INTRODUCTION TO GENERATIVE MODELING

Lecture Notes

Lea Bogensperger

extension of Chapter 3.4. of my PhD thesis
"Variational Methods for Imaging Meet Machine Learning"
supervised by Prof. Thomas Pock, June 2024

*https://github.com/leabogensperger/toy_generative_models

Contents

1 Introduction **3**
1.1 Langevin Sampling 3
1.2 Overview Models 4

2 Gaussian Mixture Models **5**

3 Variational Autoencoders **6**

4 Normalizing Flows **9**
4.1 Normalizing Flows 9
4.2 Continuous Normalizing Flows 11

5 Generative Adversarial Networks **11**

6 Diffusion Models **13**
6.1 Stochastic Differential Equations 13
6.2 Denoising Score Matching 14

7 Flow Matching **15**

8 Conclusion **17**

1 Introduction

Generative modeling is a type of machine learning that involves creating models capable of generating new data similar to the training data. Hence, given a data set $\{x_i\}_{i=1}^N$ of a data distribution of interest with $x \in \mathbb{R}^d$, the objective is to generate new samples that also come from this distribution. It is related to the concept of density estimation, which aims to determine the probability distribution from which the observed data was drawn, but generative modeling does not always explicitely estimate the underlying density (cf. generative adversarial networks (GANs)) whereas density estimation does not focus on the generation of new samples (cf. kernel density estimation (KDE)). Essentially, the data distribution is modeled using a parameterized *energy* function f_θ for $\theta \in \Theta$ such that

$$p_\theta(x) = \frac{\exp(-f_\theta(x))}{Z_\theta}. \tag{1}$$

Note that $Z_\theta > 0$ is the partition function ensuring a proper normalization such that the Boltzmann distribution p_θ is actually a probability density function (PDF). Although computing Z_θ in practice can be challenging, there are techniques in maximum likelihood learning to estimate Z_θ [Hin02]. On the other hand, normalizing flows require a constrained mapping which always yields a normalized PDF while diffusion models directly model $\nabla_x \log p(x)$. Moreover, models such as variational autoencoders (VAEs) and GANs do not directly model $p(x)$ but rather implicitely learn the statistics of the underlying data distribution.

1.1 Langevin Sampling

Given an estimated probability density $\hat{p}$ through generative modeling, the task of generating new samples can be approached using Markov chain Monte Carlo (MCMC) sampling. It offers a powerful framework, with a very prominent representative in this area being a first-order MCMC technique, namely Langevin sampling, which originates from physics to describe Brownian motions of particles in the viscous fluids.

Therefore, for Brownian motion W_t from a Wiener process, the overdamped Langevin Itô diffusion reads as

$$\mathrm{d}X_t = \nabla \log \hat{p}(X_t)\,\mathrm{d}t + \sqrt{2}\,\mathrm{d}W_t. \tag{2}$$

The probability distribution of the generated samples X_t approaches a stationary distribution $\hat{p}$ in the limit for $t \to \infty$.

Discretizing eq. (2) using the Euler-Maruyama method with a fixed step size $\tau > 0$ yields

$$x^{k+1} = x^k + \tau \nabla \log \hat{p}(x^k) + \sqrt{2\tau} z^k, \tag{3}$$

where $z^k \sim \mathcal{N}(0, \mathrm{Id})$ is Gaussian noise. This implies that the gradient term guides the direction of the samples effectively in high-dimensional spaces by generating a sequence of $\{x^k\}_k$. Note that ε denotes the step size and $\nabla \log \hat{p}(x^k)$ is the gradient of the target log-density evaluated at the current x^k. The gradient term helps in pushing the sample towards regions of higher probability, while the noise ensures exploration. Over time, these dynamics aim to produce samples that approximate the desired target distribution. The Langevin algorithm thus combines gradient information with stochasticity, making it particularly

useful for sampling from multimodal or high-dimensional distributions. For illustration purposes see Figure 1 for a visualization process of the Langevin sampling algorithm (center) using a simple probability density $p(x_1, x_2)$ (left). The right subplot shows randomly selected trajectories of the sampling process.

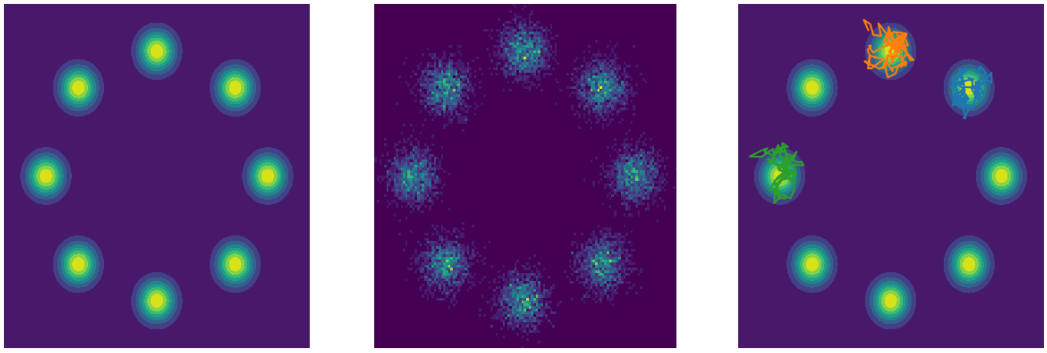

Figure 1: Example of estimated density $\hat{p}(x_1, x_2)$ (center) for a target probability density $p(x_1, x_2)$ (left) alongside with randomly selected sampling trajectories (right).

Note that the MCMC-based Langevin algorithm from (3) is also referred to as unadjusted Langevin algorithm (ULA) [RT96]. There is an entire research direction working on sampling algorithms from extensions of ULA to Metropolis-adjusted Langevin algorithm (MALA) using a Metropolis-Hastings rejection step [DCWY19] or extenting the applicability to non-smooth potentials [DMP18] and to Hamilton Monte Carlo (HMC) [N$^+$11] and higher-order sampling techniques. However, this is out of scope here.

The main advantage of Langevin sampling is rooted in its initialization independence, whereas a drawback is certainly slow convergence depending on the functions f_θ. Further, Langevin sampling requires access to the log-gradient of $\hat{p}(x)$ (i.e. the *score*), which is handled effectively by diffusion models which approximate the score through different techniques such as denoising score matching [SSDK$^+$21]. Moreover, quite some of the available generative models that are introduced in the next section do not rely on Langevin sampling to generate new samples, as they are constructed in a way such that their architecture allows to draw new samples e.g. by means of a simple forward pass. The exception of the presented models are the aforementioned diffusion models, who build on Langevin sampling in combination with a discretized reverse stochastic differential equation (SDE) to obtain samples.

