Galaxy Imaging with Generative Models: Insights from a Two-Models Framework

Jean-Eric Campagne*
Université Paris-Saclay, CNRS/IN2P3, IJCLab, 91405 Orsay, France

12 December 2024

ABSTRACT

Generative models have recently revolutionized image generation tasks across diverse domains, including galaxy image synthesis. This study investigates the statistical learning and consistency of three generative models: `light-weight-gan` (a GAN-based model), `Glow` (a Normalizing Flow-based model), and a diffusion model based on a `U-Net` denoiser, all trained on non-overlapping subsets of the SDSS DR7 dataset of 64×64 grayscale images. While all models produce visually realistic images with well-preserved morphological variable distributions, we focus on their ability to learn and generalize the underlying data distribution.

The diffusion model shows a transition from memorization to generalization as the dataset size increases, confirming previous findings by Kadkhodaie et al. (2024). Smaller datasets lead to overfitting, while larger datasets enable novel sample generation, supported by the denoising process's theoretical basis. For the flow-based model, we propose an "inversion test" leveraging its bijective nature. Similarly, the GAN-based model achieves comparable morphological consistency but lacks bijectivity. We then introduce a "discriminator test," which shows successful learning for larger datasets but poorer confidence with smaller ones. Across all models, dataset sizes below $O(10^5)$ pose challenges to learning.

Along our experiments, the "two-models" framework introduced by Kadkhodaie et al. (2024) enables robust evaluations, highlighting both the potential and limitations of these models. These findings provide valuable insights into statistical learning in generative modeling, with applications certainly extending beyond galaxy image generation.

Key words: methods: data analysis, methods: numerical, methods: statistical

1 INTRODUCTION

Image generation in Machine Learning (ML) is a challenging task, particularly in the context of non-ergodic processes. Recently, significant breakthroughs in image quality have been achieved thanks to large-scale statistical model architectures such as `DALL-E2` (Ramesh et al. 2022), `Midjourney` (Oppenlaender 2022), and `StableDiffusion` (Rombach et al. 2022), which leverage *stochastic diffusion processes*. These models have largely replaced the previous generation of approaches based on *variational autoencoders* (VAE) (Kingma & Welling 2014)[1], *adversarial networks* (GAN) (Goodfellow et al. 2014) as employed in works such as (e.g., Karras et al. 2018a; Brock et al. 2019a), and *normalizing flows*, exemplified by `Glow` (Kingma & Dhariwal 2018).

Generative models have rapidly found applications across diverse domains. For instance, in High Energy Physics, a comprehensive review can be found in Kansal et al. (2023). Regarding galaxy image generation, several architectures have been explored, often in conjunction with deblending and deconvolution tasks. For example, Ravanbakhsh et al. (2017) employed conditional VAE and GAN models, while (Schawinski et al. 2017; Fussell & Moews 2019; Hemmati et al. 2022) utilized GANs. Arcelin et al. (2020) adopted a VAE-based approach, whereas Lanusse et al. (2021) proposed a hybrid VAE-Normalizing Flow architecture. Recently, Smith et al. (2022) introduced a denoising stochastic diffusion model. These generative models aim to surpass parameterized analytical light profile simulations, such as those provided by `GalSim` (Rowe et al. 2015), by capturing complex structures and interactions between background signals and those of single or blended galaxies.

Despite the remarkable image quality achieved by modern generative models, which are widely accessible to the general public, several critical questions remain. These pertain to both mathematical foundations and practical applications. A key issue involves understanding what generative models truly learn and the statistical properties of their outputs. For example, Hataya et al. (2023) raised concerns about whether generated images could corrupt future datasets. Their study, however, focused on large-scale statistical models typically trained on billion-scale datasets sourced from the internet, which are prone to contamination by widely shared user-generated content. In contrast, galaxy image datasets remain relatively modest in size (i.e., $O(10^5)$), sourced from optical

* E-mail: jean-eric.campagne@ijclab.in2p3.fr
[1] Note: Updated in 2022.

surveys such as COSMOS (HST Advanced Camera for Surveys; Mandelbaum et al. (2019)) and the Sloan Digital Sky Survey (SDSS; York et al. 2000), specifically Data Release 7 (Abazajian et al. 2009). These datasets, used by a smaller community compared to social media, are less susceptible to such contamination.

To address the fidelity of generated galaxy images and their morphological properties relative to original datasets, Hackstein et al. (2023) and Janulewicz et al. (2024) have investigated various metrics. Furthermore, Kadkhodaie et al. (2024) tackled more mathematically oriented questions. Notably, they demonstrated the transition from memorization to generalization in diffusion generative models as dataset size increases. They also proposed an interpretation of what these networks learn, characterizing it as a form of *geometry-adaptive harmonic basis*, extending beyond the commonly used steerable wavelet basis.

To achieve this, Kadkhodaie et al. (2024) employed *denoiser* architectures, such as `U-Net` (Ronneberger et al. 2015) and `BF-CNN` networks (Mohan et al. 2020), trained on reduced datasets like `CelebA` (Liu et al. 2015) and `LSUN` Bedroom (Yu et al. 2015), comprising $O(10^5)$ grayscale images resized to 80×80 pixels. This dataset scale enables a systematic investigation of how various generative models, including normalizing flows, GANs, and denoiser-based diffusion models, perform in galaxy image generation when trained on the same dataset derived from the SDSS DR7 survey.

In the following sections, we begin by briefly describing various types of generative models, including Variational Auto-Encoders, Generative Adversarial Networks, Normalizing Flows, and Score-based Diffusion Models (Section 2). Next, we present the results of our numerical experiments (Section 3), which primarily involve pairs of models with identical architectures trained on non-overlapping datasets of the same size. This approach not only allow comparisons of morphological variable distributions, a common practice for validating the similarity between generated and real data samples using a single model, but also enables the implementation of consistency tests with two models. Finally, we provide our conclusions and discuss potential avenues for future research (Section 4). We provide the material to replay our experiment at the following location `https://github.com/jecampagne/galaxy-gen-model-compagnon`.

2 GENERATIVE MODELS

Generative models assume that observations $\boldsymbol{x} \in \mathbb{R}^d$ (where d represents the number of pixels times the number of channels in image modeling) are described by a probability distribution $p(\boldsymbol{x})$, and aim to approximate this distribution. It is further assumed that the dataset used for training consists of i.i.d. samples drawn from $p(\boldsymbol{x})$. In this section, we provide background knowledge on generative models rather than a detailed description of their architectures or implementations. We begin with the Variational Auto-Encoder (VAE), mainly for the sake of completeness and to introduce relevant terminology, although this architecture is not included in our current experiments (Section 3), which focus on GANs, normalizing flows, and score-based diffusion models. Details on some implementations used are described in Section 3.2.

2.1 Variational Auto-Encoder

The `VAE` architecture (Kingma & Welling 2014) assumes the existence of an underlying unobservable stochastic variable $\boldsymbol{z} \in \mathbb{R}^{d_\ell}$ (commonly referred to as a *latent variable*) whose distribution is selected *a priori* from a parameterized family $\pi_{\boldsymbol{\theta}_1}(\boldsymbol{z})$ that is differentiable with respect to both $\boldsymbol{z}$ and $\boldsymbol{\theta}_1$. Typically, the latent space has a much lower dimensionality than the data space, implying information compression ($d_\ell \ll d$). The data distribution is then expressed as $p(\boldsymbol{x}) = \int p(\boldsymbol{x}|\boldsymbol{z})\pi_{\boldsymbol{\theta}_1}(\boldsymbol{z})\, d\boldsymbol{z}$. However, the likelihood $p(\boldsymbol{x}|\boldsymbol{z})$ is generally unknown, necessitating the introduction of a parameterized approximation $p_{\boldsymbol{\theta}_2}(\boldsymbol{x}|\boldsymbol{z})$ with similar differentiability properties. Using the notation $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$, these parameters are optimized so that $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ aligns with the empirical distribution of the N training samples $\{\boldsymbol{x}^i\}_{i<N}$. The optimal parameters $\boldsymbol{\theta}_{ML}$ maximize $p_{\boldsymbol{\theta}}(\boldsymbol{x})$, but this requires computing the gradient of the integral

$$\int p_{\boldsymbol{\theta}_2}(\boldsymbol{x}|\boldsymbol{z})\pi_{\boldsymbol{\theta}_1}(\boldsymbol{z})\, d\boldsymbol{z}, \tag{1}$$

which is often intractable both analytically and numerically (Kingma & Welling 2014).

To address this challenge, the Bayes rule is employed:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x})p(\boldsymbol{z}|\boldsymbol{x}) = p_{\boldsymbol{\theta}_2}(\boldsymbol{x}|\boldsymbol{z})\pi_{\boldsymbol{\theta}_1}(\boldsymbol{z}), \tag{2}$$

along with a parameterized approximation $p_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ for the unknown true *a posteriori* distribution. Introducing the Kullback-Leibler (KL) divergence $\mathbb{D}_{\mathsf{KL}}(p\|q) = \mathbb{E}_{\boldsymbol{x}\sim p}[\log(p(\boldsymbol{x})/q(\boldsymbol{x}))]$, and taking the expected value according[2] to $\boldsymbol{z} \sim p_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$, we rewrite:

$$\begin{aligned}\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \log p_{\boldsymbol{\theta}_2}(\boldsymbol{x}|\boldsymbol{z}) + \log \pi_{\boldsymbol{\theta}_1}(\boldsymbol{z}) - \log p(\boldsymbol{z}|\boldsymbol{x}) \\ + \log p_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) - \log p_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}),\end{aligned} \tag{3}$$

which leads to:

$$\begin{aligned}\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{\phi}}}[\log p_{\boldsymbol{\theta}_2}(\boldsymbol{x}|\boldsymbol{z})] - \mathbb{D}_{\mathsf{KL}}(p_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\|\pi_{\boldsymbol{\theta}_1}(\boldsymbol{z})) \\ + \mathbb{D}_{\mathsf{KL}}(p_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\|p(\boldsymbol{z}|\boldsymbol{x})) \geq \mathcal{L}(\boldsymbol{x};\{\boldsymbol{\theta},\boldsymbol{\phi}\}),\end{aligned} \tag{4}$$

where we highlight the *evidence lower bound* (ELBO):

$$\mathcal{L}(\boldsymbol{x};\{\boldsymbol{\theta},\boldsymbol{\phi}\}) = \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{\phi}}}[\log p_{\boldsymbol{\theta}_2}(\boldsymbol{x}|\boldsymbol{z})] - \mathbb{D}_{\mathsf{KL}}(p_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\|\pi_{\boldsymbol{\theta}_1}(\boldsymbol{z})). \tag{5}$$

Maximizing the ELBO with respect to $\{\boldsymbol{\theta},\boldsymbol{\phi}\}$ is feasible because it eliminates dependence on the unknown distributions $p(\boldsymbol{x})$ and $p(\boldsymbol{z}|\boldsymbol{x})$. To mitigate the large variance of the gradient with respect to $\boldsymbol{\phi}$, Kingma & Welling (2014) introduced an auxiliary variable ε, replacing $\boldsymbol{z} \sim p_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ with $\boldsymbol{z} = g_{\boldsymbol{\phi}}(\boldsymbol{\varepsilon}, \boldsymbol{x})$, where $\boldsymbol{\varepsilon} \sim p(\boldsymbol{\varepsilon})$ and $g_{\boldsymbol{\phi}}$ is a differentiable function with respect to $\boldsymbol{\phi}$.

