusion equation models," *Stochastic Processes and their Applications*, vol. 12, no. 3, pp. 313–326, 1982.

[177] A. Jolicoeur-Martineau, K. Li, R. Piché-Taillefer, T. Kachman, and I. Mitliagkas, "Gotta go fast when generating data with score-based models," 2021.

[178] T. Karras, M. Aittala, T. Aila, and S. Laine, "Elucidating the design space of diffusion-based generative models," in *Advances in neural information processing systems (NeurIPS)*, 2022.

[179] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li, and J. Zhu, "Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps," 2022.

[180] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," in *International Conference on Learning Representations (ICLR)*, 2021.

[181] A. Hyvärinen and P. Dayan, "Estimation of non-normalized statistical models by score matching.," *Journal of Machine Learning Research*, vol. 6, no. 24, pp. 695–709, 2005.

[182] Y. Song, S. Garg, J. Shi, and S. Ermon, "Sliced score matching: A scalable approach to density and score estimation," in *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2020.

Appendices

Appendix A

Langevin dynamics in the latent space

A.1 Proof of Proposition 1

The proof of Proposition 1 in Section 4.3.1 combines existing results from topology and real analysis. The complete background can be found in [170] and [82]. The proof is mainly based on the following results.

Theorem 1 ([82]). *Let q_Z and q_X define probability measures on $\mathbb{R}^d$, with $\text{supp}(q_Z) \neq \text{supp}(q_X)$. For any sequence of measurable, differentiable Lipschitzian functions $f_n : \mathbb{R}^d \to \mathbb{R}^d$, if the sequence weakly converges as $f_{n\#}q_Z \xrightarrow{\mathcal{D}} q_X$, then*

$$\lim_{n\to\infty} \text{Lip}\, f_n = \infty. \tag{A.1}$$

Moreover, [81] showed the relation between the Lipschitz constant and the Jacobian of a transformation, as stated below.

Lemma 2 (Rademacher's theorem). *If $f : \mathbb{R}^m \to \mathbb{R}^n$ is Lipschitzian, then f is continuous and differentiable at almost all points of $\mathbb{R}^m$ and*

$$\text{Lip}\, f = \sup_{z\in\mathcal{Z}} \|J_f(z)\|_{\text{op}} \tag{A.2}$$

Both Theorem 1 and Lemma 2 rely on the same starting hypothesis, i.e., f is required to be continuous, differentiable and Lipschitzian. Combining these two results yields Proposition 1 following a development of the proof of the results by [82] and [81].

A.2 Properties of the Jacobian of coupling layer-based NFs

A.2.1 Structure of the Jacobian matrix and computation of its determinant

RealNVP model defines a NF by implementing a sequence of M invertible bijective transformation functions, herein referred to as coupling layers [170]. In other words, the mapping f writes as $f = f^{(M)} \circ f^{(M-1)} \circ f^{(2)} \circ f^{(1)}$. Each bijection $f^{(m)} : u \mapsto v$ associated to the mth layer splits the input $u \in \mathbb{R}^D$ into two parts of sizes d and $d - D$ ($d \leq D$), respectively, such that the output $v \in \mathbb{R}^D$ writes

$$\left\{ \begin{array}{rcl} v_{1:d} & = & u_{1:d} \\ v_{d+1:D} & = & u_{d+1:D} \odot \exp\left(h^{(m)}\left(u_{1:d}\right)\right) + t^{(m)}\left(u_{1:d}\right) \end{array} \right. \tag{A.3}$$

where $h_m(\cdot) : \mathbb{R}^d \to \mathbb{R}^{D-d}$ and $t_m(\cdot) : \mathbb{R}^d \to \mathbb{R}^{D-d}$ are scale and translation functions implemented as deep networks and $\odot$ stands for the Hadamard product. The Jacobian of the above transformation is a lower triangular matrix

$$J^{(m)}(u) = \left[\begin{array}{cc} I_d & \mathbf{0}_{d\times(D-d)} \\ A^{(m)}(u) & E^{(m)}(u) \end{array} \right] \tag{A.4}$$