1.2 Overview Models

The following paragraphs will present an introduction to a selection of some of the most well-known generative models. As this can not be covered too extensively, the interested reader is referred to [Mur22, BB24b]. Note that modeling the density in order to generate new samples from it is inherently a very challenging problem. While several methods exist that approach this task from slightly different angles by relying on specific assumptions, there is (up to now) no model yet that does not come at a disadvantage. The authors in [XKV21] identify three key requirements that a good generative model should encompass, which are

Table 1: Table summarizing selected key properties of different generative models.

criterion	GMMs	VAEs	Normalizing flows	GANs	Diffusion models	Flow matching
mode coverage	✓	✓	✓	✗	✓	✓
high sample quality	✗	✗	✗	✓	✓	✓
fast sampling	✓	✓	✓	✓	✗	✓
explicit likelihood	✓	✗	✓	✗	✗	✓
scalability	✗	✓	✗	✓	✓	✓

mode coverage, high sample quality and fast sampling. Note that fast sampling refers to both fast and computationally inexpensive sampling, which are not defined more rigorous here but rather serve as a guideline. The "generative trilema" refers to the fact that most models are usually able to fulfill only two of these requirements – although of course it is debatable on how sufficient fulfillment of any criterion is defined.

Therefore, Table 1 provides a (by no means exhaustive but) general overview on selected generative models and whether or not they fulfill some key properties that were identified. While sample quality and sampling speed are self-explanatory, mode coverage refers to a model being able to capture all significant modes of a data distribution. Moreover, scalability refers to the ability of a model to handle high-dimensional data sets and explicit likelihood computation considers the evaluation of $p_\theta(x)$. It should be mentioned that it is also more of a philosophical question of whether a model has to be scalable to high-dimensional data sets or whether an explicit likelihood evaluation is actually required – thus this should not be assessed further. This review was conducted for the vanilla version of the generative models, as each of them entails numerous improved versions, however such a comparison is out of scope.

2 Gaussian Mixture Models

Gaussian mixture models (GMMs) are perhaps the most fundamental models for generative models. Being a parametric method, they rely on assumptions of a specific functional "parametric" form of the underlying density $p(x)$, namely that it is constituted of a weighted, normalized sum of K multivariate Gaussian components. Given this assumption, the task shifts to estimating the parameters of the proposed distribution using a set of given data samples $\{x_i\}_{i=1}^N$ with $x_i \in \mathcal{X} = \mathbb{R}^n$. Therefore, for a GMM with weighted sum of K Gaussian distributions, the density is given by

$$\hat{p}(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k).$$

The estimated density $\hat{p}(x)$ is parameterized using K multivariate Gaussians, whose means $\mu_k \in \mathbb{R}^n$ and covariance matrices $\Sigma_k \in \mathbb{R}^{n\times n}$ have to be learned from the data. Moreover, the so-called mixing coefficients $\pi_k \in \mathbb{R}$ have to be fitted, which additionally live in the unit simplex Δ_K with $\Delta_K = \{\pi_k : \pi_k \geq 0, \sum_{k=1}^K \pi_k = 1\}$.

The parameters of GMMs are typically estimated using the expectation-maximization (EM) algorithm [DLR77, BN06]. Starting with initial guesses (often obtained from k-means

initializations), the expectation maximization (EM) algorithm iteratively refines these parameters to maximize the likelihood of the observed data under the model. Figure 2 shows for a given training data set the sampled results for a fitted GMM with $K = 10$ components.

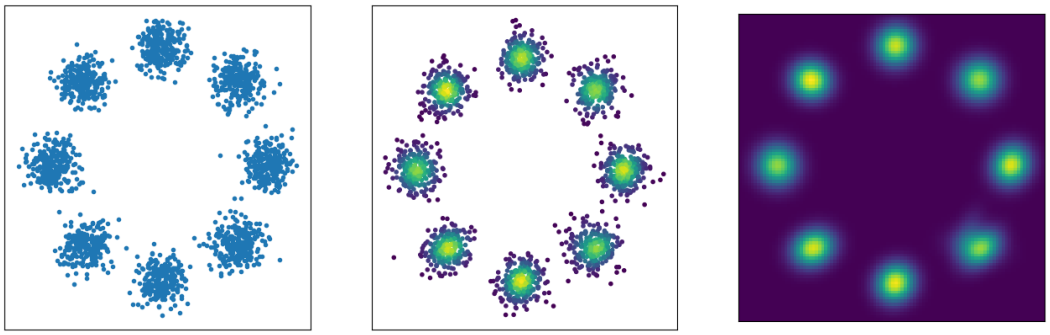

Figure 2: Generative modeling using a GMM on a simple 2D toy data set with $K = 10$ components. The trainind data (left) is used to fit the parameters of a GMM, from which new samples can be generated (center), with their corresponding likelihood (right).

3 Variational Autoencoders

One of the most well-known classes of generative models are VAEs, which are built on the concept of autoencoders [BB24a]. The core aim of autoencoders is to reconstruct input data x by learning a suitable representation z in a latent space. Thus, the first component of the model, the encoder $E_\phi(\cdot)$, has to inherently learn how to compress the data to map it to a (lower-dimensional) latent space with $z = E_\phi(x)$. This is followed by a decoder network D_θ, that maps the latent variable to x', which is required to be of the same dimensionality as the input data x. Autoencoders are deterministic and thus, their two model components, the encoder E_ϕ and decoder D_θ, are modelled by two neural networks, see Figure 3 for their basic building blocks.

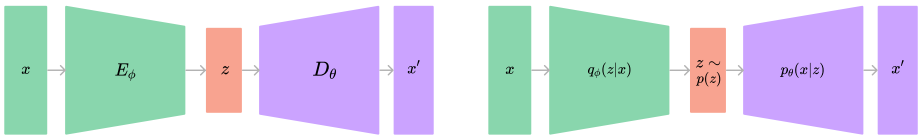

Figure 3: Block diagram showing the basic building blocks of an autoencoder (left) and a variational autoencoder (right). Note that while the overall structure looks similar, the VAE is the probabilistic version of an autoencoder, which is indicated by the encoder being the approximate posterior $q_\phi(z|x)$ and the decoder the likelihood term $p_\theta(x|z)$. The approximate posterior is used here as the true posterior is generally intractable due to the complex, high-dimensional nature of the data. Figure adapted from [Wen].