In this framework, $p_{\boldsymbol{\theta}_2}(\boldsymbol{x}|\boldsymbol{z})$ is typically referred to as the *decoder*, while $p_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ is the *encoder*. Different VAE models vary in their choice of distribution families. A common choice is Gaussian multivariate distributions, with a centered isotropic prior of unit variance $\mathcal{N}(\mathbf{0}, \mathbf{1})$ for $\boldsymbol{z}$ (where $\boldsymbol{\theta}_1$ is omitted in this scenario). Once optimization is complete, the *decoder* can generate $\boldsymbol{x}$ by sampling $\boldsymbol{z}$. However, simple VAEs often generate blurry images and may suffer from *posterior collapse*,

[2] We represent the sampling of variates $\boldsymbol{x}$ from a distribution $p(\boldsymbol{x})$ using the notation $\boldsymbol{x} \sim p(\boldsymbol{x})$.

motivating ongoing research (e.g., Engel et al. 2018; Takida et al. 2022) and modified VAE architectures, such as those proposed by Lanusse et al. (2021).

2.2 Generative Adversarial Network

The vanilla GAN (Goodfellow et al. 2014) shares with VAE the purpose of optimizing a *generator* network ($\boldsymbol{G}$) to obtain $\boldsymbol{x}$ samples from a latent variable $\boldsymbol{z}$ such that $\boldsymbol{x} = \boldsymbol{G}(\boldsymbol{z})$, with $\boldsymbol{z} \in \mathbb{R}^{d_\ell}$ (in general $d_\ell \ll d$). The *prior* $\pi(\boldsymbol{z})$ is typically a Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{1})$. A discussion on alternative choices of priors is presented by Brock et al. (2019b). Note that one can also use a prior fed by $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$, which is learned during the generator optimization.

To optimize the generator, a second network ($\boldsymbol{D}$), acting as a *discriminator*, aims to determine whether a sample $\boldsymbol{x}$ is from the model distribution or the dataset distribution. To achieve this, one solves the `min-max` problem:

$$\min_{\boldsymbol{G}} \max_{\boldsymbol{D}} \left\{ \mathbb{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}[\log \boldsymbol{D}(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim\pi(\boldsymbol{z})}[\log(1 - \boldsymbol{D}(\boldsymbol{G}(\boldsymbol{z})))] \right\}, \tag{6}$$

where p_{data} is the data-generating distribution. Both $\boldsymbol{G}$ and $\boldsymbol{D}$ are parameterized networks. Note that, at fixed parameters of $\boldsymbol{G}$, the optimal *discriminator* $\boldsymbol{D}^*$ has the following expression:

$$\boldsymbol{D}^*(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_{\boldsymbol{G}}(\boldsymbol{x})}, \tag{7}$$

with $p_{\boldsymbol{G}}(\boldsymbol{x}) = \pi(\boldsymbol{z}) |\det J_{\boldsymbol{G}}|^{-1}$, where $J_{\boldsymbol{G}}$ is the Jacobian of the *generator*. This leads to a reformulation of the `min-max` problem as follows:

$$\min_{G} \left\{ \mathbb{D}_{\mathsf{KL}}\left(p_{data} \,\middle\|\, \frac{p_{data} + p_{\boldsymbol{G}}}{2}\right) + \mathbb{D}_{\mathsf{KL}}\left(p_{\boldsymbol{G}} \,\middle\|\, \frac{p_{data} + p_{\boldsymbol{G}}}{2}\right) \right\}, \tag{8}$$

yielding the optimal solution $p_{\boldsymbol{G}} = p_{data}$, which perfectly matches the dataset distribution.

However, training the GAN model via the objective function above is generally unstable, leading to a pathology referred to in the literature as *mode-collapse*. This occurs because the *discriminator* can overfit the dataset too quickly, causing vanishing gradients (Gulrajani et al. 2017). This training instability has been extensively studied (see, e.g., reviews by Saxena & Cao 2021; Jozdani et al. 2022). Architectures like `BigGAN` (Brock et al. 2019b) and `StyleGAN2` (Karras et al. 2018b, 2020) address the gradient flow issue using various techniques, representing "state-of-the-art" improvements, albeit at the expense of computational resources.

Liu et al. (2021) addressed training stability and reduced computational resource requirements, especially for small datasets, by developing a `light-weight-GAN` structure. One key element is the hinge loss introduced by Lim & Ye (2017), which minimizes the following losses to alternately train the *discriminator*:

$$\min_{\boldsymbol{D}} \{ \mathbb{E}_{\boldsymbol{x}\sim p_{data}}[\max(0, 1 - \boldsymbol{D}(\boldsymbol{x}))] + \mathbb{E}_{\boldsymbol{z}\sim\pi(\boldsymbol{z})}[\max(0, 1 + \boldsymbol{D}(\boldsymbol{G}(\boldsymbol{z})))] \}, \tag{9}$$

and the *generator*:

$$\min_{\boldsymbol{G}} \{ -\mathbb{E}_{\boldsymbol{z}\sim\pi(\boldsymbol{z})}[\boldsymbol{D}(\boldsymbol{G}(\boldsymbol{z}))] \}. \tag{10}$$

Here, $\boldsymbol{D}$ is a linear discriminator, similar to those used in Support Vector Machines (Vapnik 1997). The second key ingredient is a strong regularization applied to the *discriminator* loss. The *discriminator* $\boldsymbol{D}$ is treated as an *encoder* (e.g., as in a VAE) and is trained alongside *decoders* to ensure that $\boldsymbol{D}$ extracts image features at different scales (e.g., 8×8, 16×16) that allow the *decoders* to reconstruct the images accurately. Finally, the *generator* employs a new skip-layer excitation module (SLE), which enables a more robust gradient flow between feature maps extracted at different scales.

GAN architectures have been used for galaxy generation in conjunction with deblending and deconvolution tasks. For instance, Schawinski et al. (2017) and Hemmati et al. (2022) employed vanilla-GANs, while Coccomini et al. (2021) utilized the `light-weight-GAN` that we use in our experiment too.

2.3 Normalizing Flows

Looking at Equation 4, one notices that the optimal solution is reached when $\mathbb{D}_{\mathsf{KL}}(p_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \| p(\boldsymbol{z}|\boldsymbol{x})) = 0$, which is true if and only if $p_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) = p(\boldsymbol{z}|\boldsymbol{x})$. However, this situation is generally unattainable because typical $p_{\boldsymbol{\phi}}$ parameterizations involve, for instance, independent Gaussian distributions. This limitation in parameterization is one of the drawbacks of VAEs. This motivates the use of normalizing flows (NF) to introduce more flexibility in the family of distributions, with the hope that the true posterior is a member of such a family (Tabak & Vanden-Eijnden 2010; Tabak & Turner 2013; Rezende & Mohamed 2015).

A flow T is defined as a diffeomorphism (i.e., a bijector) between the data space and a latent space, such that:

$$\boldsymbol{x} = T(\boldsymbol{z}) \quad \Leftrightarrow \quad \boldsymbol{z} = T^{-1}(\boldsymbol{x}). \tag{11}$$

If the distribution of $\boldsymbol{z}$ is $\pi(\boldsymbol{z})$ (e.g., $\mathcal{N}(\mathbf{0}, \mathbf{1})$), then the following relation applies:

$$p(\boldsymbol{x}) = \pi(\boldsymbol{z}) |\det J_T(\boldsymbol{z})|^{-1} = \pi(T^{-1}(\boldsymbol{x})) |\det J_{T^{-1}}(\boldsymbol{x})|, \tag{12}$$

where J_T and $J_{T^{-1}}$ are the Jacobians of the transformations T and T^{-1}, respectively. Unlike VAEs and GANs, the dimensionalities of the latent and data spaces are the same by definition in flow-based models.

The flow T is generally constructed as a composition of several individual flows $\{T_i\}_{i<n}$ such that:

$$T = T_1 \circ T_2 \circ \cdots \circ T_n \quad \Leftrightarrow \quad T^{-1} = T_n^{-1} \circ T_{n-1}^{-1} \circ \cdots \circ T_1^{-1}. \tag{13}$$

Figure 1 illustrates a schematic view of the *forward* or *generative* direction and the *backward/reverse* or *training* direction. If we denote $\boldsymbol{z}_i = T_i(\boldsymbol{z}_{i-1})$ for all $i < n$ (using $\boldsymbol{z}_0 = \boldsymbol{z}$ and $\boldsymbol{z}_n = \boldsymbol{x}$), then:

$$\log |\det J_T| = \sum_{i=0}^{n-1} \log |\det J_{T_i}(\boldsymbol{z}_{i-1})|. \tag{14}$$

Theorems exist that demonstrate, under moderate assumptions, flow-based models are capable of representing any density distribution (Bogachev et al. 2005; Huang et al. 2019). Consequently, if the flows are well-designed, one can sample $\boldsymbol{x}$ with potentially complex (multi-modal) density distributions from a simple spherical centered multivariate Gaussian distribution.

Let the flow be parametrized by a vector $\boldsymbol{\theta}$, leading to

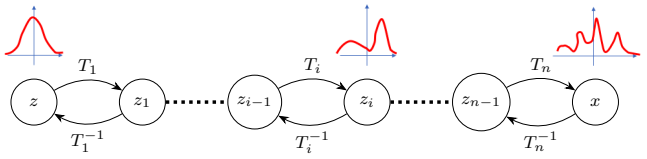


Figure 1. Schematic Normalizing Flow process. On the top the *forward* or *generative* direction from a simple $\boldsymbol{z}$ distribution to a more complex one for $\boldsymbol{x}$. On the bottom the *backward* or *training* direction from complex to simple distributions.

a parametrized flow model $p_{\boldsymbol{\theta}}(\boldsymbol{x})$. To optimize the flow, we consider the $\mathbb{D}_{\mathsf{KL}}$ divergence between $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ and the true $p(\boldsymbol{x})$, resulting in[3]:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}) &= \mathbb{D}_{\mathsf{KL}}(p(\boldsymbol{x})\|p_{\boldsymbol{\theta}}(\boldsymbol{x})) = -\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}[-\log p_{\boldsymbol{\theta}}(\boldsymbol{x})] + \text{const.} \\ &= -\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}[\log \pi(T_{\boldsymbol{\theta}}^{-1}(\boldsymbol{x})) + \log|\det J_{T_{\boldsymbol{\theta}}^{-1}}(\boldsymbol{x})|] + \text{const.}\end{aligned} \tag{15}$$

In practice, computing the expectation $\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}$ is not feasible and is replaced by a Monte Carlo approach using the data samples $\{\boldsymbol{x}^{(i)}\}_{i<N}$. Dropping the constant term, the loss function becomes:

$$\mathcal{L}(\boldsymbol{\theta}) = -\frac{1}{N}\sum_{i=0}^{N-1}\left\{\log \pi(T_{\boldsymbol{\theta}}^{-1}(\boldsymbol{x}^{(i)})) + \log|\det J_{T_{\boldsymbol{\theta}}^{-1}}(\boldsymbol{x}^{(i)})|\right\}. \tag{16}$$

The computation of the determinant of the Jacobian and its gradient can be challenging. However, it becomes more straightforward if the Jacobian matrices $\{J_{T_i}^{-1}\}_{i<n}$ (Eq. 14) are triangular. To minimize $\mathcal{L}(\boldsymbol{\theta})$ and obtain the best $\boldsymbol{\theta}$, we require T_θ^{-1} as well as its Jacobian and gradients. Additionally, T_θ is needed to generate new samples $\boldsymbol{x}$ from latent samples $\boldsymbol{z}\sim\pi(\boldsymbol{z})$. Various architectures of normalizing flows, such as *autoregressive flows*, enable such operations, as reviewed in Papamakarios et al. (2021).

The general schema of *autoregressive flows* is as follows: let $\boldsymbol{x},\boldsymbol{z}\in\mathbb{R}^d$, and denote $\boldsymbol{x}=(x_1,x_2,\ldots,x_d)$ and $\boldsymbol{z}=(z_1,z_2,\ldots,z_d)$. Then:

$$\forall i,\quad x_i = T(z_i;c_i)\quad\text{with}\quad c_i = C_i(z_1,\ldots,z_{i-1}) \triangleq C_i(z_{<i}), \tag{17}$$

where T is a strictly increasing monotonic transformation, and C_i is known as the *transformer*[4] and the *conditioner*.

