Learning multi-modal distributions and image editing with deep generative models

Thibaut Issenhuth

École des Ponts ParisTech

THÈSE DE DOCTORAT
de l'École des Ponts ParisTech

Learning Multi-Modal Distributions and Image Editing with Deep Generative Models

École doctorale MSTIC N°532

Informatique

Thèse CIFRE préparée au sein de Criteo AI Lab et du LIGM-IMAGINE, École des Ponts ParisTech

Thèse soutenue le 26 Juin 2023, par

Thibaut Issenhuth

Composition du jury:

Julie Delon Université Paris Cité	*Présidente*
Matthieu Cord Sorbonne Université	*Rapporteur*
Rémi Flamary École Polytechnique	*Rapporteur*
Jia-Bin Huang University of Maryland / Meta	*Examinateur*
Jakob Verbeek Meta	*Examinateur*
Vicky Kalogeiton École Polytechnique	*Examinatrice*
David Picard École des Ponts ParisTech	*Directeur de thèse*
Jérémie Mary Criteo AI Lab	*Co-directeur de thèse*
Ugo Tanielian Criteo AI Lab	*Invité*

Abstract

The past years have seen a great progress of deep generative models, including Generative Adversarial Networks (GANs). Notably, they can synthesize high-resolution images, sometimes indistinguishable from real images. Deep generative models were also at the root of empirical successes such as music generation or molecular discovery. However, we lack a fundamental understanding of the capabilities and limitations of deep generative models. In this thesis, we first characterize a model misspecification of generative models with connected output distributions, such as GANs or normalizing flows. Indeed, such models can not perfectly fit a target distribution composed of several disconnected modes. We analyse theoretically their best achievable performance in the setting of disconnected target distribution, and which geometrical structure can allow them to achieve best performance. Moreover, we propose methods that improve the performance of GANs by making them more amenable for disconnected data modelling.

In the second part of the thesis, we aim to improve image editing techniques thanks to deep generative models. First, we leverage a pre-trained unconditional generative model and show that it can perform a wide range of image editing tasks without re-training. Second, we build on adversarial learning to improve virtual try-on models, which consist in replacing the clothing item on an image of a person.

Key-words: GANs, generative models, adversarial training, image editing.

Résumé

Les dernières années ont vu de grands progrès des modèles génératifs profonds, et notamment des Réseaux Adversaires Génératifs (GANs). Ils peuvent notamment synthétiser des images haute résolution indiscernables des images réelles. Les modèles génératifs profonds ont également été à l'origine de succès empiriques tels que la génération de musique ou la découverte

moléculaire. Cependant, nous manquons d'une compréhension fondamentale des capacités et des limites des modèles génératifs profonds. Dans cette thèse, nous caractérisons d'abord une limitation des modèles génératifs modélisant une distribution connexe, tels que les GANs ou les flots normalisés. En effet, de tels modèles ne peuvent pas s'adapter parfaitement à une distribution cible composée de plusieurs modes déconnectés. Nous analysons théoriquement leur meilleure performance réalisable dans le cadre de distributions cibles déconnectées, et nous présentons une structure géométrique leur permettant d'atteindre une performance optimale. De plus, nous proposons des méthodes pour améliorer les GANs en les rendant plus adaptés à la modélisation de données déconnectées.

Dans la deuxième partie de la thèse, nous visons à améliorer les techniques d'édition d'images grâce aux modèles génératifs profonds. Tout d'abord, nous nous appuyons sur un modèle génératif non conditionnel et pré-entraîné, et montrons qu'il peut effectuer une large variété de tâches d'édition d'images sans réentraînement. Deuxièmement, nous nous appuyons sur l'apprentissage adversaire pour améliorer les modèles de cabine d'essayage virtuel, qui consistent à remplacer l'article vestimentaire sur une image d'une personne.

Mots-clés: GANs, modèles génératifs, entraînement adversaire, édition d'images.

Remerciements

Je tiens tout d'abord à remercier mes directeurs de thèse, Jérémie et David, pour avoir rendu cette thèse possible. Jérémie, merci de m'avoir accueilli pour un stage au sein de Criteo et de m'avoir permis de prolonger l'aventure par cette thèse. Ton esprit aguerri à la recherche et au machine learning, ta bonne humeur et ta capacité à adopter un point de vue critique m'auront énormément appris et aidé. David, je te remercie pour ta détermination et pour toutes les idées originales que tu as pu me souffler au cours de ces dernières années. Les nombreuses heures passées avec vous à proposer et décortiquer des nouvelles idées, à peaufiner et réécrire des papiers, m'ont été infiniment précieuses, tant dans le développement de cette thèse que dans celui des mes capacités à faire de la recherche.

Je tiens aussi à remercier Ugo, sans qui cette thèse n'aurait pas été la même. Tu m'auras montré la voie, et tu auras creusé un sillon fertile lors de notre première collaboration! Ton enthousiasme pour les sciences, ta rigueur et ta créativité m'ont tant aidé et inspiré. Un grand merci aussi pour tous ces moments à parler sport, cinéma ou à jouer au ping-pong, le cerveau ramolli par des sessions de travail un peu trop longues ou intenses.

Merci à Criteo de m'avoir fourni un environnement stimulant ainsi que des conditions confortables pour réaliser ma thèse, et merci à tous mes collègues et amis avec qui j'ai pu partager un café, une bière ou une partie acharnée de ping-pong entre deux deadlines. Ces moments précieux passés à décompresser, mais aussi à parler ML et papiers, ont rendu cette thèse plus riche et agréable.

Merci au laboratoire Imagine de m'avoir accueilli pour cette thèse, tout particulièrement la team des doctorants de David. Merci à tous les doctorants, post-doctorants et chercheurs pour les échanges que nous avons pu avoir. Je garderai toujours un souvenir plaisant des moments passés au laboratoire, et des retraites à la Baule et à Marseille!

Je tiens à remercier chaleureusement Rémi Flamary et Matthieu Cord pour avoir relu ce manuscrit avec attention, ainsi que Julie Delon, Jia-Bin Huang, Vicky Kalogeiton et Jakob Verbeek, d'avoir accepté de faire partie de mon jury.

Enfin, je souhaite remercier ma famille et mes proches. Merci à mes parents, qui m'auront toujours soutenu et encouragé de manière indéfectible. Maxime, merci d'avoir toujours été présent et de m'avoir tant appris. Merci à Laurine d'avoir été un soutien infaillible et de m'avoir

toujours tiré vers le haut. Merci de m'avoir toujours supporté et d'avoir rendu ces jours plus beaux.

Contents

Chapter 1

Introduction

Contents

1.1 Goals

The goal of this thesis is to understand and develop the capabilities of deep generative models for natural image generation and manipulation. Generative modelling has applications in several areas of machine learning, such as semi-supervised learning or adversarial robustness. Generative modelling aims at learning the distribution of data by having access to a discrete set of samples.

Learning natural image distributions with deep generative models. Learning the underlying distribution of natural images is a great tool for designing principled algorithms. However, natural images distributions are composed of very different visual modalities. For example, take the example of a dataset composed of animals and vehicles. It is very likely impossible to naturally interpolate between these two types of images. How do deep generative models handle these discontinuities and this multi-modality? We present new results on this questions in chapter 3. Specifically, in section 3.2, we characterize the limitations of deep generative models on multi-modal distributions and propose a heuristic method to improve pre-trained Generative

Adversarial Networks (GANs). In section 3.3, we propose a new analysis of the latent space of deep generative models. We prove that there exists an optimal geometric structure that partitions the Gaussian latent space of generative models. Moreover, we demonstrate experimentally that the appearance of this structure is positively correlated with the performance of such models, and that it can be enforced. In section 3.4, we derive a learning-based method that models a rejection mechanism in the latent space of pre-trained GANs. Interestingly, it allows to model disconnectedness in the generated distribution.

Leveraging deep generative models for image manipulation. Image manipulation encompasses a wide range of tasks, from scribble-based editing to image denoising or virtual try-on. It has exciting applications, such as creating intelligent virtual assistants for artists or developing tools for e-retailers that increase client engagement. However, it suffers from inherent difficulties: 1) their objective might be ill-defined, and tough to define; 2) the ideal data for these tasks might be complex and costly to collect. In section 4.2 we leverage a mask-based training objective and show that it allows to learn a single generative model able to tackle various image editing tasks, like image inpainting and image composition. In chapter 4.3, we approach the virtual try-on task with a teacher-student strategy, augmented with adversarial training, that allows to train the student model on a ideal synthetic dataset created by the teacher model.

1.2 Motivations

The development of deep learning methods for natural image generation and manipulation is motivated by a wide range of applications, from molecular generation in biology (Prykhodko et al., 2019) to designing SDE solvers in physics (Yang et al., 2020b). We present some of the main applications below.

Arts and advertising. The creation and editing of visual content is a highly valued skill, particularly in digital advertising, where visually appealing content is key. However, the process of creating product catalogues can be both expensive and complex, particularly when dealing with large and heavy furniture items. Furthermore, image editing demands extensive human expertise and knowledge of professional software. Therefore, the development of smart and automatic image editing techniques would greatly benefit marketing teams by expediting the process of transforming advertising ideas into visually appealing content. Additionally, automatic image editing has the potential to enhance advertising personalization. For instance,

in Figure 1.1, a virtual try-on system is showcased, which would enable e-shoppers to try on clothing items on a picture of themselves.

Fig. 1.1 **Virtual try-on**: this task consists in warping a clothing image on the image of a person. It has the potential to improve the user experience in e-shopping, since it would allow users to try clothing items on a picture of themselves.

Intelligent virtual assistants have the potential to transform the way artists work, revolutionizing the creation process of artworks through either pure generation or human-guided iterative procedures. This innovation has already sparked significant interest, with AI-designed artworks selling for high prices in 2018. Recently, large text-to-image generation models, such as Parti (Yu et al., 2022) and Stable Diffusion (Rombach et al., 2022), have achieved impressive performances, as demonstrated in Figure 1.2. Such models have been leveraged by some artists to create generated animation movies. These developments suggest that the potential of intelligent virtual assistants is vast and that they could accelerate the art creation process.

Reducing the need for labels with deep generative models. Early successes in deep learning were achieved through large-scale supervised learning, which is time-consuming and demanding of significant human resources. However, this approach is not feasible, or at least very costly, for challenging tasks that require detailed annotations, such as image segmentation, which necessitates per-pixel labeling. Deep generative modelling has demonstrated that it

Fig. 1.2 **Text-to-image generation and editing**: Hertz et al. (2023) introduce a method to improve the generated images from text-to-image models, specifically Stable Diffusion in this instance. The process involves generating an image from a given text using Stable Diffusion and allowing the user to refine the generated image by modifying the prompt. This can be achieved by adjusting the intensity of a descriptor (such as crowded or fluffy), adding new text, or modifying existing words.

can enhance the capabilities of deep learning models with minimal labeling by learning the underlying data structure without accessing any labels. For instance, Li et al. (2022) employed deep generative models to generate large-scale datasets with pixel-wise annotations of objects in images. However, the mechanisms that support the separation of data modes in the latent and feature space of deep generative models are not fully understood. Therefore, we argue that a better comprehension of these mechanisms could enable even more effective use of deep generative models in this context.

Fig. 1.3 **Reducing the need for labels:** Li et al. (2022) exploit deep generative models to construct large-scale datasets with pixel-wise annotations for image segmentation. During training, the model has only access to a few images with pixel-wise segmentation masks.

Enhancing adversarial robustness with deep generative models. Deep learning models used for discriminative tasks such as image classification or image segmentation have a well-known lack of robustness. For instance, adversarial attacks are a highly studied phenomenon in which the output of a classifier is modified by adding a small, imperceptible amount of noise to an image. Surprisingly, a deep learning-based classifier can even be fooled by modifying a single pixel of an image. Fortunately, there have been several proposed defense algorithms that leverage deep generative models. The intuition behind this approach is that adversarial attacks use the low-dimensional manifold structure of the data and send adversarial points to unseen portions of the data space. By utilizing deep generative models, we can project the attacked images back to the data manifold. For example, Song et al. (2018) use neural density estimators (PixelCNN), while Samangouei et al. (2018) leverage Generative Adversarial Networks to improve the robustness of deep learning-based classifiers.

1.3 Context

In recent years, the field of deep learning has undergone rapid progress, particularly in the development of deep generative models. GANs were invented in 2014 (Goodfellow et al., 2014), and represent a significant breakthrough in the application of deep neural networks for learning probability distributions and synthesizing images. Initially, these models were only used for grayscale 32×32 images. A few years later, Brock et al. (2019) successfully extended GANs to class-conditional synthesis on ImageNet (Deng et al., 2009) and Karras et al. (2018, 2019a) scaled GANs to high-resolution 1024×1024 images using multi-scale approaches and specific architectural design.

This thesis seeks to address and overcome the limitations of deep generative models, specifically Generative Adversarial Networks (GANs), on multi-modal target distributions. Additionally, we aim to employ these models to derive smarter algorithms for image editing. Several factors have contributed to these developments. First, the availability of large-scale datasets in computer vision has been increasing. Second, Graphics Processing Units (GPUs), used for computing deep learning experiments, have become more efficient and easier to use. This progress in performance is known as Huang's law, which posits that GPUs are advancing at a faster rate than standard Central Processing Units (CPUs). At Criteo AI Lab, we have access to a cluster with nodes consisting of two NVIDIA Tesla V100 GPUs with 16GB of RAM. For larger experiments, we also have access to a large machine with eight NVIDIA Tesla V100 GPUs, each with 32GB of RAM. Third, open-source libraries such as PyTorch (Paszke et al., 2019) or Tensorflow (Abadi et al., 2016) have greatly facilitated the prototyping of deep learning models. In this thesis, PyTorch was utilized for the majority of the experimental

work. Finally, the progress of deep generative models has been driven by both fundamental and applied research, including the development of new layers, architectures, or optimization methods. The code for recent methods is often open-sourced, along with the pre-trained model weights. All of these factors have facilitated rapid progress in deep generative models and have yielded impressive results, such as large-scale text-to-image synthesis (refer to Figure 1.2).

1.4 Challenges

When tackling the goals presented above (learning image generation and editing with deep neural networks), we face two main challenges.

Multi-modality and high diversity of natural image distributions. Looking from a historical perspective, initial computer vision datasets consisted of only a small number of classes with limited intra-class variability, such as black-and-white digits (LeCun et al., 1998). However, benchmark datasets have grown increasingly complex over time, with current large-scale datasets often containing more than a thousand classes, such as ImageNet with 21k classes (Deng et al., 2009). Consequently, such datasets exhibit high variability and diversity in the underlying and unknown target distribution. However, when working with such complex distributions, deep generative models may have two potential drawbacks. These models may occasionally overlook some modes, resulting in the well-known phenomenon of mode collapse, or they may generate low-quality data points that are located between data modes.

Lack of data for learning-based image editing. The success of deep learning models arise from well-defined training objectives optimized over large sets of training data. However, in the case of image editing tasks, such ideal combination of datasets and proper training objectives are often nonexistent. For instance, consider the task of scribble-based image editing, where a user edits an image with scribbles and the objective of the model is to produce a realistic final image while taking the user's scribbles into account. The ideal dataset for this task would comprise input images with scribbles and their corresponding edited versions. Unfortunately, constructing such a dataset would require a significant investment of time and effort from professional graphic designers. As a result, a supervised learning approach to this problem becomes infeasible. Instead, in this thesis, we investigate the use of pre-trained generative models to project images with scribbles onto the manifold of real images, thus offering a viable solution.

1.5 Thesis outline and contributions

We present the thesis outline along with the contributions:

Chapter 2: Related work. we provide a comprehensive overview of the various approaches for deep generative modelling. Specifically, we discuss the different families of models, including Generative Adversarial Networks, Autoregressive Models, and Score-Based Models. Additionally, we present standard techniques for manipulating images using deep neural networks.

Chapter 3: Learning multi-modal distributions with deep generative models. In this chapter, we focus on the challenge of learning multi-modal distributions with push-forward generative models such as GANs and VAEs. We make three distinct contributions in this area:

- **Section 3.2: learning disconnected manifolds: a no GANs' land.** We formalize the fundamental tradeoff that GANs face when learning multi-modal data, and propose a novel solution to this problem. Indeed, since GANs have a connected output distribution, they either fit all modes of the data and generate low-quality points, either ignore all but one mode and generate only good quality points. Thus, we give an upper-bound on the maximum attainable precision of GANs. This upper-bound depends on the Lipschitz constant of the generator, on the number of modes in the data and on the minimal distance between modes. Moreover, building on our theoretical analysis, we propose the 'Jacobian-Based Truncation' method for GANs. This approach involves rejecting samples with the highest Jacobian Frobenius Norm of the generator. This resulted in the following publication:

  Ugo Tanielian, **Thibaut Issenhuth**, Elvis Dohmatob, and Jérémie Mary. Learning disconnected manifolds: a no gan's land. (2020) In *International Conference on Machine Learning (ICML)*.

- **Section 3.3: unveiling the latent space geometry of push-forward generative models.** We derive a new theoretical analysis on the problem of learning disconnected manifolds with deep generative models formulated as push-forward models of a Gaussian latent distribution (such as GANs or VAEs). We study the role of the latent space geometry on the performance of such push-forward generative models. Notably, we prove a sufficient condition for their optimality: generators that structure their latent space with a specific geometry called 'simplicial cluster' are optimal. We show that this also verified experimentally with GANs. The more GANs cluster the data modes in linear regions, the better their performance. Additionally, we propose a truncation method that enforces a 'simplicial cluster' structure and improves performance of GANs. This work resulted in the following publication:

Thibaut Issenhuth, Ugo Tanielian, Jérémie Mary, David Picard. Unveiling the latent-space geometry of push-forward generative models. (2023) *International Conference on Machine Learning (ICML)*.

- **Section 3.4: latent reweighting for GANs.** Drawing from the preceding analysis, we present a novel approach for enhancing the quality of pre-trained GANs through a learning-based method. Specifically, we introduce an additional network that predicts importance weights from latent vectors, with the objective function of this network being adversarial. Our approach offers several advantages, as latent importance weights can be assigned zero values, allowing the generated distribution to model disconnected distributions effectively. Moreover, as the re-sampling mechanism occurs in the latent space, our approach is computationally efficient and can be rapidly executed at inference time. This work resulted in the following publication:

  Thibaut Issenhuth, Ugo Tanielian, David Picard, Jérémie Mary. Latent reweighting, an almost free improvement for GANs. (2022) *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*.

Chapter 4: image editing with deep neural networks. This chapter focuses on the development of deep learning methods that can perform image editing tasks without requiring direct supervision. We present two contributions. The first concerns a generalist model for image editing, and the second is designed for image-based virtual try-on:

- **Section 4.2: EdiBERT: a generative model for image editing.** The first section of this chapter introduces a novel approach that enables multiple image editing tasks to be performed using a single trained model. To achieve this, we utilize a training objective that shares similarities with most editing tasks - generating a highly realistic image from a low-quality input. Our proposed approach, named EdiBERT, is a bi-directional transformer that re-samples image patches conditioned on the entire input image. Trained with a single generic objective, EdiBERT outperforms other multi-task editing methods, such as GANs' inversion. This method is particularly well-suited for tasks that lack large-scale conditional datasets, such as image composition or scribble-based editing. This chapter is based on the following publication:

  Thibaut Issenhuth, Ugo Tanielian, Jérémie Mary, David Picard. EdiBERT: a generative model for image editing. (2022) *Transactions on Machine Learning Research (TMLR)*.

- **Section 4.3: a parser-free virtual try-on.** The virtual try-on task poses a significant challenge due to the lack of an ideal dataset, resulting in a complex pipeline comprising multiple neural networks for human parsing, pose estimation, image warping, and image

composition. The pipeline is computationally intensive, with pre-processing steps (human parsing and pose estimation) adversely impacting the final image quality since it masks important information in the input image. To address these limitations, we propose an adversarial teacher-student paradigm that eliminates the pre-processing steps during inference. Our approach reduces the inference time for virtual try-on significantly while improving the image quality.

Thibaut Issenhuth, Clément Calauzènes, and Jérémie Mary. Do not mask what you do not need to mask: a parser-free virtual try-on. (2020) In *European Conference on Computer Vision (ECCV)*.

Chapter 5: Conclusion. We summarize the contributions of this thesis and outline future directions of research, such as exploring the neural collapse phenomenon in deep generative models or investigating the latent space properties of score-based models.

Chapter 2

Related work

In this chapter, we present and discuss the recent advances in deep generative modelling, before reviewing the use of deep neural networks for image editing. This part assumes basic knowledge on machine learning and deep learning, which can be found for example in (Goodfellow et al., 2016).

2.1 Deep generative modelling

In this section, we provide an overview of the various families of deep generative models employed in this thesis, highlighting their inherent challenges and limitations with regards to the questions that we adress in this thesis. For an in-depth and comprehensive presentation of deep generative models, interested readers can refer to the works of Murphy (2023) or Tomczak (2022).

A deep generative model is a parametric family of probability distribution p_θ, where $\theta \in \Theta$ are the parameters of a neural network. It is trained to fit a target distribution $\mu^\star$ which is only accessible through an empirical distribution $\mu_n^\star$, *i.e.* a dataset of training samples $X = (x_1, \dots, x_n)$ where $x_i \in \mathbb{R}^D$ and n is the number of samples in the dataset. During training, it is optimized to minimize a distance (or divergence) D between the empirical distribution and the modelled probability distribution p_θ:

$$\theta^\star = \min_{\theta \in \Theta} \quad D(\mu_n^\star, p_\theta). \tag{2.1.1}$$

The choice of D varies depending on the family of deep generative model. For instance, Wasserstein GANs employ a Wasserstein distance as D (Arjovsky and Bottou, 2017; Gulrajani et al., 2017), while auto-regressive models rely on a Kullback-Leibler (KL) divergence.

Fig. 2.1 The evolution of images generated by GANs, from their invention in 2014 to recent developments in 2020. Source: (Zhang et al., 2021a, Figure 2.1.7).

Another notable difference between deep generative models is the way they model probability distributions. Push-forward generative models, whose denomination was introduced by Salmona et al. (2022), are a broad class of deep generative models that learn a push-forward mapping from a simple latent distribution γ, typically a Gaussian or Uniform distribution, to the modeled distribution, denoted by $p_\theta = G_\theta \sharp \gamma$. The generator G_θ is a neural network with parameters θ, and $\sharp$ represents the push-forward operator. GANs, VAEs, and normalizing flows are all examples of push-forward generative models, while score-based and diffusion models are classified as indirect push-forward generative models.

In contrast, auto-regressive models do not belong to the push-forward generative model class. Auto-regressive models utilize the decomposition $p_\theta(x) = \prod_i p_\theta(x_i|x_{\leqslant i})$, where $x \in \mathbb{R}^d$. Sampling in auto-regressive models is performed iteratively by sampling the 1-dimensional conditional distributions with a multinomial distribution. Unlike push-forward generative models, auto-regressive models do not possess a latent distribution γ. The lack of latent space makes them challenging to reuse for purposes other than sampling, such as inverse problems with generative models.

2.1.1 Families of deep generative models

2.1.1.1 Generative adversarial networks

GANs, proposed in Goodfellow et al. (2014), were the first family of deep generative models able to sample high-quality image even at high-resolution of 1024×1024 (Karras et al., 2018). On Figure 2.1, we can observe the fast progress of the quality of GANs' generated images.

GANs are based on an adversarial game between a generator network $G_\theta : \mathbb{R}^d \to \mathbb{R}^D$ and a discriminator network $D_\phi : \mathbb{R}^D \to \mathbb{R}$. The discriminator tries to distinguish real from fake images, while the generator tries to fool the discriminator. More formally, GANs rely on a minimax optimization procedure:

$$\min_{\theta \in \Theta} \max_{\phi \in \Phi} \mathbb{E}_{x \sim \mu_n^\star, z \sim \gamma} \left[f_1\big(D_\phi(x)\big) + f_2\big(D_\phi(G_\theta(z))\big) \right] \tag{2.1.2}$$

where $f_1(x) = \log(x)$ and $f_2(x) = 1 - \log(x)$. Under the assumptions of an optimal discriminator, this procedure is shown to minimize the Jenson-Shannon divergence between the empirical distribution and the generator's distribution. Surprisingly, Goodfellow et al. (2014) found that instead of minimizing $f_2\big(D_\phi(G_\theta(z))\big)$ in the generator step, minimizing $-f_1\big(D_\phi(G_\theta(z))\big)$ produced better results and reduced mode collapse. This approach is known as the non-saturating loss.

One of the primary challenges researchers faced was stabilizing the GANs training procedure. One significant observation was that, using this formulation, training the discriminator to optimality results in a discriminator with 100% accuracy and zero gradients for the generator (Arjovsky and Bottou, 2017). This led to the development of Wasserstein GANs (Arjovsky et al., 2017), where $f_1(x) = f_2(x) = x$ and discriminators are restricted to 1-Lipschitz functions. Initially, enforcing the discriminator to be 1-Lipschitz was achieved by clipping weights. Later, gradient penalty (Gulrajani et al., 2017), spectral normalization (Miyato et al., 2018), and adversarial regularization (Terjék, 2020) were introduced. However, it was later discovered that Wasserstein GANs could not converge, even on simple one-dimensional cases (Mescheder et al., 2018). The standard formulation of GANs with gradient regularization on the discriminator leads to a better-behaved optimization procedure. Overall, gradient regularization of the discriminator was critical to stabilizing the GANs' training.

A second crucial challenge is the learning of multi-modal distributions. In the seminal paper of GANs (Goodfellow et al., 2014), the mode collapse issue was already mentioned. On the opposite, when there is no mode collapse and all target modes are represented, another problem arises: images are sampled in-between modes and they are unrealistic. Indeed, as brought to the fore by Khayatkhoei et al. (2018), there is most often no realistic interpolation between two modes of image datasets, *e.g.* between a car and a dog. This is a *density misspecification problem* (Roth et al., 2017): in the case of multi-modal and disconnected target distribution, there exists no parameters $\theta \in \Theta$ such that $\mu_\theta = \mu^\star$. A solution is to use mixture distributions as latent space (Gurumurthy et al., 2017), or mixture of generators (Khayatkhoei et al., 2018). However, this introduce additional hyper-parameters and can thus complicate the training procedure. Another line of work considers rejection mechanisms from pre-trained

GANs, leveraging the discriminator scores (Azadi et al., 2019; Tanaka, 2019) or the generator's curvature (Arvanitidis et al., 2018; Humayun et al., 2022).

2.1.1.2 Variational auto-encoders

Variational auto-encoders (VAEs) (Kingma and Welling, 2014; Kingma et al., 2019) are another family of push-forward generative model, based on a likelihood training objective. However, training a latent variable model with a likelihood objective is not direct. Indeed, the modelled probability p_θ requires an intractable computation: $p_\theta(x) = \int_z p_\theta(x,z)dz$. The solution proposed by Kingma and Welling (2014) is to use variational inference, which resorts to approximating the posterior $p_\theta(z|x)$ with a stochastic encoder network $q_\phi(z|x)$. The log-likelihood can then be written:

$$\log p_\theta(x) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - \mathbb{D}_{\mathrm{KL}}\big(q_\phi(z|x)|\ p(z)\big) + \mathbb{D}_{\mathrm{KL}}\big(q_\phi(z|x)|\ p_\theta(z|x)\big) \tag{2.1.3}$$

where $\mathbb{D}_{\mathrm{KL}}$ is the KL-divergence term.

Since the true posterior $p_\theta(z|x)$ is unknown, this formulation is still not tractable. However, the last term which involves $p_\theta(x|z)$ is positive and can be dropped. This gives the Evidence Lower Bound (ELBO), which is the objective function of VAEs: $\log p_\theta(x) \geqslant \mathscr{L}_{\mathrm{ELBO}}(x;\theta,\phi) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - \mathbb{D}_{\mathrm{KL}}\big(q_\phi(z|x)|\ p(z)\big)$.

Furthermore, choosing $p(z)$ and $q_\phi(z|x)$ to be Gaussian distributions allows to get a closed-form expression of the KL-divergence term. Most often, $p(z)$ is a standard Normal distribution with a zero mean vector and identity covariance matrix. The encoder network thus predicts the mean vector $\mu_{q_\phi}(x)$ and diagonal covariance matrix $\sigma_{q_\phi}(x)$ of a Gaussian distribution. Since q_ϕ is a Gaussian distribution, a final challenge is to back-propagate the reconstruction term $\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)]$ to the encoder. This is achieved via the reparametrization trick, which consists in sampling from $\varepsilon \sim \mathbb{N}(0,I)$ and computing $x' = \sigma_{q_\phi}(x) \odot \varepsilon + \mu_{q_\phi}(x)$, where $\odot$ is an element-wise product.

Variational auto-encoders have achieved good image synthesis quality, as we observe in Figure 2.2, thanks to hierarchical approaches (Ranganath et al., 2016; Kingma et al., 2016; Vahdat and Kautz, 2020). The idea behind hierarchical variational models is to separate latent variables into different groups, so that the prior distribution has more expressiveness. However, variational model continue to lag behind GANs in terms of sampling quality.

Finally, alternative regularization methods have been proposed, such as the use of a discriminator to enforce a specific prior distribution on the encoded data points (Tolstikhin et al., 2018; Makhzani et al., 2015). Interestingly, this method allows to use a deterministic encoder, and the regularization term is applied to the aggregate posterior rather than individual samples.

Fig. 2.2 Image generations from a variational auto-encoder with a deep and hierarchical architecture. Figure from Vahdat and Kautz (2020).

2.1.1.3 Auto-regressive models

Auto-regressive models are a family of models based on the maximization of the data log-likelihood. An auto-regressive model adopts a sequence modelling approach, and models probability distribution with the following decomposition:

$$p_\theta(x) = \prod_i p_\theta(x_i|x_{<i}) \tag{2.1.4}$$

This assumes to pre-define an order on the data. Although this decomposition makes sense for natural language processing, where sentences have a natural order, it is however less natural on images, where pixels reside on a two-dimensional plane. Generally, a simple 'raster-scan' order is adopted, where pixels are processed sequentially row by row, moving from left to right and from the top of the image to the bottom.

A seminal work of autoregressive models for images was PixelCNN (Van den Oord et al., 2016; Van Den Oord et al., 2016), which proposed a deep auto-regressive neural network for images. Notably, they proposed masked convolutions, allowing each pixel to only access information from the previous ones. However, a limit of auto-regressive model for images is the large number of pixels, *e.g.* 10^6 for an image of resolution 1024×1024. To make auto-regressive models more efficient, Van Den Oord et al. (2017) derive a two-stage approach. In the first stage, an auto-encoder compresses images into sequences of discrete elements. In the second stage, an auto-regressive model learns the distribution of the sequences of discrete elements. This lead to high-quality results with the adoption of the transformer architecture (Vaswani et al., 2017) as an auto-regressive model (Esser et al., 2021b). This approach is shown in Figure 2.3. A great advantage of such approaches is their very stable training, which allows an easier scaling of models with overparametrized models and large-scale datasets.

A line of research has focused on getting rid of the arbitrary ordering of sequences, following the path of BERT (Devlin et al., 2018) where the objective is to reconstruct masked sequences.

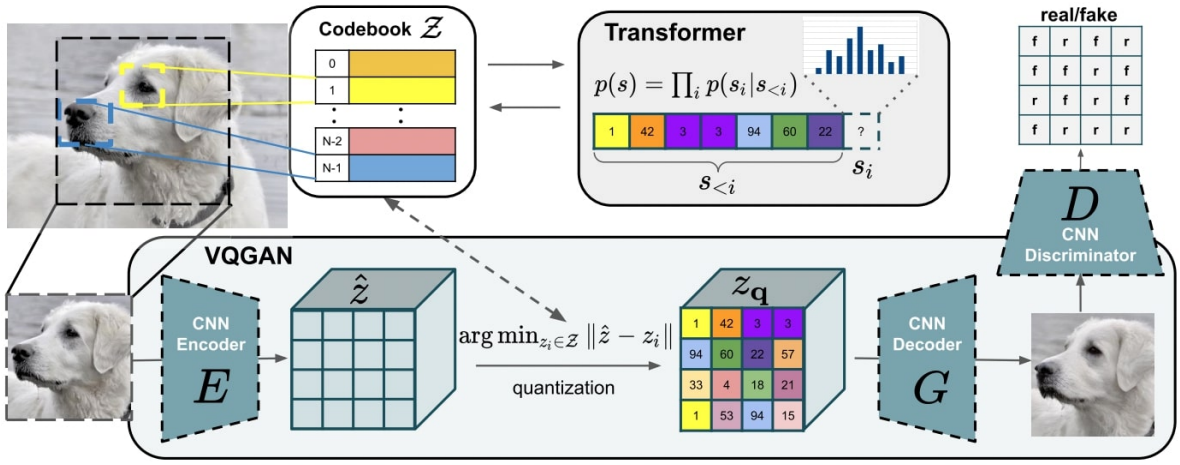


Fig. 2.3 Two-stage approach with discrete auto-encoder and auto-regressive models, parameterized with a transformer in this case. Source: Esser et al. (2021b).

Notably, MaskGIT (Chang et al., 2022) achieved state-of-the-art synthesis results by training a transformer to reconstruct masked sequences, and generating samples with a parallel decoding scheme. This was then extended to a large text-to-image model (Chang et al., 2023).

2.1.1.4 Diffusion and score-based models

Diffusion models (Sohl-Dickstein et al., 2015) are based on an iterative process that maps data to noise, and consists in learning the reverse process with a neural network. In the first years, they received less attention than other deep generative models. However, recent work by (Ho et al., 2020) has revived interest in diffusion models by drawing a fundamental connection between score-based models (Song and Ermon, 2019; Song et al., 2020) and diffusion models. The first score-based generative model (Song and Ermon, 2019) builds upon two key ingredients: learning the score $s(x) = \nabla_x \log p(x)$ of a distribution $\mu^\star$ with denoising autoencoders (Vincent, 2011), and Langevin sampling.

First, Vincent (2011) showed that the following simple denoising objective allows to learn the score of a distribution:

$$L(\theta) = \mathbb{E}_{x \sim \mu_n^\star, \tilde{x} \sim q_\sigma(x)}[\|s_\theta(\tilde{x}) - \frac{\tilde{x} - x}{\sigma^2}\|_2^2] \tag{2.1.5}$$

where s_θ is a neural network with parameters θ, x are samples from the target distribution, and $\tilde{x}$ are noisy samples $\tilde{x} = x + \varepsilon$ with $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$.

Second, Langevin sampling allows to draw samples from a given distribution $\mu^\star$ when having access to $\nabla_x \log \mu^\star(x)$. It is a Monte-Carlo Markov Chain (MCMC) composed of the

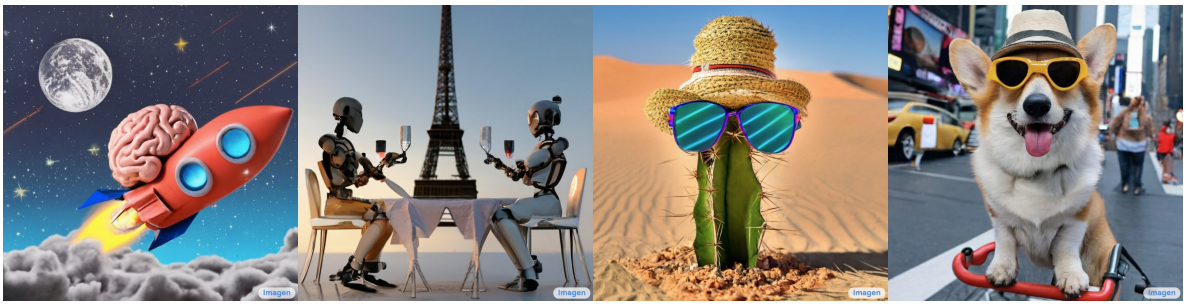


Fig. 2.4 From left to right, the text-conditioning of the generated images are: "A braing riding a rocketship heading towards the moon"; "A robot couple fine dining with Eiffel Tower in the background"; "A small cactus wearing a straw hat and neon sunglasses in the Sahara desert."; "A photo of a Corgi dog riding a bike on Time Squares. It is wearing sunglasses and a beach hat.". Source: https://imagen.research.google/.

following update steps: $x_{t+1} = x_t + \varepsilon \nabla_x \log \mu^\star(x) + \sqrt{2\varepsilon} z_t$, where z_t is a standard Gaussian random variable.

Score-based models, introduced in Song and Ermon (2019) and further developed in Song et al. (2020), rely on a hierarchy of Gaussian noise variances $\sigma_1 < \cdots < \sigma_L$ to generate high-quality images. An efficient training strategy is to use a single neural network conditioned to the noise level $s_\theta(x, \sigma_i)$. For sampling, an annealing Langevin strategy is proposed, which runs Langevin MCMC with the highest noise level σ_L and gradually reduces the noise level until the lowest level σ_1 is reached. Building upon these ideas, Song et al. (2021) propose using an infinite number of noise scales, effectively reversing a continuous-time stochastic process. This approach allows for new numerical stochastic differential equation solvers to be derived (Song et al., 2021; Karras et al., 2022).

In conjunction with architectural improvements and large-scale training, score-based and diffusion models have achieved state-of-the-art image quality in unconditional (Karras et al., 2022; Dhariwal and Nichol, 2021) and text-to-image models (Saharia et al., 2022). This can be observed in Figure 2.4. The superiority of these models over GANs is not yet fully understood, although it seems that the training objective (L2 loss) contributes to their stability, ease of optimization, and scalability. A notable feature of score-based and diffusion models is their iterative sampling procedure, which involves L forward passes on the score network to generate one image. This could contribute to their greater expressive power compared to models that rely on a single forward pass.

2.1.2 Evaluating deep generative models

The evaluation of deep generative models is a critical yet often neglected problem. The machine learning pipeline relies heavily on optimizing metrics, whether for learning the model's parameters or estimating the optimal hyperparameters. However, many generative models are trained with a differentiable loss function that does not directly reflect their ultimate objectives, namely distribution fitting and sample quality. Even when log-likelihood evaluation is possible, it may not necessarily correlate with other objectives such as sample quality, as noted by Theis et al. (2016). Also, it is worth noting that the value of deep generative models lies in their ability to generalize, which is essential for generating unseen samples or testing the likelihood of unseen data.

Standard metrics and their flaws. The initial efforts to evaluate deep generative models, particularly GANs which do not have likelihood estimates, led to the development of the Inception Score (IS) (Salimans et al., 2016) and the Fréchet Inception Distance (FID) (Heusel et al., 2017). IS leverages a pre-trained Inception classifier to compute the probability of classes per sample. It is minimized when samples are diverse and the sample-wise class distributions are peaked. On the other hand, the FID also utilizes the Inception Net as a feature extractor but only compares target and generated distributions, assuming they are Gaussian, using the Fréchet Distance. However, FID has been shown to have several limitations, such as the false assumption of Gaussianity for multimodal datasets (Luzi et al., 2023) and the potential for misleadingly high scores when there is a high intra-class variation and poor coverage by the generative model (Kynkäänniemi et al., 2019).

Metrics based on support estimation. To solve this issue, another line of work developed precision and recall metrics for generative models. These metrics have gained traction due to their clarity and interpretability. Intuitively, precision measures the quality of generated samples, while recall measures the mode coverage of the generated samples. The first precision and recall metric was proposed by Sajjadi et al. (2018). However, the algorithm had limitations, relying on clustering and discrete probability comparison, which does not account for situations with intra-cluster variability. To address this issue, Kynkäänniemi et al. (2019) developed an improved precision and recall metric by directly approximating the real and generated supports with k-nearest neighbor spheres around samples. Precision and recall can then be defined per sample. Precise samples are generated sampled that fall within the approximated support of data samples. Figure 2.5 provides an illustration of this metric. However, Naeem et al. (2020) found that the improved precision and recall metrics are sensitive to outliers and proposed density and coverage metrics.

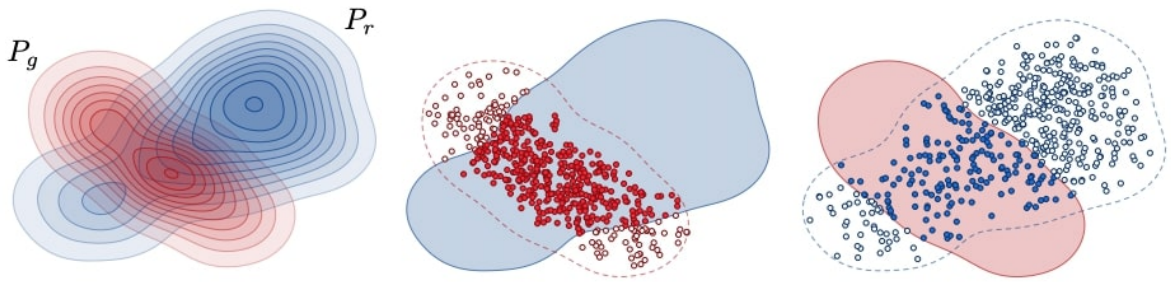


Fig. 2.5 (Left) Illustration of two distributions, where the blue one P_r is the target distribution and the red one P_g is the generated one. (Middle) Precision corresponds to the proportion of generated points that fall in the approximated support of the target distribution. (Right) Recall corresponds to the proportion of real points that fall in the approximated support of the generated distribution. Figure from Kynkäänniemi et al. (2019).

What about generalization? While the aforementioned methods provide ways to assess the quality of generative models, they fail to evaluate their generalization abilities. Suppose we compute the test-FID or the test-Precision and Recall, which would compare generated samples to test samples. A good score obtained using these metrics does not indicate good generalization abilities since using the training set (instead of generated samples) also yields a great score. To address this issue, Alaa et al. (2022) proposed a new metric called authenticity that measures the proportion of generated points that are very similar to training data, thereby providing a heuristic evaluation of a model's generalization abilities.

2.2 Image editing with deep neural networks

In this section, we provide an overview of modern methods for image editing using deep neural networks. First, we discuss neural network architectures that are particularly well-suited for image editing tasks. Next, we describe training techniques for image editing tasks that can be classified into two categories: supervised and unsupervised.

To begin, let us define image editing more precisely. Image editing involves mapping a given source image I_s to a realistic image I_r, given some user-defined conditioning C. The function that we aim to learn is of the form $f_\theta(I_s, C)$, which can be stochastic or deterministic depending on the type of I_s and C. The conditioning C can take various forms, including image-based and text-based conditioning.

Image-based conditioning encompasses a wide range of tasks, such as image inpainting or scribble-based editing. Additionally, image-based virtual try-on, which involves inserting a clothing item onto an image of a person, can also be classified as image editing with image-based conditioning.

Text-based conditioning involves editing an image based on a natural language prompt. This task was introduced in Figure 1.2 in the Introduction. With the advent of large-scale text-to-image generation models, we can expect image editing with text-based conditioning to achieve high levels of quality in the coming months or years.

2.2.1 Specific neural architectures for image editing

In this part, we describe two neural network architectures that are well suited for image editing tasks, namely U-Nets (Ronneberger et al., 2015) and spatial transformers (Jaderberg et al., 2015). Some tasks, such as virtual try-on, are based on the combination of these two architectures.

2.2.1.1 U-Nets

U-Nets, first introduced by Ronneberger et al. (2015), were initially designed for image segmentation but have proven to be particularly effective for image-to-image operations due to two key properties. Firstly, they incorporate multi-scale processing of the image, and secondly, they conserve high-resolution information. The first part of a U-Net consists of a series of convolutional blocks C_i, which are intertwined with down-sampling operations D. Given an input image x, the hidden layers $x_1 = C_1(x)$, $x_2 = C_2(D(x_1))$, and $x_3 = C_3(D(x_2))$ are defined. The second part of the network involves convolutional blocks intertwined with up-sampling operations U, and concatenation of image tensors using the cat() function. The U-Net combines low-resolution and high-resolution information by employing the operation $x_4 = C_4\Big(\text{cat}\big(x_2; U(x_3)\big)\Big)$, which is sequentially performed until reaching the input resolution of the image x.

U-Nets have proven to be crucial for the success of several deep image editing models, including CycleGAN (Zhu et al., 2017) and virtual try-on models (Han et al., 2018; Wang et al., 2018a). The primary advantage of U-Nets is their strong inductive bias towards preserving fine-grained, high-resolution details in images. This feature is particularly important for maintaining the quality of the source image, which is critical in image editing.

2.2.1.2 Spatial transformers for image warping

The Spatial Transformer Network (Jaderberg et al., 2015) is a module designed to enable differentiable image sampling. It allows spatial image manipulation, such as affine transformation, in a differentiable manner. Moreover, the parameters of the spatial transform are predicted by the neural network. Its primary purpose is to aid neural networks in learning invariance to complex geometric transformations, including translation, rotation, scaling, and warping. Originally

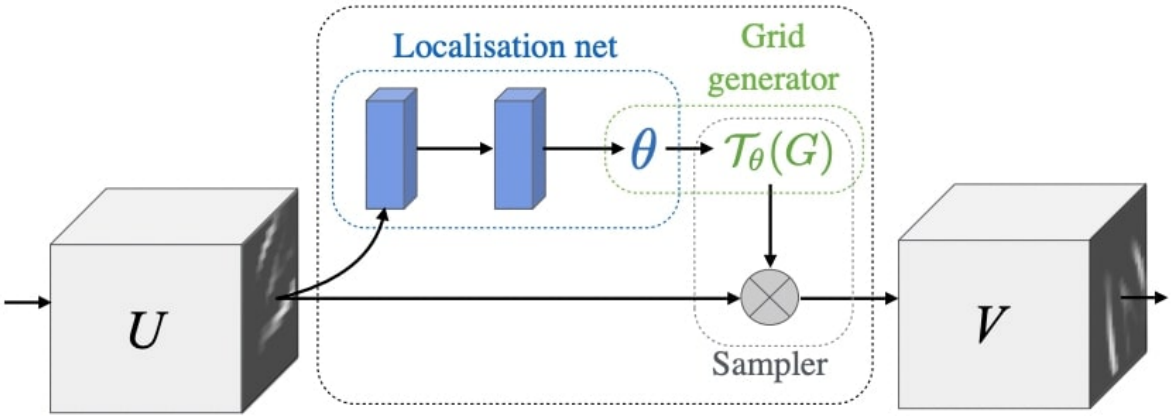


Fig. 2.6 The spatial transformer module allows to spatially transform feature maps in a differentiable manner. From an input feature map U, the module predicts the parameters θ of a spatial transform, which is then transformed into a sampling grid $\mathcal{T}_\theta(G)$, where G is the regular base sampling grid. $\mathcal{T}_\theta(G)$ is then used to re-sample pixels from the input feature map U, which gives the output feature map V. Figure from Jaderberg et al. (2015).

included in image classification and recognition networks, Spatial Transformer Networks have proven effective in enhancing the robustness of such networks to geometrical variations. An illustration of the Spatial Transformer Network module is presented in Figure 2.6. An important development of the Spatial Transformer Network concerns tasks that require attention mechanisms, such as geometric matching. In this context, Spatial Transformer Networks are learned to align two different images (Rocco et al., 2017).

For image editing, spatial transformer networks are particularly useful in tasks requiring object insertion. For example, in human pose transfer, it is used to warp the person features towards the target pose (Dong et al., 2018). In the virtual try-on task, spatial transformer networks are leveraged to deform the clothing item before inserting it on the person image (Wang et al., 2018a).

2.2.2 Supervised approaches

In the following, we present various supervised techniques for training deep neural networks in image editing tasks. The common thread among these methods is the reliance on extensive labelled datasets, which both facilitate their success and constrain their applicability. We will see that, among these methods, there are different levels of supervision required, from fully paired to unpaired datasets. These approaches benefit from the extensive history of supervised learning with deep neural networks, which has led to the refinement of crucial components such as loss functions, architectures, and initialization. However, their effectiveness is limited by the unavailability or high cost of data collection for some image editing tasks.

2.2.2.1 Regression-based methods

One of the most direct approaches for learning image editing with deep neural networks is through regression-based methods. To illustrate this, let us consider a dataset consisting of a set of source images, $X = (I_s^1, \ldots, I_s^N)$, and their corresponding target images, $Y = (I_t^1, \ldots, I_t^N)$. A neural network, denoted as f_θ, is trained on this dataset to minimize the following loss function:

$$\mathscr{L} = \frac{1}{N} \sum_{i=1}^{N} D\big(f_\theta(I_s^i), I_t^i\big) \tag{2.2.1}$$

Here, D is a function that measures the distance between images, and can be specified as L1 or L2 loss. Recent studies have shown that using perceptual loss (Zhang et al., 2018) can result in better-guided image editing or synthesis models. Perceptual loss involves computing the L2 distance in the feature space of a pre-trained deep neural network, typically the VGG network (Simonyan and Zisserman, 2014).

As previously mentioned, this setting is often limited due to the requirement of collecting pairs of source and target images, (I_s^i, I_t^i), which can be a challenging and time-consuming process. To address the issue of limited data, synthetic data creation can be employed in some cases. For instance, a synthetic dataset can be combined with a supervised regression-based objective to achieve virtual try-on tasks (Han et al., 2018) or scribble-based editing (Liu et al., 2021). However, creating synthetic data typically requires careful design and significant engineering effort.

2.2.2.2 Cycle-based methods

Suppose we have two datasets, one consisting of apples and the other of oranges, and we wish to learn an image editing model that can transform an apple to an orange, and vice versa. Unfortunately, we do not have access to explicit pairs of source and target images that demonstrate these transformations, making it impossible to use a regression objective. One solution is to use cycle-based methods (Zhu et al., 2017), which leverage adversarial training and cycle-consistency to learn the mapping between two unpaired datasets.

The idea of cycle-based methods is illustrated in Figure 2.7, and has been applied to domain adaptation (Hoffman et al., 2018), unsupervised object insertion (Zhan et al., 2019), and virtual try-on tasks (Ge et al., 2021).

2.2.2.3 Conditional distribution modelling

If we have access to pairs of data and the target function $f^\star(I_s, C)$ is stochastic, the most successful approach is typically to model the conditional distribution $p_\theta(I_t|I_s, C)$, where I_t is

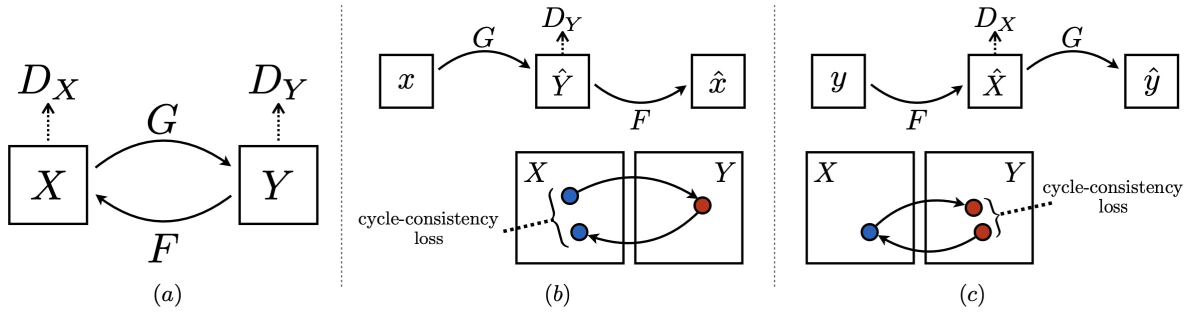


Fig. 2.7 The Cycle GAN approach consists in combining an adversarial objective with a cycle-consistency constraint to learn a mapping between unpaired datasets. The adversarial loss enforces G to map an image I_s from domain X to an image with properties of domain Y, while F maps from X to Y. The cycle-consistency constraint enforces the mapping to keep information about the original image I_s. Figure from Zhu et al. (2017).

the target image, I_s is the source image, and C is the conditioning. Standard deep generative modeling techniques, such as GANs or auto-regressive models, can be used to model conditional distributions. Depending on the approach and base architecture, incorporating conditions into the modeled distribution p_θ may require more or less adaptation. For example, when using auto-regressive transformers, the conditioning can be simply concatenated to the input sequence. In contrast, when using GANs and a style-based generator (Karras et al., 2019b), a more careful adaptation of the generator and discriminator is often necessary. For instance, Co-Modulated GANs (Zhao et al., 2020) present an adaptation of StyleGAN for image inpainting.

This approach can be very effective for tasks that fulfill two conditions: 1) the ease of data collection; and 2) the stochastic nature of the target function $f^\star(I_s, C)$. The image inpainting task satisfies both of these conditions, and state-of-the-art results have been achieved through conditional distribution modelling. Some approaches rely on conditional GANs, such as Co-Modulated GANs (Zhao et al., 2020). Others use conditional auto-regressive models, as proposed in Peng et al. (2021). More recently, some approaches have been based on diffusion models (Lugmayr et al., 2022).

2.2.3 Unsupervised approaches: leveraging pre-trained generative models

In unsupervised approaches, a prevalent strategy is to rely on pre-trained generative models and adopt an inverse problem viewpoint. The fundamental assumption is that the source images I_s in most editing tasks are perturbed and noisy. For example, in the image inpainting task, I_s corresponds to a realistic image I that has been distorted by zeroing out an area defined by a binary mask m. Consequently, we have $I_s = I \odot m$, where $\odot$ denotes the element-wise product.

The underlying idea can be cast under the Maximum A Posteriori view:

$$I_t = \arg\max_x p(x|\, I_s) = \arg\max_x p(I_s|x)p(x) \tag{2.2.2}$$

where the prior distribution $p(x)$ is estimated with a deep generative model, and the conditional distribution $p(I_s|x)$ is approximated with a distance between the two images x and I_s.

In GANs, the inversion of edited images I_s into the latent space of a pre-trained GAN can be accomplished by minimizing the distance D between the generated image $G_\theta(z)$ and the edited image I_s:

$$I_t = G_\theta(z^\star) \quad \text{where} \quad z^\star = \arg\min_z D(G_\theta(z), I_s) \tag{2.2.3}$$

The optimal latent code $z^\star$ can be obtained through optimization or by using an encoder network that maps images to their corresponding latent codes. These two methods can be combined, as proposed in the seminal work of GANs' inversion by Zhu et al. (2016). This generic approach has been adapted to new generator architectures like StyleGAN, as demonstrated in subsequent works. For instance, Abdal et al. (2019, 2020) use an optimization procedure over an intermediate feature space of StyleGAN, while Richardson et al. (2021) and Tov et al. (2021) design a specific encoder architecture that maps images into the intermediate features of StyleGAN. Figure 2.8 illustrates an example of GANs inversion.

Recently, the concept of inverting generative models has also proven to be effective in the context of diffusion and score-based models. The work by Kawar et al. (2022) has demonstrated the efficacy of pre-trained diffusion probabilistic models in solving inverse problems, such as deblurring or inpainting. In a similar vein, Meng et al. (2022) has shown that pre-trained generative models can facilitate image editing tasks. By judiciously controlling the noise that is added to the image, these models can convert sketches into realistic images, perform image inpainting or scribble-based editing using mask guided Langevin dynamics. Despite the success of diffusion models in solving inverse problems, their long sampling procedure is a significant limitation, requiring many forward passes on a neural network. Chung et al. (2022) have endeavored to reduce the number of iterations required in inverse problems, which remains a challenging task. Finally, Couairon et al. (2023) propose a text-based editing method based on diffusion models. Interestingly, it removes the burden of giving a mask as input to the editing algorithm by automatically generating a mask.

Attribute editing after inversion. Another approach for image editing is to combine image inversion with latent space traversal. The first step involves a standard inversion procedure,

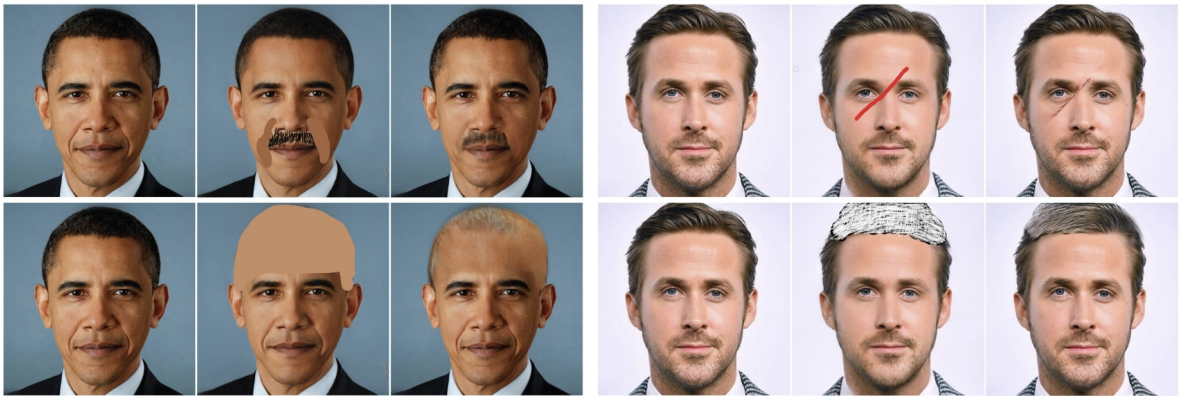


Fig. 2.8 Inverting images with scribbles in the latent space of GANs allow to recover realistic images that still takes into account the user-input (scribbles). Figure from Abdal et al. (2020).

Fig. 2.9 The latent space of GANs allow image editing along the main factors of variation in a disentangled manner. Figure from Shen et al. (2020).

which can be formulated as follows:

$$\tilde{I}_s = G_\theta(z^\star) \quad \text{where} \quad z^\star = \arg\min_z D(G_\theta(z), I_s) \tag{2.2.4}$$

Here, $\tilde{I}_s$ is an approximation of the source image I_s generated by the GAN. The primary objective of this procedure is to derive $z^\star$, which corresponds to the latent vector associated with $\tilde{I}_s$. This vector $z^\star$ can then be utilized to perform attribute editing by following specific directions in the latent space. Previous studies have demonstrated that the latent space of GANs contains interpretable and disentangled control directions (Voynov and Babenko, 2020; Härkönen et al., 2020). Consequently, the inversion process can be integrated with attribute modifications in GANs' latent space, leading to straightforward image attribute editing procedures (Shen et al., 2020). Figure 2.9 illustrates the outcomes of such a procedure. Finally, Grechka et al. (2021)

design a method that provides a tradeoff between reconstruction quality and editability of the latent vector.