Training an autoencoder given a set of data points $\{x_i\}_{i=1}^N$ can be achieved using an ℓ_2

loss between the input data and the reconstructions, i.e.

$$\mathcal{L}(x, x') = \frac{1}{N} \sum_{i=1}^{N} \|x_i - x'_i\|_2^2, \tag{4}$$

where $x'_i = D_\theta(E_\phi(x_i))$ has been obtained by feeding the sample x_i into the encoder and decoder, respectively.

As the key aim with autoencoders is to learn useful representations of the data, constraining the latent space to prevent it from memorizing the data is crucial. There are also effective strategies such as enforcing autoencoders to denoise data [VLBM08] or to recover/inpaint masked fractions of images [HCX+22] which enable learning very complex and powerful representations that benefit various downstream tasks.

Nevertheless, the latent space of an autoencoder will be sparse by construction as it learns a suitable mapping for the given data points without any further constraint or regularity imposed on its structure. This is where the concept of VAEs builds upon [KW13] as they are essentially a probabilistic version of autoencoders that assume the latent variable to follow a Gaussian normal distribution. See Figure 3 for an overview over the main building blocks which resemble a lot those of autoencoders. With the goal to model $p_\theta(x)$, this can be specified by using latent variables

$$p_\theta(x) = \int_z p_\theta(x, z) \dot{z} = \int_z p_\theta(x|z) p_\theta(z) \dot{z}.$$

Bayes' theorem can be used to model the prior $p_\theta(z)$, the likelihood $p_\theta(x|z)$ and the posterior $p_\theta(z|x)$, where the posterior is hard to compute due to intractable integrals. Thus, this is modelled using an approximate posterior $q_\phi(z|x) \approx p_\theta(z|x)$, which is commonly known as amortized inference. It is modelled by a Gaussian distribution with

$$q_\phi(z|x) = \mathcal{N}(z|E_\phi(x), \sigma_\phi^2(x)\mathrm{Id}),$$

where the encoder learns to infer its mean $E_\phi(x)$ and standard deviation $\sigma_\phi(x)$ with the latter usually being scalar due to isotropy. The variational distribution $q_\phi(z|x)$ ideally is as close as possible to $p_\theta(z|x)$, measured by the Kullback-Leibler divergence (KLD) with $D_{\mathrm{KL}}(q_\phi(z|x)||p_\theta(z|x))$. Using the definition of the KLD with $D_{\mathrm{KL}}(q||p) = \int q(x) \log \frac{q(x)}{p(x)} \dot{x}$ and the product rule, this can be rewritten to

$$D_{\mathrm{KL}}(q_\phi(z|x)||p_\theta(z|x)) = \log\ p_\theta(x) - \mathbb{E}_{z\sim q_\phi(\cdot|x)}\Big[\log \frac{p_\theta(x, z)}{q_\phi(z|x)}\Big],$$

where the evidence lower bound (ELBO) $L_{\phi,\theta}$ can be identified as

$$L_{\phi,\theta} = \mathbb{E}_{z\sim q_\phi(\cdot|x)}\Big[\log \frac{p_\theta(x, z)}{q_\phi(z|x)}\Big].$$

Thus, maximizing the ELBO with $L_{\phi,\theta} \leq p_\theta(x)$ provides a lower bound to the maximum likelihood (ML) objective. The objective to learn a VAE can finally be rewritten (using

again the product rule) as

$$\max_{\phi\in\Phi,\theta\in\Theta} L_{\phi,\theta} = \mathbb{E}_{z\sim q_\phi(\cdot|x)}[\log p_\theta(x|z)] - D_{\mathrm{KL}}(q_\phi(z|x)||p_\theta(x)),$$

which in practice can be expressed – where M denotes the size of the latent space – with

$$\max_{\phi\in\Phi,\theta\in\Theta} -\tfrac{1}{2}\mathbb{E}_{z\sim q_\phi(\cdot|x)}\left[\|x - D_\theta(z)\|_2^2\right] + M\log\sigma_\phi(x) - \tfrac{1}{2}\|E_\phi(x)\|_2^2 - \tfrac{M}{2}\sigma_\phi^2(x). \tag{5}$$

To optimize the ELBO with respect to parameters $\{\phi, \theta\}$ of the encoder E_ϕ (the approximate posterior) and decoder D_θ (the likelihood term), it is necessary to backpropagate through the sampled latent variable $z \sim q_\phi(z|x)$. The is done using the "re-parameterization trick" which converts the output of the encoder E_ϕ, which is inherently deterministic, to a tractable, stochastic latent variable $z = E_\phi(x) + \sigma_\phi(x)\eta$ with $\eta \sim \mathcal{N}(0, \mathrm{Id})$. Hence, this is inserted into z for the first term in (5), where the expectation operator is further approximated by sampling a fixed number of latents z to compute the average.

Figure 4 shows for a 2D toy data set consisting of eight Gaussians the generated samples from a trained VAE along with their densities that were computed using a KDE. This was fit to the generated samples by employing a hand-tuned bandwidth h (see Section **??**).

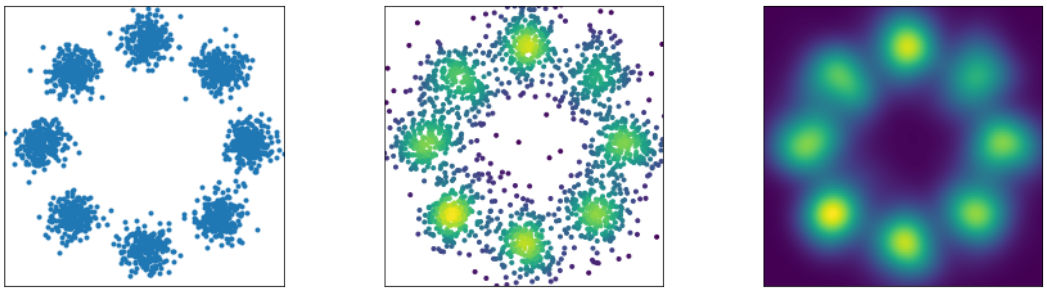

Figure 4: Generative modeling using a VAE on a simple 2D toy data set. The input data samples $\{x_i\}$ are shown on the left and the center plot depicts the generated samples coloured by their estimated density using a KDE. The right plot shows the estimated density of the generated samples using the VAE for the entire data space.

In practice, there are two major problems of VAEs that have been identified [RV18]. The first is that of blurred reconstructions arising from latent space regions where the mapping to the posterior of different data points overlap. The reconstruction is thus a combination of these data points resulting in a weighted average leading to blurred samples, which is why VAEs are known to lack high sample quality. The second issue is the "hole" problem, which occurrs in regions where the posterior $q_\phi(z|x)$ has low densities such that the ELBO objective is not constrained to learn a suitable mapping as it is not incentivized to consider these instances. If the decoder receives latent samples z_i that are from latent regions which were not covered by the input data samples $\{x_i\}_{i=1}^N$, the generated samples may be highly deficient in good quality.