The properties of such flows ensure the invertibility of the *transformer* and the triangular structure of the Jacobian:

$$J_T = \left[\frac{\partial x_i}{\partial z_j}\right] = \begin{cases} 0 & \text{for } i>j, \\ \frac{\partial x_i}{\partial z_i}(z_i;c_i) & \text{for } i=j. \end{cases} \tag{18}$$

This leads to:

$$\log|\det J_T| = \sum_{i=1}^{d}\log\left|\frac{\partial x_i}{\partial z_i}(z_i;c_i)\right|. \tag{19}$$

[3] For simplicity, we assume $\pi(\boldsymbol{z})=\mathcal{N}(\boldsymbol{0},\boldsymbol{1})$ without introducing additional unknown parameters.

[4] This term should not be confused with transformer networks used in deep learning architectures based on multi-head attention mechanisms (Vaswani et al. 2017).

Different types of transformers and conditioners yield distinct architectures.

Among the transformers, the class of affine transformations is perhaps the simplest. These are defined as:

$$T(z_i;c_i) = e^{\alpha_i}z_i + \beta_i,\quad\text{with}\quad c_i = (\alpha_i(z_{<i}),\beta_i(z_{<i})). \tag{20}$$

Using matrix notation, this can be expressed as $T(\boldsymbol{z}) = \exp(\boldsymbol{\alpha})\odot\boldsymbol{z}+\boldsymbol{\beta}$, where $\odot$ denotes the Hadamard product. The invertibility of T is guaranteed by the exponential function, and the Jacobian determinant simplifies to the e^{α_i} factor. Affine transformers are used in works such as (Dinh et al. 2015; Papamakarios et al. 2017; Dinh et al. 2017; Kingma & Dhariwal 2018). Furthermore, *spline*-based flows have gained traction, as exemplified in Crenshaw et al. (2024), where the `PZFlow` code is applied to model galaxy photometric redshift posterior distributions.

Concerning the conditioners C_i, in principle, they can be any functions (or models) that take $z_{<i}$ as input and output c_i. However, practical implementations are driven by computational cost. A particular approach is the *coupling layer*, implemented in `Real NVP`[5] as proposed in (Dinh et al. 2015, 2017). For an affine transformation T (Eq. 20) with $s<d$ (a common choice is $s=d/2$), the forward pass is defined as:

$$\text{(NVP)}\quad \begin{cases} x_{\leq s} & = z_{\leq s}, \\ x_{s+1,\ldots,d} & = z_{s+1,\ldots,d}\odot\exp(\boldsymbol{\alpha}(z_{\leq s})) + \boldsymbol{\beta}(z_{\leq s}), \end{cases} \tag{21}$$

where $\boldsymbol{\alpha}(z_{\leq s})$ and $\boldsymbol{\beta}(z_{\leq s})$ are functions of the first s components of $\boldsymbol{z}$.

The backward pass, involving the inverse transformation, has the same computational complexity—unlike other conditioners such as the masked schema used in (Germain et al. 2015)—and is given by:

$$\text{(NVP}^{-1}\text{)}\quad \begin{cases} z_{\leq s} & = x_{\leq s}, \\ z_{s+1,\ldots,d} & = (x_{s+1,\ldots,d} - \boldsymbol{\beta}(x_{\leq s}))\odot\exp(-\boldsymbol{\alpha}(x_{\leq s})). \end{cases} \tag{22}$$

The Jacobian matrix in this schema is triangular, with the diagonal given by:

$$\text{diag}(J_T) = (\boldsymbol{1}_s, \text{diag}(\exp(\boldsymbol{\alpha}(z_{\leq s})))), \tag{23}$$

where $\boldsymbol{1}_s$ is an $s\times s$ identity matrix. This leads to $d-s$ non-unity diagonal terms, reflecting the *non-volume conservation* property of the transformation (hence the "NVP" notation).

The structure of the transformation (e.g., Equation 21) involves a single split of the $\boldsymbol{z}$ vector coordinates into two parts. The first part, $z_{\leq s}$, is left untransformed, while the transformation of the second part, $z_{>s}$, depends only on $z_{\leq s}$. Notably, the conditioner functions $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ depend exclusively on $z_{\leq s}$ for both the forward and backward transformations. This single-split schema is illustrated in Figure 2. This splitting schema, implemented in a so-called *coupling layer*, makes the flow-based model computationally efficient. By using deep neural networks, one can achieve complex $(\boldsymbol{\alpha},\boldsymbol{\beta})$ functions. Designers of complete architectures can compose different *coupling layers*, alternating the order of $\boldsymbol{z}$ elements between layers to ensure that all elements are transformed by the

[5] n.b stands for *real-valued non-volume preserving* transformation.

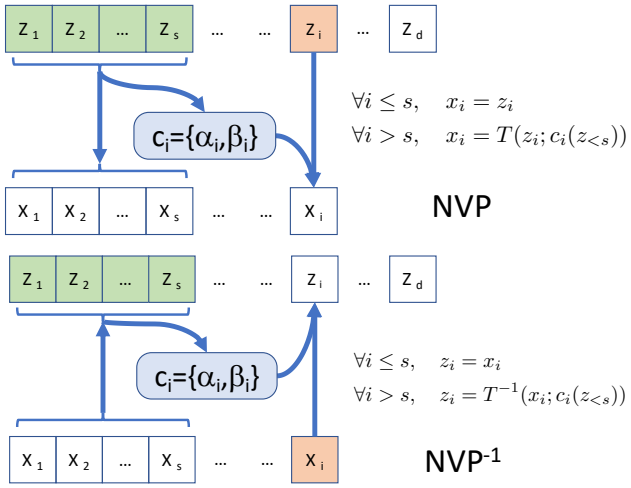
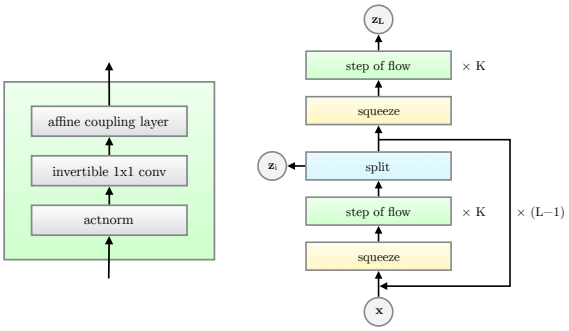


Figure 2. Affine coupling layer schema of forward and backward directions (Equations 21,22) with a single split mechanism (figure inspired by Papamakarios et al. (2021)).



(a) One step of the Glow flow.

(b) Multi-scale architecture (Dinh et al. 2017).

Figure 3. Copy of Figure 2 of (Kingma & Dhariwal 2018). The flow step consists of a scale and bias layer with data-dependent initialization (`actnorm`), a 1×1 invertible convolution layer, and an affine coupling layer where $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ are derived from a shallow convolutional neural network. Both $\boldsymbol{x}$ and $\boldsymbol{y}$ are tensors of shape $[h \times w \times c]$, where (h, w) are the spatial dimensions and c is the channel dimension. The indices (i, j) refer to spatial positions in tensors $\boldsymbol{x}$ and $\boldsymbol{y}$. This flow step is embedded within a multi-scale architecture consisting of K flow steps and L levels (Dinh et al. 2017). The squeezing operation transforms a tensor of shape $s \times s \times c$ into one of shape $s/2 \times s/2 \times 4c$, mixing spatial and channel components. At each scale, the channel image is divided into 2×2 patches ($s = 2$), and each patch undergoes the squeezing operation.

end of the sequence of operations. A generalization of this permutation can be achieved using a 1×1 convolution layer.

Dinh et al. (2017) improved the modeling further by introducing a multi-scale approach, which is too extensive to detail here. Kingma & Dhariwal (2018) utilized these extensions to build the `Glow` model, which is employed in our experiments. For completeness, the schema of this model is depicted in Figure 3 bored from Figure 2 of Kingma & Dhariwal (2018). In the `Glow` model, the prior is learned during the optimization of the flows, but to generate new samples, one still uses $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$, where $\boldsymbol{z}$ comprises tensors corresponding to the number of levels L.

2.4 Score-based diffusion models

To generate samples from $p(\boldsymbol{x})$, we can proceed as follows (e.g., Chang et al. 2023; Yang et al. 2023). Starting from a sample $\boldsymbol{x}_0$ assumed to be drawn from the distribution $p_0(\boldsymbol{x}) = p(\boldsymbol{x})$, we progressively transform it until the distribution is as simple as, for instance, $\mathcal{N}(\mathbf{0}, \mathbf{1})$. The idea is then to reverse this process (the *backward* mode) by generating a new sample from a new instance of noise. This transfer mapping is conceptually similar to normalizing flows. Their latent spaces share the same dimension as the data space, avoiding compression as seen in VAE and GAN architectures. However, the main distinction lies in normalizing flows being deterministic by nature, while *score-based diffusion* models are inherently stochastic.

Using the presentation framework of Kadkhodaie et al. (2024), the transport process in the *score-based diffusion* algorithm is modeled by the Ornstein-Uhlenbeck equation (Uhlenbeck & Ornstein 1930). The *forward* process follows this stochastic differential equation:

$$d\boldsymbol{x}_t = -\boldsymbol{x}_t\, dt + \sqrt{2}\, dB_t \quad t \in [0, T], \tag{24}$$

where dB_t represents a Wiener process (Brownian motion) and T is a finite endpoint assumed to be sufficiently large. The solution to this forward process is:

$$\boldsymbol{x}_t = \boldsymbol{x}_0 e^{-t} + \underbrace{(1 - e^{-2t})^{1/2}}_{\sigma_t} \boldsymbol{z}, \qquad \boldsymbol{z} \overset{\text{iid}}{\sim} \mathcal{N}(\mathbf{0}, \mathbf{1}). \tag{25}$$

The corresponding probability density function is expressed as:

$$p_t(\boldsymbol{x}_t) = \int p_t(\boldsymbol{x}_t, \boldsymbol{x}_0)\, d\boldsymbol{x}_0 = \int p_t(\boldsymbol{x}_t|\boldsymbol{x}_0) p_0(\boldsymbol{x}_0)\, dx_0, \tag{26}$$

where, based on the Fokker-Planck equation for the Ornstein-Uhlenbeck process:

$$p_t(\boldsymbol{x}_t|\boldsymbol{x}_0) = \frac{1}{(2\pi\sigma_t^2)^{d/2}} \exp\left\{-\frac{\|\boldsymbol{x}_t - \boldsymbol{x}_0 e^{-t}\|^2}{2\sigma_t^2}\right\}. \tag{27}$$

Thus, $p_t(\boldsymbol{x}_t)$ is the convolution of the initial distribution $p_0(\boldsymbol{x}_0)$ with a Gaussian of variance σ_t^2. This gradually blurs the distribution, leading $p_t(\boldsymbol{x}_t)$ to converge towards the normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{1})$.

In contrast to the VAE architecture, where the transformation from $\boldsymbol{x}_0$ to $\boldsymbol{z}$ is achieved via an *encoder* implemented by a deep neural network, the above stochastic process simply adds noise to the original signal (e.g., a galaxy image). Similarly, the VAE *decoder* or inverse flows are replaced in this model by a stochastic differential equation (damped Langevin dynamics):

$$d\boldsymbol{x}_{T-t} = (\boldsymbol{x}_{T-t} + 2\nabla_x \log p_{T-t}(\boldsymbol{x}_{T-t}))\, dt + \sqrt{2}\, dB_t \quad t \in [0, T], \tag{28}$$

where $s_t = \nabla_x \log p_{T-t}(\boldsymbol{x}_{T-t})$ represents the *score* of the probability density at time $T - t$ for the blurred image $\boldsymbol{x}_{T-t}$ under a white noise distribution with variance σ_{T-t}^2. This backward process constitutes the generative direction, enabling $\boldsymbol{x}$ samples to be drawn from $\mathcal{N}(\mathbf{0}, \mathbf{1})$.