The task of attribute editing can be facilitated through the natural language interaction between a user and an editing system. Notably, recent research has proposed utilizing the CLIP model (Radford et al., 2021), which evaluates the alignment between text and image, in combination with text-to-image diffusion models (Kim et al., 2022), or with unconditional StyleGAN (Patashnik et al., 2021). Interestingly, Couairon et al. (2022) encodes and manipulates images in the latent space of CLIP. These approaches have demonstrated promising results in enabling users to control and manipulate various attributes of an image, such as changing its background or altering its facial expressions, by providing textual descriptions.

Chapter 3

Learning multi-modal distributions with deep generative models

Contents

3.1 Introduction

In this chapter, our focus is on the learning of multi-modal target distributions with deep generative models, specifically, with a class of deep generative models called "push-forward generative models". These models encompass a broad range of deep generative models, including GANs, VAEs, and normalizing flows, but exclude auto-regressive models. Push-forward generative models represent a distribution p_θ as the push-forward of a latent distribution γ by a neural network G_θ, which is denoted as $p_\theta = G_\theta \sharp \gamma$. In this chapter, we demonstrate that push-forward generative models are susceptible to misspecifications when learning multi-modal target distributions. When employing standard design choices of latent distribution and neural network, the hypothesis family of generative distributions p_θ exclusively comprises connected distributions. Consequently, when attempting to cover all modes of the target distribution, the generative distribution necessarily samples in between the target modes, which raises several fundamental questions. For instance, what is the best achievable performance for push-forward generative models? How can we optimize the structure of the latent space such that low-quality

points are present only in small proportion? And how can we develop principled rejection mechanisms that enhance pre-trained models?

In Section 3.2, we present an upper bound on the precision of push-forward generative models when the target distribution is made of disconnected modes. This result is derived from the Gaussian isoperimetric inequality, which states that among all sets with a given Gaussian measure, half-spaces have the minimal Gaussian perimeter or Gaussian boundary measure. Furthermore, we leverage this analysis to develop a truncation method that enhances pre-trained GANs by removing samples where the generator has a high Jacobian Frobenius norm.

In Section 3.3, we expand upon our analysis and demonstrate the existence of an optimal geometry for the latent space. This optimal structure, referred to as a "simplicial cluster", is based on a recent mathematical breakthrough by Milman and Neeman (2022) that solves the Gaussian isoperimetric problem for partitions with more than two subsets. This result enables us to demonstrate that the simplicial cluster structure is also optimal for push-forward generative models learning disconnected target distributions. Furthermore, we propose a way to enforce this structure in the latent space of GANs and show that it leads to improved performance.

In Section 3.4, we present a novel method for learning a rejection mechanism in pre-trained GANs. This method is based on adversarial learning of importance weights. Modelling importance weights requires respecting certain constraints on the neural network's output. To enforce these constraints, we introduce regularization terms to the objective function. We provide two algorithms for sampling from these importance weights, which can be combined. We conduct experiments to demonstrate that this approach outperforms other learning-based post-processing methods for pre-trained GANs.

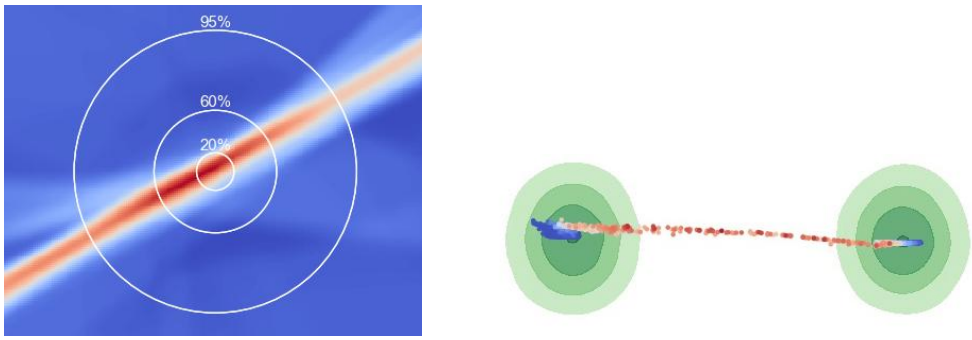
3.2 Learning disconnected manifolds: a no GAN's land

Abstract. Typical architectures of push-forward generative models make use of a unimodal latent/input distribution transformed by a continuous generator. This includes GANs, VAEs and normalizing flows. Consequently, the modeled distribution always has connected support which is cumbersome when learning a disconnected set of manifolds. We formalize this problem by establishing a "no free lunch" theorem for the disconnected manifold learning stating an upper-bound on the precision of the targeted distribution. This is done by building on the necessary existence of a low-quality region where the generator continuously samples data between two disconnected modes. Finally, we derive a rejection sampling method based on the norm of generator's Jacobian and show its efficiency on several generators including BigGAN.

3.2.1 Introduction

Push-forward generative models such as GANs (Goodfellow et al., 2014) provide a very effective tool for the unsupervised learning of complex probability distributions. For example, Karras et al. (2019b) generate very realistic human faces while Yu et al. (2017) match state-of-the-art text corpora generation. Despite some early theoretical results on the stability of GANs (Arjovsky and Bottou, 2017) and on their approximation and asymptotic properties (Biau et al., 2020), their training remains challenging. More specifically, GANs raise a mystery formalized by Khayatkhoei et al. (2018): *how can they fit disconnected manifolds when they are trained to continuously transform a unimodal latent distribution?* While this question remains widely open, we will show that studying it can lead to some improvements in the sampling quality of GANs. Indeed, training a GAN with the objective of continuously transforming samples from an unimodal distribution into a disconnected requires balancing between two caveats. On one hand, the generator could just ignore all modes but one, producing a very limited variety of high quality samples: this is an extreme case of the well known mode collapse (Arjovsky and Bottou, 2017). On the other hand, the generator could cover the different modes of the target distribution and necessarily generates samples out of the real data manifold as previously explained by Khayatkhoei et al. (2018).

As brought to the fore by Roth et al. (2017), there is a density mis-specification between the true distribution and the model distribution. Indeed, one cannot find parameters such that the model density function is arbitrarily close to the true distribution. To solve this issue, many empirical works have proposed to over-parameterize the generative distributions, as for instance, using a mixture of generators to better fit the different target modes. Tolstikhin et al. (2017) rely on boosting while Khayatkhoei et al. (2018) force each generator to target different sub-manifolds thanks to a criterion based on mutual information. Another direction is to add



(a) Heatmap of the generator's Jacobian norm. White circles: quantiles of the latent distribution $\mathcal{N}(0, I)$.

(b) Green: target distribution. Coloured dots: generated samples colored w.r.t. the Jacobian Norm using same heatmap than (a).

Fig. 3.1 Learning disconnected manifolds leads to the apparition of an area with high gradients and data sampled in between modes.

complexity in the latent space using a mixture of Gaussian distributions (Gurumurthy et al., 2017).

To better visualize this phenomenon, we consider a simple 2D motivational example where the real data lies on two disconnected manifolds. Empirically, when learning the distribution, GANs split the Gaussian latent space into two modes, as highlighted by the separation line in red in Figure 3.1a. More importantly, each sample drawn inside this red area in Figure 3.1a is then mapped in the output space in between the two modes (see Figure 3.1b). For the quantitative evaluation of the presence of out-of-manifold samples, a natural metric is the Precision-Recall (PR) proposed by Sajjadi et al. (2018) and its improved version (Improved PR) (Kynkäänniemi et al., 2019). A first contribution of this section is to formally link them. Then, taking advantage of these metrics, we lower bound the measure of this out-of-manifold region and formalize the impossibility of learning disconnected manifolds with standard GANs. We also extend this observation to the multi-class generation case and show that the volume of off-manifold areas increases with the number of covered manifolds. In the limit, this increase drives the precision to zero.

To solve this issue and increase the precision of GANs, we argue that it is possible to remove out-of-manifold samples using a truncation method. Building on the work of Arvanitidis et al. (2018) who define a Riemaniann metric that significantly improves clustering in the latent space, our truncation method is based on information conveyed by the Jacobian's norm of the generator. We empirically show that this rejection sampling scheme enables us to better fit disconnected manifolds without over-parametrizing neither the generative class of functions nor the latent distribution. Finally, in a very large high dimensional setting, we discuss the

advantages of our rejection method and compare it to the truncation trick introduced by Brock et al. (2019).

In a nutshell, our contributions are the following:

- We discuss evaluation of GANs and formally link the PR measure (Sajjadi et al., 2018) and its Improved PR version (Kynkäänniemi et al., 2019).
- We upper bound the precision of GANs with Gaussian latent distribution and formalize an impossibility result for disconnected manifolds learning.
- Using toy datasets, we illustrate the behavior of GANs when learning disconnected manifolds and derive a new truncation method based on the Jacobian's Frobenius norm of the generator. We confirm its empirical performance on state-of-the-art models and datasets.

3.2.2 Related work

Fighting mode collapse. Goodfellow et al. (2014) were the first to raise the problem of mode collapse in the learning of disconnected manifolds with GANs. They observed that when the generator is trained too long without updating the discriminator, the output distribution collapses to a few modes reducing the diversity of the samples. To tackle this issue, Salimans et al. (2016); Lin et al. (2018) suggested feeding several samples to the discriminator. Srivastava et al. (2017) proposed the use of a reconstructor network, mapping the data to the latent space to increase diversity.

In a different direction, Arjovsky and Bottou (2017) showed that training GANs using the original formulation Goodfellow et al. (2014) leads to instability or vanishing gradients. To solve this issue, they proposed a Wasserstein GAN architecture Arjovsky et al. (2017) where they restrict the class of discriminative functions to 1-Lipschitz functions using weight clipping. Pointing to issues with this clipping, Gulrajani et al. (2017); Miyato et al. (2018) proposed relaxed ways to enforce the Lipschitzness of the discriminator, either by using a gradient penalty or a spectral normalization. Albeit not exactly approximating the Wasserstein's distance (Petzka et al., 2018), both implementations lead to good empirical results, significantly reducing mode collapse. Building on all of these works, we will further assume that generators are now able to cover most of the modes of the target distribution, leaving us the problem of out-of-manifold samples (*a.k.a.* low-quality pictures).

Generation of disconnected manifolds. When learning complex manifolds in high dimensional spaces using deep generative models, Fefferman et al. (2016) highlighted the importance

of understanding the underlying geometry. More precisely, the learning of disconnected manifold requires the introduction of disconnectedness in the model. Gurumurthy et al. (2017) used a multi-modal entry distribution, making the latent space disconnected, and showed better coverage when data is limited and diverse. Alternatively, Khayatkhoei et al. (2018) studied the learning of a mixture of generators. Using a mutual information term, they encourage each generator to focus on a different submanifold so that the mixture covers the whole support. This idea of using an ensemble of generators is also present in the work of Tolstikhin et al. (2017) and Zhong et al. (2019), though they were primarily interested in the reduction of mode collapse.

In this section, we propose a truncation method to separate the latent space into several disjoint areas. It is a way to learn disconnected manifolds without relying on the previously introduced over-parameterization techniques. As our proposal can be applied without retraining the whole architecture, we can use it successfully on very larges nets. Close to this idea, Azadi et al. (2019) introduced a rejection strategy based on the output of the discriminator. However, this rejection sampling scheme requires the discriminator to be trained with a classification loss while our proposition can be applied to any generative models.

Evaluating GANs. The evaluation of generative models is an active area of research. Some of the proposed metrics only measure the quality of the generated samples such as the Inception score Salimans et al. (2016) while others define distances between probability distributions. This is the case of the Frechet Inception distance Heusel et al. (2017), the Wasserstein distance Arjovsky et al. (2017) or kernel-based metrics Gretton et al. (2012). The other main caveat for evaluating GANs lies in the fact that one does not have access to the true density nor the model density, prohibiting the use of any density based metrics. To solve this issue, the use of a third network that acts as an objective referee is common. For instance, the Inception score uses outputs from InceptionNet while the Fréchet Inception Distance compares statistics of InceptionNet activations. Since our work focuses on out-of-manifold samples, a natural measure is the PR measure (Sajjadi et al., 2018) and its Improved PR version (Kynkäänniemi et al., 2019), extensively discussed in the next section.

In the following, alongside precise definitions, we exhibit an upper bound on the precision of GANs with high recall (*i.e.* no mode collapse) and present a new truncation method.

3.2.3 Our approach

We start with a formal description of the framework of GANs and the relevant metrics. We later show a "no free lunch" theorem proving the necessary existence of an area in the latent space that generates out-of-manifold samples. We name this region the *no GAN's land* since

any data point sampled from this area will be in the frontier in between two different modes. We claim that dealing with it requires special care. Finally, we propose a rejection sampling procedure to avoid points out of the true manifold.

3.2.3.1 Notations

In the original setting of Generative Adversarial Networks (GANs), one tries to generate data that are "similar" to samples collected from some unknown probability measure $\mu_\star$. To do so, we use a parametric family of generative distribution where each distribution is the push-forward measure of a latent distribution γ and a continuous function modeled by a neural network.

Assumption 1 (γ Gaussian)**.** *The latent distribution γ is a standard multivariate Gaussian.*

Note that for any distribution μ, S_μ refers to its support. Assumption 1 is common for GANs as in many practical applications, the latent variable defined on a low dimensional space $\mathbb{R}^d$ is either a multivariate Gaussian, either a uniform distribution on a compact.

The measure $\mu_\star$ is defined on a subset E of $\mathbb{R}^D$ (potentially a highly dimensional space), equipped with the norm $\|\cdot\|$. The generator has the form of a parameterized class of functions from $\mathbb{R}^d$ (a space with a much lower dimension) to E, say $\mathscr{G} = \{G_\theta : \theta \in \Theta\}$, where $\Theta \subseteq \mathbb{R}^p$ is the set of parameters describing the model. Each function G_θ thus takes input from a d-dimensional space variable Z (Z is associated with probability distribution γ) and outputs "fake" observations with distribution μ_θ. Thus, the class of probability measures $\mathscr{P} = \{\mu_\theta : \theta \in \Theta\}$ is the natural class of distributions associated with the generator, and the objective of GANs is to find inside this class of candidates the one that generates the most realistic samples, closest to the ones collected from the unknown distribution $\mu_\star$.

Assumption 2. *Let $L > 0$. The generator G_θ takes the form of a neural network whose Lipchitz constant is smaller than L, i.e. for all (z, z'), we have $\|G_\theta(z') - G_\theta(z)\| \leqslant L\|z - z'\|$.*

This is a reasonable assumption, since Virmaux and Scaman (2018) present an algorithm that upper-bounds the Lipschitz constant of deep neural networks. Initially, 1-Lipschitzness was enforced only for the discriminator by clipping the weigths Arjovsky et al. (2017), adding a gradient penalty Gulrajani et al. (2017); Roth et al. (2017); Petzka et al. (2018), or penalizing the spectral norms Miyato et al. (2018). Nowadays, state-of-the-art architectures for large scale generators such as SAGAN Zhang et al. (2019) and BigGAN Brock et al. (2019) also make use of spectral normalization for the generator.

3.2.3.2 Evaluating GANs with Precision and Recall

When learning disconnected manifolds, Srivastava et al. (2017) proved the need of measuring simultaneously the quality of the samples generated and the mode collapse. Sajjadi et al. (2018) proposed the use of a PR metric to measure the quality of GANs. The key intuition is that precision should quantify how much of the fake distribution can be generated by the true distribution while recall measures how much of the true distribution can be re-constructed by the model distribution. More formally, it is defined as follows:

Definition 3.2.1. *(Sajjadi et al., 2018) Let X, Y be two random variables. For $\alpha, \beta \in (0,1]$, X is said to have an attainable precision α at recall β w.r.t. Y if there exists probability distributions μ, ν_X, ν_Y such that*

$$Y = \beta\mu + (1-\beta)\nu_Y \quad \textit{and} \quad X = \alpha\mu + (1-\alpha)\nu_X.$$

The component ν_Y denotes the part of Y that is "missed" by X, whereas, ν_X denotes the "noise" part of X. We denote $\bar{\alpha}$ (respectively $\bar{\beta}$) the maximum attainable precision (respectively recall). Th. 1 of Sajjadi et al. (2018) states:

$$X(S_Y) = \bar{\alpha} \quad \text{and} \quad Y(S_X) = \bar{\beta}.$$

Improved PR metric. Kynkäänniemi et al. (2019) highlighted an important drawback of the PR metric proposed by Sajjadi et al. (2018): it cannot correctly interpret situations when a large numbers of samples are packed together. To better understand this situation, consider a case where the generator slightly collapses on a specific data point, i.e. there exists $x \in E, \mu_\theta(x) > 0$. We show in Appendix A.1.1 that if $\mu_\star$ is a non-atomic probability measure and μ_θ is highly precise (*i.e.* $\alpha = 1$), then the recall β must be 0.

To solve these issues, Kynkäänniemi et al. (2019) proposed an *Improved Precision-Recall* (Improved PR) metric built on a nonparametric estimation of support of densities.

Definition 3.2.2. *(Kynkäänniemi et al., 2019) Let X, Y be two random variables and D_X, D_Y two finite sample datasets such that $D_X \sim X^n$ and $D_Y \sim Y^n$. For any $x \in D_X$ (respectively for any $y \in D_Y$), we consider $(x_{(1)}, \dots, x_{(n-1)})$, the re-ordering of elements in $D_X \setminus x$ given their euclidean distance with x. For any $k \in \mathbb{N}$ and $x \in D_X$, the precision $\alpha_k^n(x)$ of point x is defined as:*

$$\alpha_k^n(x) = 1 \iff \exists y \in D_Y, \|x-y\| \leqslant \|y_{(k)} - y\|.$$

Similarly, the recall $\beta_k^n(y)$ of any given $y \in D_Y$ is:

$$\beta_k^n(y) = 1 \iff \exists x \in D_X, \|y-x\| \leqslant \|x_{(k)} - x\|.$$

Improved precision (respectively recall) are defined as the average over D_X (respectively D_Y) as follows:

$$\alpha_k^n = \frac{1}{n}\sum_{x_i \in D_X} \alpha_k^n(x_i) \qquad \beta_k^n = \frac{1}{n}\sum_{y_i \in D_Y} \beta_k^n(y_i).$$

A first contribution is to formalize the link between PR and Improved PR with the following theorem:

Theorem 3.2.1. *Let X, Y two random variables with probability distributions μ and ν. Assume that both μ and ν are associated with uniformly continuous probability density functions f_μ and f_ν. Besides, there exists constants $a_1 > 0, a_2 > 0$ such that for all $x \in E$ we have $a_1 < f_{\mu_\star}(x) \leqslant a_2$ and $a_1 < f_{\mu_\theta}(x) \leqslant a_2$ for some $c > 0$. Also, (k,n) are such that $\frac{k}{\log(n)} \to +\infty$ and $\frac{k}{n} \to 0$. Then,*

$$\alpha_k^n \to \bar{\alpha} \ \textit{ in probability} \quad \textit{and} \quad \beta_k^n \to \bar{\beta} \ \textit{ in probability}.$$

This theorem, whose proof is delayed to Appendix A.1.2, underlines the nature of the Improved PR metric: the metric compares the supports of the modeled probability distribution μ_θ and of the true distribution $\mu_\star$. This means that Improved PR is a tuple made of both maximum attainable precision $\bar{\alpha}$ and recall $\bar{\beta}$ (e.g. Theorem 1 of Sajjadi et al. (2018)). As Improved PR is shown to have a better performance evaluating GANs sample quality, we use this metric for both the following theoretical results and experiments.

3.2.3.3 Learning disconnected manifolds

In this section, we aim to stress the difficulties of learning disconnected manifolds with standard GANs architectures. To begin with, we recall the following lemma.

Lemma 3.2.1. *Assume that Assumptions 1 and 2 are satisfied. Then, for any $\theta \in \Theta$, the support S_{μ_θ} is connected.*

There is consequently a discrepancy between the connectedness of S_{μ_θ} and the disconnectedness of $S_{\mu_\star}$. In the case where the manifold lays on two disconnected components, our next theorem exhibit a no free lunch theorem:

Theorem 3.2.2. *("No free lunch" theorem) Assume that Assumptions 1 and 2 are satisfied. Assume also that true distribution $\mu_\star$ lays on two equally measured disconnected manifolds distant from a distance $D > 0$. Then, any estimator μ_θ that samples equally in both modes must have a precision $\bar{\alpha}$ such that $\bar{\alpha} + \frac{D}{\sqrt{2\pi}L} e^{\frac{-\Phi^{-1}(\frac{\bar{\alpha}}{2})^2}{2}} \leqslant 1$, where Φ is the c.d.f. of a standard normal distribution.*

Besides, if $\bar{\alpha} \geqslant 3/4$, $\bar{\alpha} \lesssim 1 - \sqrt{\frac{2}{\pi}} W(\frac{D^2}{4L^2})$ where W is the Lambert W function.

The proof of this theorem is delayed to Appendix A.1.3. It is mainly based on the Gaussian isoperimetric inequality (Borell, 1975; Sudakov and Tsirelson, 1978) that states that among all sets of given Gaussian measure in any finite dimensional Euclidean space, half-spaces have the minimal Gaussian boundary measure. If in Figure 3.1, the generator has thus learned the optimal separation, it is not known clear how to enforce such geometrical properties in the latent space. We will show, in the next section, a method that indeed enforces a well-structured latent space.

In real world applications, when the number of distinct sub-manifolds increases, we expect the volume of these boundaries to increase with respect to the number of different classes covered by the modeled distribution μ_θ. Going in this direction, we better formalize this situation, and show an extended "no free lunch theorem" by expliciting an upper-bound of the precision $\bar{\alpha}$ in this broader framework.

Assumption 3. *The true distribution $\mu_\star$ lays on M equally-measured disconnected components at least distant from some constant $D > 0$.*

This is likely to be true for datasets made of symbol designed to be highly distinguishable (*e.g.* digits in the MNIST dataset). In very high dimension, this assumption also holds for complex classes of objects appearing in many different contexts (*e.g.* the bubble class in ImageNet, see Appendix).

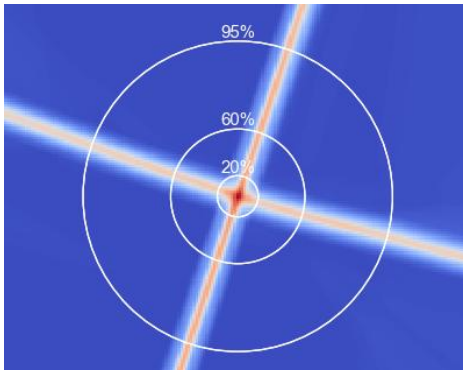
To better apprehend the next theorem, note A_m the pre-image in the latent space of mode m and A_m^r its r-enlargement: $A_m^r := \{z \in \mathbb{R}^d \mid \mathsf{dist}(\mathsf{z}, \mathsf{A_m}) \leq \mathsf{r}\}, \mathsf{r} > 0$.

Theorem 3.2.3. *(Generalized "no free lunch" theorem) Assume that Assumptions 1, 2, and 3 are satisfied, and that the pre-image enlargements A_m^ε, with $\varepsilon = \frac{D}{2L}$, form a partition of the latent space with equally measured elements.*

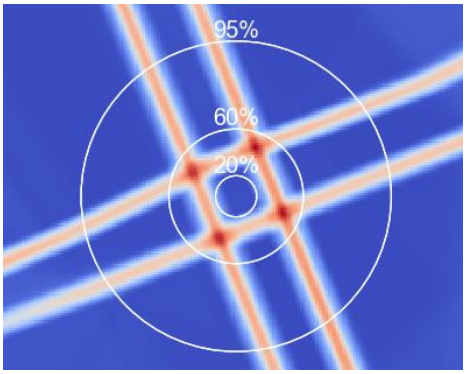
Then, any estimator μ_θ with recall $\bar{\beta} > \frac{1}{M}$ must have a precision $\bar{\alpha}$ at most $\frac{1+x^2}{x^2} e^{-\frac{1}{2}\varepsilon^2} e^{-\varepsilon x}$ where $x = \Phi^{-1}(1 - \frac{1}{\bar{\beta} M})$ and Φ is the c.d.f. of a standard normal distribution.

Theorem 3.2.3, whose proof is delayed to Appendix A.1.4, states a lower-bound the measure of samples mapped out of the true manifold. We expect our bound to be loose since no theoretical results are known, to the best of our knowledge, on the geometry of the separation that minimizes the boundary between different classes (when $M \geqslant 3$). Finding this optimal cut would be an extension of the honeycomb theorem Hales (2001). In Appendix A.1.4.2 we give a more technical statement of Theorem 3.2.3 without assuming equality of measure of the sets A_m^ε.

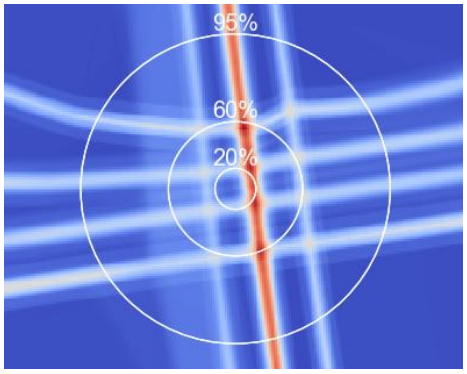
The idea of the proof is to consider the border of an individual cell with the rest of the partition. It is clear that at least half of the frontier will be inside this specific cell. Then, to



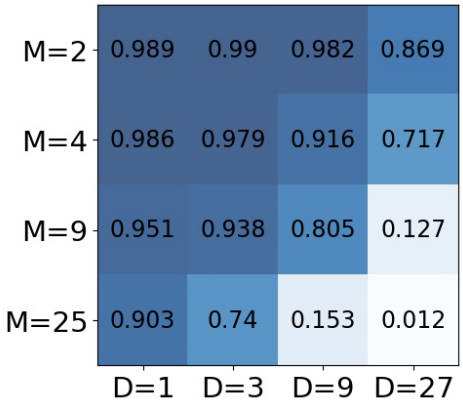
(a) WGAN 4 classes: visualisation of $\|J_G(z)\|_F$.



(b) WGAN 9 classes: visualisation of $\|J_G(z)\|_F$.



(c) WGAN 25 classes: visualisation of $\|J_G(z)\|_F$.



(d) Precision w.r.t. D (mode distance) and M (classes).

Fig. 3.2 Illustration of Theorem 3.2.3. If the number of classes $M \to \infty$ or the distance $D \to \infty$, then the precision $\bar{\alpha} \to 0$. We provide in appendix heatmaps for more values of M.

get to the final result, we sum the measures of the frontiers contained inside all of the different cells. Remark that our analysis is fine enough to keep a dependency in M which translates into a maximum precision that goes to zero when M goes to the infinity and all the modes are covered. More precisely, in this scenario where all pre-images have equal measures in the latent space, one can derive the following bound, when the recall $\bar{\beta}$ is kept fixed and M increases:

$$\bar{\alpha} \overset{M\to\infty}{\leqslant} e^{-\frac{1}{2}\varepsilon^2} e^{-\varepsilon\sqrt{2\log(\bar{\beta}M)}} \quad \text{where } \varepsilon = \frac{D}{2L}. \tag{3.2.1}$$

For a fixed generator, this equation illustrates that the precision $\bar{\alpha}$ decreases when either the distance D (equivalently ε) or the number of classes M increases. For a given ε, $\bar{\alpha}$ converges to 0 with a speed $O(\frac{1}{(\bar{\beta}M)^{\sqrt{2}\varepsilon}})$. To better illustrate this asymptotic result, we provide results from a 2D synthetic setting. In this toy dataset, we control both the number M of disconnected manifolds and the distance D. Figure 3.2 clearly corroborates (3.2.1) as we can easily get the maximum precision close to 0 ($M = 25$, $D = 27$).

3.2.3.4 Jacobian-based truncation (JBT) method

The analysis of the deformation of the latent space offers a grasp on the behavior of GANs. For instance, Arvanitidis et al. (2018) propose a distance accounting for the distortions made by the generator. For any pair of points $(z_1, z_2) \sim Z^2$, the distance is defined as the length of the geodesic $d(z_1, z_2) = \int_{[0,1]} \|J_{G_\theta}(\gamma_t)\frac{d\gamma_t}{dt}\| dt$ where γ is the geodesic parameterized by $t \in [0,1]$ and $J_{G_\theta}(z)$ denotes the Jacobian matrix of the generator at point z. Authors have shown that the use of this distance in the latent space improves clustering and interpretability. We make a similar observation that the generator's Jacobian Frobenius norm provides meaningful information.

Indeed, the frontiers highlighted in Figures 3.2a, 3.2b, and 3.2c correspond to areas of low precision mapped out of the true manifold: this is the *no GAN's land*. We argue that when learning disconnected manifolds, the generator tries to minimize the number of samples that do not belong to the support of the true distribution and that this can only be done by making paths steeper in the *no GAN's land*. Consequently, data points $G_\theta(z)$ with high Jacobian Frobenius norm (JFN) are more likely to be outside the true manifold. To improve the precision of generative models, we thus define a new truncation method by removing points with highest JFN.

However, note that computing the generators's JFN is expensive to compute for neural networks, since being defined as follows,

$$\|J_{G_\theta}(z)\|_F^2 = \sum_{i=1}^{m}\sum_{j=1}^{n}\left(\frac{\partial G_\theta(z)_i}{\partial z_j}\right)^2,$$

it requires a number of backward passes equal to the output dimension. To make our truncation method tractable, we use a stochastic approximation of the Jacobian Frobenius norm based on the following result from Rifai et al. (2011):

$$\|J_{G_\theta}(z)\|^2 = \lim_{\substack{N\to\infty \\ \sigma\to 0}} \frac{1}{N}\sum_{\varepsilon_i}^{N} \frac{1}{\sigma^2}\|G_\theta(z+\varepsilon_i) - G_\theta(z)\|^2,$$

where $\varepsilon_i \sim\sim \mathcal{N}(0, \sigma^2 I$ and I is the identity matrix of dimension d. The variance σ of the noise and the number of samples are used as hyper-parameters. In practice, σ in $[1e{-}4; 1e{-}2]$ and $N = 10$ give consistent results.

Based on the preceding analysis, we propose a new **Jacobian-based truncation** (JBT) method that rejects a certain ratio of the generated points with highest JFN. This truncation ratio is considered as an hyper-parameter for the model. We show in our experiments that our JBT can be used to to detect samples outside the real data manifold and that it consequently improves the precision of the generated distribution as measured by the Improved PR metric.

3.2.4 Experiments

In the following, we show that our truncation method, JBT, can significantly improve the performances of generative models on several models, metrics and datasets. Furthermore, we compare JBT with over-parametrization techniques specifically designed for disconnected manifold learning. We show that our truncation method reaches or surpasses their performance, while it has the benefit of not modifying the training process of GANs nor using a mixture of generators, which is computationally expensive. Finally, we confirm the efficiency of our method by applying it on top of BigGAN (Brock et al., 2019).

Except for BigGAN, for all our experiments, we use Wasserstein GAN with gradient penalty (Gulrajani et al., 2017), called WGAN for conciseness. We give in Appendix A.3 the full details of our experimental setting. The use of WGAN is motivated by the fact that it was shown to stabilize the training and significantly reduce mode collapse (Arjovsky and Bottou, 2017). However, we want to emphasise that our method can be plugged on top of any generative model fitting disconnected components.

3.2.4.1 Evaluation metrics

To measure performances of GANs when dealing with low dimensional applications - as with synthetic datasets - we equip our space with the standard Euclidean distance. However, for high dimensional applications such as image generation, Brock et al. (2019); Kynkäänniemi et al. (2019) have shown that embedding images into a feature space with a pre-trained convolutional

classifier provides more semantic information. In this setting, we consequently use the euclidean distance between the images' embeddings from a classifier. For a pair of images (a,b), we define the distance $d(a,b)$ as $d(a,b) = \|\phi(a) - \phi(b)\|_2$ where ϕ is a pre-softmax layer of a supervised classifier, trained specifically on each dataset. Doing so, they will more easily separate images sampled from the true distribution $\mu_\star$ from the ones sampled by the distribution μ_θ.

We compare performances using Improved PR (Kynkäänniemi et al., 2019). We also report the *Marginal Precision* which is the precision of newly added samples when increasing the ratio of kept samples. Besides, for completeness, we report FID (Heusel et al., 2017) and recall precise definitions in Appendix A.2.2. Note that FID was not computed with InceptionNet, but a classifier pre-trained on each dataset.

3.2.4.2 Synthetic dataset

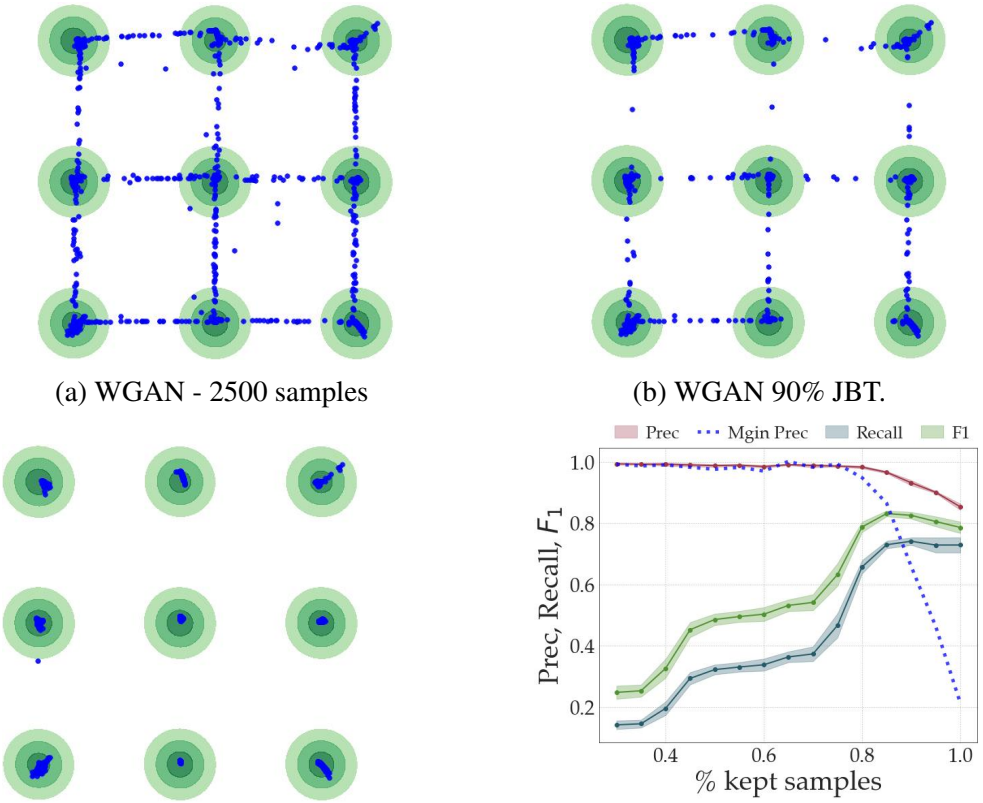
We first consider the true distribution to be a 2D Gaussian mixture of 9 components. Both the generator and the discriminator are modeled with feed-forward neural networks.

Interestingly, the generator tries to minimize the sampling of off-manifolds data during training until its JFN gets saturated (see Appendix A.2.3). One way to reduce the number of off-manifold samples is to use JBT. Indeed, off-manifold data points progressively disappear when being more and more selective, as illustrated in Figure 3.3c. We quantitatively confirm that our truncation method (JBT) improves the precision. On Figure 3.3d, we observe that keeping the 70% of lowest JFN samples leads to an almost perfect precision of the support of the generated distribution. Thus, off-manifold samples are in the 30% samples with highest JFN.

3.2.4.3 Image datasets

We further study JBT on three different datasets: MNIST (LeCun et al., 1998), FashionMNIST (Xiao et al., 2017) and CIFAR10 (Krizhevsky et al., 2009). Following (Khayatkhoei et al., 2018) implementation, we use a standard CNN architecture for MNIST and FashionMNIST while training a ResNet-based model for CIFAR10 (Gulrajani et al., 2017).

Figure 3.4 highlights that JBT also works on high dimensional datasets as the marginal precision plummets for high truncation ratios. Furthermore, when looking at samples ranked by increasing order of their JFN, we notice that samples with highest JFN are standing in-between manifolds. For example, those are ambiguous digits resembling both a "0" and a "6" or shoes with unrealistic shapes.



(a) WGAN - 2500 samples

(b) WGAN 90% JBT.

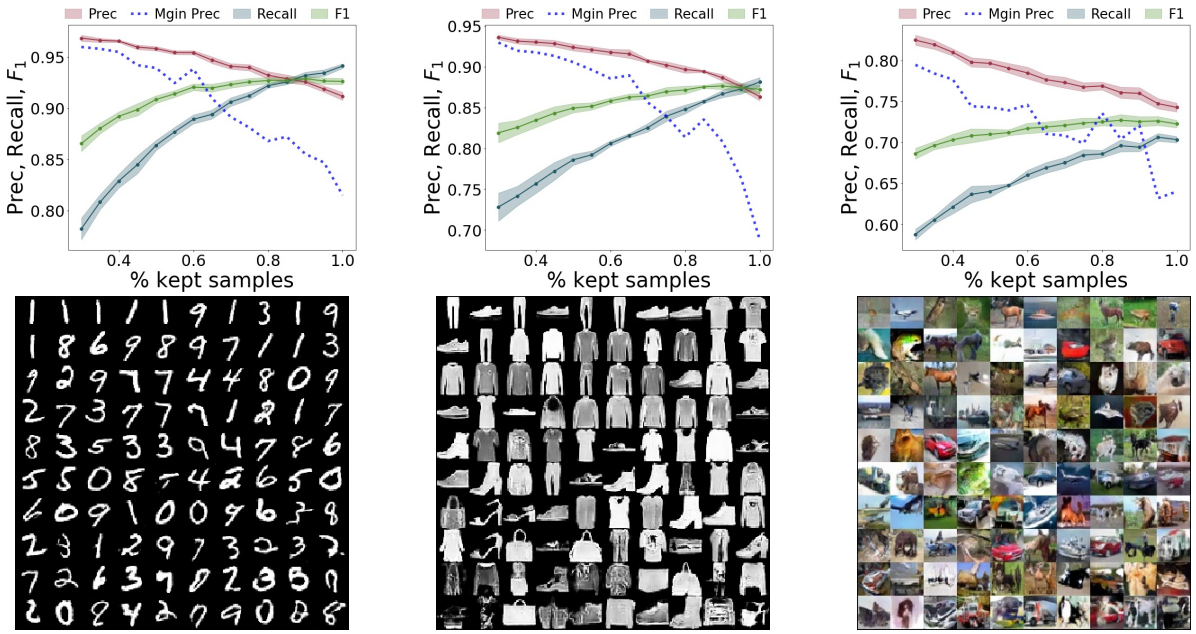
(c) WGAN 70% JBT.

(d) 97% confidence intervals .

Fig. 3.3 Mixture of 9 Gaussians in green, generated points in blue. Our truncation method (JBT) removes least precise data points as marginal precision plummets.

To further assess the efficiency of our truncation method, we also compare its performances with two state-of-the-art over-parameterization techniques that were designed for disconnected manifold learning. First, Gurumurthy et al. (2017) propose DeliGAN, a reparametrization trick to transform the unimodal Gaussian latent distribution into a mixture. The different mixture components are later learnt by gradient descent. For fairness, the re-parametrization trick is used on top of WGAN. Second, Khayatkhoei et al. (2018) define DMLGAN, a mixture of generators to better learn disconnected manifolds. In this architecture, each generator is encouraged to target a different submanifold by enforcing high mutual information between generated samples and generator's ids. Keep in mind that for DeliGAN (respectively DMLGAN), the optimal number of components (respectively generators) is not known and is a hyper-parameter of the model that has to be cross-validated.

The results of the comparison are presented in Table 3.1. In both datasets, JBT 80 % outperforms DeliGAN and DMLGAN in terms of precision while keeping a reasonnable recall. This confirms our claim that over-parameterization techniques are unnecessary. As noticed

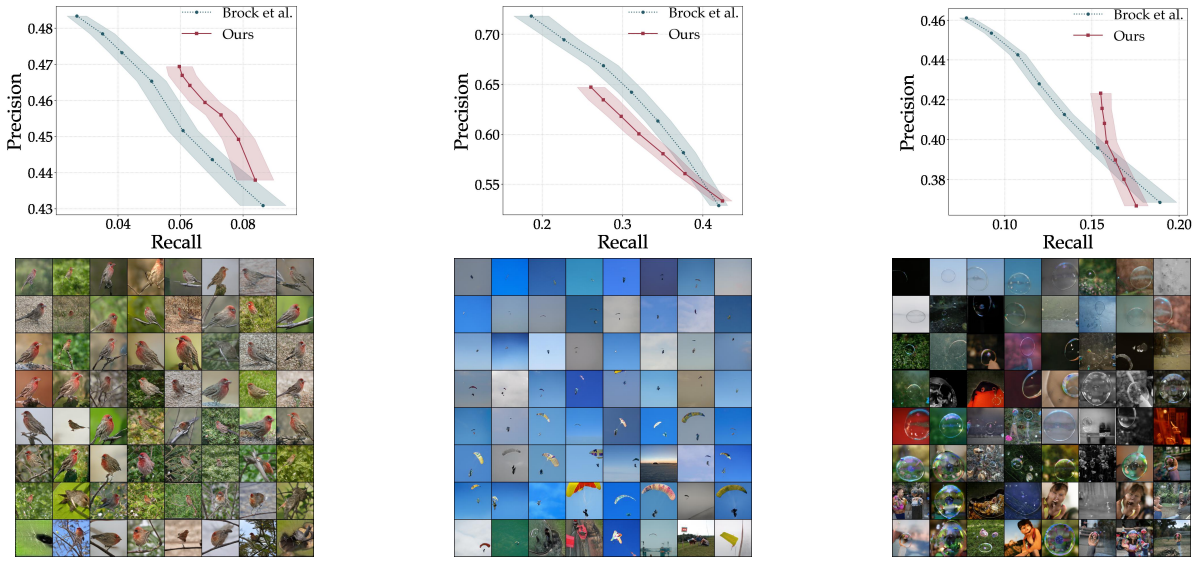


(a) MNIST dataset. (b) F-MNIST dataset. (c) CIFAR10 datatset.

Fig. 3.4 For high levels of kept samples, the marginal precision plummets of newly added samples, underlining the efficiency of our truncation method (JBT). Reported confidence intervals are 97% confidence intervals. On the second row, generated samples ordered by their JFN (left to right, top to bottom). In the last row, the data points generated are blurrier and outside the true manifold.

Table 3.1 JBT $x\%$ means we keep the $x\%$ samples with lowest Jacobian norm. Our truncation method (JBT) matches over-parameterization techniques. $\pm$ is 97% confidence interval.

MNIST	Prec.	Rec.	FID
WGAN	91.2±0.3	**93.7±0.5**	24.3±0.3
WGAN JBT 90%	92.5±0.5	92.9±0.3	26.9±0.5
WGAN JBT 80%	**93.3±0.3**	91.8±0.4	33.1±0.3
W-Deligan	89.0±0.6	**93.6±0.3**	31.7±0.5
DMLGAN	**93.4±0.2**	92.3±0.2	**16.8±0.4**
F-MNIST			
WGAN	86.3±0.4	**88.2±0.2**	259.7±3.5
WGAN JBT 90%	88.6±0.6	86.6±0.5	**257.4±3.0**
WGAN JBT 80%	**89.8±0.4**	84.9±0.5	396.2±6.4
W-Deligan	88.5±0.3	85.3±0.6	310.9±3.1
DMLGAN	87.4±0.3	**88.1±0.4**	**253.0±2.8**



(a) House finch. (b) Parachute. (c) Bubble.

Fig. 3.5 On the first row, per-class precision-recall curves comparing Brock et al. (2019)'s truncation trick and our truncation method (JBT), on three ImageNet classes generated by BigGAN. We show better results on complex and disconnected classes (*e.g.* bubble). Reported confidence intervals are 97% confidence intervals. On the second row, generated samples ordered by their JFN (left to right, top to bottom). We observe a concentration of off-manifold samples for images on the bottom row, confirming the soundness of JBT.

by Kynkäänniemi et al. (2019), we also observe that FID does not correlate properly with the Improved PR metric. Based on the Frechet distance, only a distance between multivariate Gaussians, we argue that FID is not suited for disconnected manifold learning as it approximates distributions with unimodal ones and looses many information.

3.2.4.4 Spurious samples rejections on BigGAN

Thanks to the simplicity of JBT, we can also apply it on top of any trained generative model. In this subsection, we use JBT to improve the precision of a pre-trained BigGAN model (Brock et al., 2019), which generates class-conditionned ImageNet (Krizhevsky et al., 2012) samples. The class-conditioning lowers the problem of off-manifold samples, since it reduces the disconnectedness in the output distribution. However, we argue that the issue can still exist on high-dimensional natural images, in particular complex classes can still be multi-modal (*e.g.* the bubble class). The bottom row in Figure 3.5 shows a random set of 128 images for three different classes ranked by their JFN in ascending order (left to right, top to bottom). We observe a clear concentration of spurious samples on the bottom row images.

To better assess the Jacobian based truncation method, we compare it with the truncation trick from Brock et al. (2019). This truncation trick aims to reduce the variance of the latent space distribution using truncated Gaussians. While easy and effective, this truncation has some issues: it requires to complexify the loss to enforce orthogonality in weight matrices of the network. Moreover, as explained by Brock et al. (2019) *"only 16% of models are amenable to truncation, compared to 60% when trained with Orthogonal Regularization"*. For fairness of comparison, the pre-trained network we use is optimized for their truncation method. On the opposite, JBT is simpler to apply since 100% of the tested models were amenable to the proposed truncation.

Results of this comparison are shown in the upper row of Figure 3.5. Our method can outperform their truncation trick on difficult classes with high intra-class variation, *e.g.* bubble and house finch. This confirms our claim that JBT can detect outliers within a class. However, one can note that their trick is particularly well suited for simpler unimodal classes, *e.g.* parachute and reaches high precision levels.

3.2.5 Conclusion

In this section, we provide insights on the learning of disconnected manifolds with push-forward generative models. Our analysis shows the existence of an off-manifold area with low precision. We empirically show on several datasets and GAN models that we can detect these areas and remove samples located in between two modes thanks to a newly proposed truncation method.

However, we do not have yet a clear idea of the geometrical structure of a generator's latent space. When there are two modes in the target distribution, it is clear that the best way to split the latent space is to use half-spaces (according to the standard Gaussian isoperimetric inequality). But when there are more than two modes, what is an optimal geometrical structure? In the next section, we provide an explicit characterization of an optimal latent space geometry. Moreover, we investigate if this structure is indeed learned by generative models, and if it is possible to enforce it.

3.3 Unveiling the latent space geometry of push-forward generative models

Abstract. Many deep generative models are defined as a push-forward of a Gaussian measure by a continuous generator, such as Generative Adversarial Networks (GANs) or Variational Auto-Encoders (VAEs). This work explores the latent space of such deep generative models. A key issue with these models is their tendency to output samples outside of the support of the target distribution when learning disconnected distributions. We investigate the relationship between the performance of these models and the geometry of their latent space. Building on recent developments in geometric measure theory, we prove a sufficient condition for optimality in the case where the dimension of the latent space is larger than the number of modes. Through experiments on GANs, we demonstrate the validity of our theoretical results and gain new insights into the latent space geometry of these models. Additionally, we propose a truncation method that enforces a simplicial cluster structure in the latent space and improves the performance of GANs.

3.3.1 Introduction

GANs (Goodfellow et al., 2014) and VAEs (Kingma and Welling, 2014) have shown great capacities to generate photorealistic images (Karras et al., 2021; Vahdat and Kautz, 2020). These two models are also helpful for diverse tasks such as image editing (Shen et al., 2020; Wu et al., 2021) or unsupervised image segmentation (Abdal et al., 2021; Zoran et al., 2021). GANs and VAEs rely on learning a Lipschitz-continuous transformation from a low dimensional Gaussian space. As such, they have been described as *push-forward generative models* (Salmona et al., 2022). According to the same taxonomy, score-based models can be defined as *indirect push-forward generative models* since they result from the composition of a large number of transformations and are trained with an auxiliary denoising objective.

The present section aims at making a step towards a better understanding of push-forward generative models such as GANs. In particular, the goal is to shed light on the latent space of these architectures, and to stress how it impacts the performance of both GANs and VAEs. If empirical studies such as Donahue and Simonyan (2019) have suggested the emergence of simple geometrical structure in the latent space of GANs, there is still a poor theoretical understanding of how generators organize their latent space.

To better understand the latent space of generative models, the setting of disconnected distributions learning is enlightening. Experimental and theoretical works (Khayatkhoei et al., 2018; Salmona et al., 2022) have shown a fundamental limitation of push-forward generative

models. Since the modeled distribution is connected, some areas of its support are necessarily mapped outside the true data distribution. However, when covering several modes of a disconnected distribution, generators still try to minimize the numbers of samples lying outside the true modes (*e.g.* the purple area on the right of Figure 3.6). In other words, generators aim at minimizing the measure of the existing borders between the modes in the latent space. Considering a Gaussian latent space, finding such minimizers is closely linked to Gaussian isoperimetric inequalities (Ledoux, 1996) where the goal is to derive the partitions that split a Gaussian space with minimal Gaussian-weighted perimeters. Most notably, a recent result (Milman and Neeman, 2022) shows that, as long as the number of components m in the partition and the number of dimensions d of the Gaussian space are such that $m \leq d+1$, the optimal partition is a 'simplicial cluster': a Voronoi diagram with equidistant seeds, see left of Figure 3.6 for $m = 3$ and $d = 3$.

In this section, we demonstrate the effectiveness of applying simplicial clusters to the latent space of push-forward generative models. We show both experimentally and theoretically that generators with a latent space structured as a simplicial cluster minimize the occurrence of out-of-distribution generated samples. Using the *precision* metric (Sajjadi et al., 2018; Kynkäänniemi et al., 2019), we show that generators with a simplicial cluster latent space achieve optimal precision levels and provide both an upper and a lower bound on their precision. Our experiments reveal that GANs with higher performances tend to organize their latent space as simplicial clusters. More importantly, we illustrate that enforcing this 'simplicial structure' with a truncation method can boost GANs' performance. Interestingly, simplicial clusters are highly similar to the 'simplex Equiangular Tight Frames' observed in the last-layer features of deep classification networks (Papyan et al., 2020). This study stresses that they also naturally emerge in deep push-forward generative models. Our contributions are the following:

- We are the first to build on the latest results from Gaussian isoperimetric inequalities by Milman and Neeman (2022) in the study and understanding of push-forward generative models.
- We present a new theoretical analysis, providing both an upper bound on the precision of push-forward generative models. We demonstrate that generators with a latent space organized as a simplicial cluster have an optimal precision, with lower bounds that decrease in $\sqrt{m \log m}$, where m is the number of modes.
- Experimentally, we verify that GANs tend to structure their latent space as simplicial clusters' by exploring two properties of the latent space: linear separability and convexity of classes. Also, we analyse the impact of latent space dimension on GANs, and reveal a positive correlation between GANs' performance and latent space geometry.

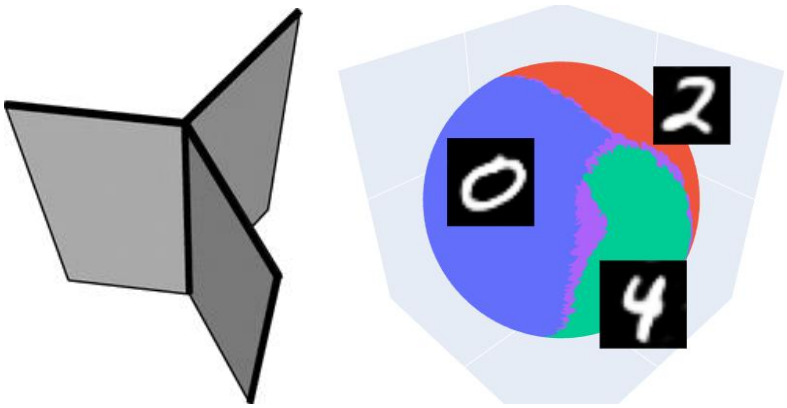


Fig. 3.6 Illustration of the capability of GANs to discover an optimal geometry of the latent space. On the left, the propeller shape represents a partition of 3D Gaussian space with the smallest Gaussian-weighted perimeter (Figure from Heilman et al. (2012)). On the right, we show the 3D Gaussian latent space of a GAN trained on three classes of MNIST. Each area colored in blue, green, or red corresponds to samples in one of the three classes. Using a pre-trained classifier, we highlight in purple the samples with low-confidence, and observe that the partition reached by the GAN (right) is close to optimality (left), as the latent space partition is similar to the intersection of the propeller on a sphere.

- Finally, we show that enforcing a simplicial structure into GANs' latent space can boost their performance and outperforms other boosting methods.

3.3.2 Related Work

3.3.2.1 Notation

Data. We consider a target distribution $\mu_\star$ defined on a Euclidean space $\mathbb{R}^D$, which may be a high-dimensional space, and equipped with the Euclidean norm $\|\cdot\|$. We use S_μ to represent the support of any distribution μ.

Push-forward generative models. We consider the set of L-Lipschitz continuous functions, denoted as $\mathscr{G}_L$, from the latent space $\mathbb{R}^d$ to the high-dimensional space $\mathbb{R}^D$. The primary goal of each generator in this set is to produce realistic samples. The distribution in the latent space, defined on $\mathbb{R}^d$, is assumed to be Gaussian and is represented as γ. For each generator $G \in \mathscr{G}_L$, we associate the push-forward distribution (or image distribution) of γ by G, and denote it $G\sharp\gamma$, where $\sharp$ denotes the push-forward operator. In the context of generative models, each distribution $G\sharp\gamma$ is now a candidate distribution to represent $\mu_\star$.

The Lipschitzness assumption on $\mathscr{G}_L$ is reasonnable: Virmaux and Scaman (2018) have shown the lipschitzness of deep neural networks, and have developed an algorithm that can

upper-bound their Lipschitz constant. While deep neural networks can have high Lipschitz constants, it is possible to constrain this in practice by techniques such as clipping the neural network's parameters (Arjovsky et al., 2017), penalizing the discriminative functions' gradient (Gulrajani et al., 2017; Kodali et al., 2017; Wei et al., 2018; Zhou et al., 2019), or penalizing the spectral norms (Miyato et al., 2018). Large-scale generators such as SAGAN (Zhang et al., 2019) and BigGAN (Brock et al., 2019) also make use of spectral normalization for the generator.

3.3.2.2 Generative models and disconnected distributions

The phenomenon of misspecification in continuous generative models, while primarily studied in the context of GANs, is also relevant to other families such as VAEs or normalizing flows (Salmona et al., 2022). This issue has been investigated both experimentally (Khayatkhoei et al., 2018) and theoretically (Salmona et al., 2022). The problem stems from a fundamental trade-off: continuous generators can either cover all modes, resulting in out-of-manifold samples, or generate only high-quality samples, neglecting some modes. To address this, various methods have been proposed, such as training disconnected distributions (Gurumurthy et al., 2017; Khayatkhoei et al., 2018) or deriving rejection mechanisms from pre-trained generators (Azadi et al., 2019; Humayun et al., 2022).

Empirical studies have provided valuable insights into the structure of the latent space of generative models. For example, Karras et al. (2019b) demonstrate that binary attributes are linearly separable in the Gaussian latent space and even more separable in an intermediate latent space. Similarly, Shen et al. (2020) find that face attributes are separated by hyperplanes in the latent space.Arvanitidis et al. (2018) and Chen et al. (2018a) view the latent space of generative models with a Riemannian perspective.

While these findings provide valuable insights into the latent space structure of generative models, they may not be sufficient for a comprehensive understanding of the latent space geometry. For instance, In section 3.2, we stress the relevance of this problem by showing that the precision of GANs can converge to 0 when the number of modes or the distance between them increases. In this section, we take a step towards a deeper understanding of the behavior of push-forward generative models and reveal an optimal latent space configuration when the number of modes m and the dimension of the latent space d are such that $m \leq d+1$.

3.3.2.3 Evaluating generative models

When learning disconnected manifolds, Sajjadi et al. (2018) illustrated the need for measures that simultaneously evaluate both the quality (Precision), and the diversity (Recall) of the

generated samples. However, Kynkäänniemi et al. (2019) pointed out an important limitation of the PR metric: it cannot accurately interpret situations when large numbers of samples are packed together. They propose an Improved PR metric based on the non-parametric estimation of manifolds to correct this.

Improved PR metric. Informally, for a generator G, precision (α_G) quantifies the proportion of generated samples that can be approximated with true samples, while recall (β_G) measures the proportion of true samples that can be approximated with generated ones. Applying this to GANs, using the target distribution $\mu^\star$ and modeled distribution $G\sharp\gamma$, the Improved PR metric was shown, by Theorem 3.2.1, to be asymptotically equivalent to:

$$\alpha_G^n \underset{n\to\infty}{\to} \alpha_G = G\sharp\gamma\big(S_{\mu^\star}\big) \text{ and } \beta_G^n \underset{n\to\infty}{\to} \beta_G = \mu^\star\big(S_{G\sharp\gamma}\big),$$

where $S_{\mu^\star}$ denotes the support of $\mu^\star$ and n is the number of samples. However, Naeem et al. (2020) have shown that the Improved PR metric (Kynkäänniemi et al., 2019) is sensitive to outlier samples of both the target and the generated distribution. To correct this and fix the overestimation of the manifold around real outliers, Naeem et al. (2020) propose the Density/Coverage metric.

Density/Coverage. Instead of counting how many fake samples belong to a real sample neighborhood, density counts how many real sample neighborhoods contain a generated sample. On the other hand, coverage counts the number of real sample neighborhoods that contain at least one fake sample.

In the next analysis both theoretical and experimental, we use both notions of precision and density defined above.

