Controlling image generative models without supervision

Antoine Plumerault

université PARIS-SACLAY

Controlling image generative models without supervision

Contrôle des modèles de génération d'image sans supervision

Thèse de doctorat de l'université Paris-Saclay

École doctorale 573, INTERFACE
Spécialité de doctorat: Informatique
Graduate School : Sciences de l'Ingénierie et des Systèmes (SIS)
Référent : CentraleSupélec

Thèse préparée dans les unités de recherche **MICS (université Paris-Saclay, CentraleSupélec) et LASTI (université Paris-Saclay, CEA)**, sous la direction de Céline HUDELOT, Professeur en informatique, le co-encadrement de Hervé LE BORGNE, chercheur au CEA LIST

Thèse soutenue à Paris-Saclay, le 20 octobre 2023, par

Antoine PLUMERAULT

Composition du jury

Membres du jury avec voix délibérative

Hugues TALBOT Professeur, CentraleSupélec	Président
Rufin VANRULLEN Directeur de Recherche, CNRS, CerCo, ANITI, TMBI, Univ. Toulouse	Rapporteur & Examinateur
Yann GOUSSEAU Professeur, Télécom Paris	Rapporteur & Examinateur
Loïc SIMON Maître de conférences, CNRS-ENSICAEN-Université Caen	Examinateur
Christian WOLF Principal Scientist, Naver Labs Europe	Examinateur

THESE DE DOCTORAT

NNT : 2023UPAST140

Acknowledgments

Tout d'abord, je tiens à exprimer ma profonde gratitude envers mes directeurs de thèse, Céline Hudelot et Hervé Le Borgne, pour leur soutien sans failles tout au long de ces années de recherche doctorale, les précieuses discussions que nous avons pu avoir et leurs conseils éclairés. Je souhaite également adresser mes remerciements sincères aux membres de mon jury pour leur engagement et leurs questions pertinentes. En particulier à mes deux rapporteurs, Yann Gousseau et Rufin VanRullen, pour leur lecture attentive de mon manuscrit de thèse et leurs commentaires constructifs.Je n'oublie pas les personnes que j'ai eu le privilège de côtoyer au fil de ces années, que ce soit dans les couloirs du CEA LIST ou au sein du laboratoire MICS de CentraleSupélec. Les échanges que j'ai pu avoir avec eux ont été précieux pour mon travail. Je tiens à exprimer ma reconnaissance envers le personnel du CEA et de CentraleSupélec, dont le professionnalisme m'a permis de mener cette thèse dans des conditions optimales. Un merci tout particulier à mes parents, qui m'ont fourni tous les moyens nécessaires pour réussir et qui ont toujours soutenu mes choix. Je souhaite également exprimer ma gratitude envers mes amis, avec qui j'ai partagé de nombreux moments mémorables au cours de cette thèse, pour certains aussi lors des années précédentes, et avec qui je passerai encore, j'en suis sûr, de bons moments dans les années à venir. Enfin, un grand merci à Pauline, ma compagne, qui m'accompagne désormais et qui a été d'un grand soutien durant les dernières années de cette thèse.

Resumé

La génération d'images a fait des progrès considérables ces dernières années, au point qu'il devient difficile de distinguer les images fausses des vraies. Ce succès est principalement dû au concept de l'entraînement adversaire : deux modèles s'entraînent pour satisfaire des objectifs contradictoires. L'un, appelé "discriminateur", essaie de distinguer les images du jeu de données des images générées par l'autre modèle, appelé "générateur", qui essaie de tromper le premier. Cette lutte conduit généralement à la génération d'images de haute qualité quasi indécelables des vraies. Cette idée a conduit à diverses applications, de l'augmentation de données aux deep-fakes. Cependant, le contrôle de ces modèles reste une question ouverte. Dans cet travail, nous proposons trois approches qui abordent ce problème. L'une d'entre elles est basée sur l'exploration de la représentation latente de ces modèles. Elle a été publiée à ICLR 2020. Notre deuxième travail vise à combiner deux types de modèles génératifs tout en conservant leurs forces respectives : GAN (Generative Adversarial Network) et VAE (Variational Auto Encoder). Les VAE sont mieux compris, leur entraînement est généralement plus facile que les GAN et leur nature d'autoencodeur est utile pour contrôler leur processus de génération. Ce deuxième travail a été accepté à ICPR 2020. Enfin, notre dernier travail présente un cadre pour générer des scènes contenant plusieurs objets avec un contrôle explicite sur les propriétés géométriques des objets sans données labelisées.

Abstract

Image generation has made substantial progress in recent years to the point where it becomes difficult to distinguish fake images from real ones. This success is mainly due to the concept of adversarial training, where two models are trained to satisfy conflicting objectives. One called the "discriminator" tries to distinguish the dataset images from images generated by the other model called the "generator" which tries to fool the first one. This fight usually leads to the generation of high-quality images nearly indistinguishable from real ones. This idea has led to various applications from data augmentation to