The central challenge in generation lies in estimating the *score*. Two main approaches have been proposed: (1) deriving a model for p_t that incorporates regularity patterns and correlations to learn the *score* via *score matching* (Hyvärinen 2005), or (2) employing a *denoising* neural network, as introduced by Bengio et al. (2013), further developed in Sohl-Dickstein

et al. (2015); Ho et al. (2020), and recently examined from a mathematical perspective in Kadkhodaie et al. (2024). The first approach has been utilized in works such as Guth et al. (2022); Lempereur & Mallat (2024) with Gibbs energy parameterization. The second has found applications in astrophysics, including galaxy image generation (Smith et al. 2022), generation of 21 cm luminosity temperature maps (Zhao et al. 2023), super-resolution of large-scale cosmic structures (Schanz et al. 2023), and Bayesian posterior sampling for weak lensing mass-mapping problems (Remy, B. et al. 2023).

The connection between a perfect *denoiser* model and the *score* has been explored by numerous authors (e.g., Tweedie 1947; Herbert 1956; Miyasawa et al. 1961) and revisited by Raphan & Simoncelli (2011) for Gaussian white noise $\mathcal{N}(0, \sigma^2)$ independent of the noiseless signal $\boldsymbol{x}$. Irrespective of the prior on $\boldsymbol{x}$, the following theorem holds: the estimator $\tilde{\boldsymbol{x}}$ of the signal $\boldsymbol{x}$, given the noisy signal $\boldsymbol{x}_\sigma = \boldsymbol{x} + \boldsymbol{z}$ with $\boldsymbol{z} \sim \mathcal{N}(0, \sigma^2)$, is:

$$\tilde{\boldsymbol{x}} = \boldsymbol{x}_\sigma + \sigma^2 \nabla_x \log p(\boldsymbol{x}_\sigma). \tag{29}$$

Thus, the integration of a *denoiser* network within a generative model operates as follows. First, the network is trained in a straightforward manner to recover original images from their blurred versions, where the blur is induced by Gaussian white noise. The noise variance is not predetermined or explicitly provided to the network during training but spans the same range (e.g., $[0, 1]$) as the image values. During the backward stochastic process at each time step t, the *denoiser* is applied to obtain a denoised estimate ($\tilde{\boldsymbol{x}}$) of the image $\boldsymbol{x}_{T-t}$ ($\boldsymbol{x}_\sigma$), which has been blurred by white noise with a known variance $\sigma^2 = \sigma^2_{T-t}$. This process yields the score s_t thanks to Equation 29, enabling progression to the next step in the backward direction.

Denoising diffusion stochastic models have gained popularity due to their performance, which rivals that of GANs while offering distinct advantages. Notably, these models do not require adversarial networks, making them simpler to train. They are grounded in clear and robust mathematical principles, and their training process is theoretically well-defined. However, they rely on an iterative schema for generation, which can be computationally intensive and slower compared to GANs. Additionally, the implementation of stochastic processes may demand a certain level of theoretical expertise, although this requirement is not unique to diffusion models, as expertise is necessary for all advanced architectures.

Further discussions comparing GANs and diffusion models can be found in Dhariwal & Nichol (2021). We do not delve into the potential integration of flow-based and diffusion-based models (e.g., Zhang & Chen 2021; Gong & Li 2021) or the use of diffusion models to enhance GAN stability, as introduced by Wang et al. (2023). These topics present intriguing opportunities for exploration in future studies.

3 EXPERIMENT

In the previous section, we briefly outlined different architectures of generative models. Here, we describe the process of generating a sample $\boldsymbol{x}$, which typically[6] begins with sampling from a centered isotropic Gaussian distribution $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$ (i.e., white noise) of dimension d_ℓ. For VAE/GAN-based models, $d_\ell \ll d$, while $d = d_\ell$ for flow-based and diffusion-based models.

An interesting question arises: if we train two generative models of the same architecture and feed them with identical latent variables sampled from the same Gaussian distribution, will the generated images be identical? This could occur if the models have learned the same transformation process to map $\pi(\boldsymbol{z})$ to $p(\boldsymbol{x})$, or if they simply replicate images from the training dataset, which is not the intended purpose.

If the generated images differ, how can we evaluate the fidelity of the learning process? More critically, are we truly sampling from the probability density $p(\boldsymbol{x})$, or are the generated samples a mixture of the original dataset images?

A partial answer to these questions is provided in Kadkhodaie et al. (2024) within the context of diffusion-based models, highlighting the impact of dataset size on the transition from *memorization* to *generation* regimes. Following their methodology, we compare two generative models of the same architecture trained on non-overlapping subsets of the same dataset. For simplicity, we limit our experiment to three models and a single dataset, which we believe suffices to derive meaningful insights.

All experiments have been conducted mainly on a single Nvidia V100-32g GPU at Jean Zay supercomputer at IDRIS[7], a national computing centre for the France's National Centre for Scientific Research (CNRS).

3.1 The dataset

For training, Kadkhodaie et al. (2024) employed widely used machine learning datasets, including `CelebA` (Liu et al. 2015), `LSUN` Bedroom (Yu et al. 2015), and `CelebA HQ` (Karras et al. 2018a). All images were grayscale (single channel) with intensity values in the range $[0, 1]$. The `LSUN` images were downsampled to a resolution of 80×80 pixels, while `CelebA` images were resized to 32×32 and `CelebA HQ` images to 40×40 pixels. The transition from memorization to generalization was observed with a training dataset consisting of 10^5 images.

In the present study, we focus on galaxy image generation. Given the requirement to train models with at least 10^5 images, we used the Sloan Digital Sky Survey (SDSS) dataset described in (Smith et al. 2022)[8] with modifications to preprocessing. Specifically, all images were cropped around target galaxy coordinates to 64×64 pixels, and a single grayscale channel was created from the (z, g, r) channels using the `scipy` (Virtanen et al. 2020) library[9].

[6] This is the usual choice, but alternatives such as uniform distributions can also be considered; see Brock et al. (2019b) for a discussion in the context of GANs.
[7] http://www.idris.fr/eng/jean-zay/jean-zay-presentation-eng.html
[8] Raw data available at https://github.com/Smith42/astroddpm.
[9] The `make_lupton_rgb` function was applied with $Q = 8$ and `stretch`=0.2 settings.

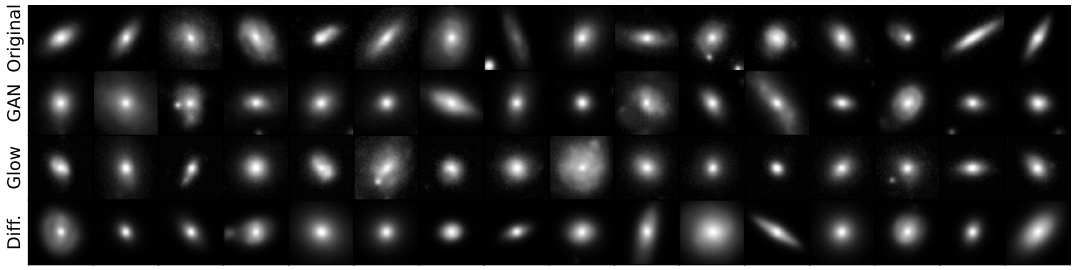


Figure 4. Examples of *validation* images from the original dataset (top row) and generated samples from different models trained with $N = 10^5$ images: the `light-weight-gan` GAN-based model (second row), the `Glow` flow-based model (third row), and the diffusion-based model with a `U-Net` denoiser (bottom row).

3.2 The models

We selected three architectures, as described in Section 2: a GAN, a flow-based model, and a diffusion-based model. We have used implementations of architectures written in `PyTorch` (Paszke et al. 2019).

For the GAN model, we used the `light-weight-gan` architecture[12]. Default settings were applied, including a latent space dimension of $d_\ell = 256$ leading to approximately 36 million parameters. Training involved resizing input images to 128×128 pixels and applying horizontal/vertical flips for data augmentation. The training process lasted about 30 hours over 100,000 iterations with a batch size of 10 images.

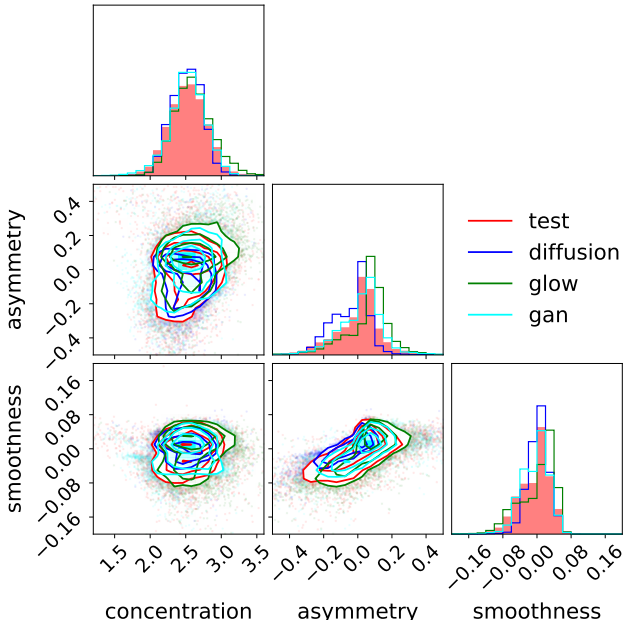
For the flow-based model, we used the `Glow` architecture[13] described in Section 2.3. The model configuration, shown in Figure 3, includes $K = 32$ flows and $L = 3$ blocks or levels by default[14], comprising 44 million parameters. Training followed default settings, with image pixel values reduced from 8-bit to 5-bit precision (Kingma & Dhariwal 2018). Optimization had required approximately 24 hours for about 200,000 iterations with a batch size of 32 images.

For the diffusion-based model, we adopted a `U-Net` architecture (Ronneberger et al. 2015) as the *denoiser*, leveraging its multi-scale structure with 7.6 million parameters[15]. Training followed the procedure in Kadkhodaie et al. (2024), with 100 epochs taking about 100 hours.

3.3 Results

3.3.1 Generated images

Figure 4 shows examples of galaxy images (*validation*) from the original dataset alongside samples generated by the three different models trained with $N = 10^5$ images. At first glance,



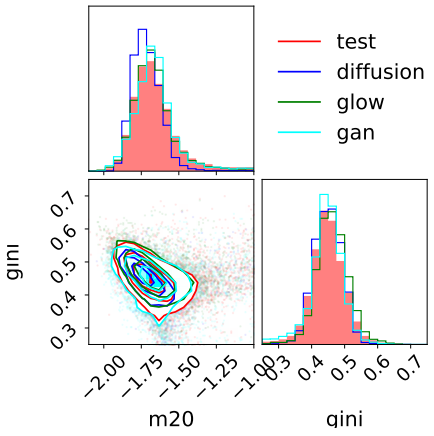


Figure 5. Corner plots[11] of morphological coefficients computed using either the validation dataset unused by any model trainings (*red, "test"*) or generated images from the diffusion-based model (*blue, "diffusion"*), the `Glow` flow-based model (*green, "glow"*), and the `light-weight-gan` GAN-based model (*cyan, "gan"*). All models were trained with $N = 10^5$ galaxy images and correspond to the results shown in Figure 4. The plot design is inspired by Hackstein et al. (2023).

[11] Corner plots generated using the library `https://corner.readthedocs.io` (Foreman-Mackey 2016).
[12] Source code: `https://github.com/lucidrains/lightweight-gan/` described in Section 2.2. Original repository: `https://github.com/odegeasslbc/FastGAN-pytorch/tree/main`.
[13] Source code: `https://github.com/rosinality/glow-pytorch`. Original repository: `https://github.com/openai/glow`.
[14] Different (K, L) values have been used with no significant differences with the default values.
[15] Source code: `https://github.com/LabForComputationalVision/memorization_generalization_in_diffusion_models`.