3.3.3 Simplicial Structure in Push-Forward Generative Models

The goal is to gain a deeper understanding of the latent space of push-forward generative models and identify which ones possess the highest precision under certain conditions. As previously mentioned, push-forward generative models map a unimodal Gaussian distribution γ through a Lipschitz-continuous function, represented by a generator G. As a result, the modeled generative distribution $G\sharp\gamma$ necessarily has a connected support.

In cases where the target distribution $\mu^\star$ contains disconnected manifolds, generators have to generate fake data points that fall outside of the true manifold. This prompts the question: given that a generator samples data points from each of the distinct modes, what is the maximum

precision that it can achieve? To begin with, let's consider a target distribution $\mu^\star$ composed of m disconnected modes.

Assumption 4 (Disconnected manifolds)**.** *The target distribution $\mu^\star$ consists of m disconnected spheres $S_i, i \in [1,m]$ of equal measure (with centers X_i and radius r_i). Additionally, the spheres satisfy the two following properties:*

- ***Small individual radius:*** *each radius r_i satisfies*

$$r_i < \min_j \frac{\|X_i - X_j\|}{2}. \tag{3.3.1}$$

- *Each distance $\|X_i - X_j\|$ satisfies:*

$$\min_{k\in[1,m],k\neq i,j} \|(X_i + X_j)/2 - X_k\| > \frac{\|X_i - X_j\|}{2}. \tag{3.3.2}$$

We believe that the assumption of disconnectedness is a reasonable one, particularly for multi-class datasets such as MNIST LeCun et al. (1998), CIFAR10 Krizhevsky et al. (2009), or STL10 Coates et al. (2011). To validate this property, we run a pre-trained CLIP (Radford et al., 2021) on the dataset, identify a certain number of clusters using a K-means algorithm, and further test the disconnectedness of these modes by training a linear classifier. The accuracy on these datasets is 98.1% on MNIST, 93.9% on CIFAR10, and 92.7% on STL10.

The second point in (3.3.2) has a direct impact on the location of the data points $X_1, \ldots, X_m$. Specifically, it implies that each cell in the Voronoi diagram with seeds $X_1, \ldots, X_m$ shares a side with all the other cells. In other words, the dual graph of this Voronoi diagram is complete. This assumption, which is further discussed with specific examples in Figure 3.7, can be justified by the concentration of distances in high-dimensional spaces: all the modes are roughly at equal distance (Beyer et al., 1999; Aggarwal et al., 2001). Furthermore, a recent work by Papyan et al. (2020) has shown that embeddings of deep neural networks trained for classification tend to collapse around means that are equidistant and maximally equiangular to one another. By using these embedded representations to measure distance, the target distribution would thus easily satisfy Assumption 4. Projected GANs (Sauer et al., 2021) is really close to this idea as the authors show the effectiveness of leveraging a pre-trained classifier when training GANs: instead of directly discriminating images, the discriminator is trained on features extracted from the classifier.

Throughout the rest of the section, we define the set of *well-balanced* generators as those mapping an equal number of data points to each mode of the data distribution:

Definition 3.3.1. *A generator G is well-balanced if for all spheres, we have $G\sharp\gamma(S_1) = \ldots = G\sharp\gamma(S_m)$.*

Considering well-balanced generators is reasonable as many empirical improvements such as WGAN-GP (Gulrajani et al., 2017) or BigBiGAN (Donahue and Simonyan, 2019) have significantly reduced mode collapse. GANs generate diverse output distributions on datasets such as CIFAR10, CIFAR100, and ImageNet. To validate the use of well-balanced generators, we conducted a small experiment and evaluated the proportion of each class generated by GANs on MNIST and CIFAR10. On MNIST, the minimum proportion of a class is 9.2 and the maximum 10.9, while on CIFAR10 it is 8.3 and 11.9 (in %). The variance-to-mean ratio is equal to 0.03 for MNIST and 0.22 for CIFAR10.

3.3.3.1 Precision and the associated partition

Now that we have defined the prerequisites for both the data and the model, we propose to establish a connection between the latent space partition and the precision of a generator. We create a link between the set of generators from $\mathbb{R}^d$ to $\mathbb{R}^D$ and the set of partitions in the latent space. Specifically, for each given partition in $\mathbb{R}^d$, there exists a set of associated generators defined as follows:

Definition 3.3.2. *For a given partition $\mathscr{A} = A_1, \ldots, A_m$ on $\mathbb{R}^d$, we say that G is associated to $\mathscr{A}$ if: for all $i \in [1,m]$, for all $z \in A_i$, $i = \underset{j\in[1,m]}{\arg\min} \|G(z) - X_j\|$.*

Each given generator G is associated with a unique partition $\mathscr{A}$ in $\mathbb{R}^d$. The geometry of the associated partition $\mathscr{A}$ plays a key role in explaining the behavior and performance of the generator G. We are interested in maximizing the precision of generative models. Points in the intersection of two cells $A_i \cap A_j, (i,j) \in [1,m]^2$ are equidistant from X_i and X_j and thus do not belong to any of these modes (since bot r_i and $r_j < \|X_i - X_j\|/2$ according to Assumption 4). Additionally, due to the generator's Lipschitzness, there is a small neighborhood around the boundary such that any points in this neighborhood are mapped out of the target manifold. This region in the latent space thus reduces the precision. For a given $\varepsilon > 0$, we now define the epsilon-boundary of the partition $\mathscr{A}$ as follows.

Definition 3.3.3. *For a given partition $\mathscr{A} = \{A_1, \ldots, A_m\}$ of $\mathbb{R}^d$ and a given $\varepsilon \in \mathbb{R}_+^\star$, we denote $\partial^\varepsilon \mathscr{A}$ the ε-boundary of $\mathscr{A}$, defined as follows.*

$$\partial^\varepsilon \mathscr{A} = \bigcup_{i=1}^{m} \left(\cup_{j\neq i} A_j\right)^\varepsilon \backslash \left(\cup_{j\neq i} A_j\right),$$

where A^ε corresponds to the ε-extension of set A. The following lemma makes the connection between the precision of a generator α_G and its associated partition $\mathscr{A}$.

Lemma 3.3.1. *Assume that Assumption 4 is satisfied and $\mathscr{A}$ be a partition in $\mathbb{R}^d$. Then, any generator $G \in \mathscr{G}_L$ associated with $\mathscr{A}$ verifies:*

$$\alpha_G \leqslant 1 - \gamma(\partial^{\varepsilon_{min}} \mathscr{A}). \tag{3.3.3}$$

where $\varepsilon_{min} = \min_{i,j} \|X_i - X_j\|/L$.

Interestingly, this result holds independently of the partition $\mathscr{A}$. It highlights that the geometry of the partition gives an upper-bound on the precision of the generator. Consequently, to properly determine this bound on the precision levels of generative models, one might be interested in determining the measure of this epsilon-boundary $\partial^\varepsilon \mathscr{A}$. By using the result from Lemma 3.3.1, we can derive an upper-bound on the precision that depends on D, L and m:

Corollary 3.3.1. *Assume that Assumption 4 is satisfied, $m \leqslant d+1$. Then, there exists L with $L \geqslant D\sqrt{\log(m)}$, such that for any well-balanced generator $G \in \mathscr{G}_L$:*

$$\alpha_G \leqslant 1 - \varepsilon_{min}\sqrt{\log m}\, e^{-3/2} \tag{3.3.4}$$

where $\varepsilon_{\min} = \min_{i,j} \|X_i - X_j\|/L$. In particular, the result in (3.3.4) gives an interesting insight when training GANs on a finite number of modes. Theorem 3.2.3 showed a similar result but for the asymptotic case when the number of modes increases:

$$\alpha_G \overset{m\to\infty}{\leqslant} e^{-\frac{1}{8}\varepsilon_{\min}^2} e^{-\varepsilon_{\min}\sqrt{\log(m)/2}}. \tag{3.3.5}$$

3.3.3.2 Optimality for push-forward generative models

To exhibit generative models with optimal precision levels, one must look at partitions with the smallest epsilon-boundary measures $\gamma(\partial^\varepsilon \mathscr{A})$. We argue that this is tightly connected to the theoretical field of Gaussian isoperimetric inequalities. Isoperimetric inequalities link the measure of sets with their perimeters. More specifically, these inequalities highlight minimizers of the perimeter for a fixed measure, *e.g.* the sphere in an euclidean space with a given Lebesgue measure. In the Gaussian space, Borell (1975) and Sudakov and Tsirelson (1978) show that in a finite-dimensional case, among all sets of a given measure, half-spaces have a minimum Gaussian perimeter. More formally, for any Borel set A in $\mathbb{R}^d$ and a half-space H, if we have $\gamma(A) \geqslant \gamma(H)$, then $\gamma(A^\varepsilon) \geqslant \gamma(H^\varepsilon)$ for any $\varepsilon > 0$, where A^ε denotes the ε-extension of A.

The Gaussian multi-bubble conjecture was formulated when looking for a way to partition the Gaussian space in m parts, with the least-weighted boundary. It was recently proved by

Milman and Neeman (2022) who showed that the best way to split a Gaussian space $\mathbb{R}^d$ in m clusters of equal measure, with $2 \leqslant m \leqslant d+1$, is by using 'simplicial clusters' obtained as the Voronoi cells of m equidistant points in $\mathbb{R}^d$. Convex geometry theory tells us that each cell is a convex cone, whose borders are hyperplanes going through the origin of $\mathbb{R}^d$. We note $\mathscr{A}^\star$ any partition corresponding to this optimal configuration, see Figure 3.6 for $m = 3$.

In the following theorem, we apply this result to the understanding of GANs. We make the connection between optimal generators (when $m \leqslant d+1$) in levels of precision and the partition $\mathscr{A}^\star$ derived in Milman and Neeman (2022).

Theorem 3.3.1 (**Optimality of generators with simplicial cluster latent space.**). *Assume that Assumption 4 is satisfied and $m \leqslant d+1$. For any $\delta > 0$, there exists C large enough (independent of δ) and $L \geqslant D\sqrt{m}\sqrt{\pi \log(Cm)}$, and a well-balanced generator $G^\star \in \mathscr{G}_L$ associated with $\mathscr{A}^\star$ such that for any other well-balanced generator $G \in \mathscr{G}_L$, we have:*

$$\alpha_{G^\star} \geqslant \alpha_G - \delta \tag{3.3.6}$$

Moreover, if $m \leqslant d$, noting $\varepsilon_{max} = \max_{i,j} \|X_i - X_j\|/L$:

$$\alpha_{G^\star} \geqslant 1 - \varepsilon_{max}\sqrt{m \log(Cm)}, \tag{3.3.7}$$

Theorem 3.3.1 shows that when L is large enough, the bound in (3.3.4) is almost tight, and thus that the given generator based on the simplicial partition $\mathscr{A}^\star$ is almost optimal. However, it is not clear whether those are the only generators with optimal precision. The proof is delayed in Appendix B.1.

What if Assumption 4 is not verified? This assumption is needed for the definition of a well-balanced generator associated with $\mathscr{A}^\star$ as in Theorem 3.3.1. As shown in Figure 3.7, the latent space configuration obtained by the GANs for 3 almost equidistant points (1st row) and 3 almost aligned data points (2nd row). We see that in the later case, the Voronoi partition of the target data points does not verify Assumption 4, and the optimal latent structure is not known. We observe in this specific case that it is made of two parallel hyperplanes, much different from $\mathscr{A}^\star$ defined by Milman and Neeman (2022) (1st row).

What if the dimension $m > d+1$? The position of the different spheres could be such that Assumption 4 is no longer valid. Second, since the result from Milman and Neeman (2022) does not hold, the optimal partition of the Gaussian space in m equal cells is unknown. In this generalized context, GANs could hint at the optimal partition geometry. Figure 3.8 stresses examples when training GANs from $\mathbb{R}^2$ to $\mathbb{R}^m$ with m equidistant modes. This

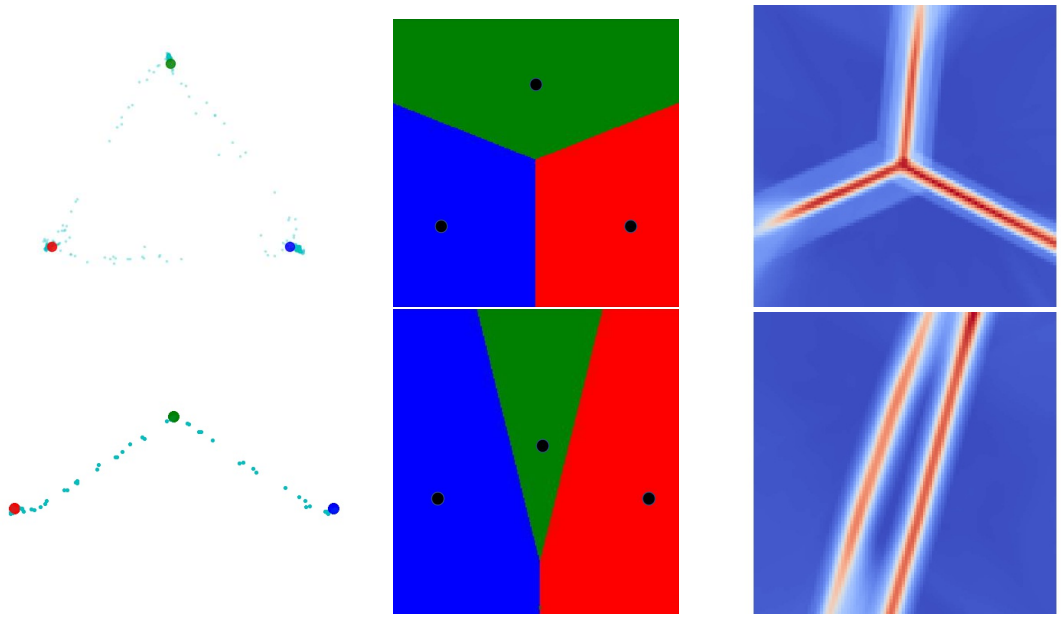


Fig. 3.7 Illustration of the impact of the geometry of data modes on the latent space of GANs. The left column shows the modes (X_1, X_2, X_3) from the target distribution and the generated points (small blue dots). In the middle, we plot the Voronoi diagram generated from (X_1, X_2, X_3). On the right column, we show the boundaries in the GANs latent space with heatmaps of the norm of the gradient of the generator. In the first row, when the data satisfies Assumption 4, GANs achieve the optimal configuration. However, when the data modes do not satisfy this assumption, as seen in the second row, this is no longer the case.

gives some insights on how to divide the Gaussian space into m equitable areas with least Gaussian-weighted perimeter.

What if the modes do not have equal measure? The fact that each mode has equal measure in the target distribution might not be verified for unbalanced datasets. First, the optimality of simplicial clusters holds because the multi-bubble theorem is still valid. However, the lower-bound (Equation 3.3.7) does not hold. Additionally, the upper-bound from Corollary 3.3.1 can be relaxed. Consider $w_1, \dots, w_m \in \mathbb{R}^m$ the weights of the different modes, and $w_{\min} = \min_i w_i$, the upper-bound becomes:

$$\alpha_G \leqslant 1 - m\varepsilon_{\min} w_{\min} \sqrt{\log(1/w_{\min})}\, e^{-3/2}.$$

We observe that this upper-bound might not be tight anymore since it depends on the minimum of the weights $w_{\min}$.

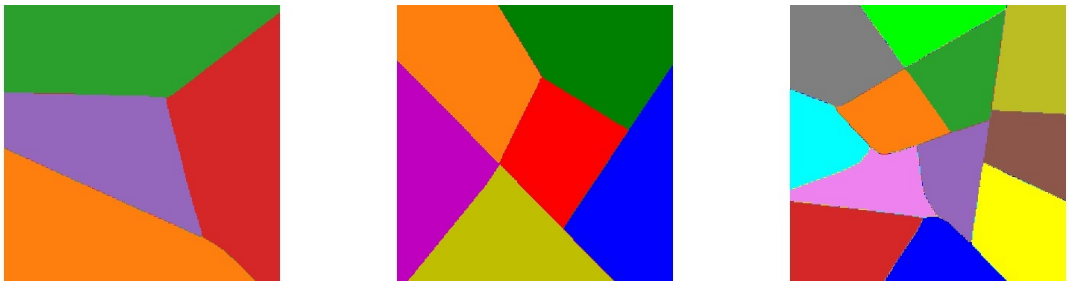


Fig. 3.8 Extension of the multi-bubble conjecture when $m > d+1$. We depict the partition of the $\mathbb{R}^2$ latent space of a GAN that maps to m equidistant points in $\mathbb{R}^m$, with $m = 4, 6, 12$. Each colored cell maps to a distinct data point in $\mathbb{R}^m$.

3.3.4 Improving generative models

Our proposed theoretical analysis offers valuable insights into the optimal structure of the latent space for push-forward generative models. We demonstrate that by leveraging this structure, it is possible to design GANs with improved performance. To achieve this, we enforce a simplicial cluster structure in the latent space of GANs during training using a novel rejection sampling procedure called *simplicial cluster truncation* that can be combined with a mutual-information loss. Note that modifying the latent space distribution of other generative models, such as VAEs or score-based models, is a more complex task.

Simplicial cluster truncation. Let us denote a simplicial cluster (Milman and Neeman, 2022) as $(u_1, \ldots, u_m) \mid u_i \in \mathbb{R}^d$. The rejection sampling procedure, based on Theorem 3.3.1, involves sampling a latent vector z from γ and accepting it if $\max_{i \in [1,\ldots,m]} (z \cdot u_i) > \tau$, where both τ and m are considered as hyper-parameters. This defines a new latent space distribution where the density is high near the unit vectors $u_i, i \in [1, m]$. As a result, the boundaries of the simplicial cluster, which are points with high distances to the centers of Voronoi cells, are rejected. The threshold parameter τ determines the ε value. With this method, the boundaries between different modes are never sampled, leading to a disconnected latent space. This approach can improve the learning of disconnected manifolds by injecting disconnectedness into the modeled generative distribution. Additionally, the use of a geometrical structure that is particularly well suited to separate several modes (Papyan et al., 2020) enhances the performance.

Mutual-information loss. The rejection sampling procedure might not be sufficient for the generator to properly use the different clusters of its latent space. To encourage the simplicial cluster structure, we also optimize the mutual information between generated samples and the

Table 3.2 Validation of linear separability (LogReg Acc.) and convexity (Convex Acc.) in GAN latent spaces. The results align with the predictions of Corollary 3.3.1, where a linearly separable and convex structure of the latent space indicates a high precision. The architecture *Transformer* refers to the TransGAN model from Jiang et al. (2021). The supervised classifiers used as oracles haves test-accuracies of 80.2% on CIFAR-10 and 61.8% on CIFAR-100.

Dataset	Architecture	Latent dim	Precision (↑)	LogReg Acc. (↑)	Convex Acc. (↑)
100 Gauss.	MLP	100	75.5	78.5	87.2
MNIST	CNN	64	93.2	90.4	98.7
CIFAR-10	ResNet	64	66.8	65.3	75.2
CIFAR-10	Transformer	256	72.8	70.7	84.3
CIFAR-100	ResNet	64	64.3	30.5	42.1
CIFAR-100	Transformer	64	64.2	26.5	39.2

corresponding cluster (Khayatkhoei et al., 2018). The loss is applied at the beginning of the training and is then dropped.

3.3.5 Experiments

In the following experiments, we validate our theoretical analysis and derive insights for GANs trained on toy and image datasets. We verify: 1) that the latent space geometry of GANs has similar properties than simplicial clusters; 2) that increasing the latent space dimension ($d+1>m$) can help improve GANs, as highlighted in the theoretical section; 3) that GANs' performance is correlated with their latent space geometry; 4) that the proposed *simplicial cluster* truncation method is effective and boost GANs' performance.

In the following experiments, we train WGANs with gradient penalty (Arjovsky et al., 2017; Gulrajani et al., 2017). For mixture of Gaussians, generator and discriminator are MLP networks. For MNIST, the generator and discriminator are standard convolutional architectures. On CIFAR-10, CIFAR-100 , and STL-10, we use either a Resnet-based (He et al., 2016) convolutional architecture with self-modulation in the generator (Chen et al., 2018b), either the transformer-based architecture from Jiang et al. (2021). To evaluate the performance of GANs, we use both the precision (Kynkäänniemi et al., 2019), the FID (Heusel et al., 2017), and the density/coverage (Naeem et al., 2020). We use a dataset-specific classifier to extract image features on MNIST, and InceptionNet pre-trained on ImageNet for CIFAR-10, CIFAR-100 and STL-10. Implementation details are given in Appendix B.2 and code is provided in Supplementary Material.

3.3.5.1 Linear separability and convexity

According to Milman and Neeman (2022), the optimal configuration in the latent space is obtained as the Voronoi cells of m equidistant points in $\mathbb{R}^d$, if $m \leq d+1$. This means that if GANs reach this optimal configuration, each of the cells must be *convex polytopes* and have the following properties: 1) the boundaries of a cell are flat; 2) each cell is convex. To investigate this, we use a labeled dataset and assess whether a simple linear model (*e.g.*, multinomial logistic regression) can map latents to labels. If the cells in the latent space are bounded by hyperplanes, then, using the hyperplane separation theorem, the linear model is expected to be a good predictor of a generated sample's label.

We use a standard multi-class labeled dataset. G_θ is a pre-trained generator and C_ϕ is a pre-trained classifier considered as an oracle. Using G_θ and C_ϕ, we construct a dataset of latent vectors $z \in \mathbb{R}^d$ and their associated labels $y = C_\phi(G_\theta(z))$. On CIFAR-10/100, similarly to Razavi et al. (2019), only data points with a confidence threshold of 0.7 or higher are accepted. This dataset is later split into 100k training points and 10k test points. We use multinomial logistic regression to learn the mapping from latent vectors z to their labels y. We can see in Table 3.2, that the *LogReg Accuracy* reaches high levels: 90% on MNIST and 70% on CIFAR-10. For the Convex accuracy, we draw two random latent vectors z_0 and z_1 that belong to the same class, and check whether linear interpolations in the latent space also belong to the same class, that is $C_\phi(G_\theta(z_0)) = C_\phi(G_\theta(z_0)) = \lambda \times C_\phi(G_\theta(z_0)) + (1-\lambda) \times C_\phi(G_\theta(z_1))$ for $\lambda \in [0,1]$. Interestingly, we see in Table 3.2 a correlation between the Logreg and Convex accuracy and the precision metric: the more the latent space behaves like a simplicial cluster, the higher the precision. For a qualitative evaluation, we show this phenomenon in Figure 3.9 and stress that linear interpolations conserve the image class.

3.3.5.2 Impact of the latent space dimension

To evaluate the impact of the latent space dimension, we train GANs with latent space dimension ranging from 2 to 128 on several datasets. In Figure 3.10, we exhibit two phases in the performance of GANs when changing the number of latent dimensions. For a fixed architecture, and a given dataset, we observe the existence of an optimal latent space dimension $d^\star$. When $d < d^\star$ the precision or density of the model falls significantly. Interestingly, when $d > d^\star$, the precision becomes constant: overparameterizing the model does not bring a significant improvement. As expected, we observe in Figure 3.10 that the maximum precision/density depends on the complexity of the dataset and its number of modes: the more complex the dataset, the lower the precision. This is also coherent with our theoretical results from both Corollary 3.3.1 and Theorem 3.3.1.

Fig. 3.9 Visualization of the convexity of classes in the latent space of GANs trained on CIFAR-10. The plot shows that latent linear interpolations within a class preserve the class label.

An interesting problem was also brought to the fore by Roth et al. (2017). When training GANs two different issues can arise: 1) *dimensional misspecification* where the true and modeled distributions do not have density functions w.r.t. the same base measure, and 2) *density misspecification*, where GANs try to fit a disconnected manifold with a unimodal distribution. To isolate the density misspecification studied in this section, we train a conditional GAN with a low-dimensional latent space $\mathbb{R}^d$ (*e.g.* $\mathbb{R}^5$ in our setting), so that the dimension of the generated manifold is at most 5. We later collect a dataset of synthetic generated samples *Synthetic CIFAR-10*, and train unconditional GANs with varying latent space dimensions. Figure 3.10 shows that GANs converge to the same limits for Precision and Density on Synthetic CIFAR-10 and CIFAR-10. This shows that the performance is more impacted by the *density misspecification* (trying to fit a disconnected target distribution with a connected one) rather than the dimensional misspecification.

3.3.5.3 The latent geometry and GANs' performance

We investigate the relationship between the performance of GANs and their latent space geometry. To do so, we train many generators with different capacities (increasing width), and study how it impacts both the latent geometry and the performance. The results in Figure 3.11 reveal a strong positive correlation between the performance of GANs and the

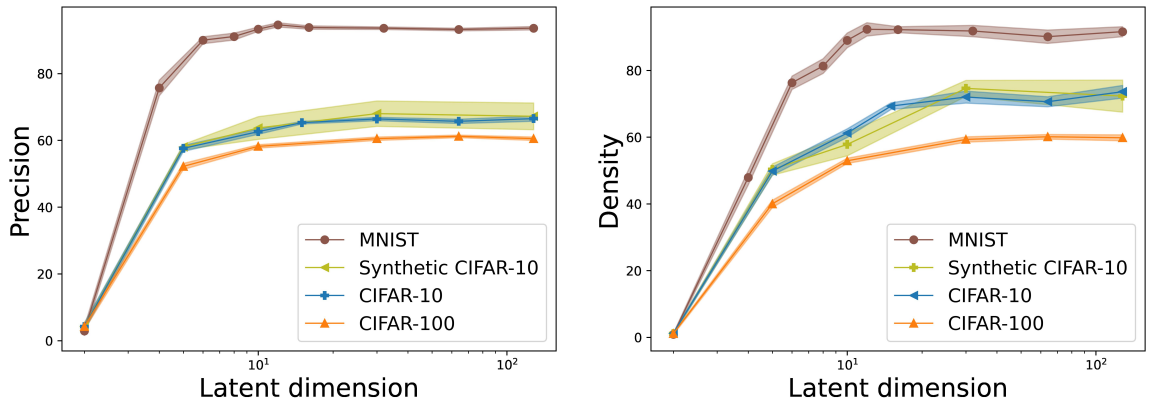


Fig. 3.10 Performance of GANs with regard to the number of modes and latent space dimensions. As the number of modes and latent space dimension increases, we observe an improvement in Precision (left) and Density (right), with a saturation point beyond a certain threshold.

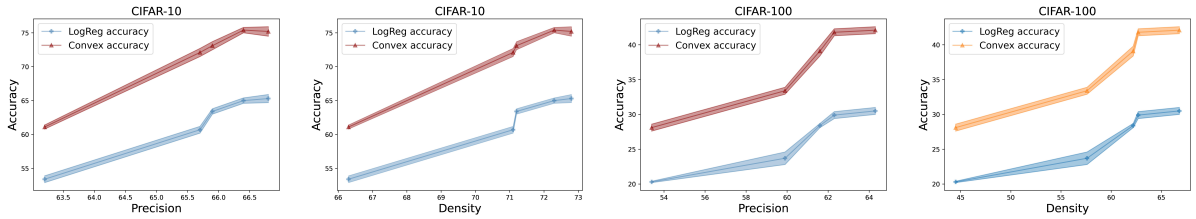


Fig. 3.11 Study of the correlation between GANs' performance and their latent space geometry. This is done by increasing the width of the generator ($w \in \{32, 64, 128, 256, 512\}$) in a fixed training setting on the CIFAR-10 and CIFAR-100 datasets. The results reveal a positive correlation between GANs' performance (measured by Precision and Density) and the linear separability and convexity of their latent space (measured by LogReg and Convex Accuracy). Confidence intervals are computed on 10 checkpoints of a training.

linearity/convexity of the latent space: the better the GANs perform, the more linearly separable and convex the latent space is. Indeed, the Pearson correlation between Precision and LogReg Accuracy is 0.98 on CIFAR-10, and 0.94 on CIFAR-100. Interestingly, overparametrization was known to help push-forward generative models in their optimization procedure (Balaji et al., 2021) and in increasing their Lipschitz constant (Salmona et al., 2022). We demonstrate here that it can help GANs in reaching an optimal latent space structure, resulting in improved performance.

3.3.5.4 Impact of the simplicial truncation method

Finally, we aim to improve GANs performance by using our theoretical results (Theorem 3.3.1). This is done by truncating the latent Gaussian distribution, as discussed in Section 3.3.4, so

Table 3.3 Improving GANs with simplicial cluster latent space. JBT stands for the Jacobian-based truncation from section 3.2; DeliG. for latent space with mixture of Gaussians (Gurumurthy et al., 2017); simp. for our proposed truncation method with simplicial cluster. These results demonstrate that generators with a simplicial cluster latent space consistently outperform the baseline generator in Precision/Density, and most of the times outperforms other boosting methods (DeliGAN and JBT).

Dataset/Model	FID ↓	Prec. ↑	Rec. ↑	Dens. ↑	Cov. ↑
CIFAR-10					
TransGAN	8.9	72.8	**62.6**	79.3	79.3
TransGAN + JBT	**8.8**	73.3	61.2	85.7	81.1
TransGAN + DeliG.	9.8	74.6	58.6	93.2	80.0
TransGAN + simp.	9.2	**74.9**	59.2	**96.4**	**82.6**
CIFAR-100					
TransGAN	15.2	64.2	**63.1**	53.4	66.0
TransGAN + JBT	**15.0**	64.8	62.9	53.6	66.2
TransGAN + DeliG.	15.9	63.5	62.2	52.6	64.4
TransGAN + simp.	15.1	**65.6**	61.5	**56.3**	**66.4**
STL-10 (32x32)					
TransGAN	10.5	75.7	60.1	87.5	83.0
TransGAN + JBT	11.0	**78.1**	57.6	**99.3**	**83.8**
TransGAN + DeliG.	10.5	76.0	**60.2**	85.5	81.5
TransGAN + simp.	**10.0**	77.8	60.1	94.1	83.5

that the generator structures its latent space with a simplicial cluster geometry. Note that the rejection threshold used at inference time can be higher than the one used at training time, since we have observed that higher rejection thresholds can help us increase both the precision and density of the models. The results in Table 3.3 demonstrate that the use of this truncation method can improve the density and precision of GANs, without lowering the coverage nor the FID. This simplicial-based truncation has thus proved to be effective at removing off-manifold samples and can help improve push-forward generative models.

3.3.6 Conclusion

In conclusion, this section takes a step towards a better understanding of push-forward generative models. When the latent space dimension is large enough, we prove the existence of an optimal latent space geometry, called 'simplicial clusters'. Through experiments, we demonstrate that generative models with sufficient capacity tend to conform to this optimal geometry and also that enforcing this latent structure can improve GANs' performance. Our analysis has potential to drive further research on generative models with both theoretical and

practical implications, such as developing new models that favor the emergence of such clusters in both latent and feature spaces. Similarly to what has been done in classification (Papyan et al., 2020), studying thoroughly the feature space of deep generative models is also an open question.

While our theoretical analysis demonstrates the existence of optimal generators, we were unable to prove their uniqueness. This limitation is associated with identifying partitions with the lowest ε-boundary measures in the Gaussian space, which is a challenging and unresolved problem in geometric measure theory. This could be a topic for future work.

Another challenge is to learn principled truncation method for pre-trained GANs. Indeed, in section 3.2, we use a heuristic method to filter generated samples. In this section, the truncation is based on the intuition that the generator naturally exploits the latent space structure. However, we have no guarantee that our truncation method will improve the performance of the generative model. In some cases, . In the next section, we investigate learning-based approaches to improve pre-trained GANs. Specifically, we propose a method that relies on an adversarial mechanism to learn importance weights in the latent space of a pre-trained GAN.

3.4 Latent reweighting, an almost free improvement for GANs

Abstract. Standard formulations of GANs, where a continuous function deforms a connected latent space, have been shown to be misspecified when fitting different classes of images. In particular, the generator will necessarily sample some low-quality images in between the classes. Rather than modifying the architecture, a line of works aims at improving the sampling quality from pre-trained generators at the expense of increased computational cost. Building on this, we introduce an additional network to predict latent importance weights and two associated sampling methods to avoid the poorest samples. This idea has several advantages: 1) it provides a way to inject disconnectedness into any GAN architecture, 2) since the rejection happens in the latent space, it avoids going through both the generator and the discriminator, saving computation time, 3) this importance weights formulation provides a principled way to reduce the Wasserstein's distance to the target distribution. We demonstrate the effectiveness of our method on several datasets, both synthetic and high-dimensional.

3.4.1 Introduction

GANs (Goodfellow et al., 2014) are an effective way to learn complex and high-dimensional distributions, leading to state-of-the-art models for image synthesis in both unconditional (Karras et al., 2019b) and conditional settings (Brock et al., 2019). However, it is well-known that a single generator with an unimodal latent variable cannot recover a distribution composed of disconnected sub-manifolds (Khayatkhoei et al., 2018). This leads to a common problem for practitioners: the existence of very low-quality samples when covering different modes. This is formalized in section 3.2 and 3.3, where we provide impossibility theorems on the learning of disconnected manifolds with standard formulations of GANs. Fitting a disconnected target distribution requires an additional mechanism inserting disconnectedness in the modeled distribution. A first solution is to add some expressivity to the model: Khayatkhoei et al. (2018) propose to train a mixture of generators, while Gurumurthy et al. (2017) make use of a multi-modal latent distribution.

A second line of research relies heavily on a variety of Monte-Carlo algorithms, such as Rejection Sampling (Azadi et al., 2019) or Metropolis-Hastings (Turner et al., 2019). Monte-Carlo methods aim at sampling from a target distribution, while having only access to samples generated from a proposal distribution. Using the previously learned generative distribution as a proposal distribution, this idea was successfully applied to GANs. However, one of the main drawbacks is that Monte-Carlo algorithms only guarantee to sample from the target distribution

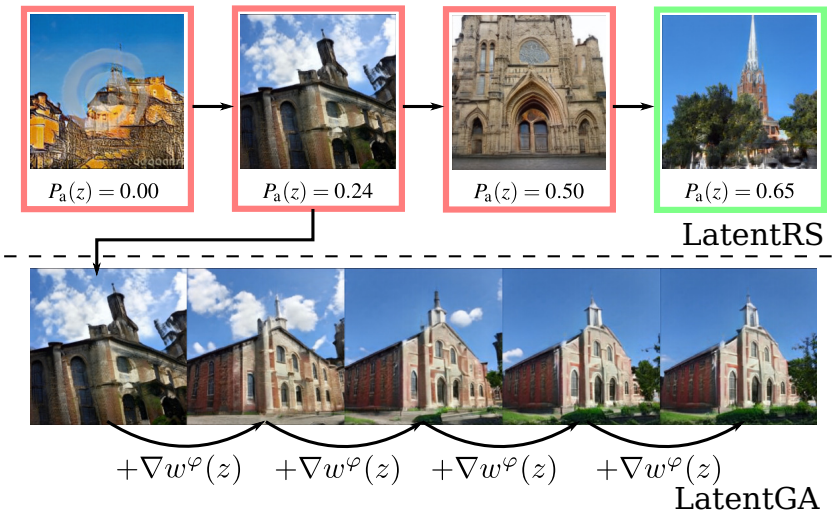


Fig. 3.12 **Overview of the proposed method**. GANs tend to produce poor images for unlucky draws of the latent variable (top row, left). We introduce importance weights $w^{\varphi}(z)$ in the latent space that allow us to use rejection sampling and accept a given latent variable z with probability $P_a(z) \propto w^{\varphi}(z)$ (LatentRS, top row), or to perform a simple gradient ascent over the importance weight (LatentGA, bottom row), leading to better images. Both strategies can be combined for improved image quality. Images generated with StyleGAN2 trained on LSUN Church.

under strong assumptions. First, we need access to the density ratios between the proposal and target distributions or equivalently to a perfect discriminator (Azadi et al., 2019). Second, these methods are efficient only if the support of the proposal distribution fully covers the one of the target distribution. This is unlikely to be the case when dealing with high-dimensional datasets (Arjovsky and Bottou, 2017).

To tackle this issue, we propose a novel method aiming at reducing the Wasserstein distance between the previously trained generative model and the target distribution. This is done via the adversarial training of a third network that learns importance weights in the latent space. Note that this network does not aim at increasing the support of the proposal distribution but at re-weighting the latent distribution, under a Wasserstein criterion. Thus, these importance weights define a new distribution in the latent space, from which we propose to sample with two complementary methods: latent rejection sampling (latentRS) and latent gradient ascent (latentGA). To better understand our approach, we illustrate its efficiency with simple examples. On the top of the Figure 3.12, we show samples coming from a pre-trained StyleGAN2 (Karras et al., 2019b) and their respective acceptance probability (latentRS). At the bottom, we exhibit a sequence of generated images while following a gradient ascent on the learned importance weights (latentGA).

Our contributions are the following:

- We propose a novel approach that trains a neural network to directly modify the latent space of a GAN. This provides a principled way to reduce the Wasserstein distance to the target distribution.
- We show how to sample from this new generative model with different methods: latent Rejection Sampling (latentRS), latent Gradient Ascent (latentGA), and latentRS+GA, a method that leverages the complementarity between the two previous solutions.
- We run a large empirical comparison between our proposed methods and previous approaches on a variety of datasets and distributions. We empirically show that all of our proposed solutions significantly reduce the computational cost of inference. More interestingly, our solutions propose a wide span of performances ranging from latentRS, optimizing speed, that matches state-of-the-art almost for free (computational cost divided by 15) and latentRS+GA (computational cost divided by 3) that outperforms previous approaches.

Notation. Before moving to the related work section, we shortly present the notation needed in the section. The goal of the generator is to generate data points that are "similar" to samples collected from some target probability measure $\mu_\star$. The measure $\mu_\star$ is defined on a potentially high-dimensional space $\mathbb{R}^D$, equipped with the euclidean norm $\|\cdot\|$. We call μ_n the empirical measure. To approach $\mu_\star$, we use a parametric family of generative distribution, where each distribution is the push-forward measure of a latent distribution Z and a continuous function modeled by a neural network. In most applications, the random variable Z defined on a low-dimensional space $\mathbb{R}^d$ is either a multivariate Gaussian distribution or uniform distribution. The generator is a parameterized class of functions from $\mathbb{R}^d$ to $\mathbb{R}^D$, say $\mathscr{G} = \{G_\theta : \theta \in \Theta\}$, where $\Theta \subseteq \mathbb{R}^p$ is the set of parameters describing the model. Each function G_θ takes input from Z and outputs "fake" observations with distribution $\mu_\theta = G_\theta \sharp Z$. On the other hand, the discriminator is described by a family of functions from $\mathbb{R}^D$ to $\mathbb{R}$, say $\mathscr{D} = \{D_\alpha : \alpha \in \Lambda\}$, $\Lambda \subseteq \mathbb{R}^Q$. Finally, for any given distribution μ, we note S_μ its support.

3.4.2 Related Work

Goodfellow et al. (2014) already stated that when training vanilla GANs, the generator could ignore modes of the target distribution: this is called mode collapse. A significant step towards understanding this phenomenon was made by Arjovsky and Bottou (2017) who explained that the standard formulation of GANs leads to vanishing or unstable gradients. The authors proposed the Wasserstein GANs (WGANs) architecture (Arjovsky et al., 2017) where, in particular, discriminative functions are restricted to the class of 1-Lipschitz functions. WGANs

aim at solving the following:

$$\sup_{\alpha \in A} \inf_{\theta \in \Theta} \mathbb{E}_{x \sim \mu_\star} D_\alpha(x) - \mathbb{E}_{z \sim \gamma} D_\alpha(G_\theta(z)) \tag{3.4.1}$$

3.4.2.1 Learning disconnected manifolds with GANs: training and evaluation

The broader drawback of standard GANs is that, since any modeled distribution is the push-forward of a unimodal distribution by a continuous transformation, it has a connected support. This means that when the generator covers multiple disconnected modes of the target distribution, it necessarily generates samples out of the real data manifold Khayatkhoei et al. (2018). Consequently, any thorough evaluation of GANs should assess simultaneously both the quality and the variety of the generated samples. To solve this issue, Sajjadi et al. (2018) and Kynkäänniemi et al. (2019) propose a Precision/Recall metric that aims at measuring both the *mode dropping* and the *mode inventing*. The precision refers to the portion of generated points that belongs to the target manifold, while the recall measures how much of the target distribution can be reconstructed by the model distribution.

Building on this metric, we highlighted the trade-off property of GANs deriving upper-bounds on the precision of standard GANs in section 3.2 and 3.3. To solve this problem, a common direction of research consists in over-parameterizing the generative model. Khayatkhoei et al. (2018) enforce diversity by using a mixture of generators, while Gurumurthy et al. (2017) suggests that a mixture of Gaussians in the latent space is efficient to learn diverse and limited data. Similarly, Balaji et al. (2020) propose importance weights that aim at robustifying the training of GANs and make it less sensitive to the target distribution's outliers.

3.4.2.2 Improving the quality of GANs post-training

Another line of research consists in improving the sampling quality of pre-trained GANs. Tanaka (2019) designed Discriminator Optimal Transport (DOT), a gradient ascent driven by a Wasserstein discriminator to improve every single sample. Similarly, Che et al. (2020) follow a discriminator-driven Langevin dynamic.

Another well-studied possibility would be to use Monte-Carlo (MC) methods (Robert and Casella, 2013). Following this path, Azadi et al. (2019) were the first to use a rejection sampling method to improve the quality of the proposal distribution μ_θ. The authors use the fact that the optimal vanilla discriminator trained with binary cross-entropy is equal to $\mu_\star/(\mu_\star + \mu_\theta)$. Thus, a parametric discriminator $D_\alpha : \mathbb{R}^D \to [0,1]$ can be used to approximate the density ratios r_α as follows:

$$r_\alpha(x) := \frac{\mu_\star(x)}{\mu_\theta(x)} = \frac{D_\alpha(x)}{1 - D_\alpha(x)}. \tag{3.4.2}$$

This density ratio can then be plugged in the Rejection Sampling (RS) algorithm. Doing so, it can be shown that sampling from μ_θ and accepting samples probabilistically is equivalent to sample from the target distribution $\mu_\star$. The acceptance probability of a given sample x is $\mathbb{P}_a(x) = \frac{r_\alpha(x)}{k}$. This is valid as long as there is a constant $k \in \mathbb{R}^+$ such that $\mu_\star(x) \leq k\mu_\theta(x)$ for all x.

Turner et al. (2019) use similar density ratios and derive MH-GAN, by using the independent Metropolis-Hasting algorithm (Hastings, 1970). Finally, Grover et al. (2019) use these density ratios r_α as importance weights and perform discrete sampling relying on the Sampling-Importance-Resampling (SIR) algorithm (Rubin, 1988). Given $X_1, \ldots, X_n \sim \mu_\theta^n$, we have:

$$\mu_{\theta,\alpha}^{\text{SIR}}(X_i) = \frac{r_\alpha(X_i)}{\sum_{j=1}^{n} r_\alpha(X_j)}. \tag{3.4.3}$$

Note that these algorithms all rely on similar density ratios and differ by the acceptance-rejection scheme chosen. Interestingly, in RS, the acceptance rate is not controlled, but we are guaranteed to sample from $\mu_\star$. Conversely, with SIR and MH, the acceptance rate is a chosen parameter, but we are sampling from an approximation of the target distribution.

3.4.2.3 Drawbacks of density-ratio-based methods

Even though these methods have the advantage of being straightforward, they suffer from one main drawback. In practice, because both the target and the proposal manifold do not have full dimension in $\mathbb{R}^D$ (Fefferman et al., 2016), (Arjovsky and Bottou, 2017, Lemma 3) show that it is highly likely that $\mu_\theta(S_{\mu_\theta} \bigcap S_{\mu_\star}) = 0$ and $\mu_\star(S_{\mu_\theta} \bigcap S_{\mu_\star}) = 0$. Consequently, when dealing with high-dimensional datasets, the proposal distribution μ_θ and the target distribution $\mu_\star$ might intersect on a null set. Thus, one would have $r_\alpha(x) = 0$ almost everywhere on S_{μ_θ}. In this setting, the assumptions of MC methods are broken, and these algorithms will not allow sampling from $\mu_\star$. This is shown in Figure 3.13.

In order to correct this drawback, our method proposes to avoid the computation of density ratios from a classifier and to directly learn how to re-weight the proposal distribution. Our proposed scheme aims at minimizing the Wasserstein distance to the empirical measure while controlling the range of these importance weights.

3.4.3 Adversarial Learning of Latent Importance weights

Similar to previous works, our method aims at improving the performance of a generative model, post-training. We assume the existence of a WGAN model (G_θ, D_α) pre-trained using (3.4.1).

The pushforward generative distribution μ_θ is assumed to be an imperfect approximation of the target distribution. The goal is now to learn how to redistribute the mass of the modeled distribution so that it best fits the target distribution.

3.4.3.1 Definition of the method

To improve the sampling quality of our pre-trained GANs, we propose to learn an importance weight function that directly learns how to avoid low-quality images and focus on very realistic ones. More formally, we over-parameterize the class of generative distributions and define a parametric class $\Omega = \{w^\varphi, \varphi \in \Phi\}$ of importance weight functions. Each function w^φ associates importance weights to latent space variables and is defined from $\mathbb{R}^d$ to $\mathbb{R}^+$. For a given latent space distribution γ and a network w^φ, a new measure γ^φ is defined on $\mathbb{R}^d$:

$$\text{for all } z \in \mathbb{R}^d,\ \mathrm{d}\gamma^\varphi(\mathrm{z}) = \mathrm{w}^\varphi(\mathrm{z})\mathrm{d}\gamma(\mathrm{z}) \tag{3.4.4}$$

Using this formulation, we can prove the following lemma:

Lemma 3.4.1. *Assume that $\mathbb{E}_\gamma\, w^\varphi = 1$, then the measure γ^φ is a probability distribution defined on $\mathbb{R}^d$.*

Consequently, we now propose a new modeled generative distribution μ_θ^φ, the pushforward distribution $\mu_\theta^\varphi = G_\theta \sharp \gamma^\varphi$. The objective is to find the optimal importance weights w^φ that minimizes the Wasserstein distance between the true distribution $\mu_\star$ and the new class of generative distributions. The proposed method can thus be seen as minimizing the Wasserstein distance to the target distribution, over an increased class of generative distributions. Denoting by Lip_1 the set of 1-Lipschitz real-valued functions on $\mathbb{R}^D$, i.e.,

$$\mathrm{Lip}_1 = \Big\{ f : \mathbb{R}^D \to \mathbb{R} : \frac{|f(x) - f(y)|}{\|x - y\|} \leqslant 1,\ (x \neq y) \in (\mathbb{R}^D)^2 \Big\},$$

we want, given a pre-trained model μ_θ, to solve:

$$\begin{aligned} \arg\min_{\varphi \in \Phi} W(\mu_\star, \mu_\theta^\varphi) &= \arg\min_{w^\varphi \in \Omega} \sup_{D \in \mathrm{Lip}_1} \mathbb{E}_{\mu_\star} D - \mathbb{E}_{\mu_\theta^\varphi} D \\ &= \arg\min_{w^\varphi \in \Omega} \sup_{D \in \mathrm{Lip}_1} \mathbb{E}_{\mu_\star} D - \mathbb{E}_{\mu_\theta} w^\varphi D \end{aligned}$$

The network w^φ, parameterized using a feed-forward neural network, thus learns how to redistribute the mass of μ_θ such that μ_θ^φ is closer to $\mu_\star$ in terms of Wasserstein distance. Similarly to the WGANs training, the discriminator D_α approximates the Wasserstein distance.

D_α and w^φ are trained adversarially, whilst keeping the weights of G_θ frozen, using the following optimization scheme:

$$\inf_{\varphi\in\Phi}\sup_{\alpha\in\Lambda}\ \mathbb{E}_{x\sim\mu_\star}D_\alpha(x)-\mathbb{E}_{z\sim Z}\ w^\varphi(z)\times D_\alpha(G_\theta(z)) \tag{3.4.5}$$

Note that our formulation can also be plugged on top of any objective function used for GANs.

3.4.3.2 Optimization procedure

However, as in the field of counterfactual estimation, a naive optimization of importance weights by gradient descent can lead to trivial solutions.

1. First, if for example, the Wasserstein critic D_α outputs negative values for any generated sample, the network w^φ could simply learn to avoid the dataset and output 0 everywhere (Swaminathan and Joachims, 2015a).

2. Second, another problem comes from the fact that (3.4.5) can be minimized not only by putting large importance weights $w^\varphi(z)$ on the examples with high likelihoods $D_\alpha(G(z))$ but also by maximizing the sum of the weights: this is the propensity overfitting (Swaminathan and Joachims, 2015b).

3. For the objective defined in (3.4.5) to be a valid Wasserstein distance minimization scheme, the measure μ_θ^φ must be a probability distribution, *i.e.* $\mathbb{E}_\gamma w^\varphi = 1$.

To tackle this, we first add a penalty term in the loss to enforce the expectation of the importance weights to be close to 1. This is similar to the self-normalization proposed by Swaminathan and Joachims (2015b). However, one still has to cope with the setting where the distribution γ^φ collapses to discrete data points:

Theorem 3.4.1. *Given a pre-trained generative distribution μ_θ absolutely continuous with respect to the Lebesgue measure on $\mathbb{R}^D$. Let Φ be the non-parametric class of continuous functions satisfying $\mathbb{E}_\gamma w^\varphi = 1$. We have that:*

$$W(\mu_n,\frac{1}{n}\sum_{i=1}^{n}\delta(\tilde{X}_i))\leqslant\inf_{\varphi\in\Phi}W(\mu_n,\mu_\theta^\varphi)$$

where δ refers to the Dirac probability distribution and $\tilde{X}_i=\underset{x\in S_{\mu_\theta}}{\arg\min}\ \|x-X_i\|$.

For clarity, the proof is delayed in Appendix. Intuitively, this theorem shows that the best way to approximate the empirical measure μ_n would be by considering a mixture of Diracs with

each mode being the projection of a training data point on the support of the learned manifold S_{μ_θ}. The network w^φ could thus be tempted to approximate this mixture of Diracs defined in Theorem 3.4.1 and collapse on some specific latent data points. This could lead to an increased time complexity at inference (see (Azadi et al., 2019, Section 3)). More importantly, this would mean *a mode collapse* and a lack of diversity in the generated samples.

To avoid such cases where small areas of z have really high $w^\varphi(z)$ values (mode collapse), we enforce a soft-clipping on the weights (Bottou et al., 2013; Grover et al., 2019). Note that this constraint on $w^\varphi(z)$ could also be implemented with a bounded activation function on the final layer, such as a re-scaled sigmoid or tanh activation. Finally, we get the following objective function for the network w^φ:

$$\sup_{\varphi\in\Phi} \mathbb{E}_{z\sim Z} \underbrace{w^\varphi(z)\big(D_\alpha(G_\theta(z)) - \Delta\big)}_{\text{discriminator reward}} - \lambda_1 \underbrace{\big(\mathbb{E}_{z\sim Z} w^\varphi(z) - 1\big)^2}_{\text{self-normalization}} - \lambda_2 \underbrace{\mathbb{E}_{z\sim Z} \max\big(0, (w^\varphi(z) - m)\big)^2}_{\text{soft-clipping}}, \tag{3.4.6}$$

where $\Delta = \min_{z\sim Z} D_\alpha(G(z))$. λ_1, λ_2, and m are hyper-parameters (values displayed in Appendix). For more details, we refer the reader to Algorithm 1.

3.4.3.3 Sampling from the latent importance weights

Given a pre-trained generator G_θ and an importance network w^φ, we now present the three proposed sampling algorithms associated with our model:

1) Latent Rejection Sampling (latentRS, Algorithm 2). The first proposed method aims at sampling from the newly learned latent distribution γ^φ defined in (3.4.4). Since the learned importance weights are capped by m defined in (3.4.6), this setting fits in the Rejection Sampling (RS) algorithm (Robert and Casella, 2013). Any sample $z \sim \gamma$ is now accepted with probability $\mathbb{P}_\mathrm{a}(z) = w^\varphi(z)/m$. Interestingly, by actively capping the importance weights as it is done in counterfactual estimation (Bottou et al., 2013; Faury et al., 2020), one controls the acceptance rates $\mathbb{P}_a(z)$ of the rejection sampling algorithm:

$$\mathbb{E}_\gamma\, \mathbb{P}_\mathrm{a}(z) = \int_{\mathbb{R}^d} \frac{w^\varphi(z)}{m} \mathrm{d}\gamma(\mathrm{z}) = \frac{1}{\mathrm{m}}.$$

2) Latent Gradient Ascent (latentGA). Inspired from (Tanaka, 2019, Algorithm 2), we propose a second method, latentGA, where we perform gradient ascent in the latent space

Algorithm 1: Adversarial learning of w^{φ}

Require: Data μ_n, Prior Z, Gen. G_θ, Disc. D_α, number of D_α updates n_d, soft-clipping param. m, regularization weights λ_1 and λ_2, batch size b;
while φ *has not converged* **do**
  for $i = 0, ..., n_d$ **do**
    Sample real data $\{x_i\}_{i=1}^{b} \sim \mu_n$;
    Sample latent vectors $\{z_i\}_{i=1}^{b} \sim Z$;
    $\text{EMD} \leftarrow \frac{1}{b}\sum_{i=1}^{b} D_\alpha(x_i) - w^{\varphi}(z_i) D_\alpha(G_\theta(z_i))$;
    $\text{GP} \leftarrow \text{Gradient-Penalty}(D_\alpha, x, G_\theta(z))$;
    $\text{grad}_\alpha \leftarrow \nabla_\alpha(-\text{EMD} + \text{GP})$;
    Update α with grad_α;
  end
  Sample $\{z_i\}_{i=1}^{b} \sim Z$;
  $\Delta \leftarrow \min_i[D_\alpha(G_\theta(z_i))]$;
  $\text{EMD} \leftarrow \frac{1}{b}\sum_{i=1}^{b} w(z_i)[D_\alpha(G_\theta(z_i)) - \Delta]$;
  $R_{norm} \leftarrow ([\frac{1}{b}\sum_{i=1}^{b} w(z_i)] - 1)^2$;
  $R_{clip} \leftarrow \frac{1}{b}\sum_{i=1}^{b} \max(0, w^{\varphi}(z_i) - m)^2$;
  $\text{grad}_\varphi \leftarrow \nabla_\varphi(\text{EMD} + \lambda_1 R_{norm} + \lambda_2 R_{clip})$;
  Update φ with grad_φ;
end

Algorithm 2: LatentRS

Requires: Prior Z, Gen. G_θ, Importance weight network w^{φ}, maximum importance weight m;
while *True* **do**
  Sample $z \sim Z$;
  Sample $\alpha \sim \text{Uniform}[0, 1]$;
  if $\frac{w^{\varphi}(z)}{m} \geq \alpha$ **then**
    break;
  end
end
$x \leftarrow G_\theta(z)$;
Result: Selected point x

(see the algorithm in Appendix). For any given sample in the latent space, we follow the path maximizing the learned importance weights. This method is denoted latentGA. Note that the learning rate and the number of updates used for this method are hyper-parameters that need to be tuned.

3) Combining latentRS with Gradient Ascent (latent RS+GA, see Appendix). Finally, we propose to combine sequentially both methods. In a first step, we avoid low-quality samples with latentRS. Then, we use latentGA to further improve the remaining generated samples. See algorithm in Appendix.

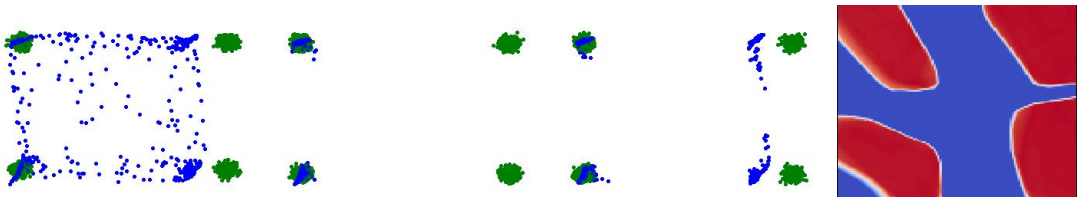
3.4.3.4 Advantages of the proposed approach

We now discuss two advantages of our method compared to previous density-ratio-based Monte-Carlo methods.

Computational cost. By using sampling algorithms in the latent space, we avoid going through both the generator and the discriminator, leading to a significant computational speed-up. This is of particular interest when dealing with high-dimensional spaces, since we do not need to pass through deep CNNs generator and discriminator (Karras et al., 2019b). In the next experimental section, we observe a computational cost decreased by a factor of 10.

Monte-Carlo methods do not properly work when the support S_{μ_θ} does not fully cover the support $S_{\mu_\star}$. To better illustrate this claim, we consider a simple 2D motivational example where the real data lies on four disconnected manifolds. We start with a proposal distribution (in blue) that does not fully recover the target distribution (Figure 3.13a). In this setting, we see in Figure 3.13b that the discriminator's density-ratio-based methods (Azadi et al., 2019) avoids half of the proposal distribution, while our proposed method learns a very different re-weighting (see Figure 3.13c).

This illustration is important since (Arjovsky and Bottou, 2017, Theorem 2.2) have shown that in high-dimension the intersection $S_{\mu_\star} \bigcap S_{\mu_\theta}$ is likely to be a negligible set under μ_θ. Knowing that S_{μ_θ} does not fully recover $S_{\mu_\star}$, there is thus no theoretical guarantee that using a sampling algorithm will improve the estimation of $\mu_\star$. On the opposite, our method looks for the optimal re-weighting of μ_θ under a well-defined criterion: the Wasserstein distance. This results in a better fit of the real data distribution (see next section).



(a) Synthetic WGAN real samples in and fake ones in blue.

(b) MC method optimizgreening for a precision criterion (Azadi et al., 2019).

(c) Optimizing for Wasserstein criterionin the latent space (in with latentRS (ours $\star$).

(d) Heatmap of the w^{φ} the blue areas, w^{φ}=0).