4 Normalizing Flows

4.1 Normalizing Flows

Normalizing flows are another prominent class of generative models [DKB14, RM15, DSDB16]. Their main idea is that they map a tractable prior distribution $p_Z(z)$ into a complex data distribution $p_X(x)$ by a so-called flow (note that subscripts are used here to differentiate the PDFs). Constructing this flow such that it is an invertible mapping allows for a proper normalization to ensure that the actual likelihood of the data can be evaluated.

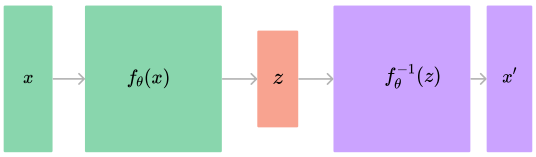

Figure 5: Block diagram showing the basic building blocks of normalizing flows. This diffeomorphism allows to map from a complex distribution $p_X(x)$ to a tractable prior distribution $p_Z(z)$ and back while maintaining explicit likelihood evaluation. Figure adapted from [Wen].

Central to this concept is the change-of-variables theorem (see Section **??**) which allows to transform one distribution into another and vice versa. Thus, given a latent variable $z \sim p_Z(z)$, an invertible mapping can be constructed such that

$$x = f_\theta(z) \leftrightarrow z = f_\theta^{-1}(x).$$

By the change-of-variables theorem one then obtains

$$p_X(x) = p_Z\big(f_\theta^{-1}(x)\big)\left|\frac{\mathrm{f}_\theta^{-1}}{\mathrm{x}}\right|,$$

which contains the Jacobian of the inverse map. As the inference path can only be meaningful if it is a function of z from whose tractable prior distribution samples can easily be drawn, this can be rewritten[1] as

$$p_X(x) = p_Z\big(z\big)\left|\frac{\mathrm{f}_\theta}{\mathrm{z}}\right|^{-1}.$$

Finally, the log-density with $p_\theta(x) \approx p_X(x)$ is evaluated using

$$\log p_\theta(x) = \log p_Z\big(z\big) - \log\left|\frac{\mathrm{f}_\theta}{\mathrm{z}}\right|.$$

This basic concept is outlined in the block diagram in Figure 5. Note that in general the function f_θ consists of a sequence of invertible mappings such that complex data distributions

[1] Note that this requires a property from the inverse function theorem which states that for $x = f(z) \leftrightarrow z = f^{-1}(x)$ it holds that $\frac{\mathrm{f}^{-1}(x)}{\mathrm{x}} = \frac{\mathrm{z}}{\mathrm{x}} = \big(\frac{\mathrm{x}}{\mathrm{z}}\big)^{-1} = \big(\frac{\mathrm{f}(z)}{\mathrm{z}}\big)^{-1}$. Moreover, the property of the Jacobian of an invertible function is used.

can be modelled. Thus, assuming a sequence of L functions f_l that map the original latent variable $z := z_0$ to x using $x := z_L = f_L \circ f_{L-1} \circ \cdots \circ f_1(z_0)$, the log-density can also be computed directly using

$$\log p_\theta(x) = \log p_0\big(z_0\big) - \sum_{l=1}^{L} \log \left| \frac{\mathrm{f}_l}{\mathrm{z}_{l-1}} \right|.$$

The transformations f_l are required to be invertible and to have a computable Jacobian. Learning suitable parameters θ for the invertible transformations f_l is done by minimizing the KLD between the model likelihood and the target distribution:

$$\min_{\theta \in \Theta} D_{\mathrm{KL}}(p_X(x), p_\theta(x)) = \mathbb{E}_{x \sim p_X(x)}[\log p_X(x)] - \mathbb{E}_{x \sim p_X(x)}[\log p_\theta(x)].$$

While the first term is independent of the model parameters θ, the expectation in the second term can be approximated with a given set of training examples $\{x_i\}_{i=1}^N \sim p_X(x)$ such that the objective reduces to the maximum likelihood of the model under the observed samples of the target distribution, i.e.

$$\max_{\theta \in \Theta} \sum_i \log p_\theta(x_i).$$

In practice, there are various methods for the construction of invertible layers [DKB14, RM15, DSDB16, KD18], which are not dealt with in more detail here. As a qualitative example, see Figure 6 where the invertible layers were constructed using the RealNVP method [DSDB16]. The mapping from samples $x \sim p_X(x)$ from the data distribution to the prior distribution $z \sim p_Z(z)$ is shown, along with the transformed samples in the intermediate layers. Note that during inference, one starts in reverse order by drawing samples $z \sim p_Z(z)$ and subsequently mapping them to the data distribution using the learned flow. Finally, Figure 7 shows qualitative results of training a normalizing flow model on a training data set $\{x_i\}$, where the generated samples and the learned density (implicit estimation using KDE) follows the original distribution closely.

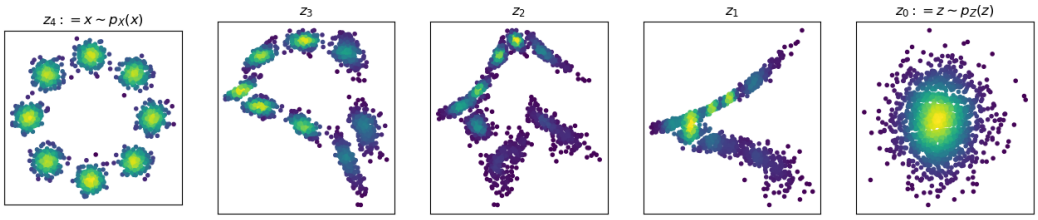

Figure 6: Generative modeling using a normalizing flow with $L = 4$ invertible layers on a simple 2D toy data set. The input data samples $\{x_i\}$ are shown on the left, the center plots depict the transformed samples in the latent space for each intermediate layer z_l and the right plot shows the final transformed samples $z_0 := z \sim p_Z(z)$ in the latent space.