where I_d and $\mathbf{0}_{d\times(D-d)}$ are the identity and zero matrices with indexed sizes, respectively, and

$$\left\{ \begin{array}{rcl} A^{(m)}(u) & = & u_{d+1:D} \odot \frac{\partial \exp h^{(m)}(u_{1:d})}{\partial u_{1:d}} + \frac{\partial t^{(m)}(u_{1:d})}{\partial u_{1:d}} \\ E^{(m)}(u) & = & \operatorname{diag}\left(\exp\left(h^{(m)}\left(u_{1:d}\right)\right)\right). \end{array} \right. \tag{A.5}$$

Thanks to the chain rule, it follows that the Jacobian of the overall NF is

$$J_f(z) = \prod_{j=1}^{J} J^{(m)}(u^{(m)}) \tag{A.6}$$

with $u^{(m)} = f^{(m-1)}(u^{(m-1)})$ and $z = u^{(0)}$.

Moreover, because of the structure of each layer, the determinant of the Jacobian $J^{(m)}(u)$ associated with the mth layer is

$$|J^{(m)}(u)| = \prod_{k=1}^{d} \exp\left(h^{(m)}\left(u_k\right)\right). \tag{A.7}$$

The determinant of the Jacobian $J_f(\cdot)$ characterizing the overall NF can be easily computed from (A.6) and (A.7).

A.2.2 Positive definiteness of the Jacobian

Property 5. *The product of two lower triangular matrices with strictly positive diagonal elements is a positive definite lower triangular matrix.*

Proof. Let $A = [a_{ij}]$ and $B = [b_{ij}]$ be two $n \times n$ lower triangular matrices with positive diagonal entries, i.e.,

$$\forall i, j \quad \text{such that } i < j, \text{ then } a_{ij} = b_{ij} = 0. \tag{A.8}$$

$$\forall i \; a_{ii} > 0 \text{ and } b_{ii} > 0 \tag{A.9}$$

Let $C = [c_{ij}]$ denote the product matrix $C = AB$ with $c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$. The upper elements c_{ij} $(i < j)$ of C can be computed as

$$c_{ij} = \sum_{k=1}^{i} a_{ik} b_{kj} + \sum_{k=i+1}^{n} a_{ik} b_{kj}. \tag{A.10}$$

In the right hand side of (A.10), if $k \leq i$ then $b_{kj=0}$. Moreover if $k > i$ then $a_{ik} = 0$. As a consequence, $c_{ij} = 0$ and C is triangular.

Moreover, the eigenvalues of a triangular matrix is its diagonal elements. It follows that C is positive definite. ■

Thanks to the structure of coupling layer-based NFs discussed in Appendix A.2.1, we have the two following corollaries.

Corollary 1. *The Jacobian matrix $J_f(\cdot)$ and its inverse $J_f^{-1}(\cdot)$ of coupling layer-based NFs are positive definite.*

Corollary 2. *The matrix $G(\cdot)$ and its inverse $G^{-1}(\cdot)$ are positive definite.*

Appendix B

From Normalizing Flow to DDPM, the continuous bridge

This appendix continues our exploration of generative modeling by transitioning from the discussion of Normalizing Flows (NF) and Denoising Diffusion Probabilistic Models (DDPM). In the preceding sections, we've explored the intricacies of NF, a deterministic approach, and DDPM, a probabilistic framework. Here, we aim to bridge these discrete models with their continuous counterparts.

Through this exploration, we delve into Continuous Normalizing Flows (CNF) and Score-Based Models (SBM), examining the implications as we extend the number of layers in NF and the number of steps in DDPM to approach infinity. This investigation uncovers a fundamental continuity between these distinct generative modeling paradigms, shedding light on their convergence and enriching our understanding of deep generative modeling. This appendix acts as a bridge, connecting the discrete and continuous aspects, providing a deeper perspective on the interaction between NF and DDPM in generative modeling.