Fig. 3.13 Synthetic experiment mimicking the setting of GANs in high-dimension, where data and generated manifolds are close but do not perfectly intersect. While DRS only selects the intersection of manifolds and ignores the rest, the latent importance weights define a rejection mechanism that minimizes the Wasserstein distance. For conciseness, WGAN stands for WGAN-GP.

3.4.4 Experiments

In this section, we illustrate the efficiency of the proposed methods, latentRS, latentGA, and latentRS+GA on both synthetic and natural image datasets. On image generation tasks, we empirically stress that latentRS slightly surpasses density-ratio-based methods with respect to the Earth Mover's distance while reducing the time complexity by a factor of around 10. The use of latentGA also gives interesting experimental visualizations and improves image quality. More importantly, when combined, we show that latenRS+GA surpasses the concurrent methods, while still being less computationally intensive. Finally, we show results with different models such as Progressive GAN (Karras et al., 2018) and StyleGAN2 (Karras et al., 2020b).

3.4.4.1 Evaluation metrics

To measure the performances of GANs when dealing with low-dimensional applications, we equip our space with the standard Euclidean distance. However, for the case of image generation, we follow Brock et al. (2019); Kynkäänniemi et al. (2019) and consider the euclidean distance between embeddings of a pre-trained network, that convey more semantic information. Thus, for a pair of images (a,b), we define the distance $d(a,b)$ as $d(a,b) = \|\phi(a) - \phi(b)\|_2$ where ϕ is a pre-softmax layer of a supervised classifier. On MNIST and F-MNIST, the classifier is pre-trained on the given dataset. On CelebA and LSUN Church, we use VGG-16 pre-trained on ImageNet.

To begin with, we report the FID (Heusel et al., 2017). We also compare the performance of the different methods with the Precision/Recall (PR) metric (Kynkäänniemi et al., 2019). It is a more robust version of the Precision/Recall metric, which was first applied in the context

Table 3.4 Comparison of latentRS with concurrent methods on two synthetic datasets in the same setting as DOT (Tanaka, 2019). Our method enables a consistent gain in EMD, surpassing other methods on Swiss Roll and slightly behind DOT on Mixture of 25 Gaussians. For conciseness, WGAN stands for WGAN-GP.

	EMD Swiss Roll	EMD 25 Gaussians
WGAN	0.030±0.002	0.044±0.001
WGAN: DRS	0.036±0.004	0.038±0.002
WGAN: SIR	0.037±0.003	0.041±0.001
WGAN: DOT	0.029±0.003	**0.035±0.002**
WGAN: latentRS ($\star$)	**0.025**±0.002	0.036±0.001

of GANs by Sajjadi et al. (2018). Finally, we approximate the Wasserstein distance using the Earth Mover's Distance (EMD) between generated and real data points. This measure is particularly suited to the study of WGANs, since it is linked to their objective function. Letting $X = \{x_1, \dots, x_n\}$ and $Y = \{y_1, \dots, y_n\}$ be two collections of n data points and $\mathscr{S}$ be the set of permutations of $[1, n]$, the Earth Mover's distance between X and Y is defined by:

$$\text{EMD}(X, Y) = \min_{\sigma \in \mathscr{S}} \sum_{i=1}^{n} \|x_i - y_{\sigma_i}\|$$

3.4.4.2 Synthetic datasets

To begin the experimental study, we test our method on 2D synthetic datasets in the same setting as Tanaka (2019). Table 3.4 compares the latentRS method with previous approaches on the Swiss roll dataset and on a mixture of 25 Gaussians. We see that the network w^φ efficiently redistributes the pre-trained distribution μ_θ since $\text{EMD}(\mu_n, \mu_\theta^\varphi)$ is significantly smaller than $\text{EMD}(\mu_n, \mu_\theta)$.

3.4.4.3 Image datasets

Implementation of baselines. We now compare latentRS, latentGA, and latentRS+GA with previous works leveraging discriminator's information on high-dimensional data. In particular, we implemented a wide set of post-processing methods for GANs: DRS (Azadi et al., 2019), MH-GAN (Turner et al., 2019), SIR (Grover et al., 2019) and DOT (Tanaka, 2019). DRS, MH-GAN and SIR use the same density ratios, and we did not see significant differences between those three methods in our experiments. Consequently, for the following experiments, we compare our algorithms to SIR and DOT. For SIR, we take the discriminator at the end of the adversarial training, fine-tune it with the binary cross-entropy loss and select the best

	Prec. (↑)	Rec. (↑)	EMD (↓)	FID (↓)	Inference (ms)
CelebA 128x128					
ProGAN	74.2±0.9	60.7±1.4	25.4±0.1	11.30±0.02	**3.6**
ProGAN: SIR	79.5±0.4	**57.3±1.0**	24.9±0.2	12.01±0.04	49.0
ProGAN: DOT	81.3±1.0	52.9±1.4	25.0±0.1	11.01±0.03	67.6
ProGAN: latentRS (⋆)	80.4±0.9	55.7±1.0	24.7±0.1	10.77±0.04	4.5
ProGAN: latentRS+GA (⋆)	**83.3±1.0**	52.7±0.9	**24.5±0.1**	**10.75±0.04**	20.5
LSUN Church 256x256					
StyleGAN2	55.6±1.2	62.4±1.1	23.6±0.1	6.91±0.02	**11.7**
StyleGAN2: SIR	60.5±1.4	**58.1±1.3**	23.4±0.1	7.36±0.01	130.0
StyleGAN2: DOT	67.4±1.4	48.3±1.0	23.1±0.1	6.85±0.02	196.7
StyleGAN2: latentRS (⋆)	63.3±0.7	57.7±1.0	23.1±0.1	6.31±0.02	16.2
StyleGAN2: latentRS+GA (⋆)	**72.6±1.1**	43.2±1.3	**22.6±0.1**	**6.27±0.03**	43.2

Table 3.5 latentRS+GA is the best performer, and latentRS matches SOTA with a significantly reduced inference cost (by an order of at least 10). ± is 97% confidence interval. Inference refers to the time in milliseconds needed to compute one image on a NVIDIA V100 GPU.

model in terms of EMD. Overall, we explicitly follow the framework used by Azadi et al. (2019); Grover et al. (2019): we keep the gradient penalty (Gulrajani et al., 2017), spectral normalization (Miyato et al., 2018) during fine-tuning and do not include an explicit mechanism to calibrate the classifier.

Description of datasets and neural architectures. We first consider two well-known image datasets that are **MNIST** (LeCun et al., 1998) and **FashionMNIST** (F-MNIST). We follow Khayatkhoei et al. (2018) and use a standard CNN architecture composed of a sequence of blocks made of 3x3 convolution layer and ReLU activations with nearest neighbor upsampling. For these datasets, the discriminator is trained using the hinge loss with gradient penalty (Hinge-GP). Finally, the architecture used for the network w^{φ} is very simple: an MLP with 4 fully-connected layers and ReLU activation (with a width $= 4 \times d$).

CelebA (Liu et al., 2015) is a large-scale dataset of faces covering a variety of poses. We use a pre-trained model of Progressive GAN (Karras et al., 2018) at 128x128 resolution. The discriminator is trained using a Wasserstein loss with gradient-penalty. Also, the architecture used for the network w^{φ} is really standard: a 5 hidden-layer MLP with a width of the same size than the latent space dimension.

LSUN Church (Yu et al., 2015) is a dataset of church images with a lot of variety. We use a pre-trained model of StyleGAN2 (Karras et al., 2020b) at 256x256 resolution. Similarly to the CelebA dataset, the discriminator is trained using a Wasserstein loss with gradient-penalty.

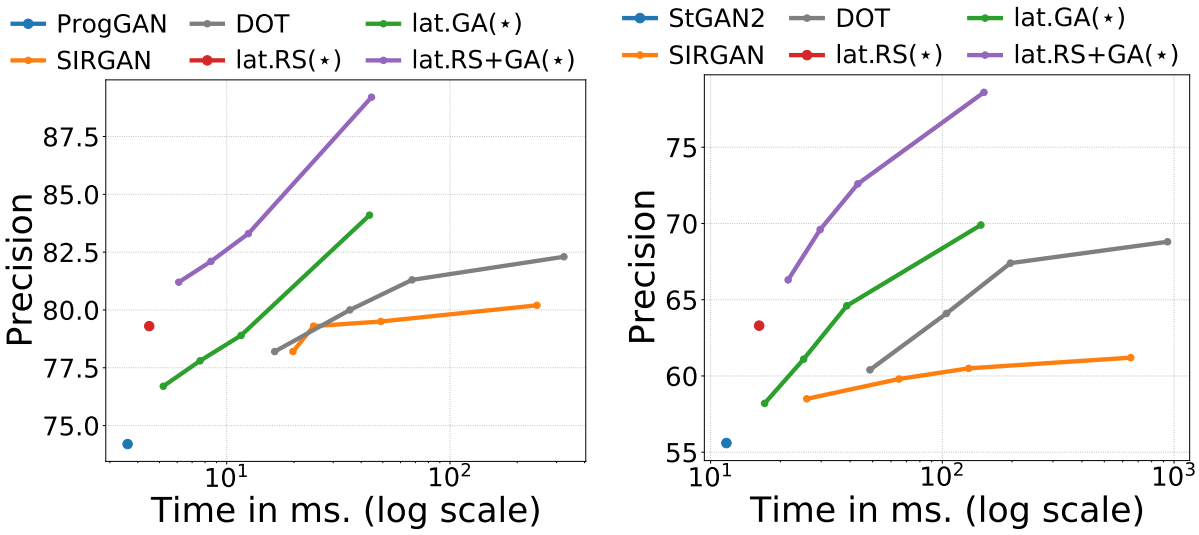


Fig. 3.14 Visualization of the trade-off between the time spent to generate an image and its average precision. Interestingly, latentRS+GA has the best Pareto front. Left: ProGAN trained on CelebA. Right: StyleGan2 trained on LSUN Church.

Also, the architecture used for the network w^{φ} is a 3 hidden-layer MLP with width equal to the latent space dimension. Note that the StyleGAN architecture already contains an 8-layer MLP network $M_{\theta} : \mathbb{R}^d \to \mathbb{R}^d$ that transforms a latent space variable to an intermediate latent variable (Karras et al., 2019b). We consequently leverage this pre-trained M_{θ} and train the network w^{φ} on top of it.

Results. The main results of this comparison are shown in Table 3.5 and Figure 3.14. On all studied datasets, our latentRS+GA outperforms every other method on the EMD with lower computational cost. Interestingly, latentRS achieves good performance on FID while being more than 15 times faster. Figure 3.14 is particularly interesting since it gives a good visualization of the trade-off between computational cost and quality of the generated samples. On this experiment ran on CelebA and LSUN, we observe that latentRS+GA can achieve a significantly better precision than both SIR and DOT while being much faster. Interestingly, even though these datasets are high-dimensional, contain only one-class, and w^{φ} has a low capacity, our proposed methods still produce interesting results.

To visualize the efficiency of the proposed method, Figure 3.15 shows generated samples along with their acceptance probabilities. As expected, we observe that higher acceptance probabilities correlate with higher quality images. Figure 3.16 stresses how generated images improve when performing latent gradient ascent on the importance weights. Finally, we provide more qualitative results and details on the experiments in supplementary material.

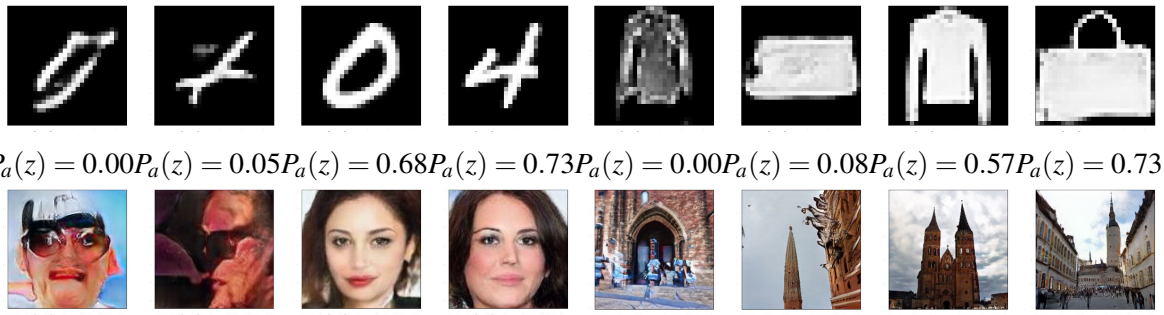


Fig. 3.15 Images drawn from the generative model and their acceptance probabilities with the latentRS algorithm, given by the network w^{φ}. As expected, the quality of images correlates with higher acceptance rates on all datasets: MNIST, F-MNIST, CelebA, and LSUN.

Fig. 3.16 Gradient ascent on latent importance weights (latentGA): the quality is gradually improved as we move to larger importance weights. Each image is generated only for visualization, and one can run this gradient ascent directly in the latent space using w^{φ}. Interestingly, this gradient ascent only involves a simple MLP network which is computationally cheap.

3.4.5 Conclusion

This section deals with improving the quality of pre-trained GANs. Conversely to concurrent methods which leverage the discriminator at inference time, we propose to train adversarially a neural network which learns importance weights in the latent space of GANs. These latent importance weights are then used with two complementary sampling methods: latentRS and latentGA. We experimentally show that this latent reweighting consistently enhances the quality of the pre-trained model. When these two methods are combined in latentRS+GA, it surpasses concurrent post-training methods while being less computationally intensive.

Chapter 4

Image editing with deep neural networks

Contents

4.1 Introduction

Designing learning algorithms for image editing using deep neural networks is a challenging task due to the scarcity of target data. Ideally, supervised learning would be the optimal approach to address image editing. This would enable the use of a standard pipeline for supervised deep learning, starting with the collection of a labeled dataset. In image editing tasks, the input data, denoted as X, is usually composed of a source image, I_s, and a conditioning variable, C, where $X = [I_s, C]$. Meanwhile, the target data represents the image to be produced, denoted as I_t. However, collecting labeled datasets for image editing tasks is often not feasible, as it can be prohibitively expensive. For instance, in the virtual try-on task discussed in Section 4.3, I_s denotes the source image of the person, C denotes the clothing item image, and I_t represents the image of the person in the exact pose as I_s, wearing the clothing item from C.

In this chapter, we present two original contributions enabling image editing tasks to be performed without complete supervision. In Section 4.2, we demonstrate the ability to address various image editing tasks, such as local denoising, image inpainting, image compositing, or scribble-based editing, using a single training objective and model per dataset. This training objective, inspired by BERT (Devlin et al., 2018), a widely-used unsupervised pretraining objective for large language models, involves predicting masked sequences. We demonstrate

its superior performance in image editing compared to other generic methods such as GANs' inversions.

In Section 4.3, we leverage adversarial training and a teacher-student distillation mechanism to enhance image-based virtual try-on algorithms. The conventional pipeline for virtual try-on entails supervised training, where the input image I_s is a masked image $I_s = M(I_t)$, with M as a masking operator that hides the clothing item. We demonstrate that this standard pipeline, coupled with adversarial training, can guide a student model that takes unmasked images I_s as input, thereby outperforming the teacher model and achieving state-of-the-art performance at the time of publication.

4.2 EdiBERT: a generative model for image editing

Abstract Advances in computer vision are pushing the limits of image manipulation, with generative models sampling highly-realistic detailed images on various tasks. However, a specialized model is often developed and trained for each specific task, even though many image edition tasks share similarities. In denoising, inpainting, or image compositing, one always aims at generating a realistic image from a low-quality one. In this section, we aim at making a step towards a unified approach for image editing. To do so, we propose EdiBERT, a bidirectional transformer that re-samples image patches conditionally to a given image. Using one generic objective, we show that the model resulting from a single training matches state-of-the-art GANs inversion on several tasks: image denoising, image completion, and image composition. We also provide several insights on the latent space of vector-quantized auto-encoders, such as locality and reconstruction capacities. The code is available at https://github.com/EdiBERT4ImageManipulation/EdiBERT.

4.2.1 Introduction

Fig. 4.1 Using a single and straightforward training, EdiBERT can tackle a wide variety of different tasks in image editing. In this image, the top row is the input, while the second and third rows are different samples from EdiBERT, showing realism, consistency, and variety.

Significant progress in image generation has been made in the past few years, thanks notably to Generative Adversarial Networks (GANs) (Goodfellow et al., 2014). For example, the StyleGAN architecture (Karras et al., 2019b, 2020b) yields state-of-the-art results in data-driven unconditional generative image modeling. Empirical studies have also shown the usefulness of GANs' architecture when it comes to image manipulation. By following specific directions in the latent space, one can modify an image attribute such as gender, age, the pose of a person (Shen et al., 2020), or the angle (Jahanian et al., 2019). However, since the whole picture is generated from a Gaussian vector, changing some undesired elements while keeping the others frozen is difficult. To solve this problem, edition algorithms involving optimization procedures have been proposed (Abdal et al., 2019, 2020) but with one main caveat: the results are not convincing when manipulating complex visuals (Niemeyer and Geiger, 2021) (cf. experimental section for visual results).

Independently, Van Den Oord et al. (2017) propose VQVAE, a promising latent representation by training an encoder/decoder using a discrete latent space. The authors demonstrate the possibility to embed images in sequences of discrete tokens borrowing ideas from vector quantization (VQ), paving the way for the generation of images with autoregressive transformer models (Ramesh et al., 2021; Esser et al., 2021b). Building on this litterature, we argue that one of the benefits of this representation is that each token in the sequence is mostly coding for a localized patch of pixels (see section 4.2.3.4), thus opening the possibility for an efficient localized latent edition.

Aiming to build a unified approach for image manipulation, we propose a method that leverages both the spatial property of the discrete vector-quantized representation and the use of model that performs attention on the whole image. To do so, we train a bi-directionnal transformer network based on ideas from the language model BERT (Devlin et al., 2018), naming EdiBERT the resulting model. During training, EdiBERT tries to recover the original tokens of a perturbed sequence through a bidirectional attention schema. In computer vision, this approach has mainly been studied in the context of self-supervised representation learning (Bao et al., 2022; He et al., 2022). We advocate that training a single model using this generic objective provides a sounded way to obtain a model able to tackle several editing tasks. Finally, to practically handle these tasks, we also derived two different sampling algorithms: one dedicated for image denoising and editing, and a second one for inpainting.

To better visualize the abilities of EdiBERT after a single training, we show in Figure 4.1 that the same model can now be used in many different image manipulation tasks such as denoising, inpainting (or completion), compositing, and scribble-based editing.

To sum up, our contributions are the following:

+ We analyze the VQ latent representations and illustrate their spatial properties, and show how to improve the reconstruction capabilities of VQGAN, using a post-processing procedure that better recovers the pixel content outside of the edited region.

+ We show how to derive two different sampling algorithms from a single bidirectional transformers: one for the task of image denoising where the locations of the edits are unknown, and a second one for inpainting or image compositing where the mask specifying the area to edit is known.

+ Finally, we show that using this generic simple training algorithm along with its companion post-processing allow us to achieve competitive results on various image manipulation tasks.

4.2.2 Related work

We start by introducing transformer models for image generation. Then, we motivate the use of the VQ representation and bidirectional models for image manipulation.

4.2.2.1 Autoregressive image generation

The use of autoregressive transformers in the field of generative modeling (Parmar et al., 2018) has been made possible by two simultaneous research branches. First, the extensive deployment of attention mechanisms such as non-local means algorithms (Buades et al., 2005), non-local neural networks (Wang et al., 2018b), and also attention layers in GANs (Zhang et al., 2019; Hudson and Zitnick, 2021). Second, the development of both classifiers and generative models sequentially inferring pixels via autoregressive convolutional networks such as PixelCNN (Van Den Oord et al., 2016; Van den Oord et al., 2016). The self-attention mechanism (Vaswani et al., 2017), which now become ubiquitous in computer vision, is quickly recalled here: a sequence $X \in \mathbb{R}^{L\times d}$, where L is length of the sequence, is mapped by a position-wise linear layer to a query $Q \in \mathbb{R}^{L\times d_k}$, a key $K \in \mathbb{R}^{L\times d_k}$ and a value $V \in \mathbb{R}^{L\times d_v}$. The self-attention layer is then:

$$\text{attn}(Q,K,V) = \text{softmax}(\frac{QK^t}{\sqrt{d_k}})V \in \mathbb{R}^{L\times d_v} \tag{4.2.1}$$

If autoregressive transformers allow a principled log-likelihood estimation of the data, attention layers have a complexity scaling with the square of the sequence length, a clear bottleneck to scale to high-resolution images. To reduce the size of these sequences, Van Den Oord et al. (2017) proposed the use of discrete representation. In this framework, an encoder E, a decoder D, and a codebook/dictionary Z are learned simultaneously to represent

images with a single sequence of tokens. Esser et al. (2021b) later trained an autoregressive model on these token sequences, stressing that high-capacity transformers can generate realistic high-resolution images. The framework consists of three steps:

1. Training simultaneously a set of encoder/decoder/codebook (E,D,Z), by combining reconstruction, commitment and adversarial losses. The reconstruction loss is a perceptual distance (Zhang et al., 2018). The commitment loss (Van Den Oord et al., 2017) pushes the codebook towards the output of the encoder using a quantization loss. The adversarial loss is the Vanilla GANs loss defined in (Goodfellow et al., 2014). The training objective becomes :

$$E^\star, D^\star, Z^\star = \arg\min_{E,D,Z} \left[L_{\text{rec.}}(E,D,Z) + L_{\text{commit.}}(E,Z) + \lambda L_{\text{adv.}}(\{E,D,Z\}) \right]. \quad (4.2.2)$$

2. Training an autoregressive transformer to maximize the log-likelihood of the encoded sequences.

3. At inference, sampling a sequence with the transformer and decoding it with the decoder D.

This vector-quantized representation was later improved by Yu et al. (2021a) and used by Yu et al. (2022) to create PARTI, a state-of-the-art text-to-image generative model. Interestingly, our work EdiBERT builds on top of the first step of VQGAN, and also requires the training of the triplet (E,D,Z) following (4.2.2).

4.2.2.2 Bidirectional attention

The main property of autoregressive models is that they only perform attention on previous tokens, making them inadequate when dealing with image manipulation (Esser et al., 2021a). Some works alleviate this bias in different ways. Yang et al. (2019) learn an autoregressive model on random permutations of the ordering. Cao et al. (2021) propose a model where missing tokens are inferred autoregressively, conditionally to the set of kept tokens. Similarly, Wan et al. (2021) use an auto-regressive procedure conditioned on the masked image, while Yu et al. (2021b) use BERT training with [MASK] tokens and Gibbs sampling. If this setting is ideal for tasks with masked tokens such as inpainting, it makes it ill-posed for scribble-editing and insertion without existing paired datasets. On the opposite, our EdiBERT tackles all tasks without any need for supervision. Finally, Esser et al. (2021a) train ImageBART, a multinomial diffusion process (Hoogeboom et al., 2021) in the discrete latent space of VQGAN. Each generated sequence is conditioned on the previous one and performs attention to the

whole image. However, this method is computationally heavy since it requires making $N \times L$ inferences, where N is the number of generated sequences and L is the number of tokens in the sequence. A more efficient way to perform bidirectional attention for image generation has been proposed in MaskGIT (Chang et al., 2022). MaskGIT consists of training with a BERT-like objective (Devlin et al., 2018) on sequences randomly perturbed with [MASK] tokens, and generating images with a parallel decoding scheme. Similarly, Zhang et al. (2021b) propose to use a masking-based strategy to perform conditional image editing with bidirectional attention mechanisms. However, they still require specific conditional data to learn their model editing model. We argue that by performing bidirectional attention over all the tokens and learning with a denoising objective (tokens perturbed by randomization instead of [MASK] tokens), it is possible to train a single model tackling many editing tasks.

4.2.2.3 Unifying image manipulation

Initially, image manipulation methods were implemented without any trainable parameters. Image completion was tackled using nearest-neighbor techniques along with a large dataset of scenes (Hays and Efros, 2007). As to image insertion, blending methods were widely used, such as the Laplacian pyramids (Burt and Adelson, 1987). In recent years, image manipulation has benefited from the advances of deep generative models. A first line of work has consisted of gathering datasets of corrupted and target images to train conditional generative models. By doing so, one can therefore learn a mapping from any corrupted image to a real one. For example, Liu et al. (2021) proposes an encoder-decoder architecture for sketch-guided image inpainting. However, in all cases, a dataset with both types of images is required, therefore limiting the applicability.

To avoid this dependency, a second idea - known as GAN inversion methods - leverages pre-trained unconditional GANs. They work by projecting edited images on the manifold of real images learned by the pre-trained GAN. It can be solved either by optimization (Abdal et al., 2019, 2020; Daras et al., 2021), or with an encoder mapping to the latent space (Chai et al., 2021; Richardson et al., 2021; Tov et al., 2021). Pros of these GAN-based methods are that one benefits from the outstanding properties of StyleGan, state-of-the-art in image generation. However, these methods rely on a task-specific loss function that needs to be defined and optimized. More recently, another line of research is based on the development of score-based models (Song et al., 2020): Meng et al. (2022) use Langevin dynamics for image edition, and (Esser et al., 2021a) combine discrete diffusion models (Hoogeboom et al., 2021; Austin et al., 2021) with the discrete vector-quantized representations from VQGANs.

4.2.3 Motivating EdiBERT for image editing

This section gives a global description of the proposed EdiBERT model. We start with notations before describing the different steps leading to the BERT-based edition.

4.2.3.1 Discrete auto-encoder VQGAN

Let I be an image with width w, a height h, and a number c of channels. I thus belongs to $\mathbb{R}^{h\times w\times c}$. Let (E,D,Z) be respectively the encoder, decoder, and codebook defined in VQVAE and VQGAN (Van Den Oord et al., 2017; Esser et al., 2021b). The codebook Z consists of a finite number of tokens with fixed vectors in an embedding space: $Z=\{t_1,\dots,t_N\}$ with $t_k\in\mathbb{R}^d$ and N being the cardinality of the codebook.

For any given image I, the encoder E outputs a vector $E(I)\in\mathbb{R}^{L\times d}$, which is then quantized and reshaped into a sequence s of length L as follows:

$$s=(\underset{z\in Z}{\arg\min}\,\|E(I)_1-z\|,\dots,\underset{z\in Z}{\arg\min}\,\|E(I)_L-z\|)=Q_Z(E(I)), \tag{4.2.3}$$

where $E(I)_l=E(I)_{l,:}\in\mathbb{R}^d$ is the feature vector of $E(I)$ at position l, and Q_Z refers to the quantization operation using the codebook Z. Recall that, after the quantization step, one gets a sequence composed of L codebook elements, thus $s\in Z^L$. After we feed the codebook embeddings to the decoder D, the reconstructed image becomes $\hat{I}=D(Q_Z(E(I)))$.

Let's note $\mathscr{D}$, the available image dataset. From a pre-trained encoder E and codebook Z, one can transform the image dataset $\mathscr{D}$ into a dataset of token-sequences $\mathscr{D}_S:=\{Q_Z(E(I)),I\in\mathscr{D}\}$. When learning transformers on sequences of tokens, the practitioner is directly working with $\mathscr{D}_S$.

4.2.3.2 Learning sequences with autoregressive models

The following sections aim at motivating the training objective for the EdiBERT model. To begin with, let p_θ be a transformer model parameterized with Θ trained on $\mathscr{D}_S$. For each position i in s, we note $p^i_\theta(.|s)$, the modeled distribution of tokens conditionally to s.

When training an autoregressive transformer on the discrete sequences of tokens $\mathscr{D}_S$ (Esser et al., 2021b), one needs to compute the likelihood $p_\theta(s)$ of each given sequence $s=(s_1,\dots,s_L)\in\mathscr{D}_S$ as follows:

$$p_\theta(s)=\prod_{i=1}^{L}p^i_\theta(s_i|s_{<i}),\quad\text{with } s_{<i}=(s_1,\dots,s_{i-1}). \tag{4.2.4}$$

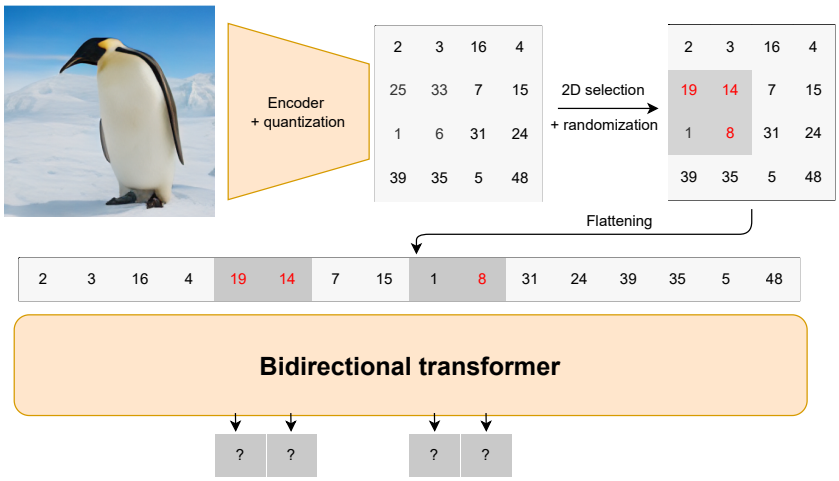


Fig. 4.2 The 2D selection and randomization strategy for the training of our bidirectional transformer: EdiBERT is trained on sequences where localized patch of tokens have been perturbed.

Conditional distributions $p_\theta^i(s_i|s_{<i})$ are computed using a causal left-to-right attention mask. The final objective of the autoregressive model is to find the best set of parameters within Θ:

$$\arg\max_{\theta\in\Theta} \mathbb{E}_{s\in\mathscr{D}_s} \log p_\theta(s). \tag{4.2.5}$$

Limitations of the model. If this setting is well suited for unconditional image generation, it is ill-posed for image manipulation tasks, as shown by Esser et al. (2021a). In the case of scribble-based editing, or inpainting, one wants to resample tokens conditionally to the whole image, so that the model has all the information at its disposal.

4.2.3.3 A unique training objective for EdiBERT.

Let us define the training objective for EdiBERT. For a sequence $s = (s_1, ..., s_L)$, a function φ randomly selects a subset of k indices $\{\varphi_1, ..., \varphi_k\}$ where $\varphi_k < L$. At each selected position φ_i, a perturbation is applied on the token s_{φ_i}. We attribute a random token with probability p, or keep the same token with probability $1-p$. Consequently, the perturbed token $\tilde{s}_{\varphi_i}$ becomes:

$$\begin{aligned}\tilde{s}_{\varphi_i} &= \mathbb{U}(Z) \quad \text{with probability } p,\\ \tilde{s}_{\varphi_i} &= s_{\varphi_i} \quad \text{with probability } 1-p,\end{aligned}$$

where $\mathbb{U}(Z)$ refers to the uniform distribution on the space of tokens Z. Similarly to Bao et al. (2022), the sampling function φ is defined with a 2D selection strategy, and the positions are selected by drawing random 2D rectangles, see in Figure 4.2. Contrarily to Bao et al. (2022) and Devlin et al. (2018), we only use random tokens from the codebook but no [MASK] tokens. We argue this setting corresponds more to the cases of denoising and editing, where tokens have to be sampled conditionally to an entire perturbed sequence.

Let us now call $\tilde{s} = (s_1, \ldots, \tilde{s}_{\varphi_1}, \ldots, \tilde{s}_{\varphi_k}, \ldots, s_L)$ the perturbed sequence, and $\tilde{\mathscr{D}}_s = \{\tilde{s},\ s \in \mathscr{D}\}$ the perturbed dataset. The training of EdiBERT optimizes the following objective :

$$\arg\max_{\theta \in \Theta}\ \mathbb{E}_{\tilde{s} \in \tilde{\mathscr{D}}_s}\ \frac{1}{k} \sum_{i=1}^{k} \log p_\theta^i(s_{\varphi_i} | \tilde{s}). \tag{4.2.6}$$

Contrary to (4.2.5), we note that the objective in (4.2.6) does not require a causal left-to-right attention. Instead, the attention can be performed over the whole input sequence.

Sampling from EdiBERT: Wang and Cho (2019) show that it is possible to generate realistic samples with a BERT model starting with random initialization. However, compared with standard autoregressive language models, the authors stress that BERT generations are more diverse but of slightly worse quality. Building on the findings of Wang and Cho (2019), we do not aim to use BERT for pure unconditional sequence generation but rather improve an already existing sequence of tokens. In our defined EdiBERT model, for any given position $i \in s$, a token will be sampled according to the multinomial distribution $p_\theta^i(.|s)$.

4.2.3.4 On the locality of Vector Quantization encoding

In this section, we argue that one of the main advantages of EdiBERT comes from the VQ latent space proposed by Van Den Oord et al. (2017) where each image is encoded in a discrete sequence of tokens. In this section, we illustrate with simple visualizations the property of this VQGAN encoding. We explore the spatial correspondence between the position of the token in the sequence and a set of pixels for the encoded image. We aim at answering the following question: do local modifications of the image lead to local modifications of the latent representation and *vice versa*?

Modifying the image. To answer this question, images are voluntarily perturbed with grey masks ($i \longrightarrow i_m$). Then, we encode the two images, quantize their representation using a pre-trained codebook, and plot the distance between the two latent representations: $\|Q_Z(E(i)) - Q_Z(E(i_m))\|_2^2$. The results are shown in the first row in Figure 4.3. Due to the large receptive field of the encoder, tokens can be influenced by distant parts of the image: the down-sampled

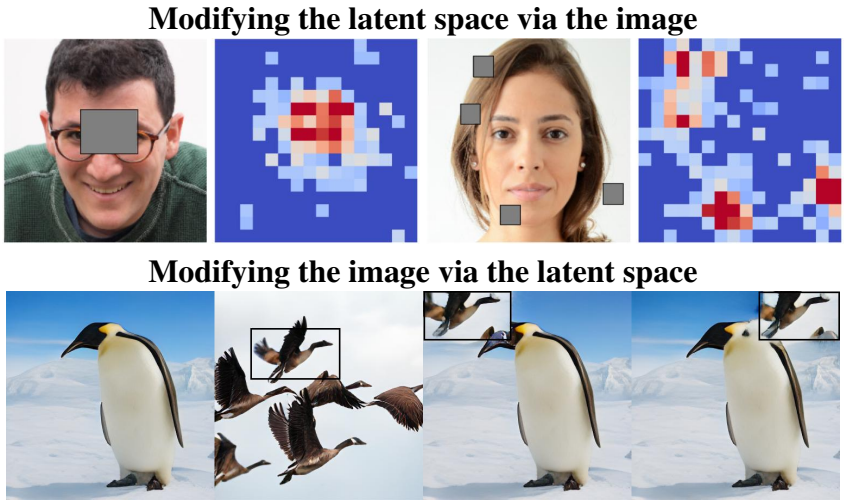


Fig. 4.3 Each token in the sequence is tied to a small spatial area in the decoded image. In the 1st row: we voluntarily perturb images and display the variations among the tokens in the latent space. The heatmaps represent the distance (red is high) between the tokens of the original image and the tokens of the perturbed image. In the 2nd row: we stress how collages of images can easily be done with this discrete latent representation: third and fourth images are generated by the decoder from a latent space collage.

mask does not recover all of the modified tokens. However, tokens that are largely modified are either inside, or very close to the down-sampled mask.

Modifying the latent space. To understand the correspondence between tokens and pixels, we stress how one can easily manipulate images using the discrete latent space. In Figure 4.3, we show that cutting a specific area of a source image to insert it in a different location of another image is possible only by replacing the corresponding tokens in both sequences. This spatial correspondence between VQGANs' latent space and the image is interesting for localized image editing tasks, *i.e.* tasks that require modifying only a subset of pixels without altering the other ones.

4.2.3.5 On the reconstruction capabilities of Vector Quantization encoding

A limit of the framework resides in the use of the vector quantization operation and the induced loss of information. Indeed, we observe in Figure 4.4 that VQGAN struggles to reconstruct high-frequency details, for example complex backgrounds on FFHQ dataset (Karras et al., 2019b). To improve the reconstruction capabilities of VQGANs, we propose a simple optimization procedure over the latent space vectors.

Original. **VQGAN + optim.** (av. LPIPS = 0.085) VQGAN. (av. LPIPS = 0.232) Id-GAN (av. LPIPS = 0.164) I2SG †++ (av. LPIPS = 0.345)

Fig. 4.4 Comparison of reconstruction capabilities of VQGAN + optimization to two GANs inversion methods such as Id-GAN (Zhu et al., 2020) and I2SG†++ (Abdal et al., 2020). Averaged LPIPS are computed on the validation set FFHQ.

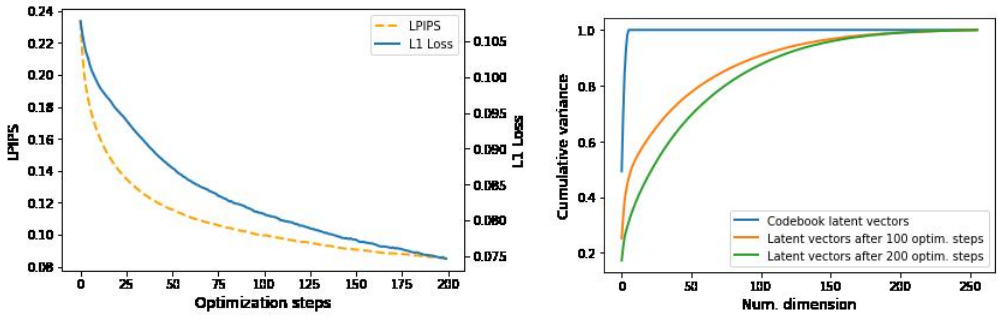


Fig. 4.5 Analysis of reconstruction capabilities of VQGAN. On the left, we see that both the L1 and perceptual loss (LPIPS) between original and reconstructed images significantly decrease when optimizing LPIPS over the latent vectors of VQGAN. This may be a consequence of a higher number of dimensions spanned by the latent vectors (on the right), after the optimization (allowing for more complex reconstructions).

The objective is to find the latent vectors that minimize the LPIPS (Zhang et al., 2018) between the target image and the decoded reconstruction. We initialize the procedure from the output of the encoder $E(I)$, and optimize the objective with gradient descent. Figure 4.4 shows how this procedure improves the inversion capabilities of VQGAN to make it better than GAN inversion methods (Abdal et al., 2020). A potential explanation of the limited reconstruction capabilities of VQGAN is displayed in Figure 4.5: the latent vectors of the codebook might suffer from a very low rank. The optimization procedure seems to solve this since the latent vectors span much more dimensions of the embedding space after a few hundred optimization steps.

4.2.4 Image editing with EdiBERT

Baselines. For each task, we run comparisons with baselines and state-of-the-art models based on GANs inversion methods. On FFHQ, we compare to ImageStyleGAN2++ (Abdal et al., 2020) on pre-trained StyleGANs: StyleGAN2 (Karras et al., 2020b) and StyleGAN2-ADA (Karras et al., 2020a). Besides, we run the solution proposed by Chai et al. (2021) where a StyleGAN2 model is inverted using a trained encoder. Finally, we use In-Domain GAN (Zhu et al., 2020), a hybrid method combining an encoder with an optimization procedure minimizing reconstruction losses. We also compare to Co-Modulated GANs (Zhao et al., 2020), a conditional GAN for inpainting.

Metrics. We follow the work of Chai et al. (2021) and use metrics assessing both fidelity and distribution fitting. The masked L1 metric (Chai et al., 2021) measures the closeness between the generated image and the source image outside the edited areas. The density/coverage metrics (Naeem et al., 2020) are robust versions of precision/recall metrics. Intuitively, density

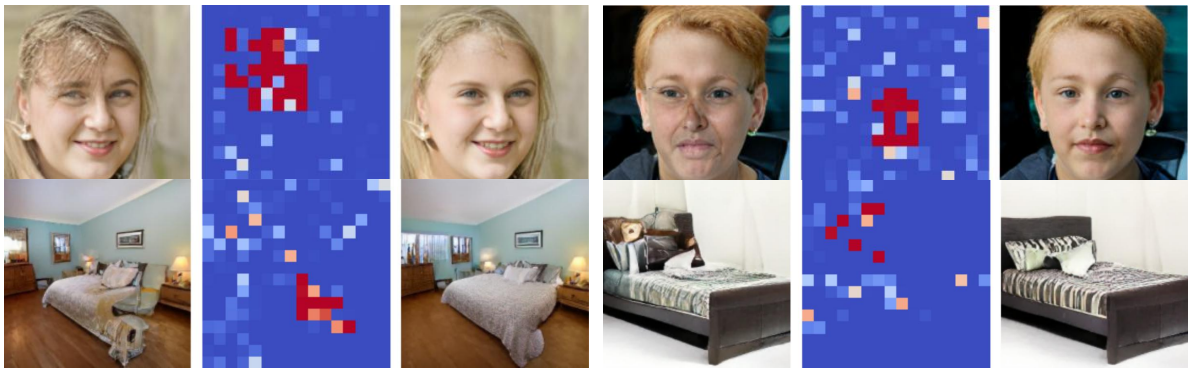


Fig. 4.6 Image denoising with EdiBERT: the color in the 4 different heatmaps is proportional to the negative likelihood of the token. Tokens with a lower likelihood appear in red in the heatmap and have a higher probability of being sampled and edited. Consequently, conditional distributions output by EdiBERT are an efficient tool to detect anomalies and artifacts in the image.

measures fidelity while coverage measures diversity. Finally, the FID (Heusel et al., 2017) quantifies the distance between generated and target distributions. Moreover, we perform a user study on FFHQ image compositing. More details and quantitative results on LSUN Bedroom are presented in Appendix.

4.2.4.1 Localized image denoising

Image denoising aims to improve the quality of a pre-generated image or improve a locally perturbed one. The model has to work without information on the localization of the perturbations. This means we need to find and replace the perturbed tokens with more likely ones to recover a realistic image. Thus, given a sequence $s = (s_1, \ldots, s_L)$, we want to:

1. Detect the tokens that do not fit properly in the sequence s.
2. Change them for new tokens increasing the likelihood of the new sequence.

We desire a significantly more likely sequence with as few as possible token amendments. To do so, we measure the likelihood of each token s_i based on the whole sequence s, aiming to compute $p(s_i|s)$, and replace the least-probable tokens considering them independently. That is, we propose to use the conditional probability output by the model in order to detect and sample the *less likely* odd tokens. Some examples of image denoising are presented in Figure 4.6, and we observe that EdiBERT is able to detect artifacts and replace them with more likely tokens. The full algorithm is given in the Algorithm 3.

Fig. 4.7 Image inpainting comparisons on FFHQ. EdiBERT performs better than inversion methods such as LC (Chai et al., 2021) and I2SG (Abdal et al., 2020). Note that Com-GAN (Zhu et al., 2020) is specialized for image inpainting and was trained on pair datasets (masked image, target image), it can not perform other image editing tasks.

Algorithm 3: Image denoising

Requires: Sequence $(s_1,\dots,s_L)$, BERT model p_θ, number of iterations T;
for *iterations in [0,T]* **do**
  Compute $p_i = \text{logit}(-p^i_\theta(s_i|s)), \forall i \in [1,L]$;
  Sample $p \sim (p_1,\dots,p_l)$ (*less likely position*);
  Sample $t \in Z \sim p^p_\theta(\cdot|s)$;
  Insert sampled token: $s_i \leftarrow t$;
end
Image $\leftarrow$ Decoder(s);
Result: Image

4.2.4.2 Image inpainting

In this setting, we have access to a masked image $i_m \in \mathbb{R}^{h\times w\times c}$ along with the location of the binary mask $m \in \mathbb{R}^{h\times w}$. i_m has been obtained by masking an image $i \in \mathbb{R}^{h\times w\times c}$ as follows: $i_m = i \odot m$ with $\odot$ pointwise multiplication. The goal of image inpainting is to generate an image $\hat{i}$ that is both realistic (high density) and conserves non-masked parts, that is $\hat{i} \odot (1-m) = i \odot (1-m)$.

Among the different tasks in image manipulation, image inpainting stands out. Indeed, when masking a specific area of an image, one shall not consider the pixels within the mask to recover the target image. The image inpainting task thus requires specific care to reach a state-of-the-art performance; this is why we added five different elements to our approach, and validated these elements with visual results in Figure 4.8.

1. **Randomization:** to erase the mask influence, the tokens within the mask are given random values.

2. **Dilation of the mask:** as shown in Figure 4.3, some tokens outside of the down-sampled mask in the latent space are also impacted by the mask on the image. Modifying only tokens inside the down-sampled mask might not be enough and could lead to images with irregularities on the borders. As a solution, we apply a dilation on the down-sampled mask and show in Figure 4.8 that it helps better blend the target image's completion since the boundaries are removed.

3. **Spiral ordering:** since there is no pre-defined ordering of positions in EdiBERT, one can look for an optimal sampling of positions. We argue that by sampling positions randomly, one does not fully leverage the spatial location of the mask. Instead, we propose a spiral ordering that goes from the border to the inside of the mask. Qualitative and quantitative results in Figure 4.8 and Table 4.2 confirm the advantage of this ordering.

4. **Periodic image collage:** to preserve fidelity to the original image, we periodically perform a collage between the masked image and the decoded image. We observed in Figure 4.8, that without this collage trick, the reconstruction can diverge too much from the input image.

5. **Online optimization on latent sequences:** to improve fidelity to the masked image i_m, the final stage of the algorithm consists in an optimization procedure on the latent sequence $s \in \mathbb{R}^{h \times w \times d}$. The objective function is defined as:

$$L = L_p\big((D(s) - i_m) \odot m\big) + L_p\big((D(s) - D(s^0)) \odot (1 - m)\big) \tag{4.2.7}$$

where L_p is a perceptual loss (Zhang et al., 2018), and s^0 is the initial sequence from EdiBERT. Intuitively, the first term enforces the decoded image to get closer to the masked image i_m, while the second term is a regularization enforcing the decoded image to stay similar to the completion proposed by the transformer's likelihood. We illustrate in Figure 4.5 and Figure 4.8 that the optimization leads to a better-preserved source image.

Analyzing the results: we see from Table 4.1 and Figure 4.7 that the specialized method com-GAN (Zhao et al., 2020) outperforms non-specialized methods on image inpainting. This was expected since it is the only method that has been trained specifically for this task. Note that the trained model co-mod GAN cannot be used in any other image manipulation task. Compared with the non-specialized method, EdiBERT always provides better fidelity to the source image (lower Masked L1) and realism (best FID and top-2 density). An ablation study is available in Table 4.2 and validates our choices. Finally, more details regarding the sampling algorithm for the task of inpainting are given in Appendix.

4.2.4.3 Image composition

In this setting, we have access to a non-realistically edited image $i_e \in \mathbb{R}^{h \times w \times c}$. The edited image i_e is obtained by a composition between a source image $i_s \in \mathbb{R}^{h \times w \times c}$ and a target image $i_t \in \mathbb{R}^{h \times w \times c}$. The target image can be a user-drawn scribble or another real image in the case of image compositing. Besides, pixels are edited inside a binary mask $m \in \mathbb{R}^{h \times w}$, which indicates the areas modified by the user. Thus, the final edited image is computed pointwise as:

$$i_e = i_s \odot m + i_t \odot (1 - m). \tag{4.2.8}$$

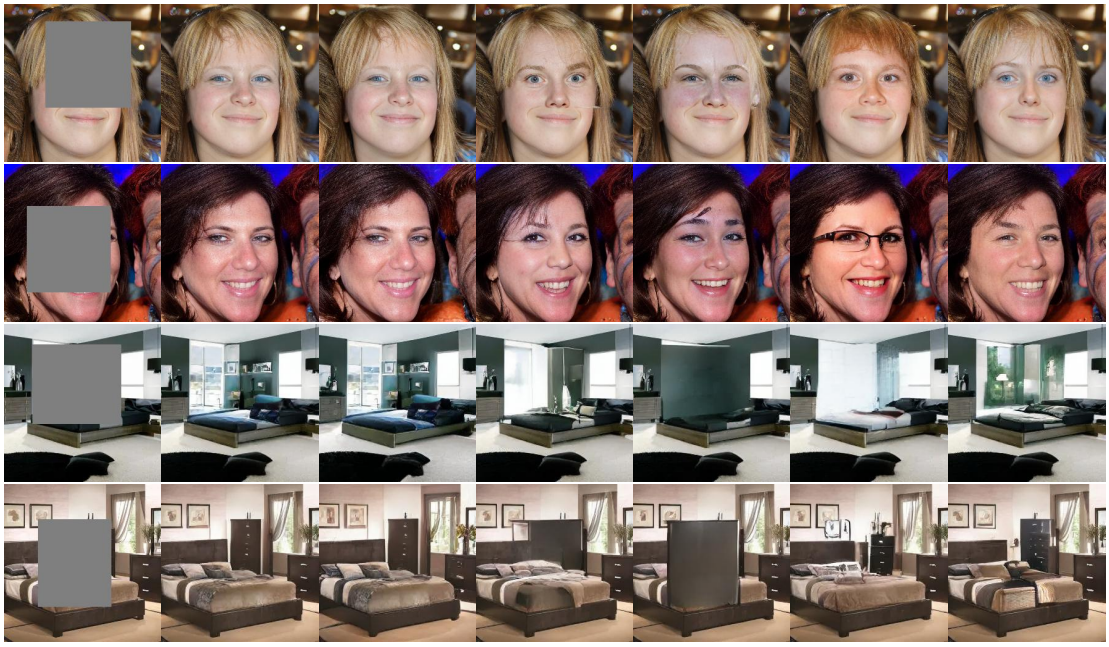


Fig. 4.8 Ablation study for inpainting. Components removed are (a) optimization, (b) dilation, (c) randomization, (d) collage, (e) spiraling (random order instead). Optimization improves fidelity to the source image, while the other components help increase image quality.

Table 4.1 Image inpainting and compositing on FFHQ 256×256. Com-GAN is a model specific for image inpainting, ID-GAN handles several editing tasks but not inpainting, while other methods handle both. I2SG†++ refers to Abdal et al. (2020) with a better GAN backbone (Karras et al., 2020a), LC to Chai et al. (2021), Com-GAN to Zhu et al. (2020), ID-GAN to Zhu et al. (2020).

	Inpainting				**Compositing**		
	Masked L1 ↓	**FID** ↓	**Dens.** ↑	**Cover.** ↑	**Masked L1** ↓	**Dens.** ↑	**User study** ↑
I2SG++	0.0767	23.6	0.99	0.88	0.0851	0.77	-
I2SG†++	0.0763	22.1	1.25	0.91	0.0866	**1.07**	8.3%
LC	0.1027	27.9	1.12	0.84	0.1116	1.00	14.8%
EdiBERT ⋆	0.0290	13.8	1.16	0.98	**0.0307**	0.94	**61.2%**
Com-GAN	**0.0086**	**10.3**	**1.42**	**1.00**	-	-	-
ID-GAN	-	-	-	-	0.0570	0.75	15.7%

Image composition aims to transform an edited image i_e to make it more realistic and faithful without limiting the changes outside the mask. We note the source image i_s outside the mask and the edits of the target image i_m for the inserted elements in the edition mask. Three tasks fall under this umbrella: *scribble-based editing*, *image compositing*, and *image crossovers*.

Results of image compositing on FFHQ are presented in Table 4.1 and Figure 4.9. EdiBERT always has the lowest masked L1. We also present the results from a user study in Table 4.1. 30 users were shown 40 original and edited images, along with four results (EdiBERT and baselines). They were asked which one is preferable, accounting for both fidelity and realism. The survey shows that on average, users prefer EdiBERT over competing approaches. We give more visual results along with the detailed answers of the user study in Appendix.

Table 4.2 Inpainting: Ablation study on the components of EdiBERT sampling algorithm. EdiBERT (1st row) shows the best tradeoff between fidelity (masked L1) and quality (FID, density/coverage).

Ordering	Optim-ization	Random-ization	Collage	Dilation	Masked L1 ↓	FID ↓	Density ↑	Coverage ↑
Spiral	✓	✓	✓	✓	0.0201	**19.4**	1.14	**0.96**
Random	✓	✓	✓	✓	0.0206	20.7	1.13	0.95
Spiral	X	✓	✓	✓	0.0299	20.3	1.20	0.94
Spiral	✓	X	✓	✓	0.0198	20.5	**1.26**	0.92
Spiral	✓	✓	X	✓	0.0252	19.9	1.11	0.95
Spiral	✓	✓	✓	X	**0.0197**	23.3	0.96	0.91

4.2.5 Discussions

EdiBERT is a bidirectional transformers model that can tackle multiple editing tasks from one single training. One of the key elements of the proposed method is that it does not require having access to paired datasets (source, target), or unpaired image datasets. This property shows how flexible EdiBERT is and why it can be easily applied to different tasks. Overall, the proposed framework is simple and tractable: 1) train a VQGAN (Esser et al., 2021b), 2) train an EdiBERT model following the objective defined in (4.2.6).

Interestingly, for simple applications, one can directly train EdiBERT based on the representations output by the VQGAN pre-trained on ImageNet. However, for more complex data or when dealing with multiple domains, one might have to train a specialized codebook, which requires a large auto-encoder and a lot of data. Another EdiBERT's drawback is related to the core interest of image editing. Since the tokens are predominantly localized, EdiBERT is perfectly suited for small manipulations that only require amending a few numbers of tokens.

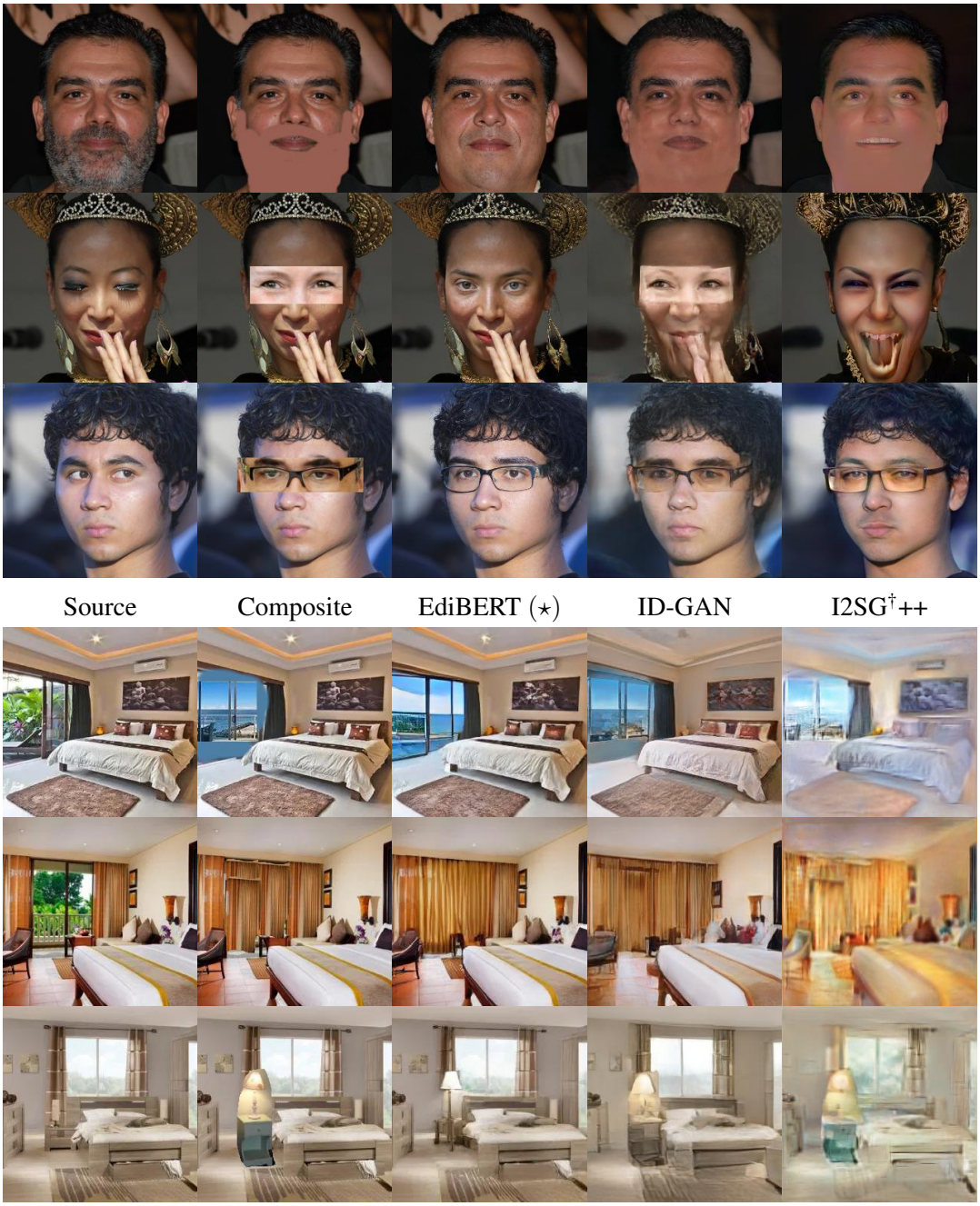


Fig. 4.9 Scribble-based editing and image compositing: comparison with ID-GAN (Zhu et al., 2020) and I2SG (Abdal et al., 2019). EdiBERT preserves better the fidelity to the source image while being also able to fit the inserted object properly. This confirms the quantitative results in Table 4.1, EdiBERT seems to be leading in both fidelity and realism.

However, some manipulations such as zooms or rotations require changing large areas of the source image. In these cases, modifying a large number of tokens might be more demanding.

4.2.6 Conclusion

In this section, we demonstrated the possibility to perform several editing tasks by using the same pre-trained model. The proposed framework is simple and aims at making a step towards a unified model able to do any conceivable manipulation task on images. An exciting direction of research would be to extend the editing capabilities of EdiBERT to global transformations (*e.g.* zoom, rotation).