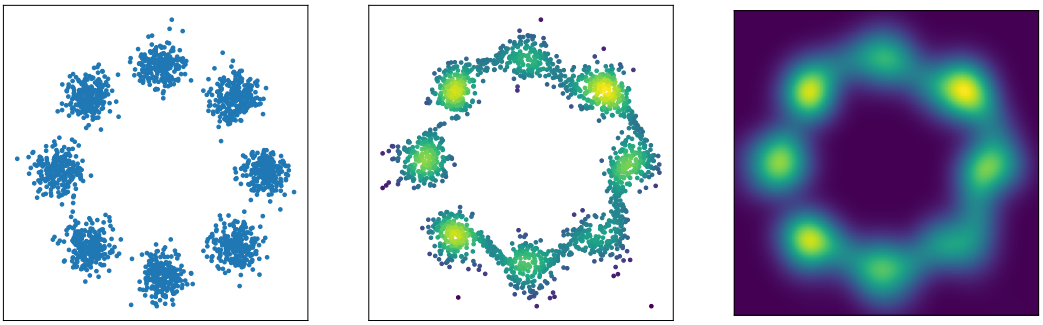

Figure 7: Generative modeling using a normalizing flow on a simple 2D toy data set: input samples $\{x_i\}$ (left), generated samples from the trained model (center) and estimated density from the samples using a KDE (right).

4.2 Continuous Normalizing Flows

As an extension of normalizing flows the authors in [CRBD18] propose continuous normalizing flows (CNFs). They characterize a target density $q = [\Psi]_\sharp p_0$ as push-forward of a tractable initial density p_0 under a deterministic transformation $\Psi\colon \mathbb{R}^d \to \mathbb{R}^d$. Typically, the transformation $\Psi\colon \mathbb{R}^d \to \mathbb{R}^d$ is learned via maximum likelihood and maps samples $x \sim p_0$ from the initial distribution to samples $\Psi(x) \sim q$ that follow the target [RM15]. However, CNFs do not directly model Ψ, but rather consider the temporal dynamics of a time-dependent transformation $\Psi_t\colon [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$ with $t \in [0,1]$. The corresponding neural ordinary differential equation (ODE) [CRBD18] is given as

$$\frac{\mathrm{d}}{\mathrm{d}t}\Psi_t(x) = v_{\theta,t}(\Psi_t(x)), \tag{6}$$

where $v_{\theta,t}\colon [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$ is a parameterized network. It defines a time-dependent vector field by assigning spatial displacements to each spatio-temporal pair $(t, \Psi_t(x))$. The overall transformation $\Psi(x) = \Psi_1(x) = x + \int_0^1 v_{\theta,t}(x)\mathrm{t}$ for the initial state x is then given by integrating the spatial displacements over the whole time domain, which can be done by utilizing off-the-shelf ODE solvers. However, the widespread adoption of CNFs as generative models has so far been hindered by the cost of maximum likelihood training. Because of this limitation, flow matching has been introduced as a feasible alternative [LCBH+22, LGL22], see Section 7.

5 Generative Adversarial Networks

GANs are a well-known type of generative models that were introduced by Goodfellow et al. [GPAM+14]. The core idea is a system of two neural networks that contest with each other: one network (the "generator" G_θ) generates samples while the second network (the "discriminator" D_ϕ) subsequently evaluates them, which ultimatively leads to the generation of data that resembles the input data set. The overall idea is also visualized in Figure 8. The task of the discriminator (left) is to decide whether the samples that it is provided with are coming from the data distribution $p(x)$ (samples x) or not (samples x'). The fake samples

x' are produced by the generator (right) from a latent vector z that is obtained by sampling from a simple distribution $z \sim p(z)$. The goal is to eventually fool the discriminator by learning to produce samples that are not distinguishable from the data distribution.

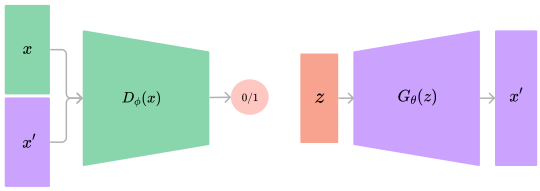

Figure 8: Block diagram showing the basic building blocks of GAN. Two networks are contesting against each other, where the generator (right) produces samples x' and the discriminator (left) evaluates them. Figure adapted from [Wen].

Assuming both players to be modelled by neural networks, the parameters of the generator G_θ and the discriminator D_ϕ can be learned using the following saddle-point objective $V(D_\phi, G_\theta)$

$$\min_{\theta\in\Theta}\max_{\phi\in\Phi} V(D_\phi, G_\theta) = \mathbb{E}_{x\sim p(x)}[\log D_\phi(x)] + \mathbb{E}_{z\sim p(z)}[\log(1 - D_\phi(G_\theta(z)))]. \tag{7}$$

The first term maximizes the probability of the discriminator to correctly distinguish real from fake samples. Meanwhile, the generator aims to minimize the probability of the discriminator to detect its fake samples. The authors in [GPAM+14] show that for arbitrary functions of D_ϕ and G_θ there exists a unique solution such that the generator produces samples that are not distinguishable from $x \sim p(x)$ which consequently means that D_ϕ outputs $\frac{1}{2}$ for all samples as it can only guess. Moreover, they show that given an optimal discriminator, the training process of the generator can be seen as minimizing the Jensen-Shanon divergence (JSD) between the learned distribution and the data distribution.

An example of using a vanilla GAN on the 2D toy data set can be seen in Figure 9. Note that while the generated samples (center, right) are reflecting the density of the data samples (left), the center plot depicting the produced samples clearly shows that they suffer from mode collapse i.e. the generator fails to capture the full diversity of the data distribution. This was also observed on a toy data set in [XKV21]. Mode collapse is a well-known issue with GANs and it is one of the main drawbacks together with the fact that they can be notoriously hard to train.

One approach that tries to improve this issue are WGANs [ACB17], that use the Wasserstein distance as a distance measure between the distributions of the generated samples and the data samples. The Wasserstein distance can be seen as the required cost to transform a probability distribution into another.

The employed objective function (see [ACB17] for a detailed explanation and derivation) is then given as follows:

$$\min_{\theta\in\Theta}\max_{\phi\in\Phi} V(D_\phi, G_\theta) = \mathbb{E}_{x\sim p(x)}[D_\phi(x)] - \mathbb{E}_{z\sim p(z)}[D_\phi(G_\theta(z))],$$

which is slightly different to the original formulation in (7). The discriminator is seen as a critic, thereby returning a score that represents the "realness" of the given data rather than a probability. Furthermore, weight clipping is introduced to enforce a Lipschitz constraint that is required for the critic to effectively estimate the Wasserstein distance. This helps with both reducing mode collapse and stabilizing the training process, as the distance metric provides meaningful gradients even when the generator produces very unrealistic samples which would result in vanishing gradients with the vanilla GAN setup.