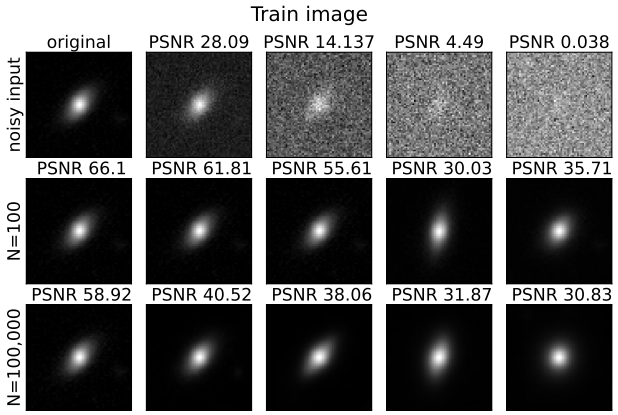


Figure 6. Denoising performances of two `U-Net` models on an image from the *training sample*. Top row: (left to right) the original image with progressively increasing noise added. The `PSNR` is defined as $10\log_{10}$ ratio of the squared dynamic range to the mean square error. Middle row: (left to right) denoised images and their PSNR values obtained using a `U-Net` trained with a dataset of size $N = 100$. Bottom row: same as the middle row but with a `U-Net` trained with a much larger dataset of $N = 100,000$ images.

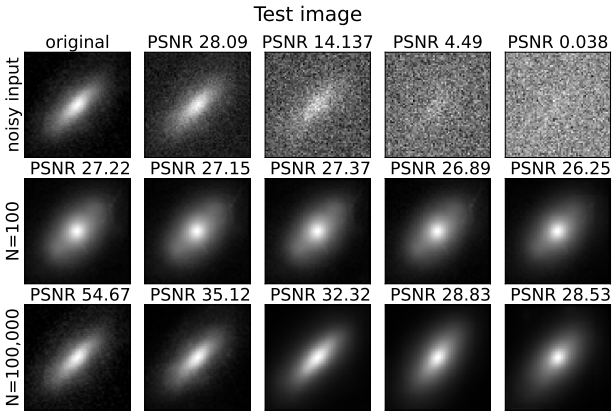


Figure 7. Similar to Figure 6, but applied to an image that was never used to train the networks (a *validation* image).

the generated images appear indistinguishable from the original images. To provide a more quantitative evaluation, Figure 5 presents various morphological coefficients computed using `Statmorph`[16] (Rodriguez-Gomez et al. 2019), including Concentration, Asymmetry, and Smoothness (Bershady et al. 2000; Conselice 2003; Lotz et al. 2004), as well as Gini and M_{20} coefficients (Lotz et al. 2004; Snyder et al. 2015).

The distributions of morphological coefficients derived from the three models show good agreement with those computed from the original images. However, we do not aim for high-quality images or state-of-the-art results, as seen in (Ravanbakhsh et al. 2017; Fussell & Moews 2019; Lanusse et al. 2021; Smith et al. 2022; Hackstein et al. 2023), which achieve better image quality. Instead, our focus is on a different question: with low-resolution and relatively simple images, do models of the same type trained on moderately sized datasets learn the same underlying probability distribution? Furthermore, if differences exist, can we identify measures to assess the reliability of the results?

3.3.2 `U-Net` Denoising Performances

Before assessing the generative performances of the models, we first present results regarding the denoising performances of two `U-Net` networks.

Figure 6 presents the denoising results of two networks trained on datasets of different sizes, using an original image shared by both training sets. In contrast, Figure 7 demonstrates the performance on an input image that was never included in the training data of either network, referred to as a *validation image*. For both figures, the peak signal-to-noise ratio (PSNR) relative to the original image is provided above each image. The top row shows the input images (original or noisy) fed to the denoiser. The middle row displays the outputs from the `U-Net` trained with $N = 100$ images, while the bottom row shows outputs from the `U-Net` trained with $N = 100,000$ images. It is important to note that the networks were trained using noisy images without access to PSNR values or noise variance information.

When using an image from the *training* dataset, both networks demonstrate similar denoising capabilities, as shown by the evolution of the output images' PSNR values relative to the PSNR of the noisy input images. However, a notable difference emerges when using the *validation image*. The `U-Net` trained with $N = 100,000$ images achieves results comparable to those observed with the training image, indicating good generalization performance. In contrast, the `U-Net` trained with $N = 100$ images shows limited generalization: the PSNR values of its output images remain nearly constant (PSNR ≈ 27), even for input images with very low noise levels.

This disparity in denoising performance between the two networks has significant implications for the generative process, which begins with a purely noisy image and progressively refines it into a noiseless output.

3.3.3 Two Diffusion-Based Models Test

Figure 8 explores the evolution of the cosine similarity metric (Tan et al. 2005), considering two distinct scenarios. The first scenario (top plots) involves $1,000$ samples generated by two diffusion-based models ("Model A" and "Model B") trained on independent datasets of the same size N, with generation *seeded by the same white noise images*. The second scenario (bottom plots) examines a single model, comparing each of its $1,000$ generated samples to the closest training image in terms of cosine similarity.

Although the training images are less complex than datasets like `CelebA` (faces) or `LSUN` (bedrooms), the following observations emerge:

- Low dataset size ($N = 100$): The generated samples are highly similar to the training images (cosine similarity close to 1), as indicated by the bottom-left histogram. However, samples generated from the same white noise image by the two models differ significantly, as the cosine similarity values are far from 1 (upper-left histogram).
- Larger dataset sizes: As the dataset size increases, generated samples become increasingly distinct from training images (progression from left to right histograms in the bottom

[16] https://github.com/vrodgom/statmorph

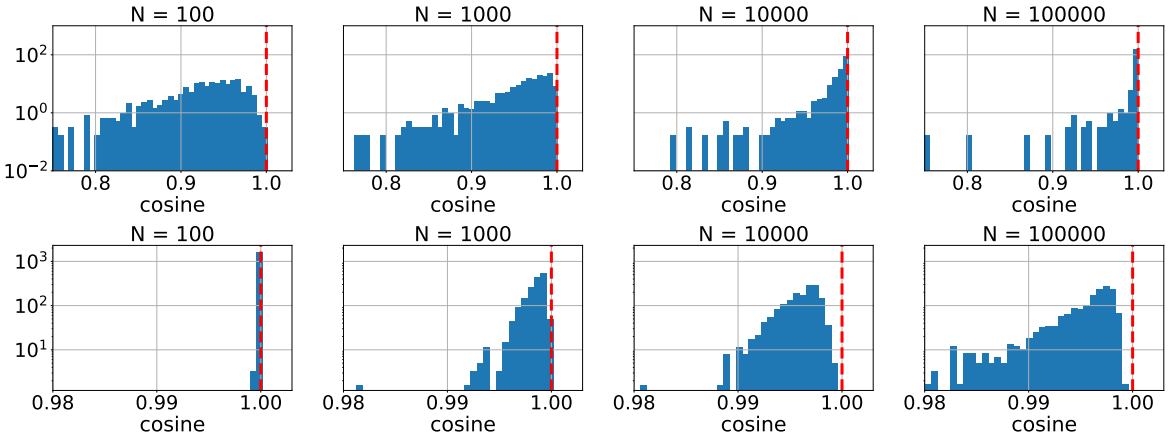


Figure 8. Evolution of the cosine similarity metric for diffusion-based models. Top row: normalized histograms of cosine similarity values between 1,000 samples generated by two models trained on independent datasets of size N, using the same white noise images as seeds. Bottom row: maximum cosine similarity between 1,000 generated samples from a single model and all images in its corresponding training dataset (i.e., the cosine of the closest training image for each generated sample). The logarithmic vertical scale and linear horizontal scale (cosine value) are consistent across all histograms on each row. The horizontal range in the bottom row is restricted to values near the maximum (cosine = 1), indicated by the dashed red vertical line. This figure is inspired by Figure 2 of Kadkhodaie et al. (2024).

row). Simultaneously, samples generated by the two models using the same noise seeds become more similar to each other (progression from left to right histograms in the top row). For $N = 100,000$, the two models produce almost identical outputs, though not perfectly (a single bin at cosine similarity of 1 has not yet been achieved).

These findings highlight two key behaviors: (i) At small dataset sizes, `U-Net`-based diffusion generative models operate in a *memorization regime*, where they predominantly replicate the training data; (ii) As the dataset size increases, the models transition into a *generation regime*, enabling them to generalize and produce novel images. These behaviors align with observations reported in Kadkhodaie et al. (2024) and are fundamentally tied to the denoising capabilities discussed in the preceding section.

Nonetheless, an examination of the morphological variable distributions (Figure 5) reveals no significant changes when the dataset size is reduced from 10^5 to 10^3 (Figure 9). However, reducing the training dataset size leads to generate samples that look similar to training images. This suggests that relying solely on morphological variable is insufficient to capture the profound differences between a model operating in a *memorization regime* and one functioning in a *generation regime*.

Interestingly, even for relatively simple, low-resolution galaxy images, approximately 10^5 samples are required to reliably achieve the transition into the *generation regime*. However, for practical applications, as emphasized in Kadkhodaie et al. (2024), producing higher-resolution images does not demand datasets that are orders of magnitude larger than those needed for 64×64 images. This is attributed to the hierarchical and multi-scale structure inherent to the images.

The "two-models test" conducted with diffusion-based architectures suggests that similar experiments could be useful to scrutinize what is learned by `Glow` flow-based and `light-weight-gan` GAN-based architectures.

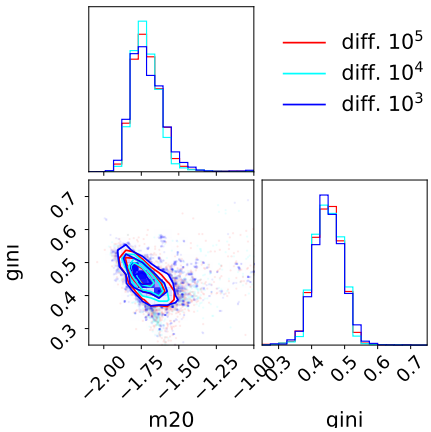


Figure 9. Some morphological coefficients (see Figure 5) computed using diffusion models trained with different dataset sizes ($N = 10^5, 10^4, 10^3$).

3.3.4 Flow-Based Models Tests

Following the two-models test described in Section 3.3.3, we trained two flow-based models ("A" and "B") with independent galaxy datasets of size $N = 10^5$. The cosine similarity results are shown in Figure 10. We can draw some key observations from these results:

- The generated samples appear distinct from any training image (or their augmented versions) as the right histogram does not peak at a cosine similarity of 1.
- Even when seeded with the same white noise, the images generated by the two models are different, as indicated by the absence of a peak at cosine similarity 1 in the left histogram.

This raises a fundamental question: what has been learned by the models? To further investigate, we utilize the bijective property of flow-based models (Section 2.3). This "inversion test" proceeds as follows:

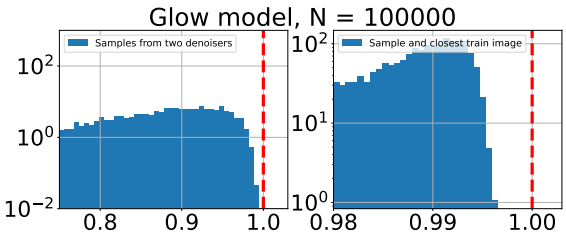


Figure 10. Histograms similar to those of Figure 8 for two `Glow` models trained on independent datasets of size $N = 100,000$ (identical to the datasets used by the diffusion models). Left: cosine similarity of $1,000$ samples generated by the two models seeded with the same white noise image. Right: maximum cosine similarity values of $1,000$ samples generated by one model compared to all training images and their augmented versions (horizontal, vertical, and horizontal+vertical flips), as these augmentations are part of the `Glow` training process.