4.3 A parser-free virtual try-on

Abstract. The 2D virtual try-on task has recently attracted a great interest from the research community, for its direct potential applications in online shopping as well as for its inherent and non-addressed scientific challenges. This task requires fitting an in-shop cloth image on the image of a person, which is highly challenging because it involves cloth warping, image compositing, and synthesizing. Casting virtual try-on into a supervised task faces a difficulty: available datasets are composed of pairs of pictures (cloth, person wearing the cloth). Thus, we have no access to ground-truth when the cloth on the person changes. State-of-the-art models solve this by masking the cloth information on the person with both a human parser and a pose estimator. Then, image synthesis modules are trained to reconstruct the person image from the masked person image and the cloth image. This procedure has several caveats: firstly, human parsers are prone to errors; secondly, it is a costly pre-processing step, which also has to be applied at inference time; finally, it makes the task harder than it is since the mask covers information that should be kept such as hands or accessories. In this section, we propose a novel student-teacher paradigm where the teacher is trained in the standard way (reconstruction) before guiding the student to focus on the initial task (changing the cloth). The student additionally learns from an adversarial loss, which pushes it to follow the distribution of the real images. Consequently, the student exploits information that is masked to the teacher. A student trained without the adversarial loss would not use this information. Also, getting rid of both human parser and pose estimator at inference time allows obtaining a real-time virtual try-on.

4.3.1 Introduction

A photo-realistic virtual try-on system would provide a significant improvement for online shopping. Whether used to create catalogs of new products or to propose an immersive environment for shoppers, it could impact e-commerce and open the door for automated image-editing possibilities.

Earlier work addresses this challenge using 3D measurements and model-based methods (Guan et al., 2012; Hahn et al., 2014; Pons-Moll et al., 2017). However, these are, by nature, computationally intensive and require expensive material, which would not be acceptable at scale for shops. Recent works aim to leverage deep generative models to tackle the virtual try-on problem (Dong et al., 2019; Han et al., 2018; Jetchev and Bergmann, 2017; Wang et al., 2018a). CAGAN (Jetchev and Bergmann, 2017) is a U-net based Cycle-GAN (Zhu et al., 2017) approach. However, this method fails to generate realistic results since such networks cannot handle large spatial deformations. In VITON (Han et al., 2018), the authors recast the

Fig. 4.10 Typical failure cases of the human parser. On the two first rows, it does not segment the person properly. On the third row, it masks the handbag which we would like to preserve in a virtual try-on. CP-VTON and our T-WUTON, which rely on the parsing information, are not robust to a bad parsing. However, the student model S-WUTON which is distilled from the human parser, pose estimator and T-WUTON, can preserve the person's attributes and does not rely on the parsing information.

virtual try-on as a supervised task. They propose to use a human parser and a pose estimator to mask the cloth in the person image and construct an agnostic person representation $p^\star$. The human parser allows segmenting the upper-body and the cloth, while the pose estimator locates the keypoints (*i.e.* shoulders, wrists, etc.) of the person. Then, with $p^\star$ and the image of the original cloth c on a white background, they train a model in a fully supervised fashion to reconstruct p. Namely, they propose a coarse-to-fine synthesis strategy with shape context matching algorithm (Belongie et al., 2002) to warp the cloth on the target person. To improve this model, CP-VTON (Wang et al., 2018a) incorporates a convolutional geometric matcher (Rocco et al., 2017), which learns geometric deformations (*i.e.* thin-plate spline transform (Bookstein, 1989)) that align the cloth with the person. State-of-the-art models are based on the supervised formulation of the virtual try-on task, which has some drawbacks. Human parsers and pose estimators are trained on other datasets and thus fail in some situations (see

Figure 4.10, two first rows). Retraining them on fashion datasets would require similar labels of semantic segmentation or unsupervised domain adaptation methods. Even though they would still be imperfect. Moreover, for a virtual try-on, one wants to preserve person's attributes like handbags or jewels. When constructing $p^\star$, these person's attributes are masked and can not be preserved, such as the partially masked handbag on the third row of Figure 4.10. Finally, the human parsing and pose estimation are the wall clock bottleneck of the pipeline.

In our work, we distill (Hinton et al., 2015) the standard pipeline of virtual try-on composed of human parser, pose estimator, and synthesis modules in the synthesis modules. Namely, we train a student synthesizer with the outputs of a pre-trained standard virtual try-on pipeline. To force the student to use information that is masked to the teacher, we also train the student with an adversarial loss. The distillation process allows us to remove the need for human parsing and pose estimation at inference time, which improves image quality and speeds up the computations from 6FPS to 77FPS. In Figure 4.10, we show visual results of a baseline CP-VTON, our teacher model T-WUTON and our student model S-WUTON. Since S-WUTON does not rely on human parsing, it is robust to parsing errors and preserves a person's attributes such as fingers or handbags.

Additionally, to build an efficient teacher model, we propose an improved architecture for virtual try-on, a Warping U-Net for a Virtual Try-On (WUTON). Our architecture is composed of two modules: a convolutional geometric matcher (Rocco et al., 2017) and a U-net generator with a siamese encoder, where the former warps the feature maps of the latter. The architecture is trained end-to-end, which leads to high-quality synthesized images.

We demonstrate the benefit of our method with several experiments on a virtual try-on dataset, with quantitative and visual results, and a user study.

4.3.2 Problem statement and related work

Given the 2D images $p \in \mathbb{R}^{h\times w\times 3}$ of a person and $c \in \mathbb{R}^{h\times w\times 3}$ of a clothing item, we want to generate the image $\tilde{p} \in \mathbb{R}^{h\times w\times 3}$ where a person p wears the cloth c. The task can be separated in two parts : the geometric deformation T required to align c with p, and the refinement that fits the aligned cloth $\tilde{c} = T(c)$ on p. These two sub-tasks can be modelled with learnable neural networks, *i.e.* spatial transformers networks STN (Jaderberg et al., 2015; Rocco et al., 2017) that output parameters $\theta = STN(p,c)$ of geometric deformations, and conditional generative networks G that give $\tilde{p} = G(p,c,\theta)$.

Because it would be costly to construct a dataset with $\{(p,c),\tilde{p}\}$ triplets, previous works (Han et al., 2018; Wang et al., 2018a) propose to use an agnostic person representation $p^\star \in \mathbb{R}^{h\times w\times c}$ where the clothing items in p are hidden but identity and shape of the persons are preserved. $p^\star$ is built with pre-trained human parsers and pose estimators : $p^\star = h(p)$. These

triplets $\{(p^\star, c), p\}$ allow to train for reconstrution. $(p^\star, c)$ are the inputs, $\tilde{p}$ the output and p the ground-truth. We finally have the conditional generative process :

$$\tilde{p} = G(\underbrace{h(p)}_{\text{agnostic person}}, \underbrace{c}_{\text{cloth}}, \underbrace{STN(h(p), c)}_{\text{geometric transform}}) \tag{4.3.1}$$

Although it eases the training of G, h is a bottleneck in the virtual try-on pipeline. We will show that we can train a student model with synthetic triplets $\{(p, c), \tilde{p}\}$, where $\tilde{p}$ comes from our pre-trained teacher generative model in Eq. 4.3.1. This allows to remove the need for h at inference time for the student model:

$$\hat{p} = G_s(\underbrace{p}_{\text{original person}}, \underbrace{c}_{\text{cloth}}, \underbrace{STN_s(p, c)}_{\text{geometric transform}}) \tag{4.3.2}$$

where G_s and STN_s are the student modules and $\hat{p}$ the generated image.

Conditional image generation. Generative models for image synthesis have shown impressive results with adversarial training (Goodfellow et al., 2014). Combined with deep networks (Radford et al., 2015), this approach has been extended to conditional image generation in Mirza and Osindero (2014) and performs increasingly well on a wide range of tasks. However, as noted by Alami Mejjati et al. (2018), these models cannot handle large spatial deformations and fail to modify the shape of objects, which is necessary for a virtual try-on.

Appearance transfer. Close to the virtual try-on task, some research focus on human appearance transfer. Given two images of different persons, the goal is to transfer the appearance of a part of the person A on the person B. Approaches using pose and appearance disentanglement (Lorenz et al., 2019; Ma et al., 2018) fit this task but others are specifically designed for it. SwapNet (Raj et al., 2018) is a dual path network which generates a new human parsing of the reference person and region of interest pooling to transfer the texture. In Wu et al. (2019), the method relies on DensePose information (Güler et al., 2018), which provides a 3D surface estimation of a human body, to perform a warping and align the two persons. The transfer is then done with segmentation masks and refinement networks. However, the warping relies on matching source and target pose, which is not feasible for the virtual try-on task.

Virtual try-on. Most of the approaches for a virtual try-on system come from computer graphics and rely on 3D measurements or representations. Drape (Guan et al., 2012) learns a deformation model to render clothes on 3D bodies of different shapes. Hahn et al. (2014)

use subspace methods to accelerate physics-based simulations and generate realistic wrinkles. ClothCap (Pons-Moll et al., 2017) aligns a 3D cloth-template to each frame of a sequence of 3D scans of a person in motion. However, the use of 3D scans is expensive and thus not doable for online users.

The task we are interested in is the one introduced in CAGAN (Jetchev and Bergmann, 2017) and further studied by VITON (Han et al., 2018) and CP-VTON (Wang et al., 2018a), which we defined in the problem statement. In CAGAN, the auhtors propose a cycle-GAN approach that requires three images as input: the reference person, the cloth worn by the person and the target in-shop cloth. Thus, it limits its practical uses. To facilitate the task, VITON introduces the supervised formulation of the virtual try-on, as described above. Their pipeline separates the task in sub-tasks: constructing the agnostic person representation (*i.e.* mask the area to replace but preserve body shape), warping the cloth and compositing the final image. Based on the agnostic person representation $p^\star$ and the cloth image c, the VITON model performs a generative composition between the warped cloth and a coarse result. The warping is done with a non-parametric geometric transform (Belongie et al., 2002). To improve this model, CP-VTON incorporates a learnable geometric matcher STN (Rocco et al., 2017). The STN is trained to align c on p with a L1 loss on paired images. However, the L1 loss is overwhelmed with the white background and the solid color parts of clothes. Thus, it faces difficulties to align patterns and to preserve inner structure of the cloth. In VTNFP (Yu et al., 2019) and ClothFlow (Han et al., 2019), a module generating the new human parsing is added. It allows to better preserve body parts and edges, but at an increased computational cost. Moreover, ClothFlow (Han et al., 2019) replaces the TPS warping by a dense flow from the target cloth to the person. All these recent works (Han et al., 2018; Wang et al., 2018a; Han et al., 2019; Yu et al., 2019) rely on pre-trained human parser and pose estimator.

Recent work MG-VTON (Dong et al., 2019) extends the task to a multi-pose virtual try-on system, where they also change the pose of the reference person. Similarly to (Dong et al., 2018; Yu et al., 2019; Dong et al., 2019), they add a module generating the new human parsing, based on input and target pose information.

4.3.3 Our approach

Our task is to build a virtual try-on system that is able to fit a given in-shop cloth on a reference person. In this work, we build a virtual try-on that does not rely on a human parser nor a pose estimator for inference. To do so, we use a teacher-student approach to distill the standard virtual try-on pipeline composed of human parser, pose estimator, and synthesis module in the synthesis module.

In Section 4.3.3.1, we detail the architecture of our synthesis module WUTON. It is trainable end-to-end and composed of two existing modules: a convolutional geometric matcher *STN* (Rocco et al., 2017) and a U-net (Ronneberger et al., 2015) with siamese encoder whose skip connections from the cloth encoder to the decoder are deformed by *STN*. We then explain its training procedure in the standard supervised setting, which gives the teacher T-WUTON.

We finally explain our distillation process. Once the first generative model is trained, the pipeline $\{h, \text{T-WUTON}\}$ becomes a teacher model for a student model S-WUTON by constructing synthetic triplets $\{(p,c),\tilde{p}\}$. These serve to supervise the training of S-WUTON, which hence does not need a human parser to pre-process the image and construct the agnostic person representation. Importantly, S-WUTON also learns from an adversarial loss so it does not only follow the teacher's distribution and it can learn to preserve a person's attributes.

4.3.3.1 WUTON architecture

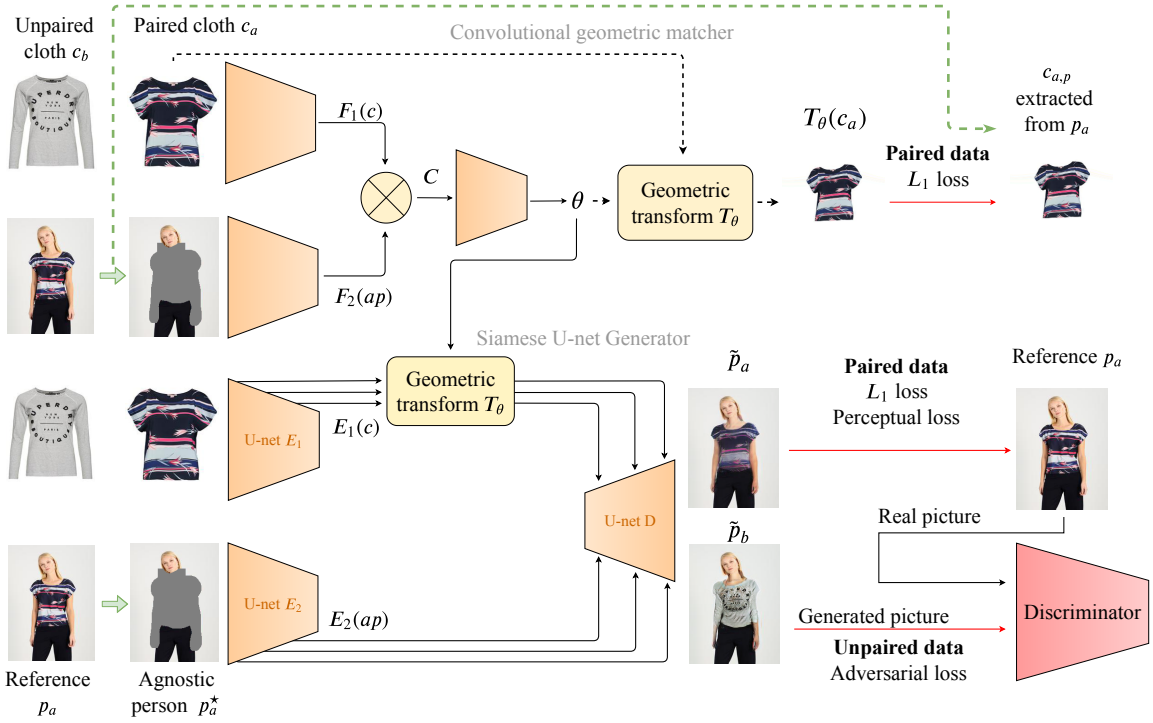


Fig. 4.11 The teacher T-WUTON : our proposed end-to-end warping U-net architecture. Dotted arrows correspond to the forward pass only performed during training. Green arrows are the human parser, red ones are the loss functions. The geometric transforms share the same parameters but do not operate on the same spaces. The different training procedure for paired and unpaired pictures is explained in Section 4.3.3.2.

Our warping U-net is composed of two connected modules, as shown in Figure 4.11. The first one is a convolutional geometric matcher, which has a similar architecture as (Rocco et al.,

2017; Wang et al., 2018a). It outputs the parameters θ of a geometric transformation, a TPS transform in our case. This geometric transformation aligns the in-shop cloth image with the reference person. However, in contrast to previous work (Dong et al., 2019; Han et al., 2018; Wang et al., 2018a), we use the geometric transformation on the feature maps of the generator rather than at a pixel-level. Thus, we learn to deform the feature maps that pass through the skip connections of the second module, a U-net generator which synthesizes the output image $\tilde{p}$.

The architecture of the convolutional geometric matcher is taken from CP-VTON (Wang et al., 2018a), which reuses the generic geometric matcher from Rocco et al. (2017). It is composed of two feature extractors F_1 and F_2, which are standard convolutional neural networks. The local vectors of feature maps $F_1(c)$ and $F_2(p^\star)$ are then L2-normalized and a correlation map C is computed as follows:

$$C_{ijk} = F_{1_{i,j}}(c) \cdot F_{2_{m,n}}(p^\star) \tag{4.3.3}$$

where k is the index for the position (m, n). This correlation map captures dependencies between distant locations of the two feature maps, which is useful to align the two images. C is the input of a regression network, which outputs the parameters θ and allows to perform the geometric transformation T_θ. We use TPS transformations (Bookstein, 1989), which generate smooth sampling grids given control points. Each scale of the U-net is transformed with the same parameters θ.

The input of the U-net generator is also the tuple of pictures $(p^\star, c)$. Since these two images are not spatially aligned, we cannot simply concatenate them and feed a standard U-net. To alleviate this, we use two different encoders E_1 and E_2 processing each image independently and with non-shared parameters. Then, the feature maps of the in-shop cloth $E_1(c)$ are transformed at each scale i: $E_1^i(c) = T_\theta(E_1^i(c))$. Then, the feature maps of the two encoders are concatenated and feed the decoder at each scale. With aligned feature maps, the generator is able to compose them and to produce realistic results. Feature maps warping was also proposed in Dong et al. (2018); Siarohin et al. (2018). We use instance normalization in the U-net generator, which is more effective than batch normalization (Ioffe and Szegedy, 2015) for image generation (Ulyanov et al., 2017).

4.3.3.2 Training procedure of the teacher model

We will now detail the training procedure of T-WUTON, *i.e.* the data representation and the different loss functions of the teacher model.

While previous works use a rich person representation with more than 20 channels representing human pose, body shape and the RGB image of the head, we only mask the upper-body of the reference person. Our agnostic person representation $p^\star$ is thus a 3-channel RGB image with a masked area. We compute the upper-body mask from pose and body parsing information provided by a pre-trained neural network from Liang et al. (2019). Precisely, we mask the areas corresponding to the arms, the upper-body cloth and a bounding box around the neck.

Using the dataset from Dong et al. (2019), we have pairs of in-shop cloth image c_a and a person wearing the same cloth p_a. Using a human parser and a human pose estimator, we generate $p_a^\star$. From the parsing information, we can also isolate the cloth on the image p_a and get $c_{a,p}$, the cloth worn by the reference person. Moreover, we get the image of another in-shop cloth c_b. The inputs of our network are the two tuples $(p_a^\star, c_a)$ and $(p_a^\star, c_b)$. The outputs are respectively $(\tilde{p}_a, \theta_a)$ and $(\tilde{p}_b, \theta_b)$.

The cloth worn by the person $c_{a,p}$ allows us to guide directly the geometric matcher with a L_1 loss:

$$L_{warp} = \|T_{\theta_a}(c_a) - c_{a,p}\|_1 \tag{4.3.4}$$

The image p_a of the reference person provides a supervision for the whole pipeline. Similarly to CP-VTON (Wang et al., 2018a), we use two different losses to guide the generation of the final image $\tilde{p}_a$, the pixel-level L_1 loss $\|\tilde{p}_a - p_a\|_1$ and the perceptual loss (Johnson et al., 2016). We focus on L_1 losses since they are known to generate less blur than L_2 for image generation (Zhao et al., 2016). The latter consists of using the features extracted with a pre-trained neural network, VGG (Simonyan and Zisserman, 2014) in our case. Specifically, our perceptual loss is:

$$L_{perceptual} = \sum_{i=1}^{5} \|\phi_i(\tilde{p}_a) - \phi_i(p_a)\|_1 \tag{4.3.5}$$

where $\phi_i(I)$ are the feature maps of an image I extracted at the i-th layer of the VGG network. Furthermore, we exploit adversarial training to train the network to fit c_b on the same agnostic person representation $p_a^\star$, which is extracted from a person wearing c_a. This is only feasible with an adversarial loss, since there is no available ground-truth for this pair $(p_a^\star, c_b)$. Thus, we feed the discriminator with the synthesized image $\tilde{p}_b$ and real images of persons from the dataset. This adversarial loss is also back-propagated to the convolutional geometric matcher, which allows to generate much more realistic spatial transformations. We use the relativistic adversarial loss (Jolicoeur-Martineau, 2019) with gradient-penalty (Arjovsky et al., 2017; Gulrajani et al., 2017), which trains the discriminator to predict relative realness of real images compared to synthesized ones. Finally, we optimize with Adam (Kingma and Ba, 2014) the

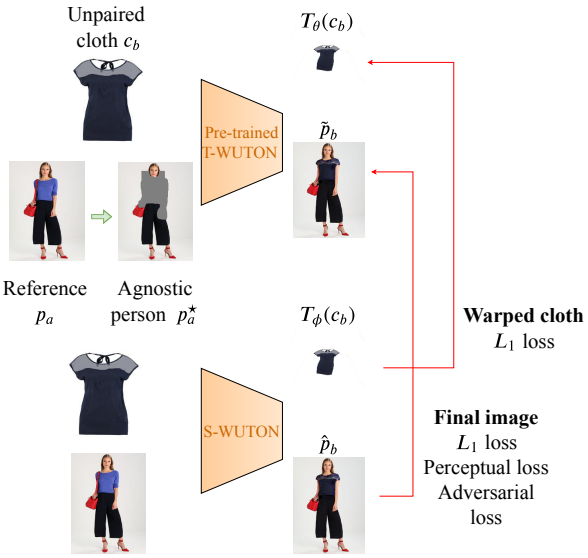


Fig. 4.12 S-WUTON: our training scheme allowing to remove the need for a human parser at inference time. We use human parser and pre-trained T-WUTON to generate synthetic ground-truth for a student model S-WUTON.

following objective function:

$$L = \lambda_w L_{warp} + \lambda_p L_{perceptual} + \lambda_{L_1} L_1 + \lambda_{adv} L_{adv} \tag{4.3.6}$$

4.3.3.3 Training procedure of the student model

We propose to use a teacher-student approach to distill the pipeline composed of $\{h, \text{T-WUTON}\}$ in a single student WUTON (S-WUTON). Indeed, our pre-trained T-WUTON is able to generate realistic images and geometric deformations of clothes on images pre-processed by h. We leverage it and use it as a way to construct generated triplets $\{(p_a, c_b), \tilde{p}_b\}$, where $\tilde{p}_b$ is the image synthesized by T-WUTON. With this pre-trained model, we can supervise the training of a student model S-WUTON. This allows to train the student model on the initial task of changing the cloth rather than reconstructing the upper-body. The student model has the exact same architecture than T-WUTON but different inputs and ground-truth. Hence, its inputs are (p_a, c_b), where p_a is the non-masked image of a person. Having this non-masked image as input, the student model does not need a human parser for pre-processing images. The ground-truth of S-WUTON are the outputs of T-WUTON, for both the warped cloth $T_\theta(c_b)$ and the final synthesized image $\tilde{p}_b$. The training scheme of the student model S-WUTON is shown in Figure 4.12.

More precisely, let us define the inputs-outputs of the teacher and student model: $(\hat{p}_b, \phi) = \text{S-WUTON}(p_a, c_b)$ and $(\tilde{p}_b, \theta) = \text{T-WUTON}(h(p_a), c_b)$. Then, the loss functions of S-WUTON

are:

$$L_{warp} = \|T_\phi(c_b) - T_\theta(c_b)\|_1 \tag{4.3.7}$$

$$L_{perceptual} = \sum_{i=1}^{5} \|\phi_i(\hat{p}_b) - \phi_i(\tilde{p}_b)\|_1 \tag{4.3.8}$$

$$L_1 = \|\hat{p}_b - \tilde{p}_b\|_1 \tag{4.3.9}$$

Finally, the total loss of the student model is:

$$L = \lambda_w L_{warp} + \lambda_p L_{perceptual} + \lambda_{L_1} L_1 + \lambda_{adv} L_{adv} \tag{4.3.10}$$

The adversarial loss L_{adv} is independant from T-WUTON. Here, we also use the relativistic loss with gradient penalty on the discriminator. The real data consists of images of persons from the dataset p_a, and the fake data corresponds to the synthesized images $\hat{p}_b$. Notice that without the adversarial loss, it would be a standard teacher-student setting, where the student is only guided by the outputs of the teacher. In our case, the discriminator (*i.e.* L_{adv}) helps S-WUTON to be close to the real data distribution, and not only to the teacher's distribution. As shown by the ablation study in Section 4.3.4.6, it is an important component and is necessary to make S-WUTON exploit the components that are masked from T-WUTON (*e.g.* hands).

4.3.4 Experiments and analysis

We first describe the dataset. We then compare our approach with CP-VTON (Wang et al., 2018a), a current state-of-the-art for the virtual try-on task. We present visual and quantitative results proving that S-WUTON achieves state-of-the-art results, and that the distillation process allows to improve image quality. We show that this stands for several metrics, and with a user study. We then provide a comparison of the runtime of virtual try-on algorithms on a Tesla NVIDIA V100 GPU. The teacher-student distillation allows to decrease the runtime by an order of magnitude. Finally, we outline the importance of the adversarial loss in our teacher-student setting.

We also show some visual comparisons with recent work VTNFP (Yu et al., 2019). Images are taken from their paper. However, since their model is not available, we could not compute the other metrics. We provide more visual comparisons with VTNFP and ClothFlow (Han et al., 2019) in supplementary material.

Fig. 4.13 On the left side, comparison of our method with CP-VTON (Wang et al., 2018a). For fairness, the two methods are trained on the same dataset and on the same agnostic person representation $p^{\star}$. More examples are provided in supplementary material. On the right side, comparison with recent work VTNFP. Except for S-WUTON's column, images are taken from their paper.

4.3.4.1 Dataset

For copyright issues, we can not use the dataset from VITON (Han et al., 2018) and CP-VTON (Wang et al., 2018a). Instead, we leverage the *Image-based Multi-pose Virtual try-on* dataset. This dataset contains 35,687/13,524 person/cloth images at 256x192 resolution. 4175 pairs are kept for test so the cloth was not seen during training. A random shuffle of these pairs produces the unpaired person/cloth images. For each in-shop cloth image, there are multiple images of a model wearing the given cloth from different views and in different poses. We remove images tagged as back images since the in-shop cloth image is only from the front. We process the images with a neural human parser and pose estimator, specifically the joint body parsing and pose estimation network (Liang et al., 2019).

4.3.4.2 Visual results

Visual results of our method and CP-VTON are shown in Figure 4.13. On the left side, images are computed from models trained on MG-VTON dataset, with $p^{\star}_{t-wuton}$ representation for T-WUTON and CP-VTON for fairness. On the right side, images are taken from VTNFP paper

(Yu et al., 2019). There, CP-VTON and VTNFP were trained on the original dataset from VITON, and CP-VTON uses $p^{\star}_{cp-vton}$. More images from S-WUTON are provided in Figure 4.10 and Figure 4.15.

CP-VTON has trouble to realistically deform and render complex patterns like stripes or flowers. Control points of the T_θ transform are visible and lead to unrealistic curves and deformations on the clothes. Also, the edges of cloth patterns and body contours are blurred.

Firstly, our proposed T-WUTON architecture allows to improve the baseline CP-VTON. Indeed, our method generates spatial transformations of a much higher visual quality, which is specifically visible for stripes (1st row). It is able to preserve complex visual patterns of clothes and produces sharper images than CP-VTON and VTNFP on the edges. Secondly, we can observe the importance of our distillation process with the visual results from S-WUTON. Since it has a non-masked image as input, it is able to preserve body details, especially the hands. Moreover, as shown in Figure 4.10, S-WUTON is robust to a bad parsing and preserves a person's attributes that are important for the virtual try-on task.

Generally, our method generates results of high visual quality while preserving the characteristics of the target cloth and of the person. However, VTNFP can surpass S-WUTON when models are crossing arms (4th row, right side), which is sometimes a failure case of our method. Note that this is not general, since on (3rd row, right side) and (4th row, left side) in Figure 4.13 and on the two last columns in Figure 4.14, models are crossing arms and S-WUTON manages to nicely compose the arms with the occluded cloth.

4.3.4.3 Quantitative results

Table 4.3 Quantitative results on paired setting (LPIPS and SSIM) and on unpaired setting (IS and FID). $\pm$ reports std. dev. T-WUTON and S-WUTON are our proposed models. The two last lines (methods with *) are the results presented in ACGPN (Yang et al., 2020a). However, it has to be taken carefully since the experiments are conducted on another dataset.

Method	LPIPS (↓)	SSIM (↑)	IS (↑)	FID (↓)
Real data	0	1	3.135	0
CP-VTON (A)	0.182±0.049	0.679±0.073	2.684	37.237
CP-VTON (B)	0.131±0.058	0.773±0.088	2.938	16.843
T-WUTON	**0.101**±0.047	**0.799**±0.089	3.114	9.877
S-WUTON	NA	NA	**3.154**	**7.927**
VTNFP*	NA	0.803	2.784	NA
ACGPN*	NA	0.845	2.829	NA

To further evaluate our method, we use four different metrics. Two are designed for the paired setting, that does not allow us to evaluate S-WUTON (because the input image is not masked), and one is for the unpaired setting. The first one for the paired setting is the linear perceptual image patch similarity (LPIPS) developed in Zhang et al. (2018), a state-of-the-art metric for comparing pairs of images. It is very similar to the perceptual loss we use in training (see Section 4.3.3.2) since the idea is to use the feature maps extracted by a pre-trained neural network to quantify the perceptual difference between two images. Different from the basic perceptual loss, they first unit-normalize each layer in the channel dimension and then learn a rescaling that match human perception.

Such as previous works, we also use the structural similarity (SSIM) (Wang et al., 2004) in the paired setting, inception score (IS) (Salimans et al., 2016) and Fréchet Inception Distance (FID) (Heusel et al., 2017) in the unpaired setting. CP-VTON (A) (Wang et al., 2018a) refers to CP-VTON trained with their agnostic person representation $p^{\star}_{cp-vton}$ (20 channels with RGB image of head and shape/pose information), while CP-VTON (B) refers to CP-VTON trained with $p^{\star}_{t-wuton}$. Results are reported in Table 4.3.

4.3.4.4 User study

We perform A/B tests on 7 users. Each one has to vote 100 times between CP-VTON and S-WUTON synthesized images, given reference person and target cloth. The user is asked to choose for the most realistic image, that preserves both person and target cloth details. The selected 100 images are a random subset of the test set in the unpaired setting. This subset is sampled for each user and is thus different for each user. There is no time limit for the users.

Let us denote p the probability that an image from S-WUTON is preferable to an image from CP-VTON. The users choose our method 88% of the time. In terms of statistical significance, it means that we can say $p > 0.85$ with a confidence level of 98.7%.

ClothFlow and VTNFP also performed user studies where they compare to CP-VTON. The authors respectively report that users prefer their method 81.2% and 77.4% of the time. Note that the experiment was not performed in the same setting (dataset, number of users, number of pictures per user).

4.3.4.5 Runtime analysis

In Table 4.4, we compare the runtime of our method to CP-VTON, ClothFlow and VTNFP. Note that the running times are estimated on a NVIDIA V100 GPU. For the human parsing and pose estimation networks, we use state-of-the-art models from Liang et al. (2019). These are based on shared neural backbones for the two tasks, which accelerates the computations.

Table 4.4 Comparison of runtime of state-of-the-art architectures for virtual try-on. The time is computed on a NVIDIA Tesla V100 GPU.

	CP-VTON	VTNFP	ClothFlow	T-WUTON	S-WUTON
Parsing + pose	168ms	168ms	168ms	168ms	**0ms**
Try-on	**9ms**	>9ms	>0ms	13ms	13ms
Total	177ms	>177ms	>168ms	181ms	**13ms**

The try-on architecture of T-WUTON and S-WUTON is slightly slower than that of CP-VTON, due to the non-shared encoder and the warping at each scale of the U-Net. However, with S-WUTON we remove the wall clock bottleneck of virtual try-on system, which is the human parsing and pose estimation. Doing so, we decrease by an order of magnitude the runtime of virtual try-on algorithms, from 6FPS to 77FPS.

We include comparisons with VTNFP and ClothFlow in the Table 4.4. Indeed, both models use human parsing and pose estimation. For VTNFP, they add a module on top of CP-VTON architecture, so their try-on architecture takes at least 9ms per image. For ClothFlow, the use of human parser and pose estimator gives a lower bound on the total runtime.

4.3.4.6 Impact of the adversarial loss in the teacher-student setting

We show the impact of the adversarial loss on S-WUTON. We train a variant student model S-WUTON without the adversarial loss. We provide a comparison of synthesized images in Figure 4.14, and IS and FID scores in Table 4.5. The adversarial loss on the student model is a constraint to make the student model closer to the real data distribution and to not only follow the teacher's distribution. Without the adversarial loss, the student model does not preserve person's attributes, even though they are not masked.

Table 4.5 Comparison of IS and FID scores of S-WUTON and S-WUTON without the adversarial loss.

	$\lambda_{adv} = 0$	$\lambda_{adv} = 1$
IS	2.912	**3.154**
FID	12.620	**7.927**

4.3.5 Conclusion

In this work, we propose a teacher-student setting to distill the standard virtual try-on pipeline and refocus on the initial task: changing the cloth of a non-masked person. This leads to a significant computational speed-up and largely improves image quality. Importantly, this allows to preserve person's attributes such as hands or accessories, which is necessary for a virtual try-on.

Fig. 4.14 Visual comparison of the student model with and without the adversarial loss. Interestingly, the student model without the adversarial loss can not exploit information that is masked to the teacher, *e.g.* arms and hands.

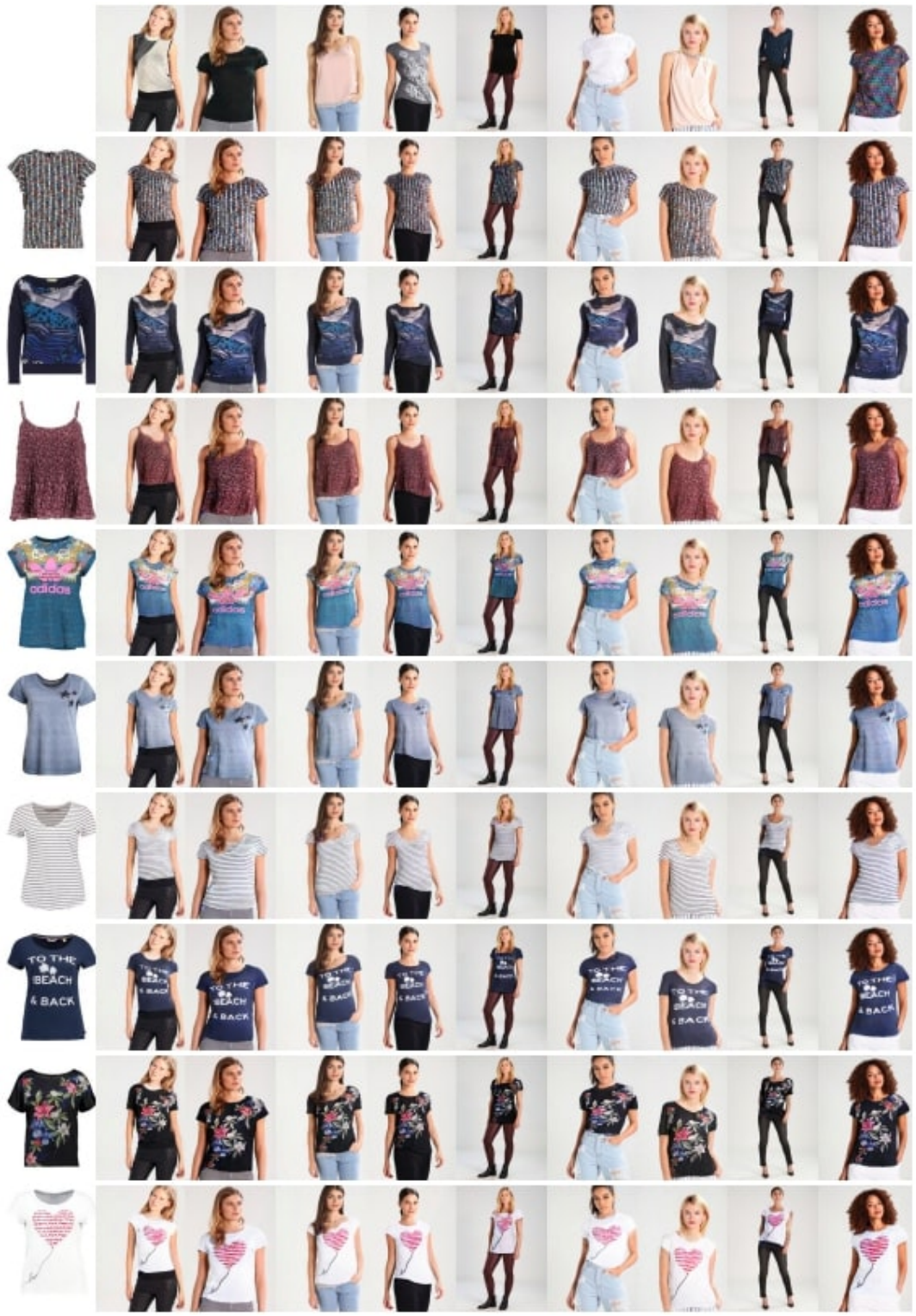


Fig. 4.15 Our student model S-WUTON generates high-quality images and preserves both person's and cloth's attributes.

Chapter 5

Conclusion

5.1 Summary of contributions

In this thesis, we contribute to a better understanding of the behavior of deep generative models on disconnected distributions, which are prevalent in real-world datasets containing various classes and modes. This research setting is crucial for advancing deep generative models. Chapter 3 formalizes a fundamental limitation of some generative models, such as GANs and VAEs, that generate samples outside the target modes because of the continuous generated distribution and disconnected target distribution. Based on our analysis, we propose a truncation method that improves sampling quality. Second, we demonstrate that when dealing with disconnected distribution, there exists an optimal geometry of the latent space which minimizes the proportion of generated samples lying outside the target modes. This geometry is characterized by linearity and convexity of the different regions and can be enforced in GANs to enhance their performance. Finally, to keep on reducing these off-manifold generated samples, we define a rejection mechanism from pre-trained generators. To do so, we train an MLP network to predict importance weights in the latent space. This method can be used on top of any pre-trained generator and helps select latent space areas corresponding to high-quality samples. Our findings highlight the importance of considering the multi-modal nature of data to design methods that improve the sampling quality of deep generative models.

The second research axis focuses on developing generative models for image editing purposes with a view to reducing the need for supervised data, which is often expensive to collect. Our work demonstrates that the use of deep generative models can be effective in this direction. In section 4.2, we propose a transformer-based generative model that handles various editing tasks such as local denoising, image inpainting, and scribble-based editing. The model is trained with a self-supervised objective of predicting randomized tokens, enabling it to learn how to modify parts of an image conditionally to the whole image. In section 4.3, we address

the image-based virtual try-on task and propose a teacher-student approach with adversarial learning to create an ideal synthetic dataset with unmasked input images. We show that this approach outperforms the suboptimal method of relying on pre-trained neural networks for human segmentation and human parsing.

Furthermore, the questions and contributions of this thesis bring interesting future research questions. We detail some of them below.

5.2 Future work

The approaches and methods developed in this thesis can shed light on recent developments in deep learning and deep generative modelling. This opens the way for future research. We describe below some interesting ideas that would be interesting follow-ups:

Disconnected latent space versus disconnected function modelling. In this thesis, we have operated under the assumption that neural networks are Lipschitz and continuous functions. To improve the sampling quality of generative models, we focused on designing the latent space and derived methods to make it non-connected, such as through different types of truncation. However, an alternative approach would be to model disconnected functions directly. This raises the question of what design choices can be made to model disconnected functions. One promising direction for investigation is the use of Sparse Mixture-of-Experts (SMoE) models, which have achieved impressive performance on large-scale discriminative tasks in Natural Language Processing (Shazeer et al., 2017; Fedus et al., 2021) and, more recently, in Computer Vision (Riquelme et al., 2021). Some theoretical advances have suggested that their disconnected nature may be responsible for their strong performance on datasets with multiple clusters (Chen et al., 2022). Exploring the use of SMoE layers for push-forward generative models, both theoretically and experimentally, would be of great interest, as it would enable these models to design disconnected distributions and overcome issues of misspecification. This opens up exciting possibilities for future research in generative modeling.

Latent space of score-based models. In this work, particularly in Section 3.3, we have demonstrated the structured nature of GANs' latent space. When learning multimodal data, different modes are clustered in linear regions of the latent space, and linear interpolations between generated samples follow a reasonably smooth path on the data manifold. In contrast, in score-based models, the interpolations are much more chaotic, and the latent space has less structure. This raises a fundamental question: why do score-based models achieve better performance than GANs? Several hypotheses can be proposed. First, the iterative generation

procedure of score-based models could make their Lipschitz constant greater than that of GANs. In fact, most attempts to distill these models show that performance decreases as the number of generation steps decreases (Salimans and Ho, 2022). Second, the latent space has the same dimension as the data space, making it more difficult to structure due to common issues in high-dimensional problems. Finally, the objective function, which is a simple L2 loss, may be more stable and have a better optimization landscape, leading to better-trained networks. Further investigation is required to fully understand the reasons behind the superior performance of score-based models over GANs.

Neural collapse in deep generative models. In Section 3.3, we establish a connection between the neural collapse phenomenon and the latent space of push-forward generative models. The neural collapse denotes the geometric structure that emerges in the final layer of deep classification networks upon achieving zero training loss. Notably, Papyan et al. (2020) have demonstrated that features in the ante-penultimate layer tend to converge towards class means, which are arranged in a regular simplex configuration featuring equidistant and maximum equiangular relationships. If we consider the class means as the seeds of a Voronoi partition, they would form a simplicial cluster that represents the optimal partition of the latent space derived in Section 3.3.

In our work, we limit the analysis of this geometrical structure to the latent space of deep generative models. However, if this structure arises in the latent space, it is very likely to be propagated through the layers of the generator. Is the generator further separating the modes in its feature space? This seems like a reasonable hyptohesis. First, this would explain methods using feature space of generators as a feature extractor for tasks such as image segmentation. Second, it could allow to exploit the feature space geometrical structure to design rejection mechanisms. We have some preliminary evidence for this. When learning latent importance weights in Section 3.4, we used the intermediate feature space of StyleGAN and it gave good performance for predicting sample quality. However, as we go through the generator's layers, up-sampling layers increase the dimensionality of the features, and we approach the data manifold which often has a highly non-linear structure. Thus, this study would raise some technical challenges.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283.

Abdal, R., Qin, Y., and Wonka, P. (2019). Image2stylegan: How to embed images into the stylegan latent space? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4432–4441.

Abdal, R., Qin, Y., and Wonka, P. (2020). Image2stylegan++: How to edit the embedded images? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8296–8305.

Abdal, R., Zhu, P., Mitra, N. J., and Wonka, P. (2021). Labels4free: Unsupervised segmentation using stylegan. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13970–13979.

Aggarwal, C. C., Hinneburg, A., and Keim, D. A. (2001). On the surprising behavior of distance metrics in high dimensional space. In *Database Theory—ICDT 2001: 8th International Conference London, UK, January 4–6, 2001 Proceedings 8*, pages 420–434. Springer.

Alaa, A., Van Breugel, B., Saveliev, E. S., and van der Schaar, M. (2022). How faithful is your synthetic data? sample-level metrics for evaluating and auditing generative models. In *International Conference on Machine Learning*, pages 290–306. PMLR.

Alami Mejjati, Y., Richardt, C., Tompkin, J., Cosker, D., and Kim, K. I. (2018). Unsupervised attention-guided image-to-image translation. *Advances in neural information processing systems*, 31.

Arjovsky, M. and Bottou, L. (2017). Towards principled methods for training generative adversarial networks. In *International Conference on Learning Representations*.

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In Precup, D. and Teh, Y., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 214–223. PMLR.

Arvanitidis, G., Hansen, L. K., and Hauberg, S. (2018). Latent space oddity: on the curvature of deep generative models. In *ICLR*.

Austin, J., Johnson, D. D., Ho, J., Tarlow, D., and van den Berg, R. (2021). Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34:17981–17993.

Azadi, S., Olsson, C., Darrell, T., Goodfellow, I., and Odena, A. (2019). Discriminator rejection sampling. In *International Conference on Learning Representations*.

Balaji, Y., Chellappa, R., and Feizi, S. (2020). Robust optimal transport with applications in generative modeling and domain adaptation. *Advances in Neural Information Processing Systems*, 33:12934–12944.

Balaji, Y., Sajedi, M., Kalibhat, N. M., Ding, M., Stöger, D., Soltanolkotabi, M., and Feizi, S. (2021). Understanding over-parameterization in generative adversarial networks. In *International Conference on Learning Representations*.

Bao, H., Dong, L., Piao, S., and Wei, F. (2022). BEit: BERT pre-training of image transformers. In *International Conference on Learning Representations*.

Belongie, S., Malik, J., and Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522.

Beyer, K., Goldstein, J., Ramakrishnan, R., and Shaft, U. (1999). When is "nearest neighbor" meaningful? In *Database Theory—ICDT'99: 7th International Conference Jerusalem, Israel, January 10–12, 1999 Proceedings 7*, pages 217–235. Springer.

Biau, G., Cadre, B., Sangnier, M., and Tanielian, U. (2020). Some theoretical properties of GANs. *The Annals of Statistics*, in press.

Biau, G. and Devroye, L. (2015). *Lectures on the nearest neighbor method*. Springer.

Bookstein, F. L. (1989). Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Transactions on pattern analysis and machine intelligence*, 11(6):567–585.

Borell, C. (1975). The brunn-minkowski inequality in gauss space. *Inventiones mathematicae*, 30(2):207–216.

Bottou, L., Peters, J., Candela, J. Q., Charles, D. X., Chickering, M., Portugaly, E., Ray, D., Simard, P. Y., and Snelson, E. (2013). Counterfactual Reasoning and Learning Systems: the example of Computational Advertising. *Journal of Machine Learning Research*, 14(1):3207–3260.

Boucheron, S., Lugosi, G., and Massart, P. (2013). *Concentration inequalities: A nonasymptotic theory of independence*. Oxford university press, address="Oxford".

Brock, A., Donahue, J., and Simonyan, K. (2019). Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*.

Buades, A., Coll, B., and Morel, J.-M. (2005). A non-local algorithm for image denoising. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 60–65. IEEE.

Burt, P. J. and Adelson, E. H. (1987). The laplacian pyramid as a compact image code. In *Readings in computer vision*, pages 671–679. Elsevier.

Cao, C., Hong, Y., Li, X., Wang, C., Xu, C., Fu, Y., and Xue, X. (2021). The image local autoregressive transformer. In *Advances in Neural Information Processing Systems*.

Chai, L., Wulff, J., and Isola, P. (2021). Using latent space regression to analyze and leverage compositionality in gans. *arXiv:2103.10426*.

Chang, H., Zhang, H., Barber, J., Maschinot, A., Lezama, J., Jiang, L., Yang, M.-H., Murphy, K., Freeman, W. T., Rubinstein, M., et al. (2023). Muse: Text-to-image generation via masked generative transformers. *arXiv preprint arXiv:2301.00704*.

Chang, H., Zhang, H., Jiang, L., Liu, C., and Freeman, W. T. (2022). Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11315–11325.

Che, T., Zhang, R., Sohl-Dickstein, J., Larochelle, H., Paull, L., Cao, Y., and Bengio, Y. (2020). Your gan is secretly an energy-based model and you should use discriminator driven latent sampling. *arXiv preprint arXiv:2003.06060*.

Chen, N., Klushyn, A., Kurle, R., Jiang, X., Bayer, J., and Smagt, P. (2018a). Metrics for deep generative models. In *International Conference on Artificial Intelligence and Statistics*, pages 1540–1550. PMLR.

Chen, T., Lucic, M., Houlsby, N., and Gelly, S. (2018b). On self modulation for generative adversarial networks. In *International Conference on Learning Representations*.

Chen, Z., Deng, Y., Wu, Y., Gu, Q., and Li, Y. (2022). Towards understanding the mixture-of-experts layer in deep learning. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K., editors, *Advances in Neural Information Processing Systems*.

Chung, H., Sim, B., and Ye, J. C. (2022). Come-closer-diffuse-faster: Accelerating conditional diffusion models for inverse problems through stochastic contraction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12413–12422.

Coates, A., Ng, A., and Lee, H. (2011). An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223. JMLR Workshop and Conference Proceedings.

Couairon, G., Grechka, A., Verbeek, J., Schwenk, H., and Cord, M. (2022). Flexit: Towards flexible semantic image translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18270–18279.

Couairon, G., Verbeek, J., Schwenk, H., and Cord, M. (2023). Diffedit: Diffusion-based semantic image editing with mask guidance. In *The Eleventh International Conference on Learning Representations*.

Daras, G., Dean, J., Jalal, A., and Dimakis, A. (2021). Intermediate layer optimization for inverse problems using deep generative models. In Meila, M. and Zhang, T., editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 2421–2432. PMLR.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv:1810.04805*.

Devroye, L. and Wise, G. (1980). Detection of abnormal behavior via nonparametric estimation of the support. *SIAM Journal on Applied Mathematics*, 38:480–488.

Dhariwal, P. and Nichol, A. (2021). Diffusion models beat gans on image synthesis. In *Advances in Neural Information Processing Systems*.

Donahue, J. and Simonyan, K. (2019). Large scale adversarial representation learning. *Advances in neural information processing systems*, 32.

Dong, H., Liang, X., Gong, K., Lai, H., Zhu, J., and Yin, J. (2018). Soft-gated warping-gan for pose-guided person image synthesis. *Advances in neural information processing systems*, 31.

Dong, H., Liang, X., Shen, X., Wang, B., Lai, H., Zhu, J., Hu, Z., and Yin, J. (2019). Towards multi-pose guided virtual try-on network. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9026–9035.

Dowson, D. and Landau, B. (1982). The fréchet distance between multivariate normal distributions. *Journal of multivariate analysis*, pages 450–455.

Dudley, R. (2004). *Real Analysis and Probability*. Cambridge University Press, Cambridge, 2 edition.

Esser, P., Rombach, R., Blattmann, A., and Ommer, B. (2021a). Imagebart: Bidirectional context with multinomial diffusion for autoregressive image synthesis. *Advances in Neural Information Processing Systems*, 34.

Esser, P., Rombach, R., and Ommer, B. (2021b). Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883.

Faury, L., Tanielian, U., Dohmatob, E., Smirnova, E., and Vasile, F. (2020). Distributionally robust counterfactual risk minimization. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 3850–3857. AAAI Press.

Federer, H. (2014). *Geometric measure theory*. Springer.

Fedus, W., Zoph, B., and Shazeer, N. (2021). Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23:1–40.

Fefferman, C., Mitter, S., and Narayanan, H. (2016). Testing the manifold hypothesis. *Journal of the American Mathematical Society*, 29:983–1049.

Ge, C., Song, Y., Ge, Y., Yang, H., Liu, W., and Luo, P. (2021). Disentangled cycle consistency for highly-realistic virtual try-on. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16928–16937.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, J. (2014). Generative adversarial nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc.

Grechka, A., Goudou, J.-F., and Cord, M. (2021). Magecally invert images for realistic editing. In *BMVC*.

Gretton, A., Borgwardt, K. M., Rasch, M., Schölkopf, B., and Smola, A. (2012). A kernel two-sample test. *Journal of Machine Learning Research*, 13:723–773.

Grover, A., Song, J., Kapoor, A., Tran, K., Agarwal, A., Horvitz, E. J., and Ermon, S. (2019). Bias correction of learned generative models using likelihood-free importance weighting. In *Advances in Neural Information Processing Systems*, pages 11056–11068.

Guan, P., Reiss, L., Hirshberg, D. A., Weiss, A., and Black, M. J. (2012). Drape: Dressing any person. *ACM Transactions on Graphics (ToG)*, 31(4):1–10.

Güler, R. A., Neverova, N., and Kokkinos, I. (2018). Densepose: Dense human pose estimation in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7297–7306.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. (2017). Improved training of Wasserstein GANs. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 5767–5777. Curran Associates, Inc.

Gurumurthy, S., Kiran Sarvadevabhatla, R., and Venkatesh Babu, R. (2017). Deligan: Generative adversarial networks for diverse and limited data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Hahn, F., Thomaszewski, B., Coros, S., Sumner, R. W., Cole, F., Meyer, M., DeRose, T., and Gross, M. (2014). Subspace clothing simulation using adaptive bases. *ACM Transactions on Graphics (TOG)*, 33(4):1–9.

Hales, T. C. (2001). The honeycomb conjecture. *Discrete & Computational Geometry*, pages 1–22.

Han, X., Hu, X., Huang, W., and Scott, M. R. (2019). Clothflow: A flow-based model for clothed person generation. In *The IEEE International Conference on Computer Vision (ICCV)*.

Han, X., Wu, Z., Wu, Z., Yu, R., and Davis, L. S. (2018). Viton: An image-based virtual try-on network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7543–7552.

Härkönen, E., Hertzmann, A., Lehtinen, J., and Paris, S. (2020). Ganspace: Discovering interpretable gan controls. *Advances in Neural Information Processing Systems*, 33:9841–9850.

Hartmann, V. and Schuhmacher, D. (2020). Semi-discrete optimal transport: a solution procedure for the unsquared euclidean distance case. *Mathematical Methods of Operations Research*, 92(1):133–163.

Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications.

Hays, J. and Efros, A. A. (2007). Scene completion using millions of photographs. *ACM Transactions on Graphics (ToG)*, 26(3).

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. (2022). Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

Heilman, S., Jagannath, A., and Naor, A. (2012). Solution of the propeller conjecture in r3. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 269–276.

Hertz, A., Mokady, R., Tenenbaum, J., Aberman, K., Pritch, Y., and Cohen-or, D. (2023). Prompt-to-prompt image editing with cross-attention control. In *The Eleventh International Conference on Learning Representations*.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in neural information processing systems*, pages 6626–6637.

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851.

Hoffman, J., Tzeng, E., Park, T., Zhu, J.-Y., Isola, P., Saenko, K., Efros, A., and Darrell, T. (2018). Cycada: Cycle-consistent adversarial domain adaptation. In *International conference on machine learning*, pages 1989–1998. Pmlr.

Hoogeboom, E., Nielsen, D., Jaini, P., Forré, P., and Welling, M. (2021). Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in Neural Information Processing Systems*, 34:12454–12465.

Hudson, D. A. and Zitnick, L. (2021). Generative adversarial transformers. In Meila, M. and Zhang, T., editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 4487–4499. PMLR.

Humayun, A. I., Balestriero, R., and Baraniuk, R. (2022). Polarity sampling: Quality and diversity control of pre-trained generative networks via singular values. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10641–10650.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134.

Jaderberg, M., Simonyan, K., Zisserman, A., et al. (2015). Spatial transformer networks. *Advances in neural information processing systems*, 28.

Jahanian, A., Chai, L., and Isola, P. (2019). On the "steerability" of generative adversarial networks. In *International Conference on Learning Representations*.

Jetchev, N. and Bergmann, U. (2017). The conditional analogy gan: Swapping fashion articles on people images. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 2287–2292.

Jiang, Y., Chang, S., and Wang, Z. (2021). Transgan: Two pure transformers can make one strong gan, and that can scale up. *Advances in Neural Information Processing Systems*, 34:14745–14758.

Johnson, J., Alahi, A., and Fei-Fei, L. (2016). Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer.

Jolicoeur-Martineau, A. (2019). The relativistic discriminator: a key element missing from standard GAN. In *International Conference on Learning Representations*.

Kallenberg, O. (2006). *Foundations of modern probability*. Springer Science & Business Media.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. (2018). Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*.

Karras, T., Aittala, M., Aila, T., and Laine, S. (2022). Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems*.

Karras, T., Aittala, M., Hellsten, J., Laine, S., Lehtinen, J., and Aila, T. (2020a). Training generative adversarial networks with limited data. *Advances in neural information processing systems*, 33:12104–12114.

Karras, T., Aittala, M., Laine, S., Härkönen, E., Hellsten, J., Lehtinen, J., and Aila, T. (2021). Alias-free generative adversarial networks. *Advances in Neural Information Processing Systems*, 34:852–863.

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

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

Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T. (2020b). Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119.

Kawar, B., Elad, M., Ermon, S., and Song, J. (2022). Denoising diffusion restoration models. In *Advances in Neural Information Processing Systems*.

Khayatkhoei, M., Singh, M. K., and Elgammal, A. (2018). Disconnected manifold learning for generative adversarial networks. In *Advances in Neural Information Processing Systems*, pages 7343–7353.

Kim, G., Kwon, T., and Ye, J. C. (2022). Diffusionclip: Text-guided diffusion models for robust image manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2426–2435.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. (2016). Improved variational inference with inverse autoregressive flow. *Advances in neural information processing systems*, 29.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. In *International Conference on Learning Representations*.

Kingma, D. P., Welling, M., et al. (2019). An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392.

Kodali, N., Abernethy, J., Hays, J., and Kira, Z. (2017). On convergence and stability of GANs. *arXiv.1705.07215*.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images. Technical report.

Krizhevsky, A., Sutskever, I., and Hinton, G. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C., Bottou, L., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.

Kynkäänniemi, T., Karras, T., Laine, S., Lehtinen, J., and Aila, T. (2019). Improved precision and recall metric for assessing generative models. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 3927–3936. Curran Associates, Inc.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324.

Ledoux, M. (1996). Isoperimetry and gaussian analysis. In *Lectures on probability theory and statistics*, pages 165–294. Springer.

Li, D., Ling, H., Kim, S. W., Kreis, K., Fidler, S., and Torralba, A. (2022). Bigdatasetgan: Synthesizing imagenet with pixel-wise annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21330–21340.

Liang, X., Gong, K., Shen, X., and Lin, L. (2019). Look into person: Joint body parsing & pose estimation network and a new benchmark. *IEEE transactions on pattern analysis and machine intelligence*, 41(4):871–885.

Lin, Z., Khetan, A., Fanti, G., and Oh, S. (2018). Pacgan: The power of two samples in generative adversarial networks. In *Advances in Neural Information Processing Systems*, pages 1498–1507.

Liu, H., Wan, Z., Huang, W., Song, Y., Han, X., Liao, J., Jiang, B., and Liu, W. (2021). Deflocnet: Deep image editing via flexible low-level controls. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10765–10774.

Liu, Z., Luo, P., Wang, X., and Tang, X. (2015). Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*.

Lorenz, D., Bereska, L., Milbich, T., and Ommer, B. (2019). Unsupervised part-based disentangling of object shape and appearance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10955–10964.

Lugmayr, A., Danelljan, M., Romero, A., Yu, F., Timofte, R., and Van Gool, L. (2022). Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11461–11471.