B.1 Continuous normalizing flows

Architectures like residual networks, based on recurrent neural networks, and certain finite normalizing flow models construct intricate transformations by combining a sequence of transformations applied to a hidden state:

$$\mathbf{h}_{t+1} = \mathbf{h}_t + f\left(\mathbf{h}_t, \theta_t\right)$$

where $t \in \{0 \dots T\}$ and $\mathbf{h}_t \in \mathbb{R}^D$,The initial data point is denoted as h_0, while h_T represents its transformation in the latent space. These sequential

updates can be thought of as discretizing continuous transformations using an Euler method. As the number of transformations approaches infinity, we describe the continuous dynamics of hidden units through an ordinary differential equation (ODE), which is determined by a neural network:

$$\frac{d\mathbf{h}(t)}{dt} = f_\theta(\mathbf{h}(t), t) \tag{B.1}$$

Beginning with the input $\mathbf{h}(0)$, we can define the output $\mathbf{h}(T)$ as the solution to this ODE initial value problem at a time T. This solution can be calculated using a differential equation solver, which operates like a black box, and evaluates the hidden unit dynamics f whenever needed to determine the solution with the desired level of accuracy.

The change in log-density under this model follows a second differential equation, called the instantaneous change of variables formula [162]:

$$\frac{\partial \log p(\mathbf{h}(t))}{\partial t} = -\operatorname{Tr}\left(\frac{\partial f_\theta}{\partial \mathbf{h}(t)}\right) \tag{B.2}$$

We can compute total change in log-density by integrating across time:

$$\log p\left(\mathbf{h}\left(t_1\right)\right) = \log p\left(\mathbf{h}\left(t_0\right)\right) - \int_{t_0}^{t_1} \operatorname{Tr}\left(\frac{\partial f_\theta}{\partial \mathbf{h}(t)}\right) dt \tag{B.3}$$

CNFs are trained to maximize (B.3). This objective involves the solution to an initial value problem with dynamics parameterized by θ. For any scalar loss function which operates on the solution to an initial value problem

$$L\left(\mathbf{h}\left(t_1\right)\right) = L\left(\int_{t_0}^{t_1} f_\theta(\mathbf{h}(t), t)dt\right) \tag{B.4}$$

then Pontryagin [171] shows that its derivative takes the form of another initial value problem

$$\frac{dL}{d\theta} = -\int_{t_0}^{t_1} \left(\frac{\partial L}{\partial \mathbf{h}(t)}\right)^T \frac{\partial f_\theta(\mathbf{h}(t), t)}{\partial \theta} dt. \tag{B.5}$$

The quantity $-\partial L/\partial \mathbf{h}(t)$ is referred to as the adjoint state of the ODE. In their work, *Chen et al.* [162] employ a black-box ODE solver to compute $\mathbf{h}\left(t_1\right)$, and then a separate call to a solver to compute (B.5) with the initial value $\partial L/\partial \mathbf{h}\left(t_1\right)$. This approach is a continuous-time analog to the backpropagation algorithm [172] and can be integrated with gradient-based optimization to adjust the parameters θ through maximum likelihood.

In general, computing $\mathrm{Tr}(\partial f_\theta / \partial \mathbf{h}(t))$ scales quadratically with dimensionality, since computing each entry of the diagonal of the Jacobian involves a separate derivative of f [173]. To mitigate this cost, [174] suggests to uses a Monte Carlo estimator of the trace of the Jacobian matrix, known as Hutchinson's trace estimator [175] :

$$\mathrm{Tr}(A) = \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\boldsymbol{\epsilon}^T A \boldsymbol{\epsilon}\right]$$

Which holds for any n-by-n matrix A and distribution $p(\epsilon)$ over n-dimensional vectors such that $\mathbb{E}[\boldsymbol{\epsilon}] = 0$ and $\mathrm{Cov}(\boldsymbol{\epsilon}) = I$.