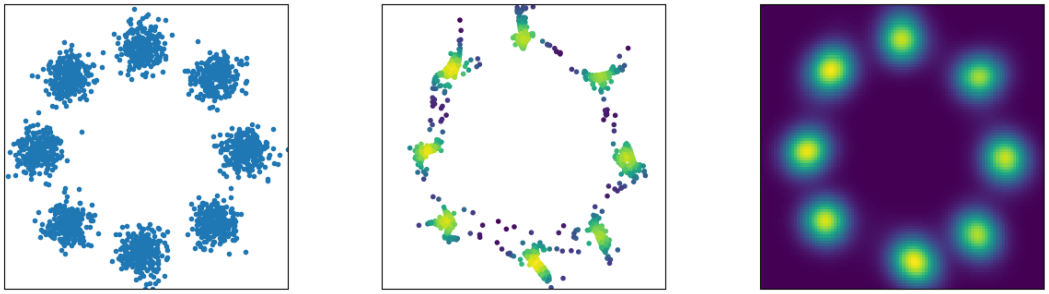

Figure 9: Generative modeling using a GAN on a simple 2D toy data set. Note that while the trained generator manages to capture the main modes of the distribution, the generated samples suffer from mode collapse. The density in the right subplot was estimated using a KDE on the entire data space with the learned GAN.

The introduction of GANs have led to many variants. The work of Radford et al. introduced deep convolutional generative adversarial networks (DCGANs) [RMC15] by proposing to replace the fully connected networks by a sophisticated convolutional architecture, which helped to better scale GANs to high-quality image synthesis. Moreover, the work of Isola et al. introduced conditional GANs [IZZE17] to enable conditional image synthesis, paving the way for image inpainting, segmentation, super-resolution and many other applications. Although no explicit likelihood evaluation is possible with GANs, they show exceptionally high sample quality while maintaining fast sampling. Such results have not been achieved with other generative models previously and this was only surpassed very recently by diffusion models [DN21].

6 Diffusion Models

6.1 Stochastic Differential Equations

Diffusion models are a younger and very promising type of generative models [SDWMG15, SSDK$^+$21, HJA20], which have different motivational perspectives such as non-equilibrium thermodynamics, score matching and SDEs. Essentially, they can be viewed as an interpolation between a data distribution of interest (this will be denoted as p_0) and a tractable prior distribution p_1, from which samples can easily be obtained. This interpolation should be bidirectional, whereas going from data to noise is a noising process (the *forward* process) and retrieving data from noise is the generative process (the *backward* process).

It turns out that this can be described mathematically using SDEs, which was described by [SSDK$^+$21] in a unifying framework to encompass all the different versions such as

denoising diffusion probabilistic models (DDPMs) and denoising score matching. In its continuous form, the forward SDE is given by

$$\mathrm{d}X_t = f(X_t, t)\,\mathrm{d}t + g(t)\,\mathrm{d}W_t, \tag{8}$$

where $f(\cdot, t) : \mathbb{R}^d \to \mathbb{R}^d$ is the so-called drift coefficient and $g(\cdot) : \mathbb{R} \to \mathbb{R}$ is the diffusion coefficient. Depending on the choice of these two terms, different instances of SDEs can be retrieved such as the variance-exploding scheme (commonly known as DDPMs) and the variance-exploding scheme (i.e. denoising score matching).

Interestingly, the time evolution of the corresponding density function p_t is governed by the Fokker-Plank equation, which is a partial differential equation and is given by

$$\partial_t p_t = -\mathrm{div}(p_t\ f(X_t, t)) + \Delta(p_t\ g(t)). \tag{9}$$

Hence, the time-related change of a PDF is given by the combination of a drift and a diffusion term: the drift term represents the deterministic part of the motion by describing the flow of the PDF due to the drift. Likewise, the diffusion term models the change of the PDF due to diffusion by incorporating a randomness into the process. If no diffusion term is present, the continuity equation is recovered (cf. Section 7).

For generative modeling, obtaining a reverse form for the SDE is of interest such that noise can be transformed into data samples. Therefore, the corresponding time-reverse SDE reads as [And82]

$$\mathrm{d}X_t = [f(X_t, t) - g(t)^2 \nabla_x \log p_t(X_t)]\,\mathrm{d}t + g(t)\,\mathrm{d}\bar{W}. \tag{10}$$

Looking closely at eq. (10) where the drift and diffusion terms are governed by the choice of SDE the only remaining term that has to be determined is the *score* $\log p_t$. Once this is known (or can be approximated suitable), the reverse SDE can be discretized using a Euler-Maruyama discretization scheme to sample from the distribution of interest. Therefore, the next section will present denoising score matching, a method to obtain an estimate for the score.

6.2 Denoising Score Matching

The concept of score matching has been introduced earlier [HD05] and popularized to denoising score matching recently [VLBM08, SE19]. The key insight is that score estimation in low density data regions is hardly feasible in its native form, thus the data is perturbed by L different levels of Gaussian noise σ_i instead with $i = 1, \ldots, L$. Note that $\sigma_1 < \sigma_2 < \cdots < \sigma_L$, which are used to perturb the data distribution $p(x)$ with Gaussian noise $\mathcal{N}(0, \sigma_i^2 \mathrm{Id})$. Therefore, samples from the noise-perturbed data distribution $\bar{x} \sim p_{\sigma_i}(x)$ can be obtained by sampling random Gaussian noise $z \sim \mathcal{N}(0, \mathrm{Id})$ and then computing

$$\bar{x} = x + \sigma_i z.$$

Estimating the score function of the perturbed samples becomes an easier task and it relies on Tweedies estimator [Efr11] which gives an estimate for the minimum mean squared error

(MMSE) of a perturbed sample $\bar{x}$:

$$x_{\text{MMSE}} = \bar{x} + \sigma_i^2 \nabla_{\bar{x}} \log p_{\sigma_i}(\bar{x}|x). \tag{11}$$

Note that by replacing the MMSE directly with the known clean sample x in training, eq. (11) can be reformulated to yield an expression for the perturbed score $\nabla_{\bar{x}} \log p_{\sigma_i}(\bar{x}|x)$. This can be contrasted to a score network $s_\theta(\cdot, \sigma_i)$ to learn optimal parameter θ. The denoising score matching objective is then given by a weighted sum of denoising score matching terms

$$\theta^* \in \arg\min_\theta \sum_{i=1}^{L} \sigma^2 \mathbb{E}_{x\sim p(x)} \mathbb{E}_{p_{\sigma_i}(\bar{x}|x)} \left[s_\theta(\bar{x}, \sigma_i) - \nabla_{\bar{x}} \log p_{\sigma_i}(\bar{x}|x) \right]. \tag{12}$$

The above loss function shows that the training process of diffusion models consists of learning a denoiser, as it compares the estimated noise by the neural network to the ground truth noise. While different variants of this exist, namely image denoising, score matching and noise predictions [KG24], they essentially represent the same idea.