Luzi, L., Marrero, C. O., Wynar, N., Baraniuk, R. G., and Henry, M. J. (2023). Evaluating generative networks using gaussian mixtures of image features. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 279–288.

Ma, L., Sun, Q., Georgoulis, S., Van Gool, L., Schiele, B., and Fritz, M. (2018). Disentangled person image generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 99–108.

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

Meng, C., He, Y., Song, Y., Song, J., Wu, J., Zhu, J.-Y., and Ermon, S. (2022). SDEdit: Guided image synthesis and editing with stochastic differential equations. In *International Conference on Learning Representations*.

Mescheder, L., Geiger, A., and Nowozin, S. (2018). Which training methods for gans do actually converge? In *International Conference on Machine learning*, pages 3481–3490. PMLR.

Milman, E. and Neeman, J. (2022). The gaussian double-bubble and multi-bubble conjectures. *Annals of Mathematics*, 195(1):89–206.

Mirza, M. and Osindero, S. (2014). Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. (2018). Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*.

Murphy, K. P. (2023). *Probabilistic machine learning: Advanced topics*. MIT Press.

Naeem, M. F., Oh, S. J., Uh, Y., Choi, Y., and Yoo, J. (2020). Reliable fidelity and diversity metrics for generative models. In *International Conference on Machine Learning*, pages 7176–7185. PMLR.

Niemeyer, M. and Geiger, A. (2021). Giraffe: Representing scenes as compositional generative neural feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11453–11464.

Pandeva, T. and Schubert, M. (2019). Mmgan: Generative adversarial networks for multi-modal distributions. *arXiv:1911.06663*.

Papyan, V., Han, X., and Donoho, D. L. (2020). Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40):24652–24663.

Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Shazeer, N., Ku, A., and Tran, D. (2018). Image transformer. In *International Conference on Machine Learning*, pages 4055–4064. PMLR.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Patashnik, O., Wu, Z., Shechtman, E., Cohen-Or, D., and Lischinski, D. (2021). Styleclip: Text-driven manipulation of stylegan imagery. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2085–2094.

Peng, J., Liu, D., Xu, S., and Li, H. (2021). Generating diverse structure for image inpainting with hierarchical vq-vae. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10775–10784.

Petzka, H., Fischer, A., and Lukovnikov, D. (2018). On the regularization of Wasserstein GANs. In *International Conference on Learning Representations*.

Pons-Moll, G., Pujades, S., Hu, S., and Black, M. J. (2017). Clothcap: Seamless 4d clothing capture and retargeting. *ACM Transactions on Graphics (ToG)*, 36(4):1–15.

Pratelli, A. (2007). On the equality between monge's infimum and kantorovich's minimum in optimal mass transportation. In *Annales de l'Institut Henri Poincare (B) Probability and Statistics*, volume 43, pages 1–13. Elsevier.

Prykhodko, O., Johansson, S. V., Kotsias, P.-C., Arús-Pous, J., Bjerrum, E. J., Engkvist, O., and Chen, H. (2019). A de novo molecular generation method using latent vector based generative adversarial network. *Journal of Cheminformatics*, 11(1):1–13.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

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

Raj, A., Sangkloy, P., Chang, H., Lu, J., Ceylan, D., and Hays, J. (2018). Swapnet: Garment transfer in single view images. In *Proceedings of the European conference on computer vision (ECCV)*, pages 666–682.

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. (2021). Zero-shot text-to-image generation. *arXiv:2102.12092*.

Ranganath, R., Tran, D., and Blei, D. (2016). Hierarchical variational models. In *International conference on machine learning*, pages 324–333. PMLR.

Razavi, A., Van den Oord, A., and Vinyals, O. (2019). Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 32.

Richardson, E., Alaluf, Y., Patashnik, O., Nitzan, Y., Azar, Y., Shapiro, S., and Cohen-Or, D. (2021). Encoding in style: a stylegan encoder for image-to-image translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2287–2296.

Rifai, S., Mesnil, G., Vincent, P., Muller, X., Bengio, Y., Dauphin, Y., and Glorot, X. (2011). Higher order contractive auto-encoder. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 645–660. Springer.

Riquelme, C., Puigcerver, J., Mustafa, B., Neumann, M., Jenatton, R., Susano Pinto, A., Keysers, D., and Houlsby, N. (2021). Scaling vision with sparse mixture of experts. *Advances in Neural Information Processing Systems*, 34:8583–8595.

Robert, C. and Casella, G. (2013). *Monte Carlo statistical methods*. Springer Science & Business Media.

Rocco, I., Arandjelovic, R., and Sivic, J. (2017). Convolutional neural network architecture for geometric matching. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6148–6157.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer.

Roth, K., Lucchi, A., Nowozin, S., and Hofmann, T. (2017). Stabilizing training of generative adversarial networks through regularization. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 2018–2028. Curran Associates, Inc.

Rubin, D. B. (1988). Using the sir algorithm to simulate posterior distribution. *Bayesian statistics*, 3:395–402.

Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E., Ghasemipour, S. K. S., Gontijo-Lopes, R., Ayan, B. K., Salimans, T., et al. (2022). Photorealistic text-to-image diffusion models with deep language understanding. In *Advances in Neural Information Processing Systems*.

Sajjadi, M., Bachem, O., Lucic, M., Bousquet, O., and Gelly, S. (2018). Assessing generative models via precision and recall. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 5228–5237. Curran Associates, Inc.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training GANs. In Lee, D., Sugiyama, M., von Luxburg, U., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems 29*, pages 2234–2242. Curran Associates, Inc.

Salimans, T. and Ho, J. (2022). Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*.

Salmona, A., De Bortoli, V., Delon, J., and Desolneux, A. (2022). Can push-forward generative models fit multimodal distributions? In *Advances in Neural Information Processing Systems*.

Samangouei, P., Kabkab, M., and Chellappa, R. (2018). Defense-GAN: Protecting classifiers against adversarial attacks using generative models. In *International Conference on Learning Representations*.

Sauer, A., Chitta, K., Müller, J., and Geiger, A. (2021). Projected gans converge faster. *Advances in Neural Information Processing Systems*, 34:17480–17492.

Schechtman, G. (2012). Approximate gaussian isoperimetry for k sets. In *Geometric Aspects of Functional Analysis: Israel Seminar 2006–2010*, pages 373–379. Springer.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. (2017). Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*.

Shen, Y., Gu, J., Tang, X., and Zhou, B. (2020). Interpreting the latent space of gans for semantic face editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9243–9252.

Siarohin, A., Sangineto, E., Lathuilière, S., and Sebe, N. (2018). Deformable gans for pose-based human image generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3408–3416.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

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

Song, Y. and Ermon, S. (2019). Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32.

Song, Y., Kim, T., Nowozin, S., Ermon, S., and Kushman, N. (2018). Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. In *International Conference on Learning Representations*.

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

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2021). Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*.

Srivastava, A., Valkov, L., Russell, C., Gutmann, M. U., and Sutton, C. (2017). Veegan: Reducing mode collapse in gans using implicit variational learning. In *Advances in Neural Information Processing Systems*, pages 3308–3318.

Sudakov, V. N. and Tsirelson, B. S. (1978). Extremal properties of half-spaces for spherically invariant measures. *Journal of Mathematical Sciences*, pages 9–18.

Swaminathan, A. and Joachims, T. (2015a). Batch Learning from Logged Bandit Feedback through Counterfactual Risk Minimization. *Journal of Machine Learning Research*, 16(1):1731–1755.

Swaminathan, A. and Joachims, T. (2015b). The self-normalized estimator for counterfactual learning. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 3231–3239. Curran Associates, Inc.

Tanaka, A. (2019). Discriminator optimal transport. In *Advances in Neural Information Processing Systems*, pages 6813–6823.

Terjék, D. (2020). Adversarial lipschitz regularization. In *International Conference on Learning Representations*.

Theis, L., Van den Oord, A., and Bethge, M. (2016). A note on the evaluation of generative models. In *International Conference on Learning Representations*.

Tolstikhin, I., Bousquet, O., Gelly, S., and Schoelkopf, B. (2018). Wasserstein auto-encoders. In *International Conference on Learning Representations*.

Tolstikhin, I., Gelly, S., Bousquet, O., Simon-Gabriel, C.-J., and Schölkopf, B. (2017). Adagan: Boosting generative models. In *Advances in Neural Information Processing Systems*, pages 5424–5433.

Tomczak, J. M. (2022). *Deep generative modeling*. Springer.

Tov, O., Alaluf, Y., Nitzan, Y., Patashnik, O., and Cohen-Or, D. (2021). Designing an encoder for stylegan image manipulation. *ACM Transactions on Graphics (TOG)*, 40(4):1–14.

Turner, R., Hung, J., Frank, E., Saatchi, Y., and Yosinski, J. (2019). Metropolis-hastings generative adversarial networks. In *International Conference on Machine Learning*, pages 6345–6353.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. (2017). Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6924–6932.

Vahdat, A. and Kautz, J. (2020). Nvae: A deep hierarchical variational autoencoder. *Advances in neural information processing systems*, 33:19667–19679.

Van Den Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. (2016). Pixel recurrent neural networks. In *International conference on machine learning*, pages 1747–1756. PMLR.

Van den Oord, A., Kalchbrenner, N., Vinyals, O., Espeholt, L., Graves, A., and Kavukcuoglu, K. (2016). Conditional image generation with pixelcnn decoders. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 4797–4805.

Van Den Oord, A., Vinyals, O., et al. (2017). Neural discrete representation learning. *Advances in neural information processing systems*, 30.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Vincent, P. (2011). A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674.

Virmaux, A. and Scaman, K. (2018). Lipschitz regularity of deep neural networks: analysis and efficient estimation. In *Advances in Neural Information Processing Systems*, pages 3835–3844.

Voynov, A. and Babenko, A. (2020). Unsupervised discovery of interpretable directions in the gan latent space. In *International conference on machine learning*, pages 9786–9796. PMLR.

Wan, Z., Zhang, J., Chen, D., and Liao, J. (2021). High-fidelity pluralistic image completion with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4692–4701.

Wang, A. and Cho, K. (2019). Bert has a mouth, and it must speak: Bert as a markov random field language model. *arXiv:1902.04094*.

Wang, B., Zheng, H., Liang, X., Chen, Y., Lin, L., and Yang, M. (2018a). Toward characteristic-preserving image-based virtual try-on network. In *Proceedings of the European conference on computer vision (ECCV)*, pages 589–604.

Wang, X., Girshick, R., Gupta, A., and He, K. (2018b). Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803.

Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612.

Wei, X., Gong, B., Liu, Z., Lu, W., and Wang, L. (2018). Improving the improved training of wasserstein gans: A consistency term and its dual effect. *arXiv:1803.01541*.

Wu, Z., Lin, G., Tao, Q., and Cai, J. (2019). M2e-try on net: Fashion from model to everyone. In *Proceedings of the 27th ACM international conference on multimedia*, pages 293–301.

Wu, Z., Lischinski, D., and Shechtman, E. (2021). Stylespace analysis: Disentangled controls for stylegan image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12863–12872.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. *arXiv:1708.07747*.

Yang, H., Zhang, R., Guo, X., Liu, W., Zuo, W., and Luo, P. (2020a). Towards photo-realistic virtual try-on by adaptively generating-preserving image content. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7850–7859.

Yang, L., Zhang, D., and Karniadakis, G. E. (2020b). Physics-informed generative adversarial networks for stochastic differential equations. *SIAM Journal on Scientific Computing*, 42(1):A292–A317.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Yu, F., Seff, A., Zhang, Y., Song, S., Funkhouser, T., and Xiao, J. (2015). Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*.

Yu, J., Li, X., Koh, J. Y., Zhang, H., Pang, R., Qin, J., Ku, A., Xu, Y., Baldridge, J., and Wu, Y. (2021a). Vector-quantized image modeling with improved vqgan. *arXiv preprint arXiv:2110.04627*.

Yu, J., Xu, Y., Koh, J. Y., Luong, T., Baid, G., Wang, Z., Vasudevan, V., Ku, A., Yang, Y., Ayan, B. K., Hutchinson, B., Han, W., Parekh, Z., Li, X., Zhang, H., Baldridge, J., and Wu, Y. (2022). Scaling autoregressive models for content-rich text-to-image generation. *Transactions on Machine Learning Research*. Featured Certification.

Yu, L., Zhang, W., Wang, J., and Yu, Y. (2017). Seqgan: Sequence generative adversarial nets with policy gradient. In *Thirty-first AAAI conference on artificial intelligence*.

Yu, R., Wang, X., and Xie, X. (2019). Vtnfp: An image-based virtual try-on network with body and clothing feature preservation. In *The IEEE International Conference on Computer Vision (ICCV)*.

Yu, Y., Zhan, F., Wu, R., Pan, J., Cui, K., Lu, S., Ma, F., Xie, X., and Miao, C. (2021b). Diverse image inpainting with bidirectional and autoregressive transformers. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 69–78.

Zhan, F., Zhu, H., and Lu, S. (2019). Spatial fusion gan for image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3653–3662.

Zhang, D., Mishra, S., Brynjolfsson, E., Etchemendy, J., Ganguli, D., Grosz, B., Lyons, T., Manyika, J., Niebles, J. C., Sellitto, M., et al. (2021a). The ai index 2021 annual report. *arXiv preprint arXiv:2103.06312*.

Zhang, H., Goodfellow, I., Metaxas, D., and Odena, A. (2019). Self-attention generative adversarial networks. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7354–7363.

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. (2018). The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595.

Zhang, Z., Ma, J., Zhou, C., Men, R., Li, Z., Ding, M., Tang, J., Zhou, J., and Yang, H. (2021b). M6-ufc: Unifying multi-modal controls for conditional image synthesis. *arXiv preprint arXiv:2105.14211*.

Zhao, H., Gallo, O., Frosio, I., and Kautz, J. (2016). Loss functions for image restoration with neural networks. *IEEE Transactions on computational imaging*, 3(1):47–57.

Zhao, S., Cui, J., Sheng, Y., Dong, Y., Liang, X., Eric, I., Chang, C., and Xu, Y. (2020). Large scale image completion via co-modulated generative adversarial networks. In *International Conference on Learning Representations*.

Zhong, P., Mo, Y., Xiao, C., Chen, P., and Zheng, C. (2019). Rethinking generative mode coverage: A pointwise guaranteed approach. In *Advances in Neural Information Processing Systems*, pages 2086–2097.

Zhou, Z., Liang, J., Song, Y., Yu, L., Wang, H., Zhang, W., Yu, Y., and Zhang, Z. (2019). Lipschitz generative adversarial nets. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 7584–7593. PMLR.

Zhu, J., Shen, Y., Zhao, D., and Zhou, B. (2020). In-domain gan inversion for real image editing. In *European conference on computer vision*, pages 592–608. Springer.

Zhu, J.-Y., Krähenbühl, P., Shechtman, E., and Efros, A. A. (2016). Generative visual manipulation on the natural image manifold. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part V 14*, pages 597–613. Springer.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232.

Zoran, D., Kabra, R., Lerchner, A., and Rezende, D. J. (2021). Parts: Unsupervised segmentation with slots, attention and independence maximization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10439–10447.

Appendix A

A.1 Technical results

A.1.1 Highlighting drawbacks of the Precision/Recall metric

Lemma A.1.1. *Assume that the modeled distribution μ_θ slightly collapses on a specific data point, i.e. there exists $x \in E, \mu_\theta(x) > 0$. Assume also that $\mu_\star$ is a continuous probability measure and that μ_θ has a recall $\beta = 1$. Then the precision must be such that $\alpha = 0$.*

Proof. Using Definition 3.2.1, we have that there exists μ such that

$$\mu_\star = \alpha\mu + (1-\alpha)\nu_{\mu_\star} \quad \text{and} \quad \mu_\theta = \mu.$$

Thus, $0 = \mu_\star(x) \geqslant \alpha\mu(x) = \alpha\mu_\theta(x)$. Which implies that $\alpha = 0$. □

A.1.2 Proof of Theorem 3.2.1

The proof of Theorem 3.2.1 relies on theoretical results from non-parametric estimation of the supports of probability distribution studied by Devroye and Wise (1980).

For the following proofs, we will require the following notation: let φ be a strictly monotonous function be such that $\lim_{n\to\infty} \frac{\varphi(n)}{n} = 0$ and $\lim_{n\to\infty} \frac{\varphi(n)}{\log(n)} = \infty$. We note $B(x,r) \subseteq E$, the open ball centered in x and of radius r. For a given probability distribution μ, S_μ refers to its support. We recall that for any x in a dataset D, $x_{(k)}$ denotes its k nearest neighbor in D. Finally, for a given probability distribution μ and a dataset D_μ sampled from μ^n, we note $R_{\min}$ and $R_{\max}$ the following:

$$R_{\min} = \min_{x\in E} \|x - x_{(\varphi(n))}\|, \quad R_{\max} = \max_{x\in E} \|x - x_{(\varphi(n))}\|. \tag{A.1.1}$$

In the following lemma, we show asymptotic behaviours for both $R_{\min}$ and $R_{\max}$.

Lemma A.1.2. *Let μ be a probability distribution associated with a uniformly continuous probability density function f_μ. Assume that there exists constants $a_1 > 0, a_2 > 0$ such that for all $x \in E$, we have $a_1 < f_\mu(x) \leqslant a_2$. Then,*

$$R_{min} \underset{n\to\infty}{\longrightarrow} 0 \ a.s. \quad and \quad R^d_{min} \underset{n\to\infty}{\longrightarrow} \infty \ a.s.$$
$$R_{max} \underset{n\to\infty}{\longrightarrow} 0 \ a.s. \quad and \quad R^d_{max} \underset{n\to\infty}{\longrightarrow} \infty \ a.s.$$

Proof. We will only prove that $R_{\max} \underset{n\to\infty}{\longrightarrow} 0$ a.s. and and $R^d_{\min} \underset{n\to\infty}{\longrightarrow} \infty$ a.s. as the rest follows.

The result is based on a nearest neighbor result from Biau and Devroye (2015). Considering the $\varphi(n)$ nearest neighbor density estimate $f_n^{\varphi(n)}$ based on a finite sample dataset D_μ, Theorem 4.2 states that if f_μ is uniformly continuous then:

$$\sup_{x\in E} \|f_n^{\varphi(n)}(x) - f_\mu(x)\| \to 0.$$

where $f_n^{\varphi(n)}(x) = \frac{\varphi(n)}{nV_d\|x-x_{\varphi(n)}\|^d}$ with V_d being the volume of the unit ball in $\mathbb{R}^d$.

Let $\varepsilon > 0$ such that $\varepsilon < a_1/2$. There exists $N \in \mathbb{N}$ such that for all $n \geqslant N$, we have, almost surely, for all $x \in E$:

$$a_1 - \varepsilon \leqslant f_n^{\varphi(n)}(x) \leqslant a_2 + \varepsilon$$
$$a_1 - \varepsilon \leqslant \frac{\varphi(n)}{nV_d\|x - x_{\varphi(n)}\|^d} \leqslant a_2 + \varepsilon$$

Consequently, for all $n \geqslant N$, for all $x \in E$ almost surely:

$$\|x - x_{\varphi(n)}\| \leqslant \left(\frac{\varphi(n)}{nV_d(a_1-\varepsilon)}\right)^{1/d}$$
$$\text{Thus,} \quad \sup_{x\in E} \|x - x_{\varphi(n)}\| \to 0 \quad \text{a.s..}$$

Also, almost surely

$$n\|x - x_{\varphi(n)}\|^d \geqslant \frac{\varphi(n)}{V_d(a_2+\varepsilon)}$$
$$\text{Thus,} \quad \inf_{x\in E} \|x - x_{\varphi(n)}\| \to \infty \quad \text{a.s..}$$

□

Lemma A.1.3. *Let μ, ν be two probability distributions associated with uniformly continuous probability density functions f_μ and f_ν. Assume that there exists constants $a_1 > 0, a_2 > 0$ such*

that for all $x \in E$, we have $a_1 < f_\mu(x) \leqslant a_2$ and $a_1 < f_\nu \leqslant a_2$. Also, let D_μ, D_ν be datasets sampled from ν^n, μ^n. If μ is an estimator for ν, then

$$(i) \text{ for all } x \in D_\mu,\ \alpha^n_{\varphi(n)}(x) \underset{n\to\infty}{\to} \mathbb{1}_{\text{supp}(\nu)}(x) \quad \text{in proba.}$$
$$(ii) \text{ for all } y \in D_\nu,\ \beta^n_{\varphi(n)}(y) \underset{n\to\infty}{\to} \mathbb{1}_{\text{supp}(\mu)}(x) \quad \text{in proba.}$$

Proof. We will only show the result for (i), since a similar proof holds for (ii).

Thus, we want to show that

$$\text{for all } x \in D_\mu,\ \alpha^n_{\varphi(n)}(x) \underset{n\to\infty}{\to} \mathbb{1}_{supp(\nu)}(x) \quad \text{a. s.}$$

First, let's assume that $x \notin S_\nu$. Biau and Devroye (2015, Lemma 2.2) have shown that

$$\lim_{n\to\infty} \|x_{(\varphi(n))} - x\| = \inf\{\|x-y\| \mid y \in S_\nu\} \quad \text{a.s.}$$

As S_ν is a closed set - e.g. (Kallenberg, 2006) - we have

$$\lim_{n\to\infty} \|x - x_{(\varphi(n))}\| > 0 \quad \text{a.s.}$$

and

$$\text{for all } y \in D_\nu, \lim_{n\to\infty} \|y - y_{(\varphi(n))}\| = 0 \quad \text{a.s.}$$

Thus, $\lim_{n\to\infty} \alpha^n_{\varphi(n)}(x) = 0$ a.s..

Now, let's assume that $x \in S_\nu$. Using Definition 3.2.2, the precision of a given data point x can be rewritten as follows:

$$\alpha^n_{\varphi(n)}(x) = 1 \iff \exists y \in D_\nu, x \in B(y, \|y - y_{(\varphi(n))}\|)$$

Using notation from (A.1.1), we note

$$R_{\min} = \min_{y\in} \|y - y_{(\varphi(n))}\|, \quad R_{\max} = \max_{y\in E} \|y - y_{(\varphi(n))}\|.$$

It is clear that :

$$\bigcup_{y\in D_\nu} B(y, R_{\min}) \subseteq S^n_\nu \subseteq \bigcup_{y\in D_\nu} B(y, R_{\max}), \tag{A.1.2}$$

where $S^n_\nu = \bigcup_{y\in D_\nu} B(y, \|y - y_{(\varphi(n))}\|))$.

Besides, combining Lemma A.1.2 with Devroye and Wise (1980, Theorem 1), we have that:

$$\nu(S_\nu \Delta \bigcup_{y \in D_\nu} B(y, R_{\min})) \underset{n \to 0}{\longrightarrow} 0 \quad \text{in proba.}$$
$$\nu(S_\nu \Delta \bigcup_{y \in D_\nu} B(y, R_{\max})) \underset{n \to 0}{\longrightarrow} 0 \quad \text{in proba.}$$

where Δ here refers to the symmetric difference.

Thus, using (A.1.2), it is now clear that, $\mu(S_\nu \Delta S_\nu^n) \to 0$ in probability. Finally, given $x \in S_\mu$, we have $\mu(x \in S_\nu^n) = \nu(\alpha_{\varphi(n)}^n(x) = 1) \to 1$ in probability. □

We can now finish the proof for Theorem 3.2.1. Recall that $\bar{\alpha} = \mu(S_\nu)$ and similarly, $\bar{\beta} = \nu(S_\mu)$.

Proof. We have that

$$|\alpha_{\varphi(n)}^n - \bar{\alpha}| = |\frac{1}{n} \sum_{x_i \in D_\mu} \alpha_{\varphi(n)}^n(x_i) - \int_E \mathbb{1}_{x \in S_\nu} \mu(\mathrm{d}x)|$$

Then,

$$\begin{aligned} |\alpha_{\varphi(n)}^n - \bar{\alpha}| &= |\frac{1}{n} \sum_{x_i \in D_\mu} (\alpha_{\varphi(n)}^n(x_i) - \mathbb{1}_{x_i \in S_\nu}) \\ &\quad + (\frac{1}{n} \sum_{x_i \in D_\mu} \mathbb{1}_{x_i \in S_\nu} - \int_E \mathbb{1}_{x \in S_\nu} \mu(\mathrm{d}x))| \\ &= |\mathbb{E}_{x_i \sim \mu_n}(\alpha_{\varphi(n)}^n(x_i) - \mathbb{1}_{x_i \in S_\nu}) \qquad \text{(A.1.3)} \\ &\quad + (\mathbb{E}_{\mu_n} \mathbb{1}_{S_\nu} - \mathbb{E}_\mu \mathbb{1}_{S_\nu})| \qquad \text{(A.1.4)} \end{aligned}$$

where μ_n is the empirical distribution of μ. As μ_n converges weakly to μ almost surely (e.g. Dudley (2004, Theorem 11.4.1)) and since $\mathbb{1}_{x \in S_\nu}$ is bounded, we can bound (A.1.4) as follows:

$$\lim_{n \to \infty} \quad \mathbb{E}_{x \sim \mu_n} \mathbb{1}_{x \in \mathrm{supp}(\mu)} - \mathbb{E}_{x \sim \mu} \mathbb{1}_{x \in \mathrm{supp}(\mu)} = 0 \quad \text{a. s.}$$

Now, to bound (A.1.3), we use the fact that for any $x \in D_\mu$, the random variable $\alpha_{\varphi(n)}^n(x)$ converges to $\mathbb{1}_{x \in S_\nu}$ in probability (Lemma A.1.3) and that for all $x \in D_\mu$, both $\alpha_{\varphi(n)}^n(x) \leqslant 1$ and

$\mathbb{1}_{x\in S_v} \leqslant 1$. Consequently, using results from the weak law for triangular arrays, we have that

$$\lim_{n\to\infty} \quad \frac{1}{n} \sum_{x_i \in D_\mu} (\alpha^n_{\varphi(n)}(x_i) - \mathbb{1}_{x_i\in S_v}) = 0 \quad \text{in proba.}$$

Finally,

$$|\alpha^n_{\varphi(n)} - \bar{\alpha}| \underset{n\to\infty}{\to} 0 \quad \text{in proba.},$$

which proves the result. The same proof works for $\lim_{k\to\infty} \beta^n_k = \bar{\beta}$. □

A.1.3 Proof of Theorem 3.2.2

This proof is based on the Gaussian isoperimetric inequality historically shown by Borell (1975); Sudakov and Tsirelson (1978).

Proof. Let $\mu_\star$ be a distribution defined on E laying on two disconnected manifolds M_1 and M_2 such that $\mu_\star(M_1) = \mu_\star(M_2) = \frac{1}{2}$ and $d(M_1, M_2) = D$. Note that for any subsets $A \subseteq E$ and $B \subseteq E$, $d(A,B) := \inf_{(x,y)\in A\times B} \|x-y\|$.

Let $G_\theta^{-1}(M_1)$ (respectively $G_\theta^{-1}(M_2)$ be the subset in $\mathbb{R}^d$ be the pre-images of M_1 (respectively M_2).

Consequently, we have for all $k \in [1,n]$

$$\gamma(G_\theta^{-1}(M_1)) = \mu_\theta(M_1) = \gamma(G_\theta^{-1}(M_2)) \geqslant \frac{\bar{\alpha}}{2}$$

We consider $(G_\theta^{-1}(M_1))^\varepsilon$ (respectively $(G_\theta^{-1}(M_2))^\varepsilon$) the ε enlargement of $G_\theta^{-1}(M_1)$ (respectively $G_\theta^{-1}(M_2)$ where $\varepsilon = \frac{D}{2L}$. We know that $(G_\theta^{-1}(M_1))^\varepsilon \bigcap (G_\theta^{-1}(M_2))^\varepsilon = \emptyset$.

Thus, we have that:

$$\gamma\big((G_\theta^{-1}(M_1))^\varepsilon\big) + \gamma\big((G_\theta^{-1}(M_2))^\varepsilon\big) \leqslant 1$$

Besides, by denoting Φ the function defined for any $t \in \mathbb{R}$ by $\Phi(t) = \int_{-\infty}^t \frac{\exp(-t^2/2)}{\sqrt{2\pi}} ds$, we have

$$\gamma\big((G_\theta^{-1}(M_1))^\varepsilon\big)+\gamma\big((G_\theta^{-1}(M_2))^\varepsilon\big)\geqslant 2\Phi\big(\Phi^{-1}(\frac{\alpha}{2})+\varepsilon\big)$$

(using Theorem 1.3 from Ledoux (1996))

$$\geqslant \alpha+\frac{2\varepsilon}{\sqrt{2\pi}}e^{-\Phi^{-1}(\frac{\alpha}{2})^2/2}$$

(since $\Phi^{-1}(\frac{\alpha}{2})+\varepsilon<0$ and Φ convex on $]-\infty,0]$)

Thus, we have that

$$\alpha+\frac{2\varepsilon}{\sqrt{2\pi}}e^{-\Phi^{-1}(\frac{\alpha}{2})^2/2}\leqslant 1$$

Thus, by noting

$$\alpha^\star=\sup\{\alpha\in[0,1]\mid \alpha+\frac{2\varepsilon}{\sqrt{2\pi}}e^{\frac{-\Phi^{-1}(\frac{\alpha}{2})^2}{2}}\leqslant 1\},$$

we have our result.

For $\alpha\geqslant 3/4$. By noting $\alpha=1-x$, we have

$$\Phi^{-1}(\frac{\alpha}{2})=\frac{\sqrt{2\pi}x}{2}+O(x^3)$$

$$\text{And, } e^{\frac{-\Phi^{-1}(\frac{\alpha}{2})^2}{2}}=e^{\frac{-\pi x^2}{4}}+O(e^{-x^4})$$

$$\text{Thus, } 1-x+\frac{2\varepsilon}{\sqrt{2\pi}}e^{\frac{-\pi x^2}{4}}+O(e^{-x^4})\leqslant 1$$

$$\iff x\geqslant\frac{2\varepsilon}{\sqrt{2\pi}}e^{\frac{-\pi x^2}{4}}+O(e^{-x^4})$$

$$\implies x\geqslant\sqrt{\frac{2}{\pi}}W(\varepsilon^2)$$

where W is the product log function. Thus, $\alpha\leqslant 1-\sqrt{\frac{2}{\pi}}W(\varepsilon^2)$. □

As an example, in the case where $\varepsilon=1$, we have that $W(1)\approx 0.5671$, $x>0.4525$ and $\alpha<0.5475$.

A.1.4 Proof of Theorem 3.2.3

A.1.4.1 Equitable setting

This result is a consequence of Theorem A.1.1 that we will assume true in this section.

We consider that the unknown true distribution $\mu_\star$ lays on M disjoint manifolds of equal measure. As specified in Section 3.2.3, the latent distribution γ is a multivariate Gaussian defined on $\mathbb{R}^d$. For each $k \in [1,M]$, we consider in the latent space, the pre-images A_k.

It is clear that $A_1,\ldots,A_M$ are pairwise disjoint Borel subsets of $\mathbb{R}^d$. We denote $\bar{M}$, the number of classes covered by the estimator μ_θ, such that for all $i \in [1,\bar{M}]$, we have $\gamma(A_i) > 0$. We know that $\bar{M} \geqslant M\bar{\beta} > 1$.

For each $i \in [1,\bar{M}]$, we denote A_i^ε, the ε-enlargement of A_i. For any pair (i,j) it is clear that $A_i^\varepsilon \bigcap A_j^\varepsilon = 0$ where $\varepsilon = \frac{D}{2L}$ (D being the minimum distance between two sub-manifolds and L being the Lipschitz constant of the generator).

As assumed, we know that $A_i^\varepsilon, i \in [1,\bar{M}]$ partition the latent space in equal measure, consequently, we assume that

$$\sum_{i=1}^{n} \gamma(A_i^\varepsilon) = 1 \quad \text{and} \quad \gamma(A_1) = \ldots = \gamma(A_{\bar{M}}) = 1/\bar{M} \tag{A.1.5}$$

Thus, we have that

$$\bar{\alpha} = \sum_{i=1}^{\bar{M}} \gamma(A_i^\varepsilon) = 1 - \gamma(\Delta^{-\varepsilon}(A_1^\varepsilon,\ldots,A_{\bar{M}}^\varepsilon))$$

Using Theorem A.1.1, we have

$$\gamma(\Delta^{-\varepsilon}(A_1^\varepsilon,\ldots,A_n^\varepsilon)) \geqslant 1 - \frac{1+x^2}{x^2} e^{-\frac{1}{2}\varepsilon^2} e^{-\varepsilon x}$$

$$\text{Thus, } \bar{\alpha} \leqslant \frac{1+y^2}{y^2} e^{-\frac{1}{2}\varepsilon^2} e^{-\varepsilon y}$$

where $y = \Phi^{-1}\left(1 - \max_{k\in[\bar{M}]} \gamma(A_k^\varepsilon)\right) = \Phi^{-1}(\frac{\bar{M}-1}{\bar{M}})$ and $\Phi(t) = \int_{-\infty}^{t} \frac{\exp(-t^2/2)}{\sqrt{2\pi}} ds$.

Knowing that $\bar{M} \geqslant \bar{\beta}M$ we have that

$$\Phi^{-1}(1 - \frac{1}{\bar{M}}) \geqslant \Phi^{-1}(1 - \frac{1}{\bar{\beta}M})$$

We conclude by saying that the function $x \mapsto \frac{1+x^2}{x^2} e^{-\varepsilon x}$ is decreasing for $x > 0$. Thus,

$$\bar{\alpha} \leqslant \frac{1+y^2}{y^2} e^{-\frac{1}{2}\varepsilon^2} e^{-\varepsilon y} \tag{A.1.6}$$

where $y = \Phi^{-1}(1 - \frac{1}{\bar{\beta}M})$ and $\Phi(t) = \int_{-\infty}^{t} \frac{\exp(-t^2/2)}{\sqrt{2\pi}} ds$.

For further analysis, when $\bar{M} \to \infty$, refer to subsection A.1.5 and note using the result in (A.1.14) that one obtains the desired upper-bound on $\bar{\alpha}$

$$\bar{\alpha} \overset{\bar{M}\to\infty}{\leqslant} e^{-\frac{1}{2}\varepsilon^2} e^{-\varepsilon\sqrt{2\log(\bar{M})}}$$

A.1.4.2 More general setting

As done previously, we denote $\bar{M}$, the number of classes covered by the estimator μ_θ, such that for all $i \in [1,\bar{M}]$, we have $\gamma(A_i) > 0$. We still assume that $\bar{M} > 1$. However, we now relax the previous assumption made in (A.1.5) and assume the milder assumption that there exists $w_1,\ldots,w_M \in [0,1]^M$ such that for all $m \in [1,M], \gamma(A_m^\varepsilon) = w_m$, $\sum_m w_m \leqslant 1$ and $\max\limits_{i\in[1,M]} w_m = w^{\max} < 1$.

Consider, $A^{\complement} = \left(\bigcup_{i=1}^{\bar{M}} A_i^\varepsilon\right)^{\complement}$ and denote $w^c = \gamma(A^{\complement}) \leqslant 1 - \bar{\alpha}$. Consequently, we have

$$\begin{aligned}
&\sum_{i=1}^{n} \gamma(A_i^\varepsilon) + \gamma(A^{\complement}) = 1 \\
&\gamma(\Delta^{-\varepsilon}(A_1^\varepsilon,\ldots,A_M^\varepsilon,A^{\complement})) + \sum_{i=1}^{M} \gamma(A_i^\varepsilon) = 1 - \gamma(A^{\complement}) \\
&\bar{\alpha} = 1 - w^{\complement} - \gamma(\Delta^{-\varepsilon}(A_1^\varepsilon,\ldots,A_M^\varepsilon,A^{\complement}))
\end{aligned}$$

In this setting, it is clear that $A_1,\ldots,A_{\bar{M}},A^{\complement}$ is a a partition of $\mathbb{R}^d$ under the measure γ. Using, result from Theorem A.1.1, we have

$$\gamma(\Delta^{-\varepsilon}(A_1^\varepsilon,\ldots,A_M^\varepsilon,A^{\complement})) \geqslant 1 - \frac{1+x^2}{x^2} e^{-\frac{1}{2}\varepsilon^2} e^{-\varepsilon x}$$

where $x = \Phi^{-1}\left(1 - \max(w^{\complement}, w^{\max})\right)$ and $\Phi(t) = \int_{-\infty}^{t} \frac{\exp(-t^2/2)}{\sqrt{2\pi}} ds$.

Finally, we have that

$$\bar{\alpha} \leqslant \frac{1+x^2}{x^2} e^{-\frac{1}{2}\varepsilon^2} e^{-\varepsilon x} - w^{\complement} \tag{A.1.7}$$

In the case where $\gamma(A^{\complement}) = 0$, we find a result similar to (A.1.6).

A.1.5 Lower-bounding boundaries of partitions in a Gaussian space

Notations and preliminaries Given $\varepsilon \geq 0$ and a subset A of euclidean space $\mathbb{R}^d = (\mathbb{R}^d, \|\cdot - \cdot\|)$, let $A^\varepsilon := \{z \in \mathbb{R}^d \mid \mathsf{dist}(\mathsf{z},\mathsf{A}) \leq \varepsilon\}$ be its ε-enlargement, where $\mathsf{dist}(\mathsf{z},\mathsf{A}) := \inf_{\mathsf{z}'\in\mathsf{A}} \|\mathsf{z}' - \mathsf{z}\|_2$ is the distance of the point $z \in \mathbb{R}^d$ from A. Let γ be the standard Gaussian distribution in $\mathbb{R}^d$

and let $A_1,\dots,A_K$ be $K \geq 2$ pairwise disjoint Borel subsets of $\mathbb{R}^d$ whose union has unit (i.e full) Gaussian measure $\sum_{k=1}^K w_k = 1$, where $w_k := \gamma(A_k)$. Such a collection $\{A_1,\dots,A_K\}$ will be called an $(w_1,\dots,w_K)$-partition of standard d-dimensional Gaussian space $(\mathbb{R}^d,\gamma)$.

For each $k \in [\![K]\!]$, define the compliment $A_{-k} := \cup_{k'\neq k} A_{k'}$, and let $\partial^{-\varepsilon} A_k := \{z \in A_k \mid \mathsf{dist}(\mathsf{z},\mathsf{A_{-k}}) \leq \varepsilon\}$ be the *inner ε-boundary* of A_k, i.e the points of A_k which are within distance ε of some other $A_{k'}$. For every $(k,k') \in [\![K]\!]^2$ with $k' \neq k$, it is an easy exercise to show that

$$\begin{aligned} \partial^{-\varepsilon} A_k \cap \partial^{-\varepsilon} A_{k'} &= \emptyset \\ \partial^{-\varepsilon} A_k \cap A_{-k} &= \emptyset \\ A_{-k}^{\varepsilon} &= \partial^{-\varepsilon} A_k \cup A_{-k} \end{aligned} \tag{A.1.8}$$

Now, let $\Delta^{-\varepsilon}(A_1,\dots,A_K) := \cup_{k=1}^K \partial^{-\varepsilon} A_k$ be the union of all the inner ε-boundaries. This is $\Delta^{-\varepsilon}(A_1,\dots,A_K)$ the set of points of $\cup_{k=1}^K A_k$ which are on the boundary between some two distinct A_k and $A_{k'}$. We want to find a lower bound in the measure $\gamma(\Delta^{-\varepsilon}(A_1,\dots,A_K))$.

Theorem A.1.1. *Given $K \geq 4$ and $w_1,\dots,w_K \in (0,1/4]$ such that $\sum_{k=1}^K w_k = 1$, we have the bound:*

$$\inf_{A_1,\dots,A_K} \gamma(\Delta^{-\varepsilon}(A_1,\dots,A_K)) \geq 1 - \frac{1+x^2}{x^2} e^{-\frac{1}{2}\varepsilon^2} e^{-\varepsilon x}$$

where the infinimum is taken over all $(w_1,\dots,w_k)$-partitions of standard Gaussian space $(\mathbb{R}^d,\gamma)$, and $x := \Phi^{-1}\left(1 - \max_{k\in[\![M]\!]} w_k\right)$.

Proof. By (A.1.8), we have the formula

$$\gamma(\Delta^{-\varepsilon}(A_1,\dots,A_K)) = \sum_{k=1}^K \gamma(\partial^{-\varepsilon} A_k) \tag{A.1.9}$$

$$= \sum_{k=1}^K \gamma(A_{-k}^{\varepsilon}) - \gamma(A_{-k}). \tag{A.1.10}$$

Let $w_{-k} := \gamma(A_{-k}) = 1 - w_k$, and assume $w_{-k} \geq 3/4$, i.e $w_k \leq 1/4$, for all $k \in [\![K]\!]$.

For example, this condition holds in the equitable scenario where $w_k = 1/K$ for all k.

Now, by standard *Gaussian Isoperimetric Inequality* (see Boucheron et al. (2013) for example), one has

$$\begin{aligned} \gamma(A_{-k}^{\varepsilon}) &\geq \Phi(\Phi^{-1}(\gamma(A_{-k}) + \varepsilon) \\ &= \Phi(\Phi^{-1}(1 - w_k) + \varepsilon). \end{aligned} \tag{A.1.11}$$

Using the bound $\frac{x}{1+x^2}\varphi(x) < 1-\Phi(x) < \frac{1}{x}\varphi(x)\ \forall x>0$ where φ is the density of the standard Gaussian law. We can further find that

$$\begin{aligned}\Phi(\Phi^{-1}(1-w_k)+\varepsilon) &\geq 1-w_k\frac{1+\Phi^{-1}(1-w_k)^2}{\Phi^{-1}(1-w_k)^2}\times\\ &\quad e^{-\frac{1}{2}\varepsilon^2}e^{-\varepsilon\Phi^{-1}(1-w_k)}\\ &\geq 1-w_k\frac{1+x^2}{x^2}e^{-\frac{1}{2}\varepsilon^2}e^{-\varepsilon x} > 0 \qquad \text{(A.1.12)}\\ &\text{(since the function } x\mapsto \frac{1+x^2}{x^2}e^{-\varepsilon x} \text{ is decreasing for } x>0)\end{aligned}$$

where $x := \min_{k\in[\![K]\!]}\Phi^{-1}(1-w_k) = \Phi^{-1}\left(1-\max_{k\in[\![K]\!]}w_k\right) \geq \Phi^{-1}(3/4) > 0.67$. Combining (A.1.9), (A.1.11), and (A.1.12) yields the following

$$\begin{aligned}\gamma(\Delta^{-\varepsilon}(A_1,\ldots,A_K)) &\geq \sum_{k=1}^{K}\Big(1-w_k\frac{1+x^2}{x^2}e^{-\frac{1}{2}\varepsilon^2}e^{-\varepsilon x}\\ &\quad -(1-w_k)\Big)\\ &= \sum_{k=1}^{K}\left(1-\frac{1+x^2}{x^2}e^{-\frac{1}{2}\varepsilon^2}e^{-\varepsilon x}\right)w_k\\ &= 1-\frac{1+x^2}{x^2}e^{-\frac{1}{2}\varepsilon^2}e^{-\varepsilon x},\end{aligned}$$

Asymptotic analysis In the limit, it is easy to check that in the case where $\max_{k\in[\![K]\!]}w_k \longrightarrow 0$, we have that $x\longrightarrow\infty$. In this setting, we thus have $\frac{1+x^2}{x^2}\longrightarrow 1$ and can now derive the following bound:

$$\inf_{A_1,\ldots,A_K}\gamma(\Delta^{-\varepsilon}(A_1,\ldots,A_K)) \overset{\max_{k\in[\![K]\!]}w_k\to 0}{\longrightarrow} 1-e^{-\frac{1}{2}\varepsilon^2}e^{-\varepsilon x}.$$

Equitable scenario In the equitable scenario where $w_k = 1/K$ for all k, we have

$$\inf_{A_1,\ldots,A_K}\gamma(\Delta^{-\varepsilon}(A_1,\ldots,A_K)) \geqslant 1-\frac{1+x^2}{x^2}e^{-\frac{1}{2}\varepsilon^2}e^{-\varepsilon x}$$

where $x=\Phi^{-1}(1-1/K)$. When $K\geq 8$ we have:

$$\Phi^{-1}(1-1/K) \geqslant \sqrt{2\log\left(\frac{K\,(q(K)^2-1)}{\sqrt{2\pi}q(K)^3}\right)} \qquad \text{(A.1.13)}$$

where $q(K)=\sqrt{2\log(\sqrt{2\pi}K)}$.

Consequently, we have when $K \to \infty$, the following behavior:

$$\gamma(\Delta^{-\varepsilon}(A_1,\ldots,A_K)) \overset{K\to\infty}{\leqslant} 1 - e^{-\frac{1}{2}\varepsilon^2} e^{-\varepsilon\sqrt{2\log(K)}} \tag{A.1.14}$$

□

Proof of the inequality (A.1.13). Set $p := 1/K$. First, for any $x > 0$, we have the following upper:

$$\int_x^\infty e^{-y^2/2}dy = \int_x^\infty \frac{y}{y} e^{-y^2/2}dy \leq \frac{1}{x}\int_x^\infty y e^{-y^2/2}dy = \frac{e^{-x^2/2}}{x}.$$

For a lower bound:

$$\int_x^\infty e^{-y^2/2}dy \quad = \int_x^\infty \tfrac{y}{y} e^{-y^2/2}dy = \tfrac{e^{-x^2/2}}{x} - \int_x^\infty \tfrac{1}{y^2} e^{-y^2/2}dy$$

and

$$\int_x^\infty \frac{1}{y^2} e^{-y^2/2}dy = \int_x^\infty \frac{y}{y^3} e^{-y^2/2}dy \leq \frac{e^{-x^2/2}}{x^3}$$

and combining these gives

$$\int_x^\infty e^{-y^2/2}dy \geq \left(\frac{1}{x} - \frac{1}{x^3}\right) e^{-x^2/2}.$$

Thus

$$\frac{1}{\sqrt{2\pi}}\left(\frac{1}{x} - \frac{1}{x^3}\right) e^{-x^2/2} \leq 1 - \Phi(x) \leq \frac{1}{\sqrt{2\pi}}\frac{1}{x} e^{-x^2/2},$$

from where

$$\frac{1}{\sqrt{2\pi}}\left(\frac{1}{\Phi^{-1}(1-p)} - \frac{1}{\Phi^{-1}(1-p)^3}\right) e^{-\Phi^{-1}(1-p)^2/2} \tag{A.1.15}$$

$$\leq p \leq \frac{1}{\sqrt{2\pi}}\frac{1}{\Phi^{-1}(1-p)} e^{-\Phi^{-1}(1-p)^2/2} \tag{A.1.16}$$

Using (A.1.16), when $\Phi^{-1}(1-p) \geq 1$ (that is $p \leqslant 0.15$ or equivalently $K \geq 8$), we have the following upper bound $\Phi^{-1}(1-p) \leqslant q(p)$ where $q(p) := \sqrt{2\log(\sqrt{2\pi}/p)}$. Then, injecting $q(p)$ in (A.1.15):

$$\frac{1}{\sqrt{2\pi}}\left(\frac{1}{q(p)} - \frac{1}{q(p)^3}\right) e^{-\Phi^{-1}(1-p)^2/2} \leq p.$$

Now when $q(p) \geq 1$ you have:

$$e^{-\Phi^{-1}(1-p)^2/2} \leq \frac{\sqrt{2\pi}pq(p)^3}{q(p)^2-1}$$

and

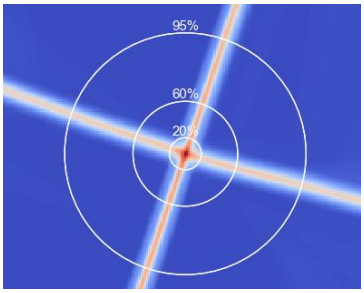
$$\Phi^{-1}(1-p) \geq \sqrt{2\log\left(\frac{q(p)^2-1}{\sqrt{2\pi}pq(p)^3}\right)}.$$

There is one additional requirement on p which is simply that the argument of the log should be ≥ 1 i.e. $q(p)^2 - 1 \geq \sqrt{2\pi}pq(p)^3$, which is true as soon as $K \geq 8$. □

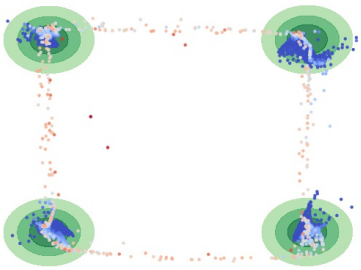
A.2 Complementary experiments

A.2.1 Visualization of Theorem 3.2.3

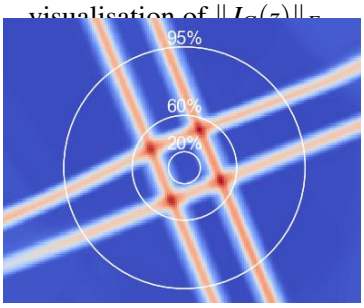
To further understand and illustrate Theorem 3.2.3, we propose in Figure A.1, an interesting visualization where we plot the manifold learned by a WGANs architecture and its corresponding latent space configuration. As expected, we observe that when the number of distinct modes increase, the number of data generated out of the manifolds increase too.



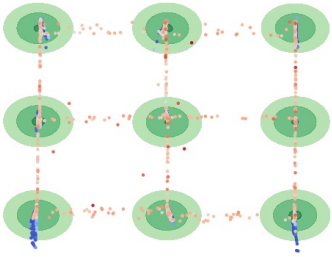
WGAN 4 classes:



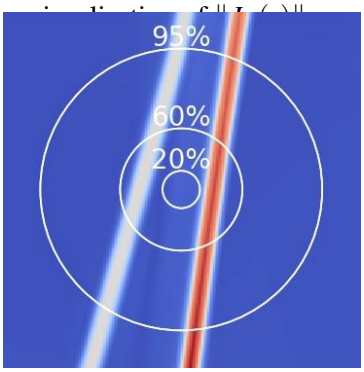
Green blobs: true densities.



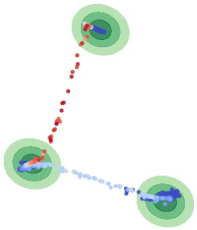
WGAN 9 classes:



Green blobs: true densities.
Dots: generated points.



WGAN 3 classes:
visualisation of $\|J_G(z)\|_F$.



Green blobs: true densities.
Dots: generated points.

Fig. A.1 Learning disconnected manifolds: visualization of the gradient of the generator (JFN) in the latent space and densities in the output space.

A.2.2 Definition of the different metrics used

In the sequel, we present the different metrics used in Section 3.2.4 of the paper to assess performances of GANs. We have:

- Improved Precision/Recall (PR) metric Kynkäänniemi et al. (2019): it has been presented in Definition 3.2.2. Intuitively, Based on a k-NN estimation of the manifold of real (resp. generated) data, it assesses whether generated (resp. real) points belong in the real (resp. generated) data manifold or not. The proportion of generated (resp. real) points that are in the real (resp. generated) data manifold is the precision (resp. recall).

- the Hausdorff distance: it is defined by

$$\text{Haus}(A,B) = \max\left\{\max_{a\in A}\min_{b\in B}\|a-b\|, \max_{b\in B}\min_{a\in A}\|a-b\|\right\}$$

  Such a distance is useful to evaluate the closeness of two different supports from a metric space, but is sensitive to outliers because of the max operation. It has been recently used for theoretical purposes by Pandeva and Schubert (2019).

- the Frechet Inception distance: first proposed by Dowson and Landau (1982), the Frechet distance was applied in the setting of GANs by Heusel et al. (2017). This distance between mutlivariate Gaussians compares statistic of generated samples to real samples as follows

$$\text{FID} = \|\nu_\star - \nu_\theta\|^2 + Tr\big(\Sigma_\star + \Sigma_\theta + 2(\Sigma_\star\Sigma_\theta)^{\frac{1}{2}}\big)$$

  where $X_\star = \mathcal{N}(\nu_\star, \Sigma_\star)$ and $X_\theta = \mathcal{N}(\nu_\theta, \Sigma_\theta)$ are the activations of a pre-softmax layer. However, when dealing with disconnected manifolds, we argue that this distance is not well suited as it approximates the distributions with unimodal one, thus loosing many information.

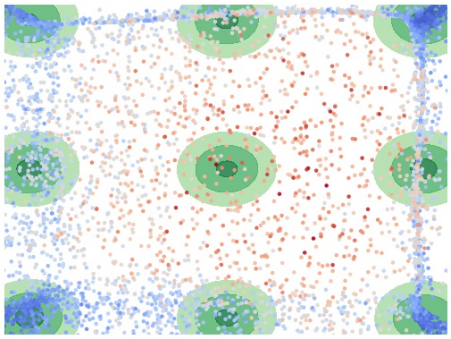
The choice of such metrics is motivated by the fact that metrics measuring the performances of GANs should not rely on relative densities but should rather be point sets based metrics.

A.2.3 Saturation of a MLP neural network

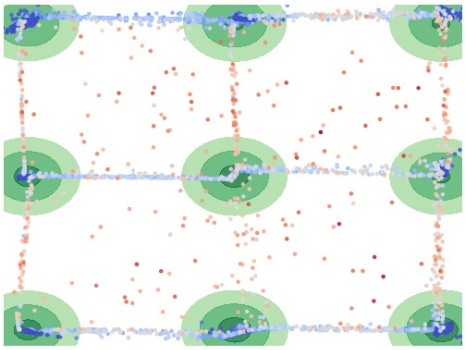
In Section 3.2.4.2, we claim that the generator reduces the sampling of off-manifold data points up to a saturation point. Figure A.2 below provides a visualization of this phenomenon. In this synthetic case, we learn a 9-component mixture of Gaussians using simple GANs architecture (both the generator and the discriminator are MLP with two hidden layers). The minimal distance between two modes is set to 9. We clearly see in Figure A.2d that the precision saturates around 80%.

A.2.4 More results and visualizations on MNIST/F-MNIST/CIFAR10

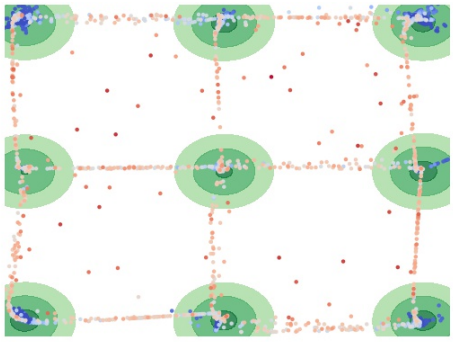
Additionally to those in Section 3.2.4.3, we provide in Figure A.3 and Table A.1 supplementary results for MNIST, F-MNIST and CIFAR-10 datasets.



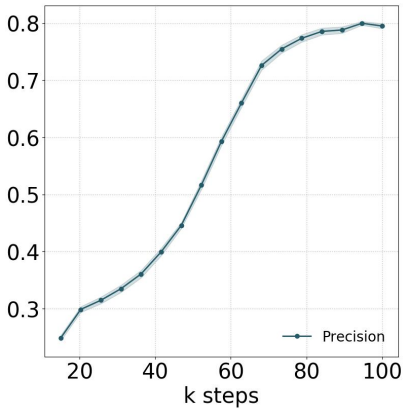
Samples after 5k training steps.



Samples after 50k training steps.



Samples after 100k training steps.



Evolution of the precision $\bar{\alpha}$ during training.

Fig. A.2 Learning 9 disconnected manifolds with a standard GANs architecture.

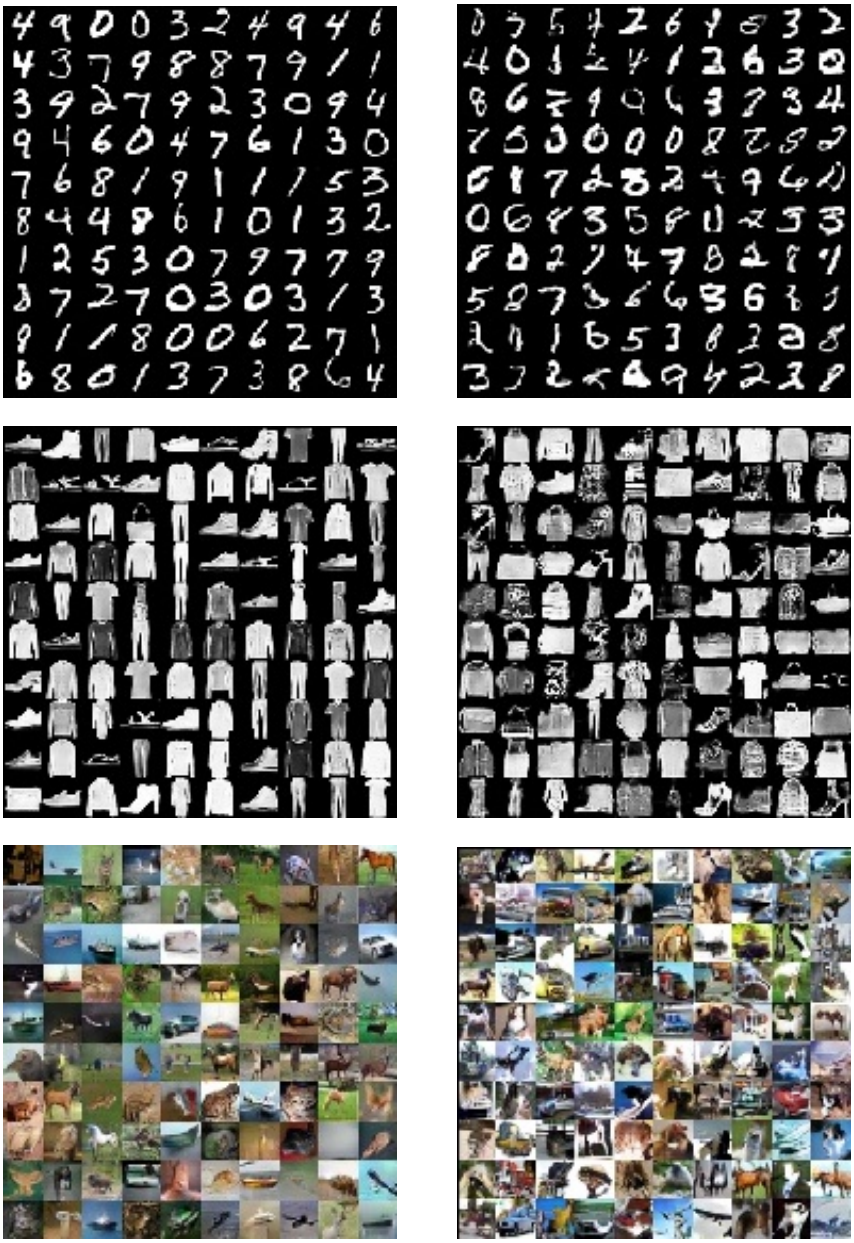


Fig. A.3 Visualization of our truncation method (JBT) on three real-world datasets. From top to bottom: MNIST, F-MNIST and CIFAR-10. Left column: examples of data points selected by our JBT with a truncation ratio of 90% (we thus removed the 10% highest gradients). Right column: examples of data points removed by our JBT with a truncation ratio of 90% (these are the 10% highest gradients data points).