To keep deterministic dynamics throughout each call to the ODE solver, it's possible to employ a constant noise vector $\boldsymbol{\epsilon}$ throughout the entire solving process without introducing bias:

$$\begin{aligned}
\log p\left(\mathbf{h}\left(t_1\right)\right) &= \log p\left(\mathbf{h}\left(t_0\right)\right) - \int_{t_0}^{t_1} \mathrm{Tr}\left(\frac{\partial f_\theta}{\partial \mathbf{h}(t)}\right) dt \\
&= \log p\left(\mathbf{h}\left(t_0\right)\right) - \int_{t_0}^{t_1} \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\boldsymbol{\epsilon}^T \frac{\partial f_\theta}{\partial \mathbf{h}(t)} \boldsymbol{\epsilon}\right] dt \\
&= \log p\left(\mathbf{h}\left(t_0\right)\right) - \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\int_{t_0}^{t_1} \boldsymbol{\epsilon}^T \frac{\partial f_\theta}{\partial \mathbf{h}(t)} \boldsymbol{\epsilon} dt\right]
\end{aligned} \tag{B.6}$$

Typical choices of $p(\boldsymbol{\epsilon})$ are a standard Gaussian or Rademacher distribution [175].

The trajectory of continuous normalizing flows are modelled by an ordinary differential equation (ODE) $\frac{d\mathbf{h}(t)}{dt} = f_\theta(\mathbf{h}(t), t)$. Recently a new class of generative models has emerged and proposed to model the trajectory using a stochastic differential equation (SDE). The next section will further discuss diffusion based model which can be seen as an SDE version of continuous normalizing flows.

B.2 Score-based diffusion models

Two successful classes of probabilistic generative models involve sequentially corrupting training data with slowly increasing noise, and then learning to reverse this corruption in order to form a generative model of the data. Score-based diffusion models (SBM) [144] estimate the score (i.e., the gradient of the log probability density with respect to data $\nabla_{\mathbf{x}} \log q_t(\mathbf{x})$) at each noise scale, and then uses Langevin dynamics to sample from a sequence of decreasing noise scales during generation. Denoising diffusion probabilistic modeling (DDPM) [48] [47] trains a sequence of probabilistic models to reverse each

step of the noise corruption, using knowledge of the functional form of the reverse distributions to make training tractable.

Score-based diffusion models are deep generative models that transform data into noise through a gradual diffusion process and generate samples by learning and simulating the reverse of this diffusion process. It constructs a diffusion process $\{\mathbf{x}(t)\}_{t=0}^{T}$ indexed by a continuous time variable $t \in [0, T]$, such that $\mathbf{x}(0) \sim p_X(x)$, for which we have a dataset of i.i.d. samples, and the prior distribution $\mathbf{x}(T) \sim \mathcal{N}(0, I)$. This diffusion process can be modeled as the solution to an Itô SDE:

$$\mathrm{d}\mathbf{x} = f(\mathbf{x}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w} \tag{B.7}$$

here $f(\mathbf{x}, t)$ and $g(t)$ represent the diffusion and drift functions of the SDE, and $\mathbf{w}$ denotes a standard Wiener process (a.k.a., Brownian motion). It naturally appears that continuous normalizing flows correspond to the special case where $g(t) = 0$. In the following discussion, we use $q_t(\mathbf{x})$ to refer to the probability density of $\mathbf{x}(t)$, and employ $q_{st}(\mathbf{x}(t) \mid \mathbf{x}(s))$ to denote the transition kernel from $\mathbf{x}(s)$ to $\mathbf{x}(t)$, where $0 \leqslant s < t \leqslant T$.

There are multiple ways of designing the SDE in Eq.(B.7) such that it diffuses the data distribution into a Normal distribution. One of the most widely recognized methods in the literature is Variance Preserving (VP), which is given by

$$\mathrm{d}\mathbf{x} = -\frac{1}{2}\beta(t)\mathbf{x}\mathrm{d}t + \sqrt{\beta(t)\left(1 - e^{-2\int_0^t \beta(s)\mathrm{d}s}\right)}\mathrm{d}\mathbf{w} \tag{B.8}$$

where $\beta(t) \in (0, 1)$ is a predefined function which plays a key role. It regulates the level of noise throughout the process, where larger values result in more pronounced noise in the samples. Conventionally, it's common to use a linearly increasing function [47]. More recent techniques have suggested to use cosine-based functions [49].