Figure 10 qualitatively depicts for five selected time steps of the discretized reverse SDE the generated samples by a diffusion model that was trained on the 2D toy data set. Thereby, a simple prior distribution is gradually turned into the data distribution of interest (from left to right) to obtain samples.

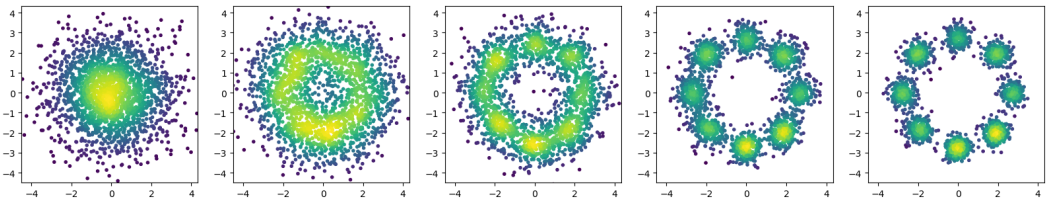

Figure 10: Generative modeling using a diffusion model on a simple 2D toy data set. The variance-exploding SDE scheme was used (i.e. denoising score matching) thus the sampling process is given by the discretized reverse SDE. The five selected time steps show intermediate generated samples during the sampling process.

7 Flow Matching

Finally, it remains to study in a bit more detail the very recently introduced generative model of flow matching [LGL22, LCBH$^+$22]. Reconsidering the limitations of CNFs, which were briefly introduced in Section 4.2, their main limitation was the lack of simulation-free training. It turns out that this necessity can be targeted by considering conditional flow matching. Again, as with CNFs the goal is to learn a parameterized neural network $v_{\theta,t} \colon [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$ which satisfies the neural ODE [CRBD18]

$$\frac{\mathrm{d}}{\mathrm{d}t} \Psi_t(x) = v_{\theta,t}(\Psi_t(x)). \tag{13}$$

Note that this means that a network $v_{\theta,\cdot}$ is learned to encompass the temporal dynamics of the transformation Ψ_t.

The essential insight is that any time-dependent vector field $u_t \colon [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$ which is related to the temporal dynamics of a given density path $p_t \colon [0,1] \times \mathbb{R}^d \to [0,\infty)$ via the ubiquitous continuity equation [V+09]

$$\frac{\mathrm{d}}{\mathrm{d}t} p_t(x) + \operatorname{div}(p_t(x) u_t(x)) = 0, \tag{14}$$

can be used as a regression target. The corresponding flow matching objective is thus given as

$$\mathcal{L}_{\mathrm{FM}}(\theta) := \mathbb{E}_{t \sim \mathcal{U}_{[0,1]}} \mathbb{E}_{x \sim p_t} \left[\tfrac{1}{2} \| v_{\theta,t}(x) - u_t(x) \|^2 \right]. \tag{15}$$

This objective is justified by identifying the density path in Eq. (14) with the one that is induced by the continuous transformation, i.e., $p_t = [\Psi_t]_\sharp p_0$. Thus, it becomes evident that minimizing Eq. (15) indeed yields a vector field $v_{\theta,t}(\cdot)$ characterizing Ψ_t via Eq. (13). However, as the marginal ground truth vector field u_t is not known in general and p_t can take a complicated form, directly minimizing Eq. (15) is impossible. Instead, Lipman et al. [LCBH+22] show that u_t can be constructed by superimposing many conditional vector fields $u_t(\cdot|x_1)$ that depend on available training samples $x_1 \sim q$. The simpler conditional flow matching objective is thus given as

$$\mathcal{L}_{\mathrm{CFM}}(\theta) = \mathbb{E}_{t \sim \mathcal{U}_{[0,1]}} \mathbb{E}_{x_1 \sim q} \mathbb{E}_{x \sim p_t(\cdot|x_1)} \left[\tfrac{1}{2} \| v_{\theta,t}(x) - u_t(x|x_1) \|^2 \right], \tag{16}$$

which yields the same set of minimizers as Eq. (15) and reveals similarities to score-based optimization [HD05]. To obtain an explicit expression for the regression target $u_t(x|x_1)$, they propose conditional affine Gaussian flows $\Psi_t(x|x_1) = \sigma_t x + \mu_t(x|x_1)$. According to Eq. (13) it is of the form $u_t(x|x_1) = \frac{x - \mu_t(x|x_1)}{\sigma_t} \frac{\mathrm{d}}{\mathrm{d}t}\sigma_t + \frac{\mathrm{d}}{\mathrm{d}t}\mu_t(x|x_1)$. In addition, the corresponding density path has the form $p_t(x|x_1) = \mathcal{N}\big(x|\mu_t(x|x_1), \sigma_t^2 \operatorname{Id}\big)$. Following [LCBH+22], it starts from the tractable multivariate standard normal distribution $p_0(\cdot|x_1) = \mathcal{N}\big(\cdot|0, \operatorname{Id}\big)$ and ends in a Dirac peak at the conditioning sample x_1, i.e., $p_1(\cdot|x_1) = q(\cdot|x_1) = \mathcal{N}\big(\cdot|x_1, \sigma_{\min}^2 \operatorname{Id}\big)$, $\sigma_{\min}^2 \approx 0$. In contrast, the temporal interpolation between (μ_0, σ_0) and (μ_1, σ_1) is not uniquely defined and offers some degree of freedom. In this work, we adopt the best-performing variant from [LCBH+22] that induces conditional optimal transport paths via $\mu_t(x|x_1) = t x_1$ and $\sigma_t = 1 - (1 - \sigma_{\min})t$.

Finally, Figure 11 shows the samples at selected time steps during the sampling process, which essentially amounts to solving the discretized ODE that characterizes the dynamics where the learned network $v_{\theta,\cdot}$ is used.

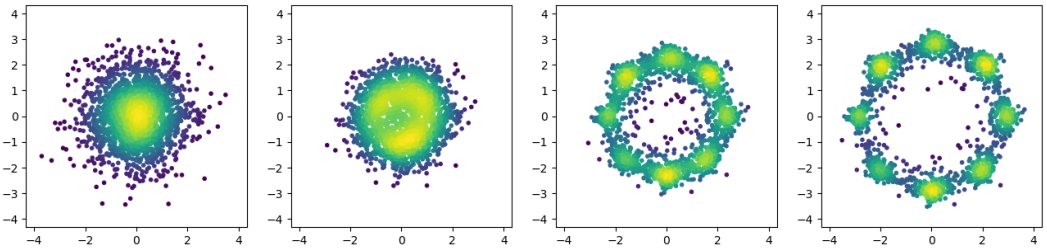

Figure 11: Generative modeling using flow matching on a simple 2D toy data set. The sampling process is given by the discretized integrated ODE. The selected time steps show intermediate generated samples during the sampling process.