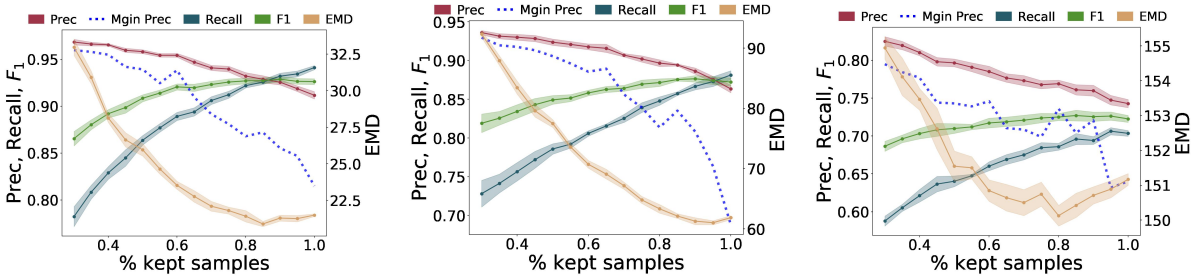


Fig. A.4 For high levels of kept samples, the marginal precision plummets of newly added samples, underlining the efficiency of our truncation method (JBT). Reported confidence intervals are 97% confidence intervals. On the second row, generated samples ordered by their JFN (left to right, top to bottom). In the last row, the data points generated are blurrier and outside the true manifold.

Table A.1 Scores on MNIST and Fashion-MNIST. JFN stands for Jacobian Frobenius norm. $\pm$ is 97% confidence interval.

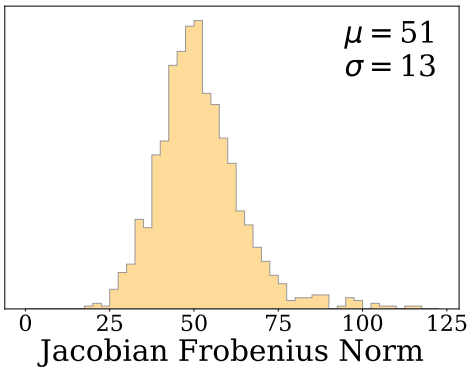
MNIST	Prec.	Rec.	F1	Haus.	FID	EMD
WGAN	91.2 ± 0.3	$\mathbf{93.7 \pm 0.5}$	$\mathbf{92.4 \pm 0.4}$	49.7 ± 0.2	24.3 ± 0.3	21.5 ± 0.1
WGAN 90% JFN	92.5 ± 0.5	92.9 ± 0.3	$\mathbf{92.7 \pm 0.4}$	$\mathbf{48.1 \pm 0.2}$	26.9 ± 0.5	21.3 ± 0.2
WGAN 80% JFN	$\mathbf{93.3 \pm 0.3}$	91.8 ± 0.4	$\mathbf{92.6 \pm 0.4}$	50.6 ± 0.4	33.1 ± 0.3	21.4 ± 0.4
W-Deligan	89.0 ± 0.6	$\mathbf{93.6 \pm 0.3}$	91.2 ± 0.5	50.7 ± 0.3	31.7 ± 0.5	22.4 ± 0.1
DMLGAN	$\mathbf{93.4 \pm 0.2}$	92.3 ± 0.2	$\mathbf{92.8 \pm 0.2}$	$\mathbf{48.2 \pm 0.3}$	$\mathbf{16.8 \pm 0.4}$	$\mathbf{20.7 \pm 0.1}$
Fashion-MNIST						
WGAN	86.3 ± 0.4	$\mathbf{88.2 \pm 0.2}$	$\mathbf{87.2 \pm 0.3}$	140.6 ± 0.7	259.7 ± 3.5	61.9 ± 0.3
WGAN 90% JFN	88.6 ± 0.6	86.6 ± 0.5	$\mathbf{87.6 \pm 0.5}$	$\mathbf{138.7 \pm 0.9}$	$\mathbf{257.4 \pm 3.0}$	$\mathbf{61.3 \pm 0.6}$
WGAN 80% JFN	$\mathbf{89.8 \pm 0.4}$	84.9 ± 0.5	$\mathbf{87.3 \pm 0.4}$	146.3 ± 1.1	396.2 ± 6.4	63.3 ± 0.7
W-Deligan	88.5 ± 0.3	85.3 ± 0.6	86.9 ± 0.4	141.7 ± 1.1	310.9 ± 3.1	$\mathbf{60.9 \pm 0.4}$
DMLGAN	87.4 ± 0.3	$\mathbf{88.1 \pm 0.4}$	$\mathbf{87.7 \pm 0.4}$	141.9 ± 1.2	$\mathbf{253.0 \pm 2.8}$	$\mathbf{60.9 \pm 0.4}$
CIFAR10						
WGAN	74.3 ± 0.5	$\mathbf{70.3 \pm 0.4}$	$\mathbf{72.3 \pm 0.5}$	334.7 ± 3.5	$\mathbf{634.8 \pm 4.6}$	151.2 ± 0.2
WGAN 90% JFN	$\mathbf{76.0 \pm 0.7}$	69.4 ± 0.5	$\mathbf{72.5 \pm 0.6}$	$\mathbf{318.1 \pm 3.7}$	$\mathbf{631.3 \pm 4.5}$	150.7 ± 0.2
WGAN 80% JFN	$\mathbf{76.9 \pm 0.5}$	68.6 ± 0.5	$\mathbf{72.5 \pm 0.5}$	$\mathbf{323.5 \pm 4.0}$	725.0 ± 3.5	$\mathbf{150.1 \pm 0.3}$
W-Deligan	71.5 ± 0.7	$\mathbf{69.8 \pm 0.7}$	70.6 ± 0.7	328.7 ± 2.1	727.8 ± 3.9	154.0 ± 0.3
DMLGAN	74.1 ± 0.5	65.7 ± 0.6	69.7 ± 0.6	328.6 ± 2.7	967.2 ± 4.1	152.0 ± 0.4

A.2.5 More results on BigGAN and ImageNet

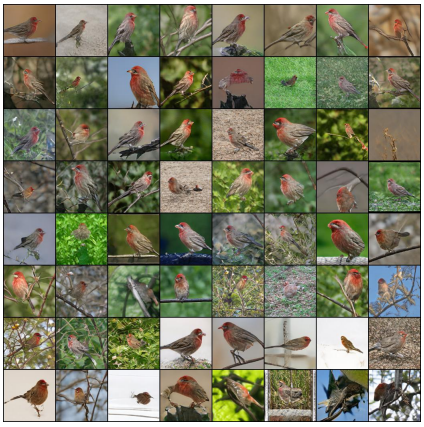
In Figure A.5, we show images from the Bubble class of ImageNet. It supports our claim of manifold disconectedness, even within a class, and outlines the importance of studying the learning of disconnected manifolds in generative models. Then, in Figure A.6 and Figure A.7, we give more exemples from BigGAN 128x128 class-conditionned generator. We plot in the same format than in 3.2.4.4. Specifically, for different classes, we plot 128 images ranked by JFN. Here again, we see a concentration of off-manifold samples on the last row, proving the efficiency of our method. Example of classes responding particularly well to our ranking are House Finch c, Monnarch Butterfly e or Wood rabbit c. For each class, we also show an histogram of JFN based on 1024 samples. It shows that the JFN is a good indicator of the complexity of the class. For example, classes such as Cornet (see Figure A.7e) or Football helmet (see Figure A.7a) are very diverse and disconnected, resulting in high JFNs.

Fig. A.5 Images from the Bubble class of ImageNet showing that the class is complex and slightly multimodal.

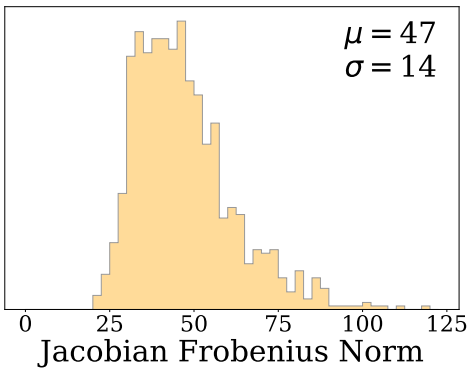
'Black swan' class.



'Black swan' class histogram.

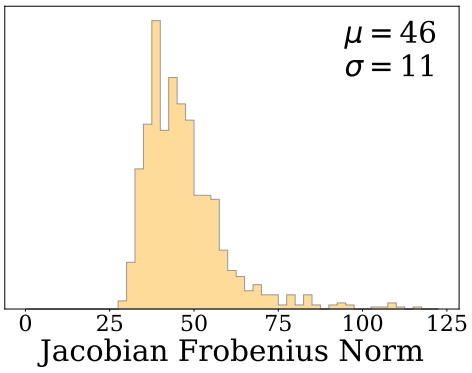


'House finch' class.



'House finch' class histogram.

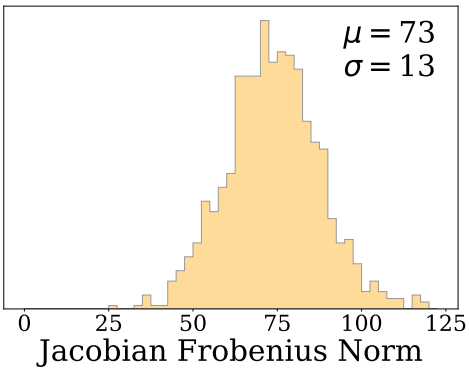
'Monarch butterfly' class.



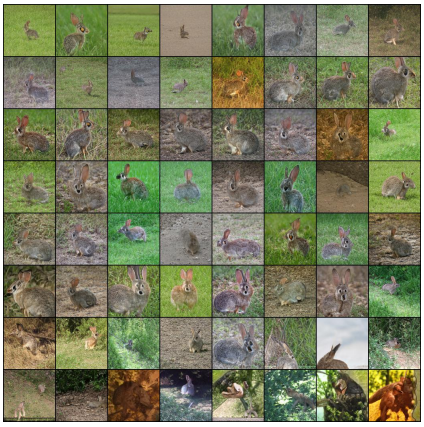
'Monarch butterfly' class histogram.

Fig. A.6 Images from BigGAN class-conditional generator, along with an histogram of class-specific Jacobian Frobenius Norms.

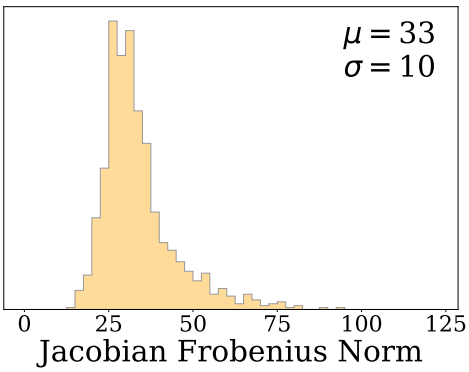
'Football helmet' class.



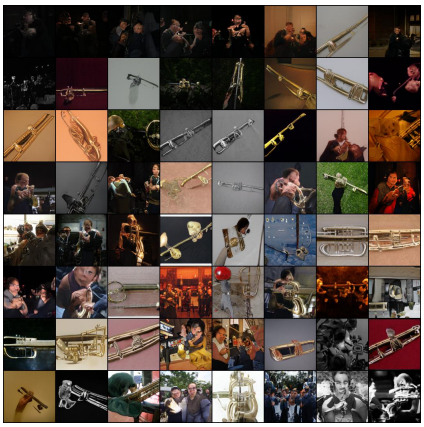
'Football helmet' class histogram.



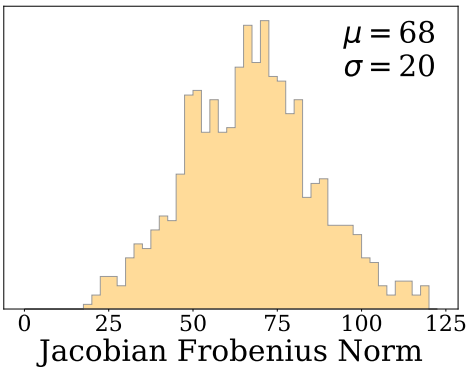
'Wood rabbit' class.



'wood rabbit' class histogram.



'Cornet' class.



'Cornet' class histogram.

Fig. A.7 Images from BigGAN class-conditional generator, along with an histogram of class-specific Jacobian Frobenius Norms.

A.3 Supplementary details

We now provide the different network's architecture used and their corresponding hyperparameters.

Table A.2 Models for Synthetic datasets

Operation	Feature Maps	Activation
G(z): $z \sim \mathcal{N}(0,1)$	2	
Fully Connected - layer1	20	ReLU
Fully Connected - layer2	20	ReLU
D(x)		
Fully Connected - layer1	20	ReLU
Fully Connected - layer2	20	ReLU
Batch size	32	
Leaky ReLU slope	0.2	
Gradient Penalty weight	10	
Learning Rate	0.0002	
Optimizer	Adam: $\beta_1 = 0.5 \beta_2 = 0.5$	

For DeliGan, we use the same architecture and simply add 50 Gaussians for the reparametrization trick. For DMLGAN, we re-use the architecture of the authors.

Table A.3 WGAN for MNIST/Fashion MNIST

Operation	Kernel	Strides	Feature Maps	Activation
G(z): $z \sim \mathrm{N}(0, Id)$			100	
Fully Connected			$7 \times 7 \times 128$	
Convolution	3×3	1×1	$7 \times 7 \times 64$	LReLU
Convolution	3×3	1×1	$7 \times 7 \times 64$	LReLU
Nearest Up Sample			$14 \times 14 \times 64$	
Convolution	3×3	1×1	$14 \times 14 \times 32$	LReLU
Convolution	3×3	1×1	$14 \times 14 \times 32$	LReLU
Nearest Up Sample			$14 \times 14 \times 64$	
Convolution	3×3	1×1	$28 \times 28 \times 16$	LReLU
Convolution	5×5	1×1	$28 \times 28 \times 1$	Tanh
D(x)			$28 \times 28 \times 1$	
Convolution	4×4	2×2	$14 \times 14 \times 32$	LReLU
Convolution	3×3	1×1	$14 \times 14 \times 32$	LReLU
Convolution	4×4	2×2	$7 \times 7 \times 64$	LReLU
Convolution	3×3	1×1	$7 \times 7 \times 64$	LReLU
Fully Connected			1	-
Batch size	256			
Leaky ReLU slope	0.2			
Gradient Penalty weight	10			
Learning Rate	0.0002			
Optimizer	Adam $\beta_1 : 0.5 \beta_2 : 0.5$			

Table A.4 DMLGAN for MNIST/Fashion MNIST

Operation	Kernel	Strides	Feature Maps	BN	Activation
G(z): $z \sim \mathrm{N}(0, Id)$			100		
Fully Connected			$7 \times 7 \times 128$	-	
Convolution	3×3	1×1	$7 \times 7 \times 64$	-	Leaky ReLU
Convolution	3×3	1×1	$7 \times 7 \times 64$	-	Leaky ReLU
Nearest Up Sample			$14 \times 14 \times 64$	-	
Convolution	3×3	1×1	$14 \times 14 \times 32$	-	Leaky ReLU
Convolution	3×3	1×1	$14 \times 14 \times 32$	-	Leaky ReLU
Nearest Up Sample			$14 \times 14 \times 64$	-	
Convolution	3×3	1×1	$28 \times 28 \times 16$	-	Leaky ReLU
Convolution	5×5	1×1	$28 \times 28 \times 1$	-	Tanh
Encoder Q(x), Discriminator D(x)			$28 \times 28 \times 1$		
Convolution	4×4	2×2	$14 \times 14 \times 32$	-	Leaky ReLU
Convolution	3×3	1×1	$14 \times 14 \times 32$	-	Leaky ReLU
Convolution	4×4	2×2	$7 \times 7 \times 64$	-	Leaky ReLU
Convolution	3×3	1×1	$7 \times 7 \times 64$	-	Leaky ReLU
D Fully Connected			1	-	-
Q Convolution	3×3		$7 \times 7 \times 64$	Y	Leaky ReLU
Q Convolution	3×3		$7 \times 7 \times 64$	Y	Leaky ReLU
Q Fully Connected			$n_g = 10$	-	Softmax
Batch size	256				
Leaky ReLU slope	0.2				
Gradient Penalty weight	10				
Learning Rate	0.0002				
Optimizer	Adam $\beta_1 = 0.5 \beta_2 = 0.5$				

Table A.5 WGAN for CIFAR10, from Gulrajani et al. (2017)

Operation	Kernel	Strides	Feature Maps	BN	Activation
G(z): $z \sim \mathrm{N}(0, Id)$			128		
Fully Connected			$4 \times 4 \times 128$	-	
ResBlock	$[3 \times 3] \times 21 \times 1$		$4 \times 4 \times 128$	Y	ReLU
Nearest Up Sample			$8 \times 8 \times 128$	-	
ResBlock	$[3 \times 3] \times 21 \times 1$		$8 \times 8 \times 128$	Y	ReLU
Nearest Up Sample			$16 \times 16 \times 128$	-	
ResBlock	$[3 \times 3] \times 21 \times 1$		$16 \times 16 \times 128$	Y	ReLU
Nearest Up Sample			$32 \times 32 \times 128$	-	
Convolution	3×3	1×1	$32 \times 32 \times 3$	-	Tanh
Discriminator D(x)			$32 \times 32 \times 3$		
ResBlock	$[3 \times 3] \times 21 \times 1$		$32 \times 32 \times 128$	-	ReLU
AvgPool	2×2	1×1	$16 \times 16 \times 128$	-	
ResBlock	$[3 \times 3] \times 21 \times 1$		$16 \times 16 \times 128$	-	ReLU
AvgPool	2×2	1×1	$8 \times 8 \times 128$	-	
ResBlock	$[3 \times 3] \times 21 \times 1$		$8 \times 8 \times 128$	-	ReLU
ResBlock	$[3 \times 3] \times 21 \times 1$		$8 \times 8 \times 128$	-	ReLU
Mean pooling (spatial-wise)	-	-	128	-	
Fully Connected			1	-	-
Batch size	64				
Gradient Penalty weight	10				
Learning Rate	0.0002				
Optimizer	Adam	$\beta_1 = 0.\beta_2 = 0.9$			
Discriminator steps	5				

Table A.6 DMLGAN for CIFAR10, from Gulrajani et al. (2017)

Operation	Kernel	Strides	Feature Maps	BN	Activation
G(z): $z \sim \mathrm{N}(0, Id)$			128		
Fully Connected			$4 \times 4 \times 128$	-	
ResBlock	$[3 \times 3] \times 21 \times 1$		$4 \times 4 \times 128$	Y	ReLU
Nearest Up Sample			$8 \times 8 \times 128$	-	
ResBlock	$[3 \times 3] \times 21 \times 1$		$8 \times 8 \times 128$	Y	ReLU
Nearest Up Sample			$16 \times 16 \times 128$	-	
ResBlock	$[3 \times 3] \times 21 \times 1$		$16 \times 16 \times 128$	Y	ReLU
Nearest Up Sample			$32 \times 32 \times 128$	-	
Convolution	3×3	1×1	$32 \times 32 \times 3$	-	Tanh
Encoder Q(x), Discriminator D(x)			$32 \times 32 \times 3$		
ResBlock	$[3 \times 3] \times 21 \times 1$		$32 \times 32 \times 128$	-	ReLU
AvgPool	2×2	1×1	$16 \times 16 \times 128$	-	
ResBlock	$[3 \times 3] \times 21 \times 1$		$16 \times 16 \times 128$	-	ReLU
AvgPool	2×2	1×1	$8 \times 8 \times 128$	-	
ResBlock	$[3 \times 3] \times 21 \times 1$		$8 \times 8 \times 128$	-	ReLU
D ResBlock	$[3 \times 3] \times 21 \times 1$		$8 \times 8 \times 128$	-	ReLU
D Mean pooling (spatial-wise)	2×2	1×1	128	-	
D Fully Connected			1	-	-
Q ResBlock	$[3 \times 3] \times 21 \times 1$		$8 \times 8 \times 128$	-	ReLU
Q Mean pooling (spatial-wise)	2×2	1×1	128	-	
Q Fully Connected			$n_g = 10$	-	Softmax
Batch size	64				
Gradient Penalty weight	10				
Learning Rate	0.0002				
Optimizer	Adam	$\beta_1 = 0.\beta_2 = 0.9$			
Discriminator steps	5				

Appendix B

B.1 Technical results: proofs

B.1.1 Proof of Lemma 3.3.1

We want to show that generator $G \in \mathscr{G}_L^{\mathscr{A}}$ is such that $\alpha_G \leqslant 1-\gamma(\partial^{\varepsilon_{\min}}\mathscr{A})$, where

$$\partial^{\varepsilon_{\min}}\mathscr{A} = \bigcup_{i=1}^{m} \left(\cup_{j\neq i} A_j\right)^{\varepsilon_{\min}} \backslash \left(\cup_{j\neq i} A_j\right).$$

Proof by contradiction.

Assume a generator G such that there exists $z \in \partial^{\varepsilon_{\min}}\mathscr{A}$ and $i \in [1,m]$ such that $G(z) \in M_i$. Since G is associated with $\mathscr{A}$, we have using Definition 3.3.2, that there exists z' and $j \in [1,m], j \neq i$ such that $\|z-z'\| < \varepsilon_{\min}/2$ and $j = \underset{k\in[1,m]}{\arg\min} \|G(z') - M_k\|$. Thus, we have:

$$\begin{aligned} \|G(z)-G(z')\| &\geqslant d(G(z'),M_i), \\ &\geqslant d(M_i,M_i)/2, \\ &\geqslant D_{\min}/2. \\ \text{And, } \frac{\|G(z)-G(z')\|}{\|z-z'\|} &> D_{\min}/\varepsilon_{\min}, \\ &> L. \end{aligned}$$

This contradicts G being in $\mathscr{G}_L^{\mathscr{A}}$.

B.1.2 Proof of Corollary 3.3.1.

Let L, D be such that $L \geqslant D\sqrt{\log(m)}$. Let's prove that for any well-balanced generator $G \in \mathscr{G}_L$, we have:

$$\alpha_G \leqslant 1 - \varepsilon_{\min}\sqrt{\log m}\, e^{-3/2}.$$

Using the method from Schechtman (2012), we have the measure of the border of cell i:

$$\begin{aligned}
\gamma\Big(\big(\cup_{j\neq i} A_j\big)^{\varepsilon} \backslash \big(\cup_{j\neq i} A_j\big)\Big) &\geqslant \frac{1}{\sqrt{2\pi}} \int_t^{t+\varepsilon} e^{-s^2/2} ds, \\
&\geqslant \frac{\varepsilon}{\sqrt{2\pi}} e^{-(t+\varepsilon)^2/2}, \\
&\geqslant \frac{\varepsilon\sqrt{\log m}}{m} e^{-\varepsilon t - \varepsilon^2/2}, \\
&\geqslant \frac{\varepsilon\sqrt{\log m}}{m} e^{-\varepsilon\sqrt{\log m} - \varepsilon^2/2}.
\end{aligned}$$

In first line, t is such that $\frac{1}{\sqrt{2\pi}} \int_t^{\infty} e^{-s^2/2} ds = 1/m$. In third line, we use $\sqrt{\log m} \leq t \leq \sqrt{2\log m}$).

Thus:

$$\gamma(\partial^{\varepsilon_{\min}} \mathscr{A}) = \sum_{i=1}^{m} \gamma\Big(\big(\cup_{j\neq i} A_j\big)^{\varepsilon} \backslash \big(\cup_{j\neq i} A_j\big)\Big) \geqslant \varepsilon_{\min}\sqrt{\log m}\, e^{-\varepsilon_{\min}\sqrt{\log m} - \varepsilon_{\min}^2/2}.$$

Thus, we have

$$\begin{aligned}
\alpha_G &\leqslant 1 - \gamma(\partial^{\varepsilon_{\min}} \mathscr{A}), \\
&\leqslant 1 - \varepsilon_{\min}\sqrt{\log m}\, e^{-\varepsilon_{\min}\sqrt{\log m} - \varepsilon_{\min}^2/2}.
\end{aligned}$$

Moreover, using $\varepsilon_{\min} = \frac{D}{L}$ and $L \geqslant D\sqrt{\log m}$, so we get $\varepsilon_{\min}\sqrt{\log m} \leqslant 1$:

$$\alpha_G \leqslant 1 - \varepsilon_{\min}\sqrt{\log m}\, e^{-3/2}.$$

Proof of Corollary 3.3.1 with w_i (non-equal measure of modes). Let L, D be such that $L \geqslant D\sqrt{\log(m)}$. Let's prove that for any well-balanced generator $G \in \mathscr{G}_L$, we have:

$$\alpha_G \leqslant 1 - m w_{\min} \varepsilon_{\min}\sqrt{\log 1/w_{\min}}\, e^{-3/2}.$$

Using the method from Schechtman (2012), we have the measure of the border of cell i:

$$\begin{aligned}\gamma\Big(\big(\cup_{j\neq i}A_j\big)^{\varepsilon}\backslash\big(\cup_{j\neq i}A_j\big)\Big) &\geqslant \frac{1}{\sqrt{2\pi}}\int_t^{t+\varepsilon} e^{-s^2/2}ds,\\ &\geqslant \frac{\varepsilon}{\sqrt{2\pi}}e^{-(t+\varepsilon)^2/2},\\ &\geqslant w_{\min}\varepsilon\sqrt{\log 1/w_{\min}}e^{-\varepsilon t-\varepsilon^2/2},\\ &\geqslant w_{\min}\varepsilon\sqrt{\log 1/w_{\min}}e^{-\varepsilon\sqrt{\log 1/w_{\min}}-\varepsilon^2/2}.\end{aligned}$$

In first line, t is such that $\frac{1}{\sqrt{2\pi}}\int_t^{\infty} e^{-s^2/2}ds = w_{\min}$. In third line, we use $\sqrt{\log 1/w_{\min}} \leq t \leq \sqrt{2\log 1/w_{\min}}$).

Thus:

$$\gamma(\partial^{\varepsilon_{\min}}\mathscr{A}) = \sum_{i=1}^{m}\gamma\Big(\big(\cup_{j\neq i}A_j\big)^{\varepsilon}\backslash\big(\cup_{j\neq i}A_j\big)\Big) \geqslant m w_{\min}\varepsilon_{\min}\sqrt{\log 1/w_{\min}}\; e^{-\varepsilon_{\min}\sqrt{\log 1/w_{\min}}-\varepsilon_{\min}^2/2}.$$

Thus, we have

$$\begin{aligned}\alpha_G &\leqslant 1-\gamma(\partial^{\varepsilon_{\min}}\mathscr{A}),\\ &\leqslant 1-m w_{\min}\varepsilon_{\min}\sqrt{\log 1/w_{\min}}\; e^{-\varepsilon_{\min}\sqrt{\log 1/w_{\min}}-\varepsilon_{\min}^2/2}.\end{aligned}$$

Moreover, using $\varepsilon_{\min} = \frac{D}{L}$ and $L \geqslant D\sqrt{\log 1/w_{\min}}$, so we get $\varepsilon_{\min}\sqrt{\log 1/w_{\min}} \leqslant 1$:

$$\alpha_G \leqslant 1 - m\varepsilon_{\min}w_{\min}\sqrt{\log 1/w_{\min}}\; e^{-3/2}.$$

B.1.3 Proof of Theorem 3.3.1

Let $\mu^\star$ be the target distribution. We know that $\mu_\star$ lays on m disconnected components contained in spheres $S_i, i \in [1,m]$. We note $M_i, i \in [1,m]$ the centers, and r_i the radius of each sphere. We also assume that the spheres verify Assumption 4. For each pair $(i,j) \in [1,m]^2$, we define $X_{ij} \in S_i$ and $X_{ji} \in S_j$ the points verifying

$$X_{ij} = \underset{x\in S_i}{\arg\min}\; d(x,S_j) \quad \text{and} \quad X_{ji} = \underset{x\in S_j}{\arg\min}\; d(x,S_i).$$

We consider the optimal partition $\mathscr{A}^\star$ in the Gaussian latent space. For each given latent point $z \in \mathbb{R}^d$, we define:

$$N_z = \{j \in [1,m], z \in A_j^{\varepsilon}\}.$$

We then distinguish two different cases:

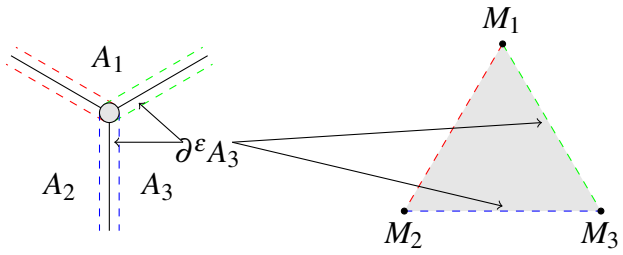


Fig. B.1 An optimal generator maps a 2D latent space to a 2D output space with three modes (M_1, M_2, M_3). The latent space has an optimal 'simplicial cluster' geometry. In the latent space, all the ε-boundaries intersect each other in the gray circle, which is mapped in the output space in the convex hull of the three modes.

1. $|N_z| = 1$: the point z belongs to the interior of a single cell, $z \in A_i^{-\varepsilon}$.

2. $|N_z| \geqslant 2$: the point z is in the neighborhood of at least two different cells.

Interestingly, a point can only belong at most to the interior of one cell, but it can be at the intersection of several boundaries. We are now ready to define the optimal generator.

First, we set

$$G(z) = X_{i,j} \text{ for all } z \in \{z \in \mathbb{R}^d, |N_z| = 2, z \in \overline{A_i^{-\varepsilon}} \cap A_i^{\varepsilon} \cap A_j^{\varepsilon} \text{ where } N_z = \{i, j\}\}.$$

Second, we define the generator in the interior of the cells, i.e. $N_z = \{i\}$. For each $z \in A_i^{-\varepsilon}$ and for a given unit vector $u \in \mathbb{R}^d$, we assume that the generator is constant along the parametric line $z = k \times u, k \in \mathbb{R}$.

Finally, we define the generator when z does not belong to the interior of any cell, i.e. $|N_z| \geqslant 2$:

$$G_\varepsilon^\star(z) = \sum_{i \in [1,m]} \sum_{j \neq i} w_{i,j}(z)\, X_{i,j}\, \mathbb{1}_{j \in N_z}\, \mathbb{1}_{i \in N_z} \tag{B.1.1}$$

where

$$w_{i,j}(z) = \frac{d(z, (A_i^\varepsilon)^\complement)}{\sum_{i \in [1,m]} \sum_{j \neq i} d(z, (A_j^\varepsilon)^\complement)\, \mathbb{1}_{j \in N_z}\, \mathbb{1}_{i \in N_z}}$$

where $d(z, A) = \min_{a \in A} \|z - a\|$. An illustration of the optimal generator is given in Figure B.1. When z belongs to the intersection of two ε-boundaries, $G_\varepsilon(z)$ is a simple linear combination of 2 points. It is only when $|N_z| \geqslant 3$ that more complex samples are generated. A simple illustration of $G_\varepsilon^\star$ for $d = 2$ and $m = 3$ is given in Figure B.1. Interestingly, one can also show that the image of $G_\varepsilon^\star$ is equal to the convex hull of the Diracs $X_i, i \in [1, m]$. In particular, there

exists a particularly interesting neighborhood ν of 0 where $G^\star_\varepsilon(\nu)$ is equal to the whole convex hull of the points $X_i, i \in [1,m]$.

Proof that $G^\star_\varepsilon$ is well-balanced. We recall that a generator is *well-balanced* if we have $G\sharp\gamma(M_1) = \ldots = G\sharp\gamma(M_m)$. By construction (B.1.1), we have that for any $i \in [1,m]$

$$\begin{aligned}\|G^\star_\varepsilon(z) - X_i\| &= \|\sum_{k\neq i} w_k(X_k - X_i)\|,\\ &= D \times (1 - w_i).\end{aligned}$$

So, for any $z \in A_i$, we have that

$$i = \underset{j\in[1,m]\ w_j}{\arg\min} = \underset{j\in[1,m]\ \|G(z)-X_j\|}{\arg\min}$$

Thus $G^\star_\varepsilon$ is associated with the optimal partition $\mathscr{A}^\star$, .

Besides, for a given radius r of the different modes, since everything is symmetrical, we have that $\gamma(\{z \in \mathbb{R}^d, \|G(z) - X_1\| \leqslant r\} = \ldots = \gamma(\{z \in \mathbb{R}^d, \|G(z) - X_m\| \leqslant r\}$. Thus, the generator is well-balanced.

Showing that $G^\star_{\varepsilon^\star}$ is in $\mathscr{G}_L$. It is clear that when $|N_z| = 1$, we have that $G^\star_\varepsilon(z)$ is a L-Lipshitz continuous function.

Now, assume that $|N_z| \geqslant 2$. Consider z, z' such that $N_z = N'_z$. Let $\alpha = (\alpha_1, \ldots, \alpha_m)$ and $\beta = (\beta_1, \ldots, \beta_m)$ be two vectors, both in $\mathbb{R}^m$, such that for all $i \in [1,m]$:

$$\alpha_i = \frac{d(z, (A_i^\varepsilon)^\complement)}{\sum_{j\in\mathscr{A}_z} d(z, (A_j^\varepsilon)^\complement)} \quad \text{and} \quad \beta_i = \frac{d(z', (A_i^\varepsilon)^\complement)}{\sum_{j\in\mathscr{A}_z} d(z', (A_j^\varepsilon)^\complement)}$$

We have that

$$\begin{aligned}\|G(z) - G(z')\| &= \|(1 - \sum_{i\neq 1}\alpha_i)X_1 - (1 - \sum_{i\neq 1}\beta_i)X_1 + \sum_{i\neq 1}\alpha_i X_i - \sum_{i\neq 1}\beta_i X_i\|\\ &= \|\sum_{i\neq 1}(\alpha_i - \beta_i)(X_1 - X_i)\|\\ &\leqslant \max_{(i,j)\in[1,m]^2}\|X_i - X_j\|\ \|\alpha - \beta\|,\\ &\leqslant \max_{(i,j)\in[1,m]^2}\|X_i - X_j\|\ \|h(z) - h(z')\|,\end{aligned}$$

where h is the function from $\mathbb{R}^d \to \mathbb{R}^m$ defined as:

$$h(z) = \left(\frac{d(z,(A_1^\varepsilon)^\complement)}{\sum_{i\in\mathscr{A}_z} d(z,(A_i^\varepsilon)^\complement)}, \ldots, \frac{d(z,(A_m^\varepsilon)^\complement)}{\sum_{i\in\mathscr{A}_z} d(z,(A_i^\varepsilon)^\complement)}\right).$$

We can write $h = f \circ g$ with f the function defined from $\mathbb{R}^d \to \mathbb{R}^m$ by

$$f(z) = \Big(d(z,(A_1^\varepsilon)^\complement), \ldots, d(z,(A_k^\varepsilon)^\complement)\Big),$$

and g the function defined on $\mathbb{R}^m \setminus \{0\}$ by

$$g(z) = \frac{z}{\|z\|_1}$$

We have that f is a $\sqrt{m}$-Lipschitz functions (given that $z \mapsto d(z,(A_m^\varepsilon)^\complement)$ is 1-Lipschitz). Besides, we know that outside the ball $B_{\varepsilon/2}(0)$, the function g is $(2/\varepsilon)$-Lipschitz. Since it is clear that for every point z such that $|N_z| \geqslant 2$, we have that $|f(z)| \geqslant \varepsilon/2$. Finally, the function h is $\frac{2\sqrt{m}}{\varepsilon}$-Lipschitz. Thus, we have that:

$$\|G_\varepsilon^\star(z) - G_\varepsilon^\star(z')\| \leqslant \frac{2D\sqrt{m}}{\varepsilon}\|z - z'\|,$$

with $D = \max_{i,j} \|X_i - X_j\|, (i,j) \in [1,m]^2, i \neq j$.

Now, by noting $\varepsilon_{\max} = \frac{D}{L}$, and considering $\varepsilon^\star = 2\sqrt{m}\,\varepsilon_{\max}$, we have:

$$\|G_{\varepsilon^\star}^\star(z) - G_{\varepsilon^\star}^\star(z')\| \leqslant L\|z - z'\|.$$

Now, consider two latent vectors z, z' in the same cell $\overline{A_i^{-\varepsilon}}$. There exists $i \in [1,m]$, and a pairs $(j,j') \in [1,m]^2$ (note that j could be equal to j') such that $G(z) = X_{i,j}$ and $G(z') = X_{i,j'}$. Using a similar reasoning as before, we can show that:

$$\|G_{\varepsilon^\star}^\star(z) - G_{\varepsilon^\star}^\star(z')\| \leqslant L\|z - z'\|,$$

with $D = 2\max_{i\in[1,m]}\ r_i$.

We can now conclude on the Lipschitzness of $G^\star$ on $\mathbb{R}^d$.

Proving that: for $m \leqslant d+1$, for any $\delta > 0$, if L is large enough, then, for any well-balanced $G \in \mathscr{G}_L$, we have $\alpha_{G^\star_{\varepsilon_{\max}}} \geqslant \alpha_G - \delta$. Let G be a well-balanced generator and $\mathscr{A}$ the partition associated with G. Let us first define the gaussian boundary measure P_γ of a partition $\mathscr{A}$ of $\mathbb{R}^d$. For partitions with smooth boundaries, it coincides with the $(d-1)$-dimensional gaussian

measure of the boundary, defined as follows:

$$P_\gamma(\mathscr{A}) = \liminf_{\varepsilon\to 0} \frac{\gamma(\partial^\varepsilon \mathscr{A}) - \gamma(\mathscr{A})}{\sqrt{2/\pi}\varepsilon}$$

Moreover, for sets with smooth boundaries, we have from Federer (2014, Theorem 3.2.29):

$$\liminf_{\varepsilon\to 0} \frac{\gamma(\partial^\varepsilon \mathscr{A}) - \gamma(\mathscr{A})}{\sqrt{2/\pi}\varepsilon} = \lim_{\varepsilon\to 0} \frac{\gamma(\partial^\varepsilon \mathscr{A}) - \gamma(\mathscr{A})}{\sqrt{2/\pi}\varepsilon}$$

Let us denote $\mathscr{A}^\star$, the optimal partition defined in Milman and Neeman (2022), based on simplicial clusters. $A^\star$ is a standard partition where $\gamma(A_1^\star) = \ldots = \gamma(A_m^\star)$ for all i, and $\sum_i \gamma(A_i) = 1$. By the multi-bubble theorem (Milman and Neeman, 2022), simplicial clusters (such as $\mathscr{A}^\star$) are the unique minimizers of the gaussian isoperimetric problem, thus:

$$\begin{aligned} P_\gamma(\mathscr{A}^\star) &\leqslant P_\gamma(\mathscr{A}) \\ \lim_{\varepsilon\to 0} \frac{\gamma(\partial^\varepsilon \mathscr{A}^\star)}{\varepsilon} &\leqslant \lim_{\varepsilon\to 0} \frac{\gamma(\partial^\varepsilon \mathscr{A})}{\varepsilon} \\ L_{\mathscr{A}} &\leqslant L_{\mathscr{A}^\star} \end{aligned}$$

where $L_{\mathscr{A}} = \lim_{\varepsilon\to 0} \frac{\gamma(\partial^\varepsilon \mathscr{A}^\star)}{\varepsilon}$ and $L_{\mathscr{A}^\star} = \lim_{\varepsilon\to 0} \frac{\gamma(\partial^\varepsilon \mathscr{A}^\star)}{\varepsilon}$.

Then, for any $\delta > 0$, there exists $\varepsilon' > 0$ such that, for any $\varepsilon < \varepsilon'$,

$$\left|\frac{\gamma(\partial^\varepsilon \mathscr{A}^\star)}{\varepsilon} - L_{\mathscr{A}^\star}\right| < \delta \quad , \quad \left|\frac{\gamma(\partial^\varepsilon \mathscr{A})}{\varepsilon} - L_{\mathscr{A}}\right| < \delta \quad \text{and} \quad L_{\mathscr{A}^\star} \leqslant L_{\mathscr{A}}$$

Thus, for any $\delta > 0$, there exists $\varepsilon' > 0$ such that, for any $\varepsilon < \varepsilon'$,

$$\gamma(\partial^\varepsilon \mathscr{A}^\star) \leqslant \gamma(\partial^\varepsilon \mathscr{A}) + 2\delta\varepsilon \tag{B.1.2}$$

Besides, we know that

$$\alpha_G \leqslant 1 - \gamma(\partial^{\varepsilon_{\min}} \mathscr{A})$$

Consequently, we have that:

$$\begin{aligned} \alpha_G &\leqslant 1 - \gamma(\partial^{\varepsilon_{\min}} \mathscr{A}) \\ &\leqslant 1 - \gamma(\partial^{\varepsilon_{\min}} \mathscr{A}^\star) + 2\delta\varepsilon_{\min} \quad \text{using (B.1.2).} \end{aligned}$$

Now, by construction of $G^\star_{\varepsilon_{\max}}$, we have that

$$\alpha_{G^\star_{\varepsilon_{\max}}} \geqslant 1 - \gamma(\partial^{\varepsilon_{\max}} \mathscr{A}^\star).$$

Consequently,

$$\begin{aligned}
\alpha_G &\leqslant 1-\gamma(\partial^{\varepsilon_{\min}}\mathscr{A}^\star)+2\delta\varepsilon_{\max}+\gamma(\partial^{\varepsilon_{\max}}\mathscr{A}^\star)-\gamma(\partial^{\varepsilon_{\max}}\mathscr{A}^\star)\\
&\leqslant \alpha_{G^\star_\varepsilon}+2\delta\varepsilon_{\max}+\gamma(\partial^{\varepsilon_{\max}}\mathscr{A}^\star)-\gamma(\partial^{\varepsilon_{\min}}\mathscr{A}^\star)\\
&\leqslant \alpha_{G^\star_\varepsilon}+2\delta\varepsilon_{\max}+\gamma(\partial^{\varepsilon_{\max}}\mathscr{A}^\star)-2L_{\mathscr{A}^\star}\varepsilon_{\max}-\gamma(\partial^{\varepsilon_{\min}}\mathscr{A}^\star)+2L_{\mathscr{A}^\star}\varepsilon_{\min}+2L_{\mathscr{A}^\star}(\varepsilon_{\max}-\varepsilon_{\min})\\
&\leqslant \alpha_{G^\star_\varepsilon}+4\delta\varepsilon_{\max}+2L_{\mathscr{A}^\star}\varepsilon_{\max},\\
&\leqslant \alpha_{G^\star_\varepsilon}+\varepsilon_{\max}(4\delta+2L_{\mathscr{A}^\star}).
\end{aligned}$$

We conclude by choosing L big enough such that $\varepsilon_{\max}$ is strictly smaller than $\frac{\delta}{4\delta+2L_{\mathscr{A}^\star}}$.

Proving the lower-bound 3.3.7 of Theorem 3.3.1. Let's consider $G_{\varepsilon^\star}$ defined using (B.1.1) and $\varepsilon^\star = 2\sqrt{m}\varepsilon_{\max}$. The precision of $G^\star_{\varepsilon^\star}$ is thus such that:

$$\alpha_{G^\star_{\varepsilon^\star}} \geqslant 1-\gamma(\partial^{\varepsilon^\star}\mathscr{A}).$$

However, since $\partial^\varepsilon\mathscr{A}\subset\bigcup_{i=1}^n A_i^\varepsilon$, we have that for any ε:

$$\gamma(\partial^\varepsilon\mathscr{A})\leqslant\sum_{i=1}^n\gamma(A_i^\varepsilon).$$

Using results from Schechtman (2012, Proposition 1), when $m\leq d$, there exists C large enough, such that

$$\gamma(A_i^{\varepsilon^\star})\leqslant\frac{\varepsilon^\star}{m}\big(\sqrt{\pi\log(Cm)}\big).$$

Thus, we have

$$\alpha_{G^\star_{\varepsilon^\star}}\geqslant 1-\varepsilon^\star\sqrt{\pi\log(Cm)},$$

To have $\alpha_{G^\star_{\varepsilon_{\max}}}\geqslant 0$, we must have $^\star\leqslant 1/\sqrt{\pi\log(Cm)}$. This is the case since we have

$$^\star = 2D\sqrt{m}/L \quad\text{and}\quad L\geqslant D\sqrt{m}\sqrt{\pi\log(Cm)},$$

where $D=\max_{i,j}\ \|X_i-X_j\|$.

Table B.1 GANs training details on MNIST

Operation	Kernel	Strides	Feature Maps	Activation
Generator G(z)				
$z \sim \mathrm{N}(0, I)$			$dim(z)$	
Fully Connected			$7 \times 7 \times 128$	
Convolution	3×3	1×1	$7 \times 7 \times 64$	LReLU
Convolution	3×3	1×1	$7 \times 7 \times 64$	LReLU
Nearest Up Sample			$14 \times 14 \times 64$	
Convolution	3×3	1×1	$14 \times 14 \times 32$	LReLU
Convolution	3×3	1×1	$14 \times 14 \times 32$	LReLU
Nearest Up Sample			$14 \times 14 \times 32$	
Convolution	3×3	1×1	$28 \times 28 \times 16$	LReLU
Convolution	3×3	1×1	$28 \times 28 \times 1$	Tanh
D(x)			$28 \times 28 \times 1$	
Convolution	4×4	2×2	$14 \times 14 \times 512$	LReLU
Convolution	3×3	1×1	$14 \times 14 \times 512$	LReLU
Convolution	4×4	2×2	$7 \times 7 \times 512$	LReLU
Convolution	3×3	1×1	$7 \times 7 \times 512$	LReLU
Fully Connected			1	-
Batch size	256			
Leaky ReLU slope	0.2			
Gradient Penalty weight	10			
Learning Rate Discriminator	1×10^{-4}			
Learning Rate Generator	5×10^{-5}			
Disciminator steps	2			
Optimizer	Adam	$\beta_1 : 0.5 \beta_2 : 0.5$		

B.2 Experiments

B.2.1 Implementation details

First, let us note that we share our code in Supplementary Material for reproducibility.

Training. We use the Wasserstein loss with gradient-penalty on interpolations of fake and real data. At each iteration, the discriminator is trained 2 steps and the generator 1 step with Adam optimizer. The batch size is 256. The learning rate of the discriminator is two times larger (Heusel et al., 2017), *i.e.* 5×10^{-5} for the generator and 1×10^{-4} for the discriminator. GANs are trained for 80k steps on MNIST and for 100k steps on CIFAR datasets. Architectures of generator and discriminator are described in Table B.1 and Table B.2.

Table B.2 GANs training details on CIFAR datasets. BN stands for batch-normalization.

Operation	Kernel	Strides	Feature Maps	Conditional BN	Activation
Generator G(z)					
$z \sim \mathrm{N}(0, Id)$			128		
Fully Connected			$4 \times 4 \times 128$	-	
ResBlock	$[3 \times 3] \times 2$	1×1	$4 \times 4 \times 128$	Y	ReLU
Nearest Up Sample			$8 \times 8 \times 128$	-	
ResBlock	$[3 \times 3] \times 2$	1×1	$8 \times 8 \times 128$	Y	ReLU
Nearest Up Sample			$16 \times 16 \times 128$	-	
ResBlock	$[3 \times 3] \times 2$	1×1	$16 \times 16 \times 128$	Y	ReLU
Nearest Up Sample			$32 \times 32 \times 128$	-	
Convolution	3×3	1×1	$32 \times 32 \times 3$	-	Tanh
Discriminator D(x)			$32 \times 32 \times 3$		
ResBlock	$[3 \times 3] \times 2$	1×1	$32 \times 32 \times 256$	-	ReLU
AvgPool	2×2	1×1	$16 \times 16 \times 256$	-	
ResBlock	$[3 \times 3] \times 2$	1×1	$16 \times 16 \times 256$	-	ReLU
AvgPool	2×2	1×1	$8 \times 8 \times 256$	-	
ResBlock	$[3 \times 3] \times 2$	1×1	$8 \times 8 \times 256$	-	ReLU
ResBlock	$[3 \times 3] \times 2$	1×1	$8 \times 8 \times 256$	-	ReLU
Mean spatial pooling	-	-	256	-	
Fully Connected			1	-	-
Batch size	256				
Gradient Penalty weight	10				
Learning Rate Discriminator	1×10^{-4}				
Learning Rate Generator	5×10^{-5}				
Discriminator steps	2				
Optimizer	Adam	$\beta_1 = 0.\beta_2 = 0.999$			

For TransGAN (Jiang et al., 2021), we follow the implementation from the authors available at https://github.com/VITA-Group/TransGAN. TransGAN is trained with a WGAN-GP loss, 4 discriminator steps for 1 generator step, and Adam optimizer with a learning rate of 10^{-4}.

Evaluation. For evaluation metrics, we follow the setting proposed by the authors. For FID (Heusel et al., 2017), we use 50k real images and 50k fake images. For precision, recall, density and coverage (Kynkäänniemi et al., 2019; Naeem et al., 2020), we use 10k real images and 10k fake images with nearest-k$=5$.

GPUs. For all datasets, the training of GANs was run on NVIDIA Tesla V100 GPUs (16 GB). The training of ResNet GANs for 100k steps on CIFAR takes around 30 hours. For TransGAN models, the training is done for 250k steps on two NVIDIA Tesla V100 GPUs, which takes around $35 \times 2 = 70$ GPU hours.

B.2.2 Correlation between latent space geometry and GANs' performance (Details for Section 3.3.5.3)

We present the full results of this study in Table B.3.

Table B.3 Correlation between GANs' performance and their latent space geometry. Increasing the capacity of GANs tend to structure their latent space in simplicial clusters (better LogReg accuracy) and improve their performance on precision, density and coverage. Confidence intervals are computed on several sets of generated/training points from a given generator.

Dataset	Width	LogReg Acc. ↑	Convex Acc. ↑	FID ↓	Prec. ↑	Rec. ↑	Dens. ↑	Cov. ↑
CIFAR-10 (Resnet)	32	53.4 ± 0.5	61.1 ± 0.3	28.3 ± 0.6	63.2 ± 0.6	58.6 ± 0.9	66.3 ± 1.5	61.3 ± 1.1
	64	60.7 ± 0.5	72.1 ± 0.6	20.6 ± 0.3	65.7 ± 0.5	62.0 ± 0.6	71.4 ± 1.7	71.5 ± 1.0
	128	63.4 ± 0.4	73.1 ± 0.6	17.0 ± 0.3	65.9 ± 0.4	64.5 ± 0.9	71.2 ± 1.5	74.6 ± 0.9
	256	65.0 ± 0.4	**75.4** ± 0.4	**16.1** ± 0.3	66.4 ± 0.5	**66.2** ± 1.0	72.3 ± 1.5	75.6 ± 1.0
	512	**65.3** ± 0.6	75.2 ± 0.7	**16.1** ± 0.3	**66.8** ± 1.0	66.1 ± 1.3	**72.8** ± 2.9	**76.1** ± 1.4
CIFAR-100 (Resnet)	32	20.3 ± 0.1	28.1 ± 0.5	28.3 ± 0.3	53.4 ± 0.7	63.5 ± 0.8	44.5 ± 1.3	56.1 ± 1.2
	64	23.7 ± 0.9	33.4 ± 0.5	23.4 ± 0.3	59.9 ± 0.5	64.6 ± 0.7	57.6 ± 1.7	67.6 ± 0.5
	128	28.4 ± 0.3	39.1 ± 0.7	21.1 ± 0.4	61.6 ± 0.4	63.8 ± 0.5	62.2 ± 1.0	70.2 ± 0.5
	256	29.9 ± 0.5	41.8 ± 0.6	21.0 ± 0.4	62.3 ± 0.6	**65.6** ± 0.6	62.7 ± 2.0	70.1 ± 0.9
	512	**30.5** ± 0.5	**42.1** ± 0.5	**19.7** ± 0.4	**64.3** ± 0.8	64.8 ± 0.8	**66.8** ± 1.9	**72.2** ± 0.6

B.2.3 Details on simplicial truncation method (Details for Section 3.3.5.4)

We provide here more details about our truncation method. First, the rejection sampling in the latent space $\mathbb{R}^d$ of GANs procedure is the following:

- Define hyper-parameters threshold τ, number of clusters N, latent space dimension d.
- Initialize N equidistant points in $\{(u_0, \dots, u_N) \mid u_i \in \mathbb{R}^d\}$. This can be done easily when $N \leq d$.
- When sampling latent vectors $z \in \mathbb{R}^d$, compute a softmax over the negative distances between z and u_i: $p_i(z) = \frac{e^{-d(z,u_i)}}{\sum_j e^{-d(z,u_j)}}$. Then, z is selected if $\max_i \big(p_i(z)\big) > \tau$.

Second, we add a classification loss to encourage the generator to use this latent structure. This loss is motivated by the need to maximize mutual information between the latent cluster and the modes of the generator Khayatkhoei et al. (2018), and can be written as:

$$L_c = -\mathbb{E}_{z\sim\gamma}[\ln q_\phi \big(i(z)|G_\theta(z)\big)]$$

where q_ϕ is parametrized by a second classification head added to the discriminator; $i(z) = \arg\max_i \big(p_i(z)\big)$ is the index of the latent cluster of the sample. This loss is added during training, at each step of generator's and discriminator's training, during the first 20 epochs. It is then dropped, since we noticed that it harms the GANs performance if it is kept until the end of the training.

Training hyper-parameters: for $N = 10$, we use a latent dimension of $d = 64$ and training threshold of $\tau = 0.135$; for $N = 100$, we use $d = 128$ and $\tau = 0.08$.

During inference, if the generator has properly learned to use the different clusters of the latent space, we observe that augmenting the threshold τ leads to an increased density and precision.

We present full results in Table B.4 and Figure B.2.

Table B.4 Density/Coverage curves comparing TransGAN and boosting methods for multi-modal datasets and different threshold ratios. Our simplicial truncation method (TransGAN + simp.) consistently outperforms the TransGAN and TransGAN + DeliGAN baselines.

Dataset	Model	FID	Prec	Rec	Dens.	Cov.
CIFAR-10	TransGAN	8.9 ± 0.1	72.8 ± 0.8	62.6 ± 0.7	79.3 ± 0.9	79.3 ± 1.2
	TransGAN + 90% JBT	8.7 ± 0.1	73.0 ± 0.6	61.9 ± 0.8	83.5 ± 1.7	80.0 ± 1.6
	TransGAN + 80% JBT	8.8 ± 0.1	73.3 ± 0.8	61.2 ± 1.0	85.7 ± 2.8	81.1 ± 0.6
	TransGAN + DeliGAN N=10	9.8 ± 0.1	74.6 ± 0.8	58.6 ± 0.9	93.2 ± 2.8	80.0 ± 0.6
	TransGAN + lin. N=10					
	(0.23,0.23)	9.2 ± 0.1	73.1 ± 1.1	61.9 ± 1.2	78.1 ± 2.7	79.4 ± 0.9
	(0.23,0.29)	9.2 ± 0.1	73.7 ± 0.8	61.5 ± 1.2	83.4 ± 3.3	79.8 ± 0.5
	(0.23,0.31)	9.3 ± 0.1	74.0 ± 0.5	61.0 ± 0.7	86.1 ± 1.8	81.3 ± 0.7
	(0.23,0.4)	9.8 ± 0.1	75.1 ± 0.8	59.8 ± 1.2	89.5 ± 1.6	80.5 ± 1.2
	TransGAN + simp. N=10					
	(0.135,0.135)	9.0 ± 0.1	72.9 ± 0.5	61.8 ± 0.9	82.7 ± 1.9	80.4 ± 0.7
	(0.135,0.14)	9.0 ± 0.1	74.2 ± 1.5	60.7 ± 1.0	88.5 ± 3.1	81.3 ± 1.4
	(0.135,0.1445)	9.3 ± 0.1	75.3 ± 0.6	58.8 ± 0.8	98.6 ± 1.3	82.9 ± 0.5
CIFAR-100	TransGAN	15.2 ± 0.1	64.2 ± 0.5	63.1 ± 0.9	53.4 ± 1.3	66.0 ± 1.1
	TransGAN + 90% JBT	15.1 ± 0.2	64.8 ± 1.0	62.9 ± 1.3	53.6 ± 2.4	66.2 ± 1.4
	TransGAN + 80% JBT	14.8 ± 0.2	65.4 ± 1.7	61.7 ± 1.2	55.0 ± 4.0	65.6 ± 2.3
	TransGAN Deligan 10	15.9 ± 0.2	63.5 ± 0.8	62.2 ± 0.7	52.6 ± 1.3	64.4 ± 0.6
	TransGAN DeliGAN 100	15.3 ± 0.1	64.2 ± 0.5	61.9 ± 0.9	52.6 ± 0.6	65.9 ± 0.8
	TransGAN + simp. N=10					
	(0.135,0.135)	15.1 ± 0.1	65.1 ± 0.6	62.3 ± 0.5	55.6 ± 0.6	67.1 ± 0.5
	(0.135, 0.14)	15.1 ± 0.1	64.8 ± 0.2	61.1 ± 0.5	55.3 ± 1.3	66.8 ± 1.1
	(0.135,0.1445)	15.1 ± 0.1	65.6 ± 1.3	61.5 ± 0.8	56.3 ± 1.5	66.4 ± 1.4
STL-10 (32x32)	TransGAN	10.5 ± 0.1	75.7 ± 0.6	60.1 ± 0.8	87.5 ± 1.9	83.0 ± 0.2
	TransGAN + 90% JBT	10.5 ± 0.1	76.9 ± 0.7	58.8 ± 0.5	91.9 ± 1.9	82.1 ± 0.8
	TransGAN + 80% JBT	11.0 ± 0.1	78.1 ± 0.3	57.6 ± 1.3	99.3 ± 2.8	83.8 ± 0.7
	TransGAN DeliGAN 10	12.1 ± 0.1	74.2 ± 1.2	60.2 ± 0.5	81.5 ± 1.5	79.6 ± 0.8
	TransGAN DeliGAN 100	10.5 ± 0.2	76.0 ± 0.5	60.2 ± 1.6	85.5 ± 2.8	81.5 ± 1.4
	TransGAN + simp. N=100					
	(0.08,0.08)	10.1 ± 0.1	76.5 ± 0.9	60.2 ± 0.8	90.0 ± 1.7	83.0 ± 0.5
	(0.08,0.15)	10.0 ± 0.1	76.9 ± 0.8	59.9 ± 0.6	91.4 ± 1.1	83.8 ± 0.3
	(0.08,0.20)	10.0 ± 0.1	77.8 ± 0.6	59.8 ± 0.8	94.1 ± 0.9	83.5 ± 0.8

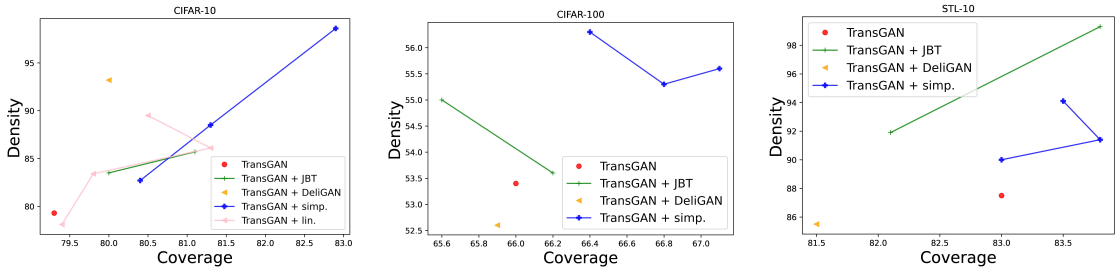


Fig. B.2 Density/Coverage curves comparing TransGAN and boosting methods for multi-modal datasets and different threshold ratios. Our simplicial truncation method (TransGAN + simp.) consistently outperforms the TransGAN and TransGAN + DeliGAN baselines.