- Sample $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$.
- Use Model "A" in *reverse/backward* mode to generate new samples $\boldsymbol{x}_A = G_A(\boldsymbol{z})$.
- Apply either Model "A" in *forward* mode to get $\boldsymbol{z}'_A = G_A^{-1}(\boldsymbol{x}_A)$, or similarly Model "B" to get $\boldsymbol{z}'_B = G_B^{-1}(\boldsymbol{x}_A)$.

The central question is whether $\boldsymbol{z}'_{\texttt{model}}$ (where `"model"` stands for "A" or "B") matches $\boldsymbol{z}$ in terms of probability density, and more strictly, whether $\boldsymbol{z}'_{\texttt{model}} = \boldsymbol{z}$ in a one-to-one correspondence. Figure 11 presents the results of the inversion test:

- In the left panel, $\boldsymbol{z}'_A = \boldsymbol{z}$ (blue dots), confirming the consistency of Model A with the flow property $G_A^{-1} \circ G_A = \mathbb{1}$. Conversely, $\boldsymbol{z}'_B$ and $\boldsymbol{z}$ are independent variables, indicating $G_B^{-1} \circ G_A \neq \mathbb{1}$.
- The right panel shows that for all levels of the `Glow` architecture, $\boldsymbol{z}'_B$ closely follows a $\mathcal{N}(\mathbf{0}, \mathbf{1})$ density distribution. While not perfect, $\boldsymbol{z}'_B$ approximately retains the same probability density as $\boldsymbol{z}$, which is a satisfactory outcome.

These results explain why models "A" and "B" cannot render the same image if they are fed with the same latent $\boldsymbol{z}$: they differ by a bijection from $\mathcal{N}(\mathbf{0}, \mathbf{1})$ to itself. In fact, flow-based models cannot guarantee more than this property. Fortunately, in this situation, both models account for the same sample probability density. However, there is a caveat: while the distributions of the latent variables match, this does not ensure that the output probability density matches the underlying density of the input dataset. To verify this, it is essential to analyze, for instance, the distributions of morphological variables as shown in Figure 5.

This raises the question: how does reducing the dataset size affect the results? To quantify this, we computed the 1-Wasserstein distance[17], denoted as $W_1(X, Y)$, for two sets of samples $X = \{x_i\}_i$ and $Y = \{y_j\}_j$. Here, X corresponds to the input z samples, while Y corresponds to the output z'_A or z'_B samples at all levels of the `Glow` architecture (denoted $W_1(\boldsymbol{z}'_A, \boldsymbol{z})$ and $W_1(\boldsymbol{z}'_B, \boldsymbol{z})$). Table 1 summarizes the results for varying dataset sizes ($N = 10^3, 10^4, 10^5$).

Decreasing the dataset size significantly degrades the results, as visually evident in Figures 12 and 13. This effect is particularly pronounced for the smallest dataset size, $N = 10^3$. However, even with $N = 10^4$, the morphological distributions deteriorate, despite the W_1 metrics showing relatively small changes compared to the $N = 10^5$ dataset results.

[17] Using the `wasserstein_distance` function from the `scipy` library.

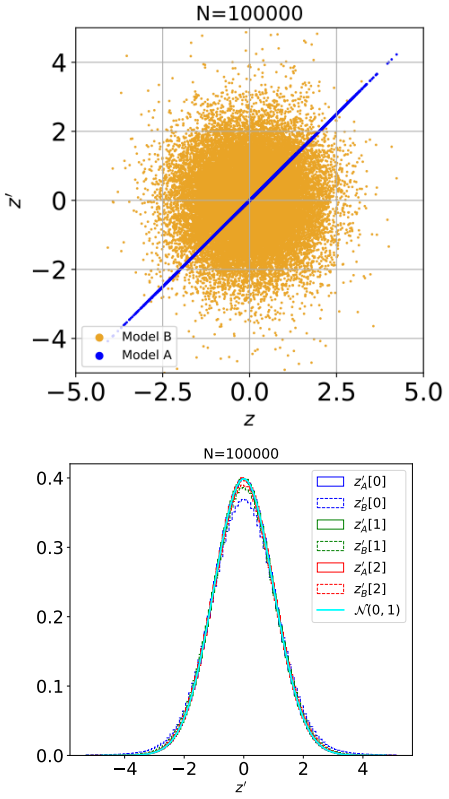


Figure 11. Results of the "inversion test" using $1,000$ samples $\boldsymbol{z}$ (see text). Top panel: Comparison of $z'_{\texttt{model}}$ vs. z at all levels of the `Glow` architecture (see Figure 3) for Model A (blue dots) and Model B (orange dots). Bottom panel: Distributions of $z'_{\texttt{model}}[\ell]$ for different levels $\ell = \{0, 1, 2 = L\}$, where solid (resp. dashed) curves correspond to Model A (resp. Model B). Both models were trained on independent datasets of 10^5 galaxy images.

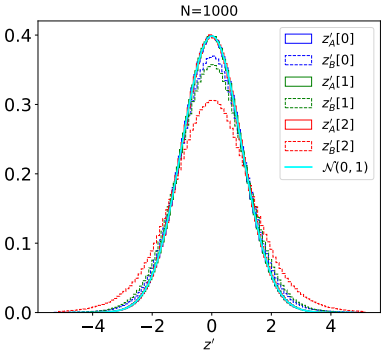


Figure 12. Same legend as bottom panel of Figure 11 but the models have been trained with independent datasets composed of 10^3 galaxy images. One notices that de z'_B distributions significantly tend to deviate from the z (i.e. $\mathcal{N}(\mathbf{0}, \mathbf{1})$) contrary to the z'_A distributions.

Table 1. Values of the 1-Wasserstein metric ($W_1(X,Y)$) computed for `Glow` models trained in the "two-models test" context with three different training dataset sizes. For $K = 32$ flows and $L = 3$ levels, and considering 1,000 generated images, one obtains 6×10^6 (resp. 3×10^6) z and z' samples at level 0 (resp. levels 1, 2). Using i.i.d. samples (X, Y) from $\mathcal{N}(0, 1)$ yields $W_1(X, Y) \approx 7 \times 10^{-4}$ (resp. $\approx 10^{-3}$) for level 0 (resp. levels 1, 2). The very low value of $W_1(z'_A, z) = O(10^{-6})$ is a direct consequence of the $z'_A = z$ property shown in the left panel of Figure 11.

	$N = 10^5$		$N = 10^4$		$N = 10^3$	
Level	z'_A vs z	z'_B vs z	z'_A vs z	z'_B vs z	z'_A vs z	z'_B vs z
0	3.7×10^{-6}	8.7×10^{-2}	2.3×10^{-6}	7.6×10^{-2}	2.6×10^{-6}	7.9×10^{-2}
1	2.6×10^{-6}	3.6×10^{-2}	4.0×10^{-6}	4.8×10^{-2}	2.7×10^{-6}	1.1×10^{-1}
2	4.1×10^{-6}	1.9×10^{-2}	3.9×10^{-6}	2.7×10^{-2}	5.1×10^{-6}	2.9×10^{-2}

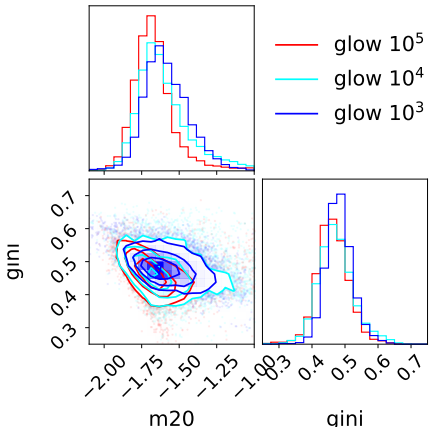


Figure 13. Some morphological coefficients (see Figure 5) computed using `Glow` models trained with different dataset sizes ($N = 10^5, 10^4, 10^3$). We can notice a degradation of the M_{20} distribution when the number of training images decreases.

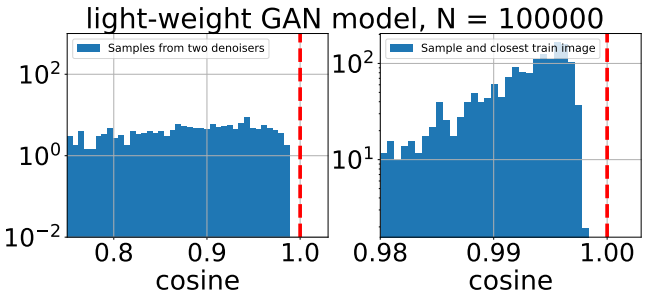


Figure 14. Histograms similar to those of Figures 8, 10 for two `light-weight` GAN models using two different datasets of size $N = 100,000$ (nb. they are the same as the ones used by the diffusion models). Left: the cosine similarity of 1,000 samples generated by the two models seeded by the same white noise image. Right: the maximum cosine similarity values of 1,000 samples generated by one of the models and the training images and their flipped versions (horizontal, vertical and horizontal+vertical) as this data augmentation is used in the training process.

3.3.5 GAN-based models tests

Following the tests done using the diffusion-based models (Section 3.3.3) and the flow-based models (Section 3.3.4), we trained two GAN-based models ("A" and "B") with independent galaxy datasets of size $N = 10^5$. The cosine similarity results are shown in Figure 14. The results closely resemble those obtained with flow-based models (Figure 10), exhibiting similar observations: the generated images differ from the training images, and the two models seeded with identical latent variables produce distinct samples. This raises a legitimate question: can we be confident in the model learning?

Although GAN-based models present a different structure compared to flow-based models, as the latent space and dataset space have different dimensionalities, the "inversion test" described in Section 3.3.4 cannot be applied directly. Instead, we propose an alternative approach based on discriminator performance (Section 2.2), hereafter referred to as the "discriminator test". According to Theorem 3.1 in Lim & Ye (2017), the hinge loss (Equation 9) should theoretically converge to a value of 2, provided that both the discriminator and generator are well optimized. This convergence indicates that the probability density of generated samples matches the empirical probability density. To evaluate this, we define the following loss function:

$$\ell(a, b) = \max(0, 1 - a) + \max(0, 1 + b) \tag{30}$$

where a and b are selected from the set $\{D_M(\boldsymbol{x}_M), D_M(\boldsymbol{x}_{M'}), D_M(\boldsymbol{x}_{\text{train}})\}$. Here, D_M represents the discriminator of model M (either "A" or "B"), $\boldsymbol{x}_M = G_M(\boldsymbol{z})$ are samples generated by the generator G_M using latent variables $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{1})$ (identical for both models), and $\boldsymbol{x}_{\text{train}}$ refers to training dataset samples. For example, we examine whether a sample $\boldsymbol{x}_A$ generated by G_A produces the same discriminator score when replaced by either $\boldsymbol{x}_B$ or $\boldsymbol{x}_{\text{train}}$.