For any diffusion process structured as (B.7), Anderson [176] demonstrates that it can be reversed by solving the following reverse-time SDE

$$\mathrm{d}\mathbf{x} = \left[f(\mathbf{x}, t) - g(t)^2\nabla_{\mathbf{x}} \log q_t(\mathbf{x})\right]\mathrm{d}t + g(t)\mathrm{d}\overline{\mathbf{w}} \tag{B.9}$$

where $\overline{\mathbf{w}}$ is a standard Wiener process when time flows backwards, and $\mathrm{d}t$ denotes an infinitesimal negative time step. The solution trajectories of this reverse SDE share the same marginal densities as those of the forward SDE, except that they evolve in the opposite time direction [146]. Intuitively, solutions to the reverse-time SDE are diffusion processes that gradually convert noise to data. Moreover, *Song et al.* [146] prove the existence of an ODE,

namely the probability flow ODE, whose trajectories have the same marginals as the reverse-time SDE. The probability flow ODE is given by

$$\mathrm{d}\mathbf{x} = \left[f(\mathbf{x}, t) - \frac{1}{2} g(t)^2 \nabla_{\mathbf{x}} \log q_t(\mathbf{x}) \right] \mathrm{d}t. \tag{B.10}$$

Both the reverse-time SDE and the probability flow ODE allow sampling from the same data distribution as their trajectories have the same marginals.

Once accessed to the score function at each time step, one unlock both the reverse-time SDE (B.9) and the probability flow ODE (B.10). Subsequently, one can generate samples by solving these equations using various numerical techniques. These techniques encompass methods like annealed Langevin dynamics [144], numerical SDE solvers [177, 146], numerical ODE solvers [178, 179, 180], and predictor-corrector methods (combination of MCMC and numerical ODE/SDE solvers) [144].

The score of a distribution can be approximated by training a neural network with score matching [181, 182]. In order to estimate $\nabla_{\mathbf{x}} \log q_t(\mathbf{x})$, one can train a time-dependent neural network $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)$ using:

$$\mathbb{E}_{t \sim \mathcal{U}[0,T], \mathbf{x}_0 \sim p_X, \mathbf{x}_t \sim q(\mathbf{x}_t | \mathbf{x}_0)} \left[\lambda(t) \left\| \mathbf{s}_\theta \left(\mathbf{x}_t, t \right) - \nabla_{\mathbf{x}_t} \log q_{0t} \left(\mathbf{x}_t \mid \mathbf{x}_0 \right) \right\|^2 \right] \tag{B.11}$$

Here $\lambda : [0, T] \to \mathbb{R}_{>0}$ is a positive weighting function, t is uniformly sampled over $[0, T]$, $\mathbf{x}(0) \sim p_X(\mathbf{x})$ and $\mathbf{x}(t) \sim q_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0))$. Given sufficient data and model capacity, score matching ensures that the optimal solution to (B.11), denoted by $\mathbf{s}_{\theta^*}(\mathbf{x}, t)$, equals $\nabla_{\mathbf{x}} \log q_t(\mathbf{x})$ for almost all $\mathbf{x}$ and t. This learned score $\mathbf{s}_{\theta^*}(\mathbf{x}, t)$ will then be utilized in (B.9) to sample from the model.

SBM and DDPM

Recall that DDPM have a discretized version of the backward process modelled by a markov chain with a gaussian kernel (1.35). More specifically this Gaussian kernel is parametrized by a neural network ϵ_θ which predict the noise present in an arbitrary x_t. Comparing (B.11) with (1.35), it is clear that the training objectives of SBM and DDPM are equivalent, once we set $\boldsymbol{\epsilon}_\theta \left(\mathbf{x}_t, t \right) = \mathbf{s}_\theta \left(\mathbf{x}_t, t \right)$. Consequently, DDPM can be regarded as an explicit discretization of a continuous Stochastic Differential Equation (SDE)-based Scored-Based Diffusion Model.