B.2.4 Impact of the number of modes: a synthetic example (Details for Section 3.3.5.2)

To illustrate our theoretical results, we propose to vary the number of modes of the data distribution. On real-world data, the number of modes is set but usually unknown, and removing/adding classes as a proxy for modes usually does not give insightful results since some classes can be much more complex than others. We thus use a synthetic setting, where we can easily control both the number of modes and their complexity. Figure B.3 stresses that as the number of modes increase, the precision decrease. Interestingly, using large latent space dimension can relieve the problem, even if the latent space dimension is clearly below that of the target. Recall the two problems that arise when training GANs: i) *dimensional misspecification* where the true and modeled distributions do not have density functions w.r.t. the same base measure, and ii) *density misspecification*, where GANs try to fit a disconnected manifold with a unimodal disitribution. From the results we conclude that:

- With very low latent space dimensions, both problems i) and ii) have to be addressed and this leads to poor precision as the number of modes increases.
- With larger latent space dimensions, the problem i) is less of a burden even when there is a clear dimensional misspecification and thus the GANs' performance is more tied to problem ii).

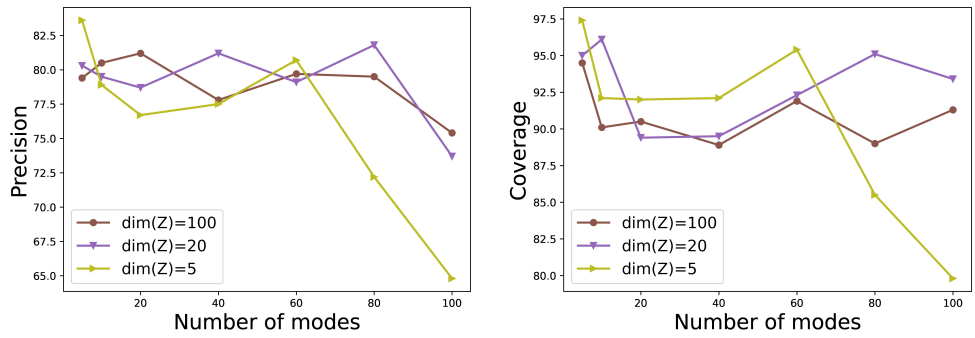


Fig. B.3 Training on a mixture of Gausians in $\mathbb{R}^{100}$ with varying number of modes and varying latent space dimension. The bigger the number of modes, the lower the precision. Increasing the latent space dimension helps up to a limit depending on the number of modes.

Appendix C

C.1 Proof of Lemma 3.4.1

Let's prove that $\mathbb{E}_{\gamma^\varphi} = 1$. We have that:

$$\int_{\mathbb{R}^d} 1\gamma^\varphi(\mathrm{dz}) = \int_{\mathbb{R}^\mathrm{d}} \mathrm{w}^\varphi(\mathrm{z})\gamma(\mathrm{dz}) = 1,$$

by assumption. Consequently, the measure γ^φ is a well-defined probability distribution on $\mathbb{R}^d$.

C.2 Proof of Theorem 3.4.1

It is clear that the network w^φ is a density function with respect to the distribution γ defined on $\mathbb{R}^d$. Consequently, the measure μ_θ^φ is absolutely continuous with respect to μ_θ and thus with respect to the Lebesgue measure.

We start the proof by stating that for any absolutely continuous distribution μ_θ^φ, there exists an optimal transport T_θ^φ, (Pratelli, 2007, Theorem B), such that $W(\mu_n, \mu_\theta^\varphi) = \int_{\mathbb{R}^D} \|x - T_\theta^\varphi(x)\| \mathrm{d}\mu_\theta^\varphi$ Hartmann and Schuhmacher (2020). Recall that for any $x \in \mathbb{R}^D$, there exists $X_i \in [1,n]$ such that $T_\theta^\varphi(x) = X_i$. Since μ_θ^φ is absolutely continuous, there exists a ball $B(z,r)$ centered in $z \in \mathbb{R}^d$ with radius $r > 0$ such that $\mu_\theta^\varphi(B(z,r)) > 0$ and, we have:

1. there exists $i \in [1,n]$ such that for all $x \in B(z,r)$, $T_\theta^\varphi(x) = X_i$,
2. for all $x \in B(z,r)$, $\|x - X_i\| > \|X_i - \tilde{X}_i\|$, recall that $\tilde{X}_i = \underset{z \in \mathbb{R}^d}{\arg\min} \|X_i - G_\theta(z)\|$.

Consequently, we have:

$$
\begin{aligned}
W(\mu_n, \mu_\theta^\varphi) &= \int_{\mathbb{R}^D} \|x - T_\theta^\varphi(x)\| \mu_\theta^\varphi(\mathrm{d}x) \\
&= \int_{\mathbb{R}^D \setminus B(z,r)} \|x - T_\theta^\varphi(x)\| \mu_\theta^\varphi(\mathrm{d}x) + \int_{B(z,r)} \|x - T_\theta^\varphi(x)\| \mu_\theta^\varphi(\mathrm{d}x) \\
&> \int_{\mathbb{R}^D \setminus B(z,r)} \|x - T_\theta^\varphi(x)\| \mu_\theta^\varphi(\mathrm{d}x) + \int_{B(z,r)} \|X_i - \tilde{X}_i\| \mu_\theta^\varphi(\mathrm{d}x) \\
&\geqslant \int_{\mathbb{R}^D \setminus B(z,r)} \|\tilde{T}_\theta^\varphi(x) - T_\theta^\varphi(x)\| \mu_\theta^\varphi(\mathrm{d}x) + \int_{B(z,r)} \|X_i - \tilde{X}_i\| \mu_\theta^\varphi(\mathrm{d}x) \\
&\text{(where } \tilde{T}_\theta^\varphi(x) = \min_{z \in \mathbb{R}^d} \|z - T_\theta^\varphi(x)\|) \\
&= \frac{1}{n} \sum_{i=1}^n \|X_i - \tilde{X}_i\| \\
&= W(\mu_n, \frac{1}{n} \sum_{i=1}^n \delta(\tilde{X}_i)),
\end{aligned}
$$

where δ refers to the Dirac probability distribution.

Finally, when taking the infinimum over all continuous functions φ, we have that:

$$
W(\mu_n, \frac{1}{n} \sum_{i=1}^n \delta(\tilde{X}_i)) \leqslant \inf_{\varphi \in \Phi} W(\mu_n, \mu_\theta^\varphi)
$$

C.3 Evaluation details

Precision recall metric. For the precision-recall metric, we use the algorithm from Khayatkhoei et al. (2018). Namely, when comparing the set of real data points $(x_1, ..., x_n)$ with the set of fake data points $(y_1, ..., y_n)$:

A point x_i has a recall $r(x_i) = 1$ if there exists y_j, such that $\|x_i - y_j\| \leq \|y_j - y_j(k)\|$, where $y_j(k)$ is the k-nearest neighbor of n. Finally, the recall is the average of individual recall: $\frac{1}{n} \sum_i r(x_i)$.

A point y_i has a precision $p(y_i) = 1$ if there exists x_j, such that $\|y_i - x_j\| \leq \|x_j - x_j(k)\|$, where $x_j(k)$ is the k-nearest neighbor of n. Finally, the precision is the average of individual precision: $\frac{1}{n} \sum_i p(x_i)$.

Parameters. For all datasets, we use $k = 3$ (3rd nearest neighbor). For MNIST and F-MNIST, we use a set of $n = 2048$ points. For CelebA and LSUN Church, we use a set of $n = 1024$ points. This is also valid for the EMD. For FID, we use the standard protocol with $n = 50000$ points and Inception Net. We run 10 evaluations of each metric (each evaluation is

done with a different set of random points), report the average and the 97% confidence interval by considering that we have 10 i.i.d. samples from a normal distribution.

C.4 Sampling algorithms: latentRS, latentGA, and latentRS+GA

We present here the three sampling algorithms associated with our importance weight function w^{φ}. See Section C.5 for details on the hyper-parameters used in latentGA and latentRS+GA.

Algorithm 4: LatentRS

Requires: Prior Z, Gen. G_θ, Importance weight network w^{φ}, maximum importance weight m;
while *True* **do**
  Sample $z \sim Z$;
  Sample $\alpha \sim \text{Uniform}[0,1]$;
  if $\frac{w^{\varphi}(z)}{m} \geq \alpha$ **then**
    break;
  end
end
$x \leftarrow G_\theta(z)$;
Result: Selected point x

Algorithm 5: Latent Gradient Ascent (latentGA)

Requires: Prior Z, number of dimensions of the prior d, Gen. G_θ, Importance weight network w^{φ}, number of steps N, step size ε;
Sample $z \sim Z$;
for $n = 1 : N$ **do**
  $\text{grad}_z \leftarrow \nabla_z w^{\varphi}(z)$;
  if $Z == \mathcal{N}(0, I)$ ***and*** $d >> 1$ **then**
    ## Projection step for high-dimensional gaussians ## ;
    $\text{grad}_z \leftarrow \text{grad}_z - (\text{grad}_z \cdot z)z/\sqrt{d}$;
  end
  $z \leftarrow z + \varepsilon * \text{grad}_z$;
end
$x \leftarrow G_\theta(z)$;
Result: Selected point x

Algorithm 6: Latent RS+GA

Requires: Prior Z, Number of dimensions of the prior d, Gen. G_θ, Importance weight network w^φ, maximum importance weight m, number of steps N, step size ε;
while *True* **do**
    Sample $z \sim Z$;
    Sample $\alpha \sim \text{Uniform}[0;1]$;
    if $\frac{w^\varphi(z)}{m} \geq \alpha$ **then**
        break;
    end
end
for $n = 1 : N$ **do**
    $\text{grad}_z \leftarrow \nabla_z w^\varphi(z)$;
    if $Z = \mathcal{N}(0, I)$ ***and*** $d >> 1$ **then**
        ## Projection step for high-dimensional gaussians ## ;
        $\text{grad}_z \leftarrow \text{grad}_z - (\text{grad}_z \cdot z)z/\sqrt{d}$;

    end
    $z \leftarrow z + \varepsilon * \text{grad}_z$;
end
Compute $x = G_\theta(z)$;
Result: Selected point x

C.5 Hyper-parameters.

SIR Grover et al. (2019): Model selection: we fine-tune with a binary cross-entropy loss the discriminator from the end of the adversarial training and select the best model in terms of EMD. We tested with/without regularizing the discriminator during the fine-tuning (with gradient penalty or spectral normalization). Without regularization, the performance drops fast. Best results are obtained by regularizing the discriminator, thus we report these results.

We use then use Sampling-Importance-Resampling algorithm. In SIR, we sample N points from the generator, compute their importance weights according density ratios, and accept one of them (each point is accepted with a probability proportional to its importance weight). The hyper-parameter of SIR algorithm is N. Results for grid search on N are shown below in Table C.2 and Table C.3. In Table 3.5, results are shown with $N = 10$.

DOT Tanaka (2019): Model selection: we fine-tune with the WGAN-GP loss the discriminator from the end of the adversarial training and select the best model in terms of EMD, when running DOT. We perform a projected gradient descent as described in Tanaka (2019) with SGD. Hyper-parameters are the number of steps N_{steps} and the step size ε. We made the following grid search: $N_{steps} = [2,5,10,50]$ and $\varepsilon = [0.01,0.05,0.1]$. Results for grid search on N_{steps} are shown below in Table C.2 and Table C.3. In Table 3.5, results are shown with $N_{steps} = 10$ and $\varepsilon = 0.05$ or $\varepsilon = 0.01$ depending on the dataset (we select the best one).

Training of w^{φ}: For MNIST and F-MNIST, we use the same hyper-parameters: $\lambda_1 = 10$, $\lambda_2 = 3$ and $m = 3$. w^{φ} is a standard MLP with 4 hidden layers, each having 400 nodes (4x dimension of latent space), and relu activation. The output layer is 1-dimensional and with a relu activation. The learning rate of the discriminator is $4 * 10^{-4}$, the learning rate of w^{φ} is 10^{-4}. The two networks are optimized with Adam algorithm, where we set $\beta = (0.5,0.5)$. We use 1 step of importance weight optimization for 1 step of discriminator optimization.

For Progressive GAN on CelebA (128x128), we use: $\lambda_1 = 20$, $\lambda_2 = 5$ and $m = 3$. w^{φ} is a standard MLP with 4 hidden layers, each having 512 nodes (1x dimension of latent space), and leaky-relu activation (0.2 of negative slope). The output layer is 1-dimensional and with relu activation. Since we do not have the pre-trained discriminator, we first train a WGAN-GP discriminator between ProGAN and CelebA images for 500 steps, and then start the adversarial training of w^{φ}. The learning rate of the discriminator is 10^{-4}, the learning rate of w^{φ} is 10^{-5}. The two networks are optimized with Adam algorithm, where we set $\beta = (0.,0.999)$. During optimization, we perform iteratively 3 w^{φ} updates and 1 discriminator's updates.

For StyleGAN2 on LSUN Church (256x256), we use: $\lambda_1 = 30$, $\lambda_2 = 5$ and $m = 2$. w^{φ} is a standard MLP with 3 hidden layers, each having 512 nodes (1x dimension of latent space), and leaky-relu activation (0.2 of negative slope). The output layer is 1-dimensional and with a relu activation. Since we do not have the pre-trained discriminator, we first train a WGAN-GP

discriminator between StyleGAN2 and LSUN Church images for 500 steps, and then start the adversarial training of w^{φ}. The learning rate of the discriminator is 10^{-4}, the learning rate of w^{φ} is 10^{-5}. The two networks are optimized with Adam algorithm, where we set $\beta = (0., 0.999)$. During optimization, we perform 3 w^{φ} updates for 1 discriminator's updates.

LatentRS: Once the network w^{φ} is trained (see above), there is no hyper-parameter for latentRS algorithm.

LatentGA and latentRS+GA: We use the same neural network than in LatentRS. The hyper-parameters for this method are similar to DOT: number of steps of gradient ascent N_{steps} and step size ε. With the model selected on LRS, we make the following grid search: $N_{steps} = [2, 5, 10, 50]$ and $\varepsilon = [0.01, 0.05, 0.1]$. Best results were obtained with $\varepsilon = 0.05$ on all datasets. Results for grid search on N_{steps} are shown below in Table C.2 and Table C.3. In Table 3.5, results are shown with $N_{steps} = 10$ and $\varepsilon = 0.05$.

C.6 Comparisons with concurrent methods on synthetic and real-world datasets

In this section, we provide more quantitative results: a comparison of SIR, DOT, SIR, LatentRS and latentRS+GA on MNIST and F-MNIST in Table C.1; an ablation study on the impact of number of points (respectively gradient ascent steps) in SIR (respectively DOT, latentGA and latentRS+GA), on ProGAN trained on CelebA in Table C.2 and StyleGAN2 trained on Lsun Church in Table C.3.

Table C.1 latentRS+GA is the best performer and latentRS matches SOTA with a significantly reduced inference cost (by an order of at least 10). FID was computed using the same dataset-specific classifier used for the Precision/Recall metric. $\pm$ is 97% confidence interval. Inference refers to the time in milliseconds needed to compute one image on a NVIDIA V100 GPU.

	Prec. (↑)	Rec. (↑)	EMD (↓)	FID (↓)	Inference (ms)
MNIST					
Hinge-GP	87.4±0.9	94.6±0.4	24.9±0.3	53.6±7.2	**0.7**
HGP: SIR	88.8±1.0	94.3±0.5	24.2±0.2	38.7±3.1	10.0
HGP: DOT	89.5±0.6	94.0±0.3	24.8±0.2	43.3±3.4	15.7
HGP: latentRS (⋆)	89.0±0.4	**94.7±0.7**	24.1±0.3	**36.3±3.2**	**1.6**
HGP: latentRS+GA (⋆)	**91.8±1.0**	92.8±0.4	**23.4±0.2**	38.2±3.8	8.6
F-MNIST					
Hinge-GP	86.4±0.6	86.8±0.6	68.6±0.4	598.9±55.5	**0.7**
HGP: SIR	86.6±1.1	**88.0±0.5**	68.0±0.5	499.6±31.1	10.0
HGP: DOT	**88.7±0.6**	86.6±0.7	67.7±0.5	508.3±45.7	15.7
HGP: latentRS (⋆)	86.8±0.8	87.5±0.9	67.6±0.6	**438.3±50.2**	1.6
HGP: latentRS+GA (⋆)	88.4±0.7	86.8±0.7	**67.0±0.9**	475.5±58.5	8.6

Table C.2 Comparison of the proposed methods (latentRS, latentGA, and latentRS+GA) with concurrent methods on ProgressiveGan (CelebA 128x128). For this specific study, we explore different computational budgets for SIR, DOT, latentGA, and latentRS+GA. latentRS+GA enables a consistent gain in both EMD and precision for a reasonable computational overhead.

CelebA 128x128	Precision	Recall	EMD	Inference Time
ProGAN	74.2±0.9	60.7±1.4	25.4±0.1	3.6
ProGAN: SIR (n=2)	78.2±1.0	58.4±1.3	25.0±0.1	9.8
ProGAN: SIR (n=5)	79.3±0.6	57.6±1.3	24.9±0.1	24.5
ProGAN: SIR (n=10)	79.5±0.4	57.3±1.0	24.9±0.2	49.0
ProGAN: SIR (n=50)	80.2±1.0	57.4±1.4	25.0±0.1	245.0
ProGAN: DOT (n=2)	78.2±1.1	58.6±1.1	24.9±0.1	16.4
ProGAN: DOT (n=5)	80.0±1.0	56.0±1.1	24.8±0.1	35.6
ProGAN: DOT (n=10)	81.3±1.0	52.9±1.4	25.0±0.1	67.6
ProGAN: DOT (n=50)	82.3±0.7	52.1±1.3	25.0±0.1	323.6
ProGAN: latentGA (n=2) (⋆)	76.7±1.2	**59.4±0.9**	25.2±0.1	5.2
ProGAN: latentGA (n=5) (⋆)	77.8±1.2	58.4±0.7	25.1±0.1	7.6
ProGAN: latentGA (n=10) (⋆)	78.9±1.2	57.4±0.7	25.0±0.1	11.6
ProGAN: latentGA (n=50) (⋆)	84.1±1.2	49.0±1.3	24.8±0.1	43.6
ProGAN: latentRS+GA (n=2) (⋆)	81.2±0.8	55.3±1.5	24.7±0.1	6.1
ProGAN: latentRS+GA (n=5) (⋆)	82.1±0.7	54.3±1.2	24.6±0.2	8.5
ProGAN: latentRS+GA (n=10) (⋆)	83.3±1.0	52.7±0.9	**24.5±0.1**	12.5
ProGAN: latentRS+GA (n=50) (⋆)	**89.2±0.8**	36.1±0.7	25.0±0.1	44.5
ProGAN: latentRS (⋆)	79.3±1.0	56.5±1.2	24.8±0.2	**4.5**

Table C.3 Comparison of the proposed methods (latentRS, latentGA, and latentRS+GA) with concurrent methods on StyleGAN2 (LSUN Church 256x256). For this specific study, we explore different computational budgets for SIR, DOT, latentGA, and latentRS+GA. latentRS+GA enables a consistent gain in both EMD and precision with a reasonable computational overhead.

LSUN Church (256x256)	Precision	Recall	EMD	Inference Time
StyleGAN2	55.6±1.2	62.4±1.1	23.6±0.1	11.7
StyleGAN2: SIR (n=2)	58.5±0.7	60.7±1.3	23.5±0.1	26.0
StyleGAN2: SIR (n=5)	59.8±1.1	59.0±1.2	23.5±0.1	65.0
StyleGAN2: SIR (n=10)	60.5±1.4	58.1±1.3	23.4±0.1	130.0
StyleGAN2: SIR (n=50)	61.2±1.2	57.8±0.9	23.4±0.1	650.0
StyleGAN2: DOT (n=2)	60.4±1.4	57.0±1.1	23.4±0.1	48.7
StyleGAN2: DOT (n=5)	64.1±0.9	52.2±1.0	23.2±0.1	104.2
StyleGAN2: DOT (n=10)	67.4±1.4	48.3±1.0	23.1±0.1	196.7
StyleGAN2: DOT (n=50)	68.8±0.9	37.0±1.1	23.6±0.1	937.7
StyleGAN2: latentGA (n=2) ($\star$)	58.2±1.0	**61.4±1.2**	23.4±0.1	17.1
StyleGAN2: latentGA (n=5) ($\star$)	61.1±0.9	58.5±1.1	23.2±0.1	25.2
StyleGAN2: latentGA (n=10) ($\star$)	64.6±0.9	55.9±1.5	23.0±0.1	38.7
StyleGAN2: latentGA (n=50) ($\star$)	69.9±1.1	47.2±1.4	22.8±0.1	146.7
StyleGAN2: latentRS+GA (n=2) ($\star$)	66.3±1.2	54.8±1.3	23.0±0.1	21.6
StyleGAN2: latentRS+GA (n=5) ($\star$)	69.6±1.0	50.6±0.9	22.8±0.2	29.7
StyleGAN2: latentRS+GA (n=10) ($\star$)	72.6±1.1	43.2±1.3	**22.6±0.1**	43.2
StyleGAN2: latentRS+GA (n=50) ($\star$)	**78.6±1.2**	34.1±0.9	**22.6±0.1**	151.2
StyleGAN2: latentRS ($\star$)	63.3±0.7	57.7±1.0	23.1±0.2	**16.2**

Table C.4 Inference time for one pass of different computational graphs. With the acceptance rate of the different methods, it allows to compute the runtime of these methods.

	Inference Time
MNIST/F-MNIST 28x28	
Generator	0.7
Generator + Discriminator	1.0
$\nabla_z D(G(z))$ (gradient for latent DOT)	1.5
Network w^{φ} ($\star$)	0.3
$\nabla_z W(z)$ (gradient for latent GA on IW) ($\star$)	0.7
CelebA 128x128	
ProGAN Generator	3.6
ProGAN Generator + Discriminator	4.9
ProGAN $\nabla_z D(G(z))$ (gradient for latent DOT)	6.4
ProGAN Network w^{φ} ($\star$)	0.3
ProGAN: $\nabla_z W(z)$ (gradient for latent GA on IW) ($\star$)	0.8
LSUN Church 256x256	
StyleGAN2 Generator	11.7
StyleGAN2 Generator + Discriminator	13.0
StyleGAN2 $\nabla_z D(G(z))$ (gradient for latent DOT)	18.5
StyleGAN2 Network w^{φ} ($\star$)	1.5
StyleGAN2: $\nabla_z W(z)$ (gradient for latent GA on IW) ($\star$)	2.7

C.7 Qualitative results of latentGA.

Fig. C.1 Gradient ascent on latent importance weights (latentGA), on StyleGAN2 trained on LSUN Church.

Fig. C.2 Gradient ascent on latent importance weights (latentGA), on StyleGAN2 trained on LSUN Church.

Fig. C.3 Gradient ascent on latent importance weights (latentGA), on Progressive GAN trained on CelebA.

Appendix D

D.1 Implementation details

The code for the implementation of EdiBERT is available on GitHub at the following link https://github.com/EdiBERT4ImageManipulation/EdiBERT.

A pre-trained model on FFHQ is available on a linked Google Drive. Notebooks to showcase the model have also been developped.

D.1.1 Training hyper-parameters

We use the same architecture than Esser et al. (2021b) for both VQGAN and transformer. On LSUN Bedroom and FFHQ, we use a codebook size of 1024. For the transformer, we use a model with 32 layers of width 1024.

To train the transformer with 2D masking strategy, we generate random rectangles before flattening $Q(E(I))$. The height of rectangles is drawn uniformly from $[0.2 \times h, 0.5 \times h]$. Similarly, the width of rectangles is drawn uniformly from $[0.2 \times w, 0.5 \times w]$. In our experiments, since we work at resolution 256×256 and follow the downsampling factor of 4 from Esser et al. (2021b), we have $h = w = 256/16 = 16$.

Tokens outside the rectangle are used as input, to give context to the transformer, but not for back-propagation. Tokens inside the rectangle are used for back-propagation. $p_{\text{rand}} = 90\%$ of tokens inside the mask are put to random tokens, while $p_{\text{same}} = 1 - p_{\text{rand}} = 10\%$ are given their initial value. Although we did not perform a large hyper-parameter study on this parameter, we feel it is an important one. The lower p_{rand}, the more the learned distributions $p^i_\theta(.|s)$ will be biased towards the observed token s_i. However, setting $p_{\text{rand}} = 1$ leads to a model that diverges too fast from the observed sequence.

D.1.2 Inference hyper-parameters

D.1.2.1 Image inpainting.

We set the number of epochs to 2, collage frequency to 4 per epoch, top-k sampling to 100, dilation to 1, and number of optimization steps to 200. We apply a gaussian filter on the mask for the periodic image collage.

Additionally, we use these two implementation details. 1) We use two latent masks: the latent down-sampled mask latent_mask$_1$, and the dilated mask latent_mask$_2$, obtained by a dilation of latent_mask$_1$. The randomization is done with latent_mask$_1$, such that no information from the unmasked parts of the image is erased. However, the selection of positions that are re-sampled by EdiBERT is done with latent_mask$_2$. 2) At the second epoch, we randomize the token value, at the position that is being replaced. This is only done for image inpainting.

D.1.2.2 Image compositing.

We set the number of epochs to 2, collage frequency to 4 per epoch, top-k sampling to 100, dilation to 1, and number of optimization steps to 200. We apply a gaussian filter on the mask for the periodic image collage. Contrarily to inpainting, we do not randomize such that EdiBERT samples stay closer to the original sequence.

The full algorithm is presented below in Algorithm 7.

D.2 Additional experimental results

We give additional comparisons on FFHQ and LSUN Bedroom, for the following tasks: image inpainting in Table D.1, image crossovers in Table D.2, and image composition in Table D.3. All these experiments are run on the test-set of EdiBERT. Note that concurrent methods based on StyleGAN2 were trained on the full dataset, which advantages them.

Inpainting. We use 2500 images. On FFHQ, we provide results for free-form masks and rectangular masks. The height of rectangular masks is drawn uniformly from $[0.4 \times h, 0.6 \times h]$ with $h = 256$, and similarly for the width. For non-rectangular masks generations, we follow the procedure of Chai et al. (2021): we draw a binary mask at low-resolution 6×6 and uspsample it to 256×256 with bilinear interpolation.

The ablation study in Table 4.2 of main paper is performed on free-form masks. Results in Table 4.1 of main paper are on rectangular masks. On LSUN Bedroom, we provide results for rectangular masks.

Algorithm 7: Image inpainting/composition

Requires: Masked (or edited) image i_m, mask m, Encoder E, Decoder D, BERT model p_θ, epochs e, periodic collage c, optimization steps optim_steps;
$s \leftarrow Q(E(i_m))$;
latent_mask $\leftarrow$ get_mask_in_latent_space(m);
if *task is inpainting* **then**
  $s \leftarrow s \times \text{latent\_mask} + \text{rand} \times (1 - \text{latent\_mask})$;
end
for *e in [0,epochs]* **do**
  for *p in chosen_order*$(latent\_mask)$ **do**
    Sample token $t \in Z \sim p_\theta^p(\cdot|s)$;
    Insert sampled token: $s_p \leftarrow t$;
    if *p%c=0 (collage)* **then**
      Encode image post-collage: $s \leftarrow E(i_m \odot m + D(s) \odot (1-m))$;
    end
  end
end
$s^0 \leftarrow s$;
for *i in [0, optim_steps]:* **do**
  $L = L_p\big((D(s) - i_m) \odot m\big) + L_p\big((D(s) - D(s^0)) \odot (1-m)\big)$;
  $s \leftarrow s + \varepsilon * \text{Adam}(\nabla_s L, s)$;
end
Image $\leftarrow$ Decoder(s);
Result: Image

Crossovers. We generate 2500 crossovers from random pairs of images, on both FFHQ and LSUN Bedroom.

Editing/Compositing. We create small datasets of 100 images from the test set of EdiBERT for FFHQ scribble-based editing, FFHQ compositing and LSUN Bedroom compositing. A user study on FFHQ compositing is presented in main paper with statistically significant number of votes. We also provide some metrics in D.3. Because of the small size of the dataset, we only report masked L1 and density. For density, the support of the real distribution is estimated with 2500 real points, and density is averaged over the individual density of the 100 generated images.

Table D.1 Image inpainting.

	Masked L1 ↓	FID ↓	Density ↑	Coverage ↑
FFHQ: rect. masks				
I2SG++ (Abdal et al., 2020)	0.0767	23.6	0.99	0.88
I2SG†++ (Abdal et al., 2020)	0.0763	22.1	**1.25**	0.91
LC (Chai et al., 2021)	0.1027	27.9	1.12	0.84
EdiBERT ($\star$)	0.0290	13.8	1.16	0.98
Co-mod. GAN (Zhao et al., 2020)	**0.0128**	**4.7**	1.24	**0.99**
FFHQ: free-form masks				
I2SG++ (Abdal et al., 2020)	0.0440	22.3	0.92	0.89
I2SG†++ (Abdal et al., 2020)	0.0435	21.1	1.17	0.91
LC (Chai et al., 2021)	0.0620	27.9	1.22	0.85
EdiBERT ($\star$)	0.0201	19.4	1.14	0.96
Com-GAN (Zhao et al., 2020)	**0.0086**	**10.3**	**1.42**	**1.00**
LSUN Bedroom: rect. masks				
I2SG (Abdal et al., 2019)	0.1125	50.2	0.04	0.04
MaskGIT (Chang et al., 2022)	**0.0209**	11.4	**1.09**	0.96
EdiBERT ($\star$)	0.0288	**11.3**	0.89	**0.97**

Table D.2 Image crossover.

	Masked L1 ↓	FID ↓	Density ↑	Coverage ↑
FFHQ				
I2SG++ (Abdal et al., 2020)	0.1141	29.4	0.97	0.78
I2SG†++ (Abdal et al., 2020)	0.1133	26.9	**1.35**	0.82
ID-GAN (Zhu et al., 2020)	0.0631	23.2	0.88	0.83
LC (Chai et al., 2021)	0.1491	31.9	1.17	0.77
EdiBERT ($\star$)	**0.0364**	**19.7**	1.05	**0.88**
LSUN Bedroom				
I2SG (Abdal et al., 2019)	0.1123	45.7	0.12	0.20
ID-GAN (Zhu et al., 2020)	0.0682	21.4	0.35	0.57
EdiBERT ($\star$)	**0.0369**	**12.4**	**0.64**	**0.84**

Table D.3 Image editing.

	Masked L1 ↓	Density ↑
FFHQ scribble-edits		
I2SG++ (Abdal et al., 2020)	0.7811	0.91
I2SG†++ (Abdal et al., 2020)	0.0777	1.11
ID-GAN (Zhu et al., 2020)	0.0461	0.79
LC (Chai et al., 2021)	0.1016	**1.14**
EdiBERT ($\star$)	**0.0281**	0.96
FFHQ compositing		
I2SG++ (Abdal et al., 2020)	0.0851	0.77
I2SG†++ (Abdal et al., 2020)	0.0866	**1.07**
ID-GAN (Zhu et al., 2020)	0.0570	0.75
LC (Chai et al., 2021)	0.1116	1.00
EdiBERT ($\star$)	**0.0307**	0.94
LSUN Bedroom compositing		
I2SG (Abdal et al., 2019)	0.1285	0.25
ID-GAN (Zhu et al., 2020)	0.0484	1.45
EdiBERT ($\star$)	**0.0247**	**1.49**

D.3 Baselines

We use the implementation and pre-trained models from the following repositories.

ID-GAN (Zhu et al., 2020): https://github.com/genforce/idinvert_pytorch, which has pre-trained models on FFHQ 256x256 and LSUN Bedroom 256x256.

I2SG++ and I2SG†++(Karras et al., 2020b,a; Abdal et al., 2020): https://github.com/NVlabs/stylegan2-ada-pytorch. We tested projections with the following pre-trained models on FFHQ: StyleGAN2 (Karras et al., 2020b) at resolution 256x256, and StyleGAN2-Ada (Karras et al., 2020a) at resolution FFHQ 1024x1024. For evaluation, we downsample the 1024x1024 generated images to 256x256.

LC (Chai et al., 2021): https://github.com/chail/latent-composition. We use the pre-trained encoder and StyleGAN2 generator, for FFHQ at resolution 1024x1024. For evaluation, we downsample the 1024x1024 generated images to 256x256.

Com-GAN (Zhao et al., 2020): https://github.com/zsyzzsoft/co-mod-gan. We use the pre-trained network for image inpainting on FFHQ at resolution 512x512. We downsample the generated images to 256x256 for evaluation.

MaskGIT (Chang et al., 2022): https://github.com/google-research/maskgit. We use the tokenizer and transformer trained for conditional image generation and editing on ImageNet 256x256. To perform comparisons with EdiBERT on LSUN Bedroom Image Inpainting, we condition the transformer to the ImageNet class '843: studio couch, day bed'.

D.4 Qualitative results on image compositon

We present more examples of image compositions, with image compositing and scribble-based editing on FFHQ and LSUN Bedroom in Figure 4.9, D.1, and D.2.

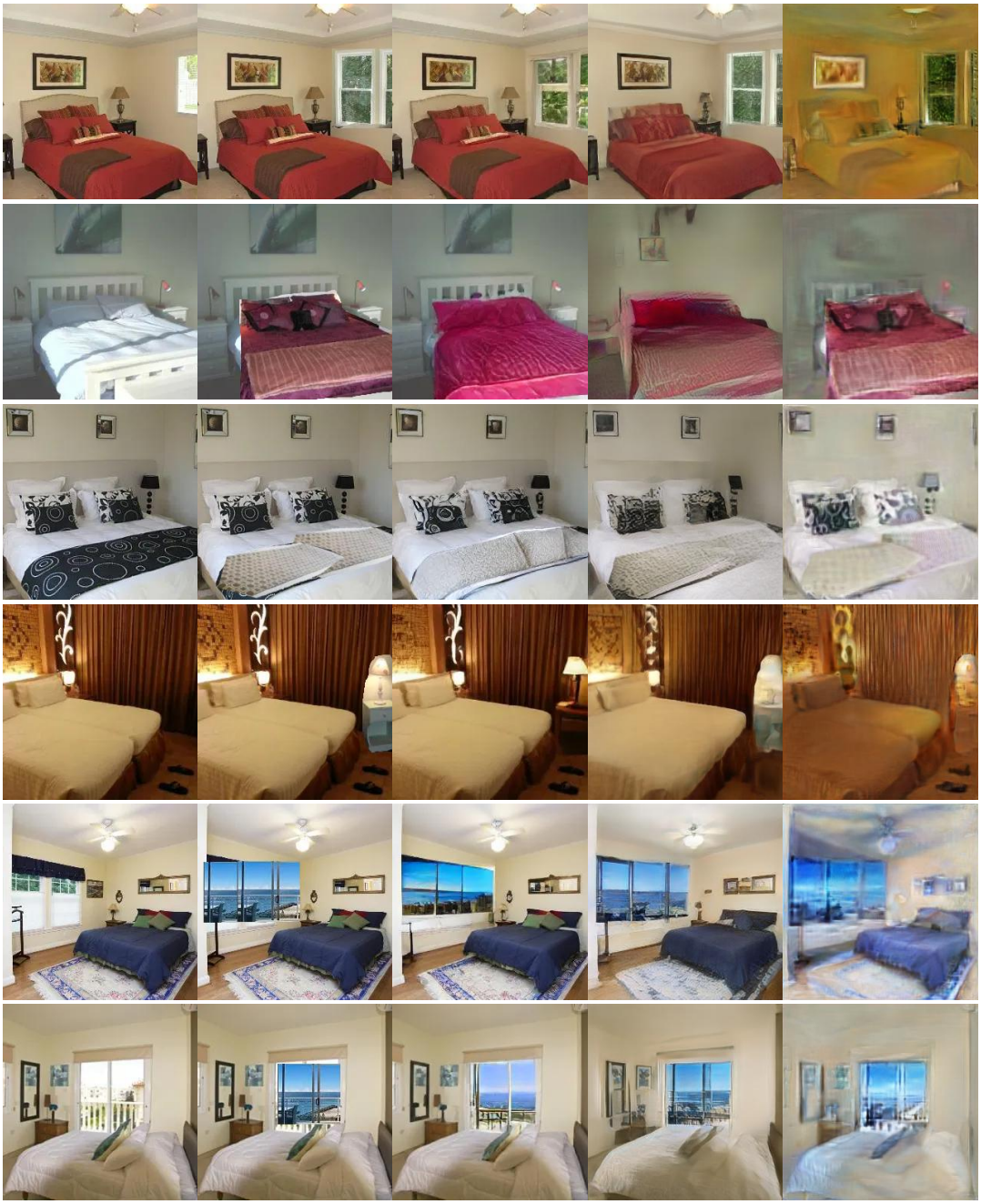
Preservation of non-masked parts. Thanks to its VQGAN auto-encoder, EdiBERT generally better conserves areas outside the mask than GANs inversion methods. This is particularly visible for images with complex backgrounds on FFHQ (Figure D.1, 5th and last rows).

Insertion of edited parts. Since EdiBERT is a probabilistic model and the tokens inside the modified area are resampled, the inserted object can be modified and mapped to a more likely object given the context. It thus generates more realistic images, but can alter the fidelity to the inserted object. For example, on row 1 of Figure D.2, the green becomes lighter and the perspective of the inserted window is improved. Although it can be a downside for image compositing, note that this property is interesting for scribble-based editing, where the scribbles have to be largely transformed to get a realistic image. Contrarily, GANs inversion methods

tend to conserve the inserted object too much, even if it results in a highly unrealistic generated image. We can observe this phenomenon on last row of Figure D.1.

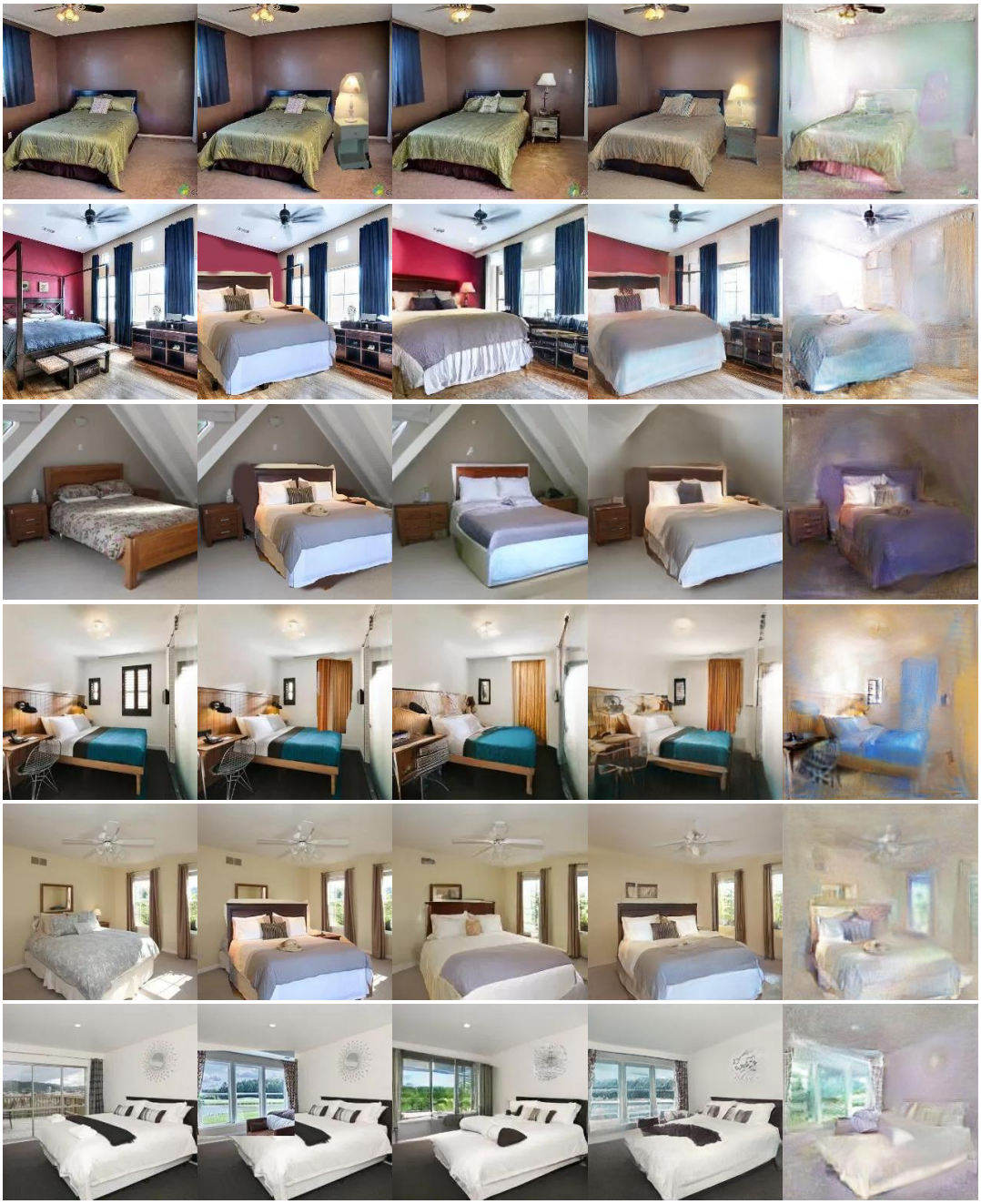
Source Composite EdiBERT ID-GAN I2SG†++

Fig. D.1 Image compositing and scribble-based editing on FFHQ. ID-GAN refers to, while I2SG†++ refers to Abdal et al. (2020) with the backbone from Karras et al. (2020a).



Source Composite EdiBERT ID-GAN I2SG

Fig. D.2 Image compositing on LSUN Bedroom. ID-GAN refers to Zhu et al. (2020), and I2SG to Abdal et al. (2019).



Source Composite EdiBERT ID-GAN I2SG

Fig. D.3 Image compositing on LSUN Bedroom. ID-GAN refers to Zhu et al. (2020), I2SG to Abdal et al. (2019).

Masked EdiBERT MaskGIT (Chang et al., 2022)

Fig. D.4 Comparisons of image inpainting on LSUN Bedroom between EdiBERT and MaskGIT.

D.5 Survey on FFHQ image compositing

The survey was presented as a Google Form with 40 questions. For each question, the user was shown 6 images: Source, Composite, EdiBERT, ID-GAN (Zhu et al., 2020), I2SG†++ (Abdal et al., 2019) based on pre-trained network from Karras et al. (2020a), LC (Chai et al., 2021). The different generated images were referred as Algorithm 1, ..., Algorithm 4. The user was asked to vote for its preferred generated image, by taking into account realism and fidelity criterions. The user had no time limit for the poll. 30 users answered our poll. We provide the detailed answers for each image in Table D.4.

Table D.4 Detailed results of the user study. Each line corresponds to an image, with the associated number of votes per method.

EdiBERT	ID-GAN (Zhu et al., 2020)	LC (Chai et al., 2021)	I2SG†++ (Abdal et al., 2019)
17	5	7	1
15	1	8	6
22	4	1	3
19	4	5	2
22	0	4	4
6	7	8	9
21	1	5	3
23	1	4	2
20	5	5	0
11	13	6	0
27	0	3	0
12	3	3	12
16	4	6	4
25	2	1	2
18	8	1	3
8	13	9	0
26	0	4	0
7	0	21	2
14	9	1	6
27	0	1	2
11	19	0	0
14	9	4	3
16	14	0	0
21	1	3	5
8	2	18	2
19	3	3	5
22	7	0	1
23	2	1	4
18	0	2	10
27	2	1	0
22	2	1	5
24	0	5	1
3	25	2	0
28	0	2	0
24	0	6	0
27	0	3	0
27	1	2	0
22	7	1	0
9	15	6	0
14	0	14	2
Total 735: 61.25%	189: 15.75%	177: 14.75%	99: 0.0825%

Appendix E

E.1 Implementation details

Convolutional geometric matcher. To extract the feature maps, we apply five times one standard convolution layer followed by a 2-strided convolution layer which downsamples the maps. The depth of the feature maps at each scale is (16,32,64,128,256). The correlation map is then computed and feeds a regression network composed of two 2-strided convolution layers, two standard convolution layers and one final fully connected layer predicting a vector $\theta \in \mathbb{R}^{50}$. We use batch normalization Ioffe and Szegedy (2015) and relu activation. The parameters of the two feature maps extractors are not shared.

Siamese U-net generator. We use the same encoder architecture as in the convolutional geometric matcher, but we store the feature maps at each scale. The decoder has an architecture symmetric to the encoder. There are five times one standard convolution layer followed by a 2-strided deconvolution layer which upsamples the feature maps. After a deconvolution, the feature maps are concatenated with the feature maps passed through the skip connections. In the generator, we use instance normalization, which shows better results for image and texture generation Ulyanov et al. (2017), with relu activation.

Discriminator. We adopt the fully convolutional discriminator from Pix2Pix Isola et al. (2017), but with five downsampling layers instead of three in the original version. Each of it is composed of: 2-strided convolution, batch normalization, leaky relu, 1-strided convolution, batch normalization, leaky relu.

Adversarial loss. We use the relativistic formulation of the adversarial loss Jolicoeur-Martineau (2019). In this formulation, the discriminator is trained to predict that real images are more real than synthesized ones, rather than trained to predict that real images are real and synthesized images are synthesized. We also use gradient penalty on the discriminator.

Optimization. We use the Adam optimizer Kingma and Ba (2014) with $\beta_1 = 0.5$, $\beta_2 = 0.999$, a learning rate of $10e^{-3}$ and a batch size of 8. Also, we use $\lambda_p = \lambda_{L1} = \lambda_w = \lambda_{adv} = 1$.

Hardware. We use a NVIDIA Tesla V100 with 16GB of RAM. The training takes around 2 days for T-WUTON, and around 3 days for S-WUTON. For inference, S-WUTON processes $\sim$77 frames per second.

E.2 More visual examples on the importance of distillation

In Figure E.1, we show more visual results proving the soundness of our teacher-student approach. Visually, our student model solves two kinds of problems: it is robust to human parser errors; it preserves important information that is masked to the standard virtual try-ons (hands, skin, handbags).

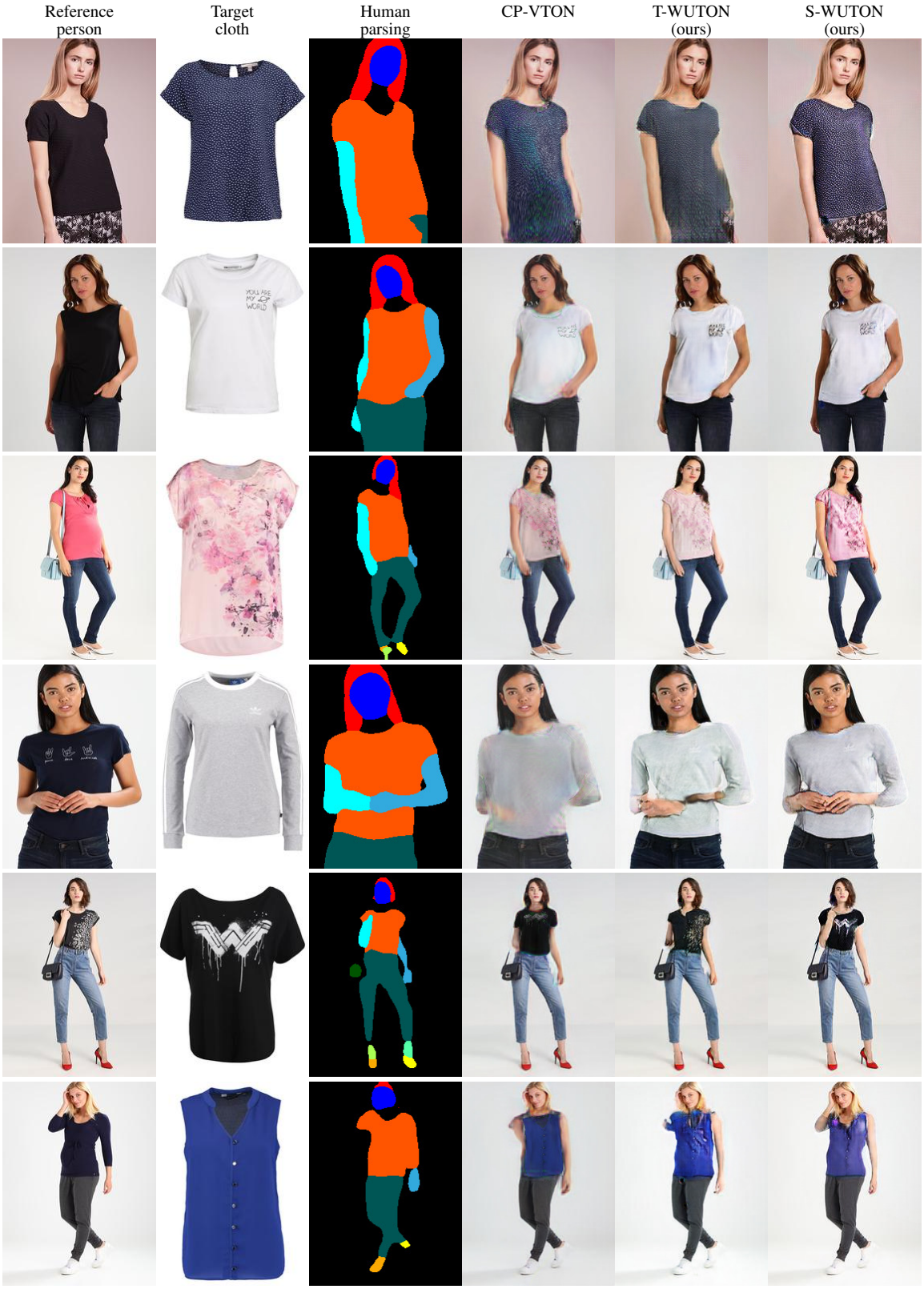


Fig. E.1 Visual results proving the importance of the student-teacher approach. It is robust to parsing errors and preserves person's attributes such as arms, hands, and handbags.

E.3 Ablation studies on T-WUTON

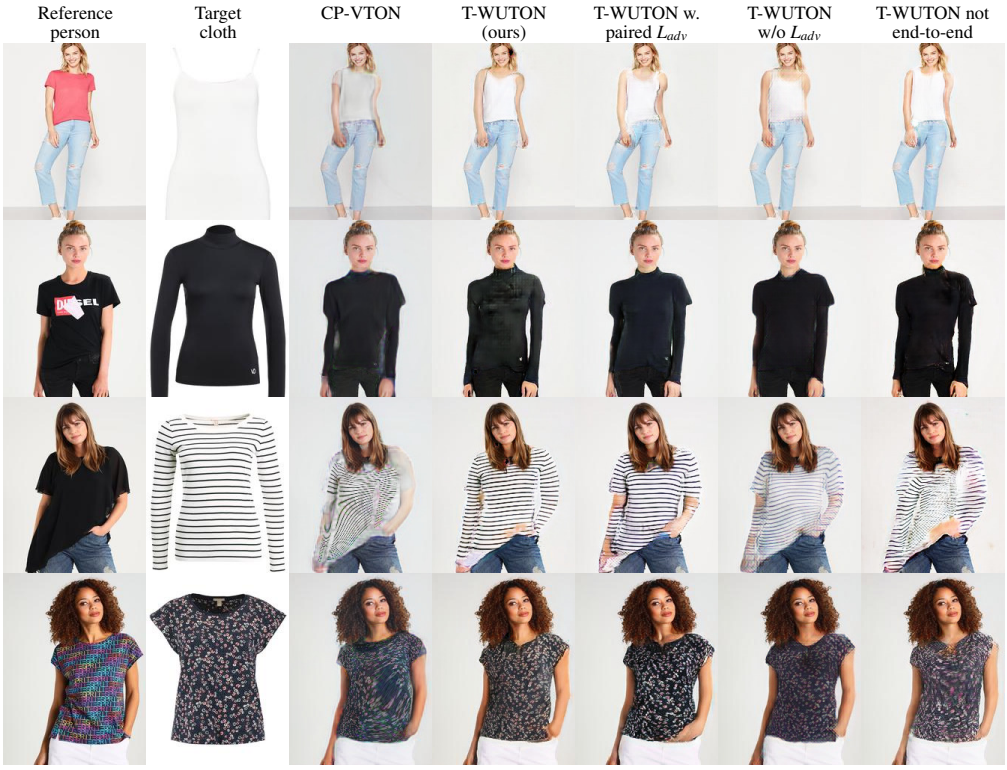


Fig. E.2 Impact of loss functions on T-WUTON: The unpaired adversarial loss function improves the performance of T-WUTON in the case of significant shape changes from the source cloth to the target cloth. Specifically, when going from short sleeves to long sleeves, it tends to gum the shape of the short sleeves. With the paired adversarial loss, we do not observe this phenomenon since the case never happens during training. Without the adversarial loss, images are blurry and less sharp. Finally, the end-to-end training is key to realistic geometric deformations (see last column).

To investigate the effectiveness of T-WUTON's components, we perform several ablation studies. In Figure E.2, we show visual comparisons of different variants of our approach: S-WUTON and T-WUTON; T-WUTON with an adversarial loss on paired data (*i.e.* the adversarial loss is computed with the same synthesized image as the L1 and VGG losses); T-WUTON without the adversarial loss; T-WUTON without back-propagating the loss of the synthesized images ($L_1, L_{perceptual}, L_{adv}$) to the geometric matcher.

The results in Figure E.2 as well as FID and LPIPS metrics in Table E.1 show the importance of our end-to-end learning of geometric deformations. When the geometric matcher only benefits from L_{warp}, it only learns to align c with the masked area in $p^\star$. However, it does not preserve the inner structure of the cloth. Back-propagating the loss computed on the synthesized images $\tilde{p}$ alleviates this issue. The quantitative results of IS and SSIM scores on

Table E.1 Ablation studies on T-WUTON. Quantitative metrics on paired setting (LPIPS and SSIM) and on unpaired setting (IS and FID). For LPIPS and FID, the lower is the better. For SSIM and IS, the higher is the better. $\pm$ reports std. dev.

Method	T-WUTON	W/o L_{adv}	Paired L_{adv}	Not end-to-end
Paired L_{adv}			✓	
Unpaired L_{adv}	✓			✓
End-to-end	✓	✓	✓	
LPIPS	0.101 ± 0.047	0.107 ± 0.049	$\mathbf{0.099} \pm 0.046$	0.112 ± 0.053
SSIM	0.799 ± 0.089	0.799 ± 0.088	$\mathbf{0.800} \pm 0.089$	0.799 ± 0.089
IS	$\mathbf{3.114} \pm 0.118$	2.729 ± 0.091	3.004 ± 0.091	3.102 ± 0.077
FID	9.877	13.020	**8.298**	11.125

the not end-to-end variant show that these metrics are less suited to the virtual try-on task than LPIPS.

The adversarial loss generates sharper images and improves the contrast. This is confirmed by the LPIPS, IS and FID metrics in Table E.1 and with visual results in Figure E.2. With the unpaired adversarial setting, the system better handles large variations between the shape of the cloth worn by the person and the shape of the new cloth. On metrics in the paired setting (LPIPS and SSIM), the best model is the variant using adversarial loss on paired data, which is logical. However, visual investigation suggests that the unpaired adversarial loss is better in the real use case of our work (see Figure E.2).