Figure 15 shows the results of the "discriminator test" at various optimization stages (*epochs*) for models "A" and "B", both trained on datasets of size $N = 10^5$, using 1,000 samples of $\boldsymbol{z}$. Some key observations include:

- Initial state (*epoch* $= 0$): The discriminators D_A and D_B, along with their associated generators, are clearly under-optimized, as the mean values of ℓ are far from the theoretical target of 2. Additionally, for model "A", the distributions of $\ell(D_A(\boldsymbol{x}_A), D_A(\boldsymbol{x}_B))$ and $\ell(D_A(\boldsymbol{x}_A), D_A(\boldsymbol{x}_{\text{train}}))$ are identical, and a similar pattern is observed for model "B". However, the mean values not only differ from the target value, but they differ between the two models, likely due to the differences in their training datasets.
- Progressive optimization (increasing *epochs*): As training

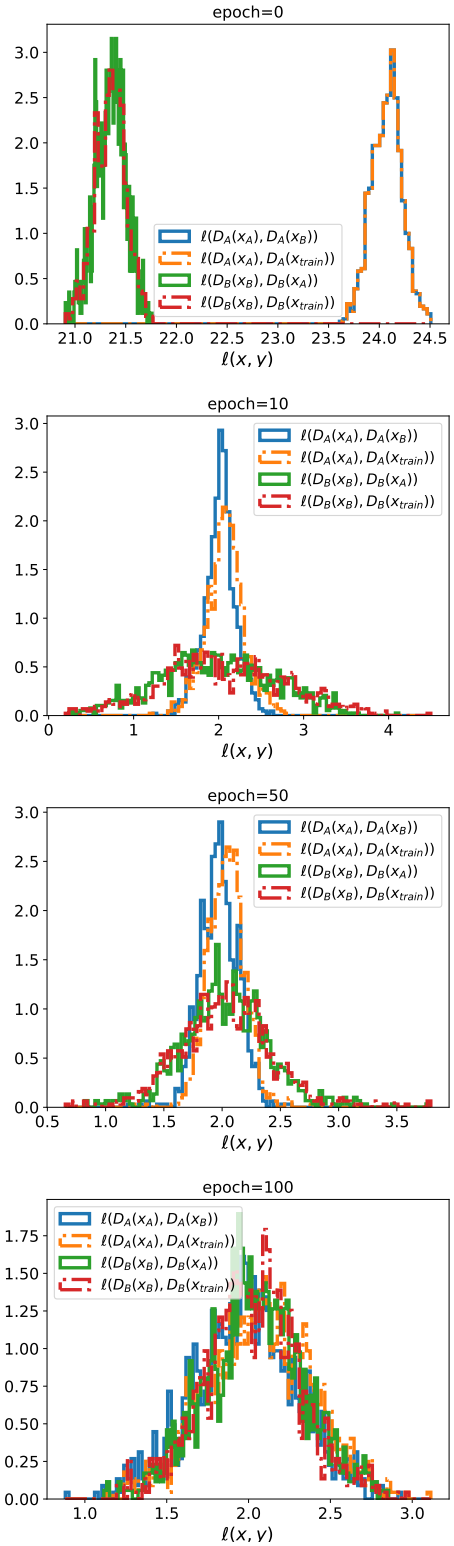


Figure 15. Evolution of the hinge loss (Equation 30) for the "discriminator test" across various training epochs for models "A" and "B". Here, $D_M(\boldsymbol{x}_{M'})$ represents the output of the discriminator of model M applied to samples generated by the generator of model M', while $\boldsymbol{x}_{\text{train}}$ refers to samples from the respective training dataset used for both models. All models were trained using datasets consisting of $N = 10^5$ images.

progresses, all loss distributions gradually converge to a single distribution with a mean value of 2. This indicates that the discriminators and generators of both models are reaching their optimal states. When the four losses converge, it signifies that the discriminator for either model "A" or "B" no longer distinguishes between (i) samples from its associated generator, (ii) samples from the other model's generator, and (iii) training samples.

Furthermore, we observe that the rate of convergence differs between the two models. If we focus solely on model "A", it might seem reasonable to stop optimization around *epoch* 10, where $\ell(D_A(\boldsymbol{x}_A), D_A(\boldsymbol{x}_B))$ and $\ell(D_A(\boldsymbol{x}_A), D_A(\boldsymbol{x}_{\text{train}}))$ have converged satisfactorily to the target value of 2. However, this would prematurely halt optimization, as the stronger convergence criterion—the alignment of all four loss distributions—emerges only around *epoch* 100. This highlights the utility of the two-models test in providing a more robust measure of convergence.

Regarding the impact of reducing the training dataset size below $N = 10^5$, we first analyze the morphological variables (Section 3.3.1). As observed in Figure 9 for diffusion-based models, Figure 16 illustrates that the distributions remain comparable across training dataset sizes $N = \{10^5, 10^4, 10^3\}$. This observation would likely lead to the conclusion that it is possible to achieve good sampling of the underlying probability density with models trained on small batches of images.

However, Figure 17 presents the results of the "discriminator test" for models trained with datasets of sizes $N = 10^4$ (panel (a)) and $N = 10^3$ (panel (b)), evaluated at *epoch* 100. Readers are encouraged to compare these findings with the corresponding panel in Figure 15. When training with $N = 10^4$ or $N = 10^3$, the results indicate that the discriminators are not entirely agnostic to the origin of the image samples, as evidenced by the distinct distributions of $\ell(D_A(\boldsymbol{x}_A), D_A(\boldsymbol{x}_B))$ and $\ell(D_A(\boldsymbol{x}_A), D_A(\boldsymbol{x}_{\text{train}}))$ (and similarly for the D_B discriminator). Moreover, the discrepancies in these loss distributions persist even when training continues beyond *epoch* 100, suggesting that the observed divergences are not merely attributable to early convergence issues. It could be argued that these inconsistencies may arise from suboptimal hyperparameter choices in designing the `light-weight-gan` architecture used for the experiment, rather than solely from the limited dataset size. Notably, in the absence of convergence across the four loss distributions, there is no *a priori* method to guide further investigations or remedial measures effectively.

4 CONCLUSION

In this study, we investigated the behavior of generative models applied to galaxy image generation, focusing on their performance under controlled experimental conditions. We compared three architectures: the `light-weight-gan` GAN-based model, the `Glow` flow-based model, and the diffusion model based on a `U-Net` denoiser (Section 3.2). These models were trained and evaluated using identical datasets sourced from the SDSS DR7 survey (Section 3.1).

Even with relatively small datasets, the generated images from all models appeared visually indistinguishable from the original ones (Figure 4). However, this study did not aim to achieve the highest possible image quality, as accomplished in prior works mentioned in the introduction. Instead, we focused

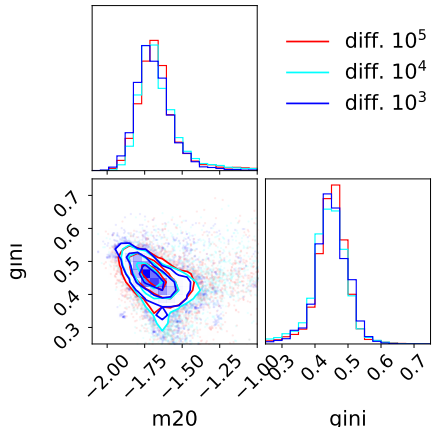
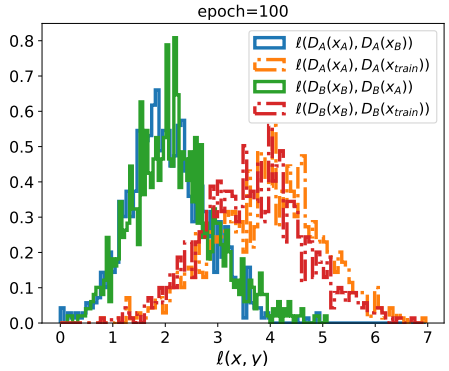
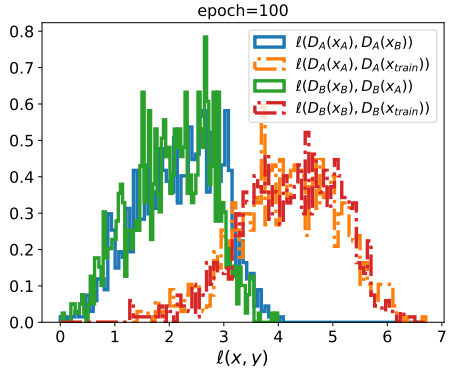


Figure 16. Some morphological coefficients (see Figure 5) computed using `light-weight-gan` models trained with different dataset sizes ($N = 10^5, 10^4, 10^3$).



(a) $N = 10^4$



(b) $N = 10^3$

Figure 17. Same legend as for Figure 15 but with different dataset sizes: $N = 10^4$ (a) and $N = 10^3$ (b).

on a different question: do models of the same type learn the same underlying probability distribution? The similarity in morphological features (Figure 5)—used to assess whether the generated images are samples from the underlying data probability distribution function (pdf)—suggests that this is indeed the case. The main objective of this article was to approach this question from a novel perspective, revisiting the reliability of the generation processes. To this end, we designed experiments (Section 3) involving pairs of models trained on non-overlapping subsets of the data and seeded by identical latent variables, inspired by the methodology of Kadkhodaie et al. (2024).

For the diffusion-based model (Section 3.3.3), our study of the cosine similarity metric between images revealed a transition from *memorization* to *generation* as the dataset size increased. This transition, rooted in the denoiser's capabilities, aligns with findings from Kadkhodaie et al. (2024). Small datasets led to models that primarily memorized the training data, while larger datasets enabled generalization and the creation of novel images. This transition is essential for understanding the limits of such generative models and their ability to produce novel outputs. However, morphological distributions alone were insufficient to fully capture the nuances of this transition. It is worth noting that the forward and reverse processes in diffusion models are theoretically grounded in the Ornstein-Uhlenbeck equation and damped Langevin dynamics, adding robustness to their probabilistic framework.

For the flow-based `Glow` model (Section 3.3.4), while it generated samples with good morphological features, the cosine similarity metric indicated that the outputs were distinct from the training images, a positive outcome. However, two models trained on non-overlapping datasets of the same size produced different samples. Leveraging the bijective property of flow-based models, we proposed a consistency test called the "inversion test". Specifically, given two models "A" and "B," and a latent variables $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$, we investigated whether latent variables $\boldsymbol{z}' = G_B^{-1}(G_A(\boldsymbol{z}))$ were identical to $\boldsymbol{z}$ or at least shared the same pdf. Our findings showed that the latter was true only when the dataset size was sufficiently large, highlighting challenges faced by normalizing flows in capturing the true underlying data distribution with small datasets. This issue was also reflected in the degradation of morphological features with smaller dataset sizes.

For the `light-weight-gan` model (Section 3.3.5), the cosine similarity results were comparable to those of the flow-based model, but the GAN architecture lacks bijectivity. This distinction is inherent to the design of GANs. Nonetheless, we proposed a consistency test based on the hinge loss in the context of the "two-models" scenario. Using losses such as $\ell(D_M(\boldsymbol{x}_M), D_M(\boldsymbol{x}_{M'}))$ and $\ell(D_M(\boldsymbol{x}_M), D_M(\boldsymbol{x}_{\rm train}))$, where D_M is the discriminator of model M and $\boldsymbol{x}$ are either generated by a model or issued from the training datasets, we assessed whether the generators successfully learned the underlying data distribution. As training progressed, the "discriminator test" showed convergence of the mean loss to the theoretical value of 2, suggesting successful learning. However, discrepancies in the discriminator loss distributions, particularly with smaller datasets, revealed differences in generalization capabilities. Models trained on smaller datasets exhibited poorer generalization, even though their morphological features remained consistent, emphasizing the need for evaluations beyond morphological distributions.

In conclusion, while modern generative models can produce high-quality galaxy images with reasonably sized datasets, much remains to be explored regarding the mechanisms governing their learning processes and the factors influencing their generalization. The "two-models" methodology proved to be a valuable approach, allowing us to assess both the similarity of

generated images to real-world data and the consistency of the learned probability distributions. This framework represents a key takeaway of this study to study generative modelling. The methods developed here are certainly not limited to generating images of galaxies.

ACKNOWLEDGEMENTS

We thank Prafulla Dhariwal who have accepted the reproduction of Figure 2 of Kingma & Dhariwal (2018) (see Figure 3). We thank Kim Seonghyeon for fruitful discussion on `Glow` model implementation. We acknowledge the use of Nvidia's V100 resources from French GENCI–IDRIS (Grant 2024-AD010413957R1).

DATA AVAILABILITY

Codes, trained models and results, and the complete dataset of 250,000 galaxy images are publicly available on request to the corresponding author. A GitHub repository is also accompanying this article, see details at https://github.com/jecampagne/galaxy-gen-model-compagnon.