8 Conclusion

A final comparison of the generated samples on the 2D toy data set is shown in Figure 12. Moreover, to additionally have a quantitative comparison, see Table 2 for the KLD between the generated samples and the original data samples $x \sim p(x)$ that were used to train each of the generative models. In general, the metrics reflect what can be observed qualitatively in Figure 12, but this should be taken with caution. Only the baseline models were used here, whereas numerous enhancements exist that potentially yield improved performances. Hence, this does not imply that one model is favorable over the others, as all have their strengths and weaknesses (see Table 1) and essentially, the choice of model will heavily depend on the specific application.

Table 2: Table showing the computed KLD between the generated samples and the data samples for each of the generative models.

	GMM	VAE	Normalizing flow	GAN	Diffusion model	Flow matching
KLD	0.0099	0.0950	0.0165	0.1502	0.0149	0.0164

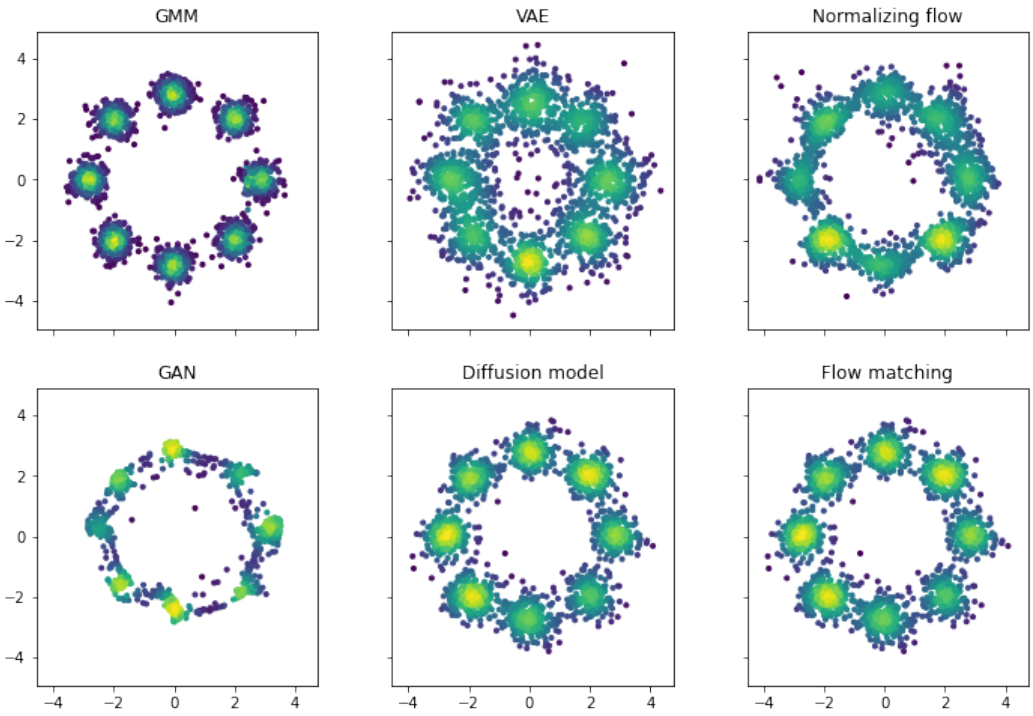

Figure 12: Comparison of the generated samples from the generative models, namely GMMs, VAEs, normalizing flows, GANs, diffusion models and flow matching (left to right, top-down) on the toy data set.

References

[ACB17] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223. PMLR, 2017.

[And82] Brian DO Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

[BB24a] Christopher M. Bishop and Hugh Bishop. *Autoencoders*, pages 563–579. Springer International Publishing, Cham, 2024.

[BB24b] Christopher M. Bishop and Hugh Bishop. *Deep learning : foundations and concepts*. Springer, 2024.

[BN06] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[CRBD18] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[DCWY19] Raaz Dwivedi, Yuansi Chen, Martin J Wainwright, and Bin Yu. Log-concave sampling: Metropolis-hastings algorithms are fast. *Journal of Machine Learning Research*, 20(183):1–42, 2019.

[DKB14] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

[DLR77] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society: series B (methodological)*, 39(1):1–22, 1977.

[DMP18] Alain Durmus, Eric Moulines, and Marcelo Pereyra. Efficient bayesian computation by proximal markov chain monte carlo: when langevin meets moreau. *SIAM Journal on Imaging Sciences*, 11(1):473–506, 2018.

[DN21] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

[DSDB16] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. In *International Conference on Learning Representations*, 2016.

[Efr11] Bradley Efron. Tweedie's formula and selection bias. *Journal of the American Statistical Association*, 106(496):1602–1614, 2011.

[GPAM+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. volume 27, 2014.

[HCX+22] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 16000–16009, 2022.

[HD05] Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

[Hin02] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[HJA20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. volume 33, pages 6840–6851, 2020.

[IZZE17] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1125–1134, 2017.

[KD18] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. volume 31, 2018.

[KG24] Diederik P Kingma and Ruiqi Gao. Understanding diffusion objectives as the elbo with simple data augmentation. In *Advances in Neural Information Processing Systems*, 2024.

[KW13] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[LCBH+22] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[LGL22] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

[Mur22] Kevin P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022.

[N+11] Radford M Neal et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.

[RM15] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538. PMLR, 2015.

[RMC15] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[RT96] Gareth O Roberts and Richard L Tweedie. Exponential convergence of langevin distributions and their discrete approximations. *Bernoulli*, pages 341–363, 1996.

[RV18] Danilo Jimenez Rezende and Fabio Viola. Taming vaes. *arXiv preprint arXiv:1810.00597*, 2018.

[SDWMG15] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

[SE19] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. volume 32, 2019.

[SSDK+21] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

[V+09] Cédric Villani et al. *Optimal transport: old and new*, volume 338. Springer, 2009.

[VLBM08] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *International Conference on Machine Learning*, ICML '08, page 1096–1103, New York, NY, USA, 2008. Association for Computing Machinery.

[Wen] Lilian Weng. What are diffusion models? `"https://lilianweng.github.io/posts/2021-07-11-diffusion-models/"`. Accessed: 2024-02-05.

[XKV21] Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the generative learning trilemma with denoising diffusion gans. In *International Conference on Learning Representations*, 2021.