REFERENCES

Abazajian K. N., et al., 2009, The Astrophysical Journal Supplement Series, 182, 543

Arcelin B., Doux C., Aubourg E., Roucelle C., Collaboration) T. L. D. E. S., 2020, Monthly Notices of the Royal Astronomical Society, 500, 531

Bengio Y., Yao L., Alain G., Vincent P., 2013, in Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1. NIPS'13. Curran Associates Inc., Red Hook, NY, USA, p. 899–907

Bershady M. A., Jangren A., Conselice C. J., 2000, AJ, 119, 2645

Bogachev V. I., Kolesnikov A. V., Medvedev K. V., 2005, Sbornik: Mathematics, 196, 309

Brock A., Donahue J., Simonyan K., 2019a, in International Conference on Learning Representations. https://openreview.net/forum?id=B1xsqj09Fm

Brock A., Donahue J., Simonyan K., 2019b, in 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net, https://openreview.net/forum?id=B1xsqj09Fm

Chang Z., Koulieris G. A., Shum H. P. H., 2023, arXiv e-prints, p. arXiv:2306.04542

Coccomini D., Messina N., Gennaro C., Falchi F., 2021, arXiv e-prints, p. arXiv:2111.11578

Conselice C. J., 2003, ApJS, 147, 1

Crenshaw J. F., Kalmbach J. B., Gagliano A., Yan Z., Connolly A. J., Malz A. I., Schmidt S. J., Collaboration T. L. D. E. S., 2024, The Astronomical Journal, 168, 80

Dhariwal P., Nichol A., 2021, Advances in neural information processing systems, 34, 8780

Dinh L., Krueger D., Bengio Y., 2015, in Bengio Y., LeCun Y., eds, 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings. http://arxiv.org/abs/1410.8516

Dinh L., Sohl-Dickstein J., Bengio S., 2017, in 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net, https://openreview.net/forum?id=HkpbnH9lx

Engel J., Hoffman M., Roberts A., 2018, in International Conference on Learning Representations. https://openreview.net/forum?id=Sy8XvGb0-

Foreman-Mackey D., 2016, The Journal of Open Source Software, 1, 24

Fussell L., Moews B., 2019, Monthly Notices of the Royal Astronomical Society, 485, 3203

Germain M., Gregor K., Murray I., Larochelle H., 2015, in Bach F., Blei D., eds, Proceedings of Machine Learning Research Vol. 37, Proceedings of the 32nd International Conference on Machine Learning. PMLR, Lille, France, pp 881–889, https://proceedings.mlr.press/v37/germain15.html

Gong W., Li Y., 2021, in ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models. https://openreview.net/forum?id=jxsmOXCDv9l

Goodfellow I., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley D., Ozair S., Courville A., Bengio Y., 2014, Advances in neural information processing systems, 27

Gulrajani I., Ahmed F., Arjovsky M., Dumoulin V., Courville A., 2017, in Proceedings of the 31st International Conference on Neural Information Processing Systems. NIPS'17. Curran Associates Inc., Red Hook, NY, USA, p. 5769–5779

Guth F., Coste S., De Bortoli V., Mallat S., 2022, Advances in neural information processing systems, 35, 478

Hackstein S., Kinakh V., Bailer C., Melchior M., 2023, Astronomy and Computing, 42, 100685

Hataya R., Bao H., Arai H., 2023, in 2023 IEEE/CVF International Conference on Computer Vision (ICCV). IEEE Computer Society, Los Alamitos, CA, USA, pp 20498–20508, doi:10.1109/ICCV51070.2023.01879, https://doi.ieeecomputersociety.org/10.1109/ICCV51070.2023.01879

Hemmati S., et al., 2022, The Astrophysical Journal, 941, 141

Herbert R., 1956, in Proceedings of the third berkeley symposium on mathematical statistics and probability. pp 157–163

Ho J., Jain A., Abbeel P., 2020, in Proceedings of the 34th International Conference on Neural Information Processing Systems. NIPS '20. Curran Associates Inc., Red Hook, NY, USA

Huang C.-W., Dinh L., Courville A., 2019, in ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations. https://openreview.net/forum?id=cfKpOiUzF

Hyvärinen A., 2005, Journal of Machine Learning Research, 6, 695

Janulewicz P., Perreault-Levasseur L., Webb T., 2024, in ICML 2024 Workshop on Structured Probabilistic Inference & Generative Modeling. https://openreview.net/forum?id=lzLMJ6KkiS

Jozdani S., Chen D., Pouliot D., Alan Johnson B., 2022, International Journal of Applied Earth Observation and Geoinformation, 108, 102734

Kadkhodaie Z., Guth F., Simoncelli E. P., Mallat S., 2024, in The Twelfth International Conference on Learning Representations. https://openreview.net/forum?id=ANvmVS2Yr0

Kansal R., Li A., Duarte J., Chernyavskaya N., Pierini M., Orzari B., Tomei T., 2023, Phys. Rev. D, 107, 076017

Karras T., Aila T., Laine S., Lehtinen J., 2018a, in 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings. OpenReview.net, https://openreview.net/forum?id=Hk99zCeAb

Karras T., Laine S., Aila T., 2018b, 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp 4396–4405

Karras T., Laine S., Aittala M., Hellsten J., Lehtinen J., Aila T., 2020, in 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp 8107–8116, doi:10.1109/CVPR42600.2020.00813

Kingma D. P., Dhariwal P., 2018, in Bengio S., Wallach H., Larochelle H., Grauman K., Cesa-Bianchi N., Garnett R., eds, NIPS '18 Vol. 31, Advances in Neural Information Processing Systems. Curran Associates,

Inc., https://proceedings.neurips.cc/paper_files/paper/2018/file/d139db6a236200b21cc7f752979132d0-Paper.pdf
Kingma D. P., Welling M., 2014, in 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings. (http://arxiv.org/abs/1312.6114)
Lanusse F., Mandelbaum R., Ravanbakhsh S., Li C.-L., Freeman P., Póczos B., 2021, Monthly Notices of the Royal Astronomical Society, 504, 5543
Lempereur E., Mallat S., 2024, arXiv e-prints, p. arXiv:2405.03468
Lim J. H., Ye J. C., 2017, arXiv e-prints, p. arXiv:1705.02894
Liu Z., Luo P., Wang X., Tang X., 2015, in 2015 IEEE International Conference on Computer Vision (ICCV). pp 3730–3738, doi:10.1109/ICCV.2015.425
Liu B., Zhu Y., Song K., Elgammal A., 2021, in International Conference on Learning Representations. https://openreview.net/forum?id=1Fqg133qRaI
Lotz J. M., Primack J., Madau P., 2004, AJ, 128, 163
Mandelbaum R., Lackner C., Leauthaud A., Rowe B., 2019, COSMOS real galaxy dataset, doi:10.5281/zenodo.3242143, https://doi.org/10.5281/zenodo.3242143
Miyasawa K., et al., 1961, Bull. Inst. Internat. Statist, 38, 1
Mohan S., Kadkhodaie Z., Simoncelli E. P., Fernandez-Granda C., 2020, in International Conference on Learning Representations. https://openreview.net/forum?id=HJlSmC4FPS
Oppenlaender J., 2022, in Proceedings of the 25th International Academic Mindtrek Conference. Academic Mindtrek '22. Association for Computing Machinery, New York, NY, USA, p. 192–202, doi:10.1145/3569219.3569352, https://doi.org/10.1145/3569219.3569352
Papamakarios G., Pavlakou T., Murray I., 2017, in Proceedings of the 31st International Conference on Neural Information Processing Systems. NIPS'17. Curran Associates Inc., Red Hook, NY, USA, p. 2335–2344
Papamakarios G., Nalisnick E., Rezende D. J., Mohamed S., Lakshminarayanan B., 2021, Journal of Machine Learning Research, 22, 1
Paszke A., et al., 2019, arXiv e-prints, p. arXiv:1912.01703
Ramesh A., Dhariwal P., Nichol A., Chu C., Chen M., 2022, arXiv e-prints, p. arXiv:2204.06125
Raphan M., Simoncelli E. P., 2011, Neural Comput., 23, 374–420
Ravanbakhsh S., Lanusse F., Mandelbaum R., Schneider J., Poczos B., 2017, in Proceedings of 31st AAAI Conference on Artificial Intelligence (AAAI '17). pp 1488 – 1494 (arXiv:1609.05796)
Remy, B. Lanusse, F. Jeffrey, N. Liu, J. Starck, J.-L. Osato, K. Schrabback, T. 2023, A&A, 672, A51
Rezende D. J., Mohamed S., 2015, in Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37. ICML'15. JMLR.org, p. 1530–1538
Rodriguez-Gomez V., et al., 2019, MNRAS, 483, 4140
Rombach R., Blattmann A., Lorenz D., Esser P., Ommer B., 2022, in 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp 10674–10685, doi:10.1109/CVPR52688.2022.01042
Ronneberger O., Fischer P., Brox T., 2015, in Navab N., Hornegger J., Wells W. M., Frangi A. F., eds, Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. Springer International Publishing, Cham, pp 234–241
Rowe B., et al., 2015, Astronomy and Computing, 10, 121
Saxena D., Cao J., 2021, ACM Comput. Surv., 54
Schanz A., List F., Hahn O., 2023, arXiv e-prints, p. arXiv:2310.06929
Schawinski K., Zhang C., Zhang H., Fowler L., Santhanam G. K., 2017, Monthly Notices of the Royal Astronomical Society: Letters, 467, L110
Smith M. J., Geach J. E., Jackson R. A., Arora N., Stone C., Courteau S., 2022, Monthly Notices of the Royal Astronomical Society, 511, 1808
Snyder G. F., et al., 2015, Monthly Notices of the Royal Astronomical Society, 454, 1886
Sohl-Dickstein J., Weiss E., Maheswaranathan N., Ganguli S., 2015, in Bach F., Blei D., eds, Proceedings of Machine Learning Research Vol. 37, Proceedings of the 32nd International Conference on Machine Learning. PMLR, Lille, France, pp 2256–2265, https://proceedings.mlr.press/v37/sohl-dickstein15.html
Tabak E. G., Turner C. V., 2013, Communications on Pure and Applied Mathematics, 66, 145
Tabak E. G., Vanden-Eijnden E., 2010, Communications in Mathematical Sciences, 8, 217
Takida Y., Liao W.-H., Lai C.-H., Uesaka T., Takahashi S., Mitsufuji Y., 2022, Neurocomput., 509, 137–156
Tan P.-N., Steinbach M. S., Kumar V., 2005, Introduction to Data Mining. Addison-Wesley
Tweedie M. C. K., 1947, Mathematical Proceedings of the Cambridge Philosophical Society, 43, 41–49
Uhlenbeck G. E., Ornstein L. S., 1930, Phys. Rev., 36, 823
Vapnik V. N., 1997, in Gerstner W., Germond A., Hasler M., Nicoud J.-D., eds, Artificial Neural Networks — ICANN'97. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 261–271
Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser L., Polosukhin I., 2017, in Proceedings of the 31st International Conference on Neural Information Processing Systems. NIPS'17. Curran Associates Inc., Red Hook, NY, USA, p. 6000–6010
Virtanen P., et al., 2020, Nature Methods, 17, 261
Wang Z., Zheng H., He P., Chen W., Zhou M., 2023, in The Eleventh International Conference on Learning Representations. https://openreview.net/forum?id=HZf7UbpWHuA
Yang L., et al., 2023, ACM Comput. Surv., 56
York D. G., et al., 2000, AJ, 120, 1579
Yu F., Seff A., Zhang Y., Song S., Funkhouser T., Xiao J., 2015, arXiv e-prints, p. arXiv:1506.03365
Zhang Q., Chen Y., 2021, in Beygelzimer A., Dauphin Y., Liang P., Vaughan J. W., eds, Advances in Neural Information Processing Systems. https://openreview.net/forum?id=x1Lp2bOlVIo
Zhao X., Ting Y.-S., Diao K., Mao Y., 2023, MNRAS, 526, 1699

This paper has been typeset from a TEX/LATEX file prepared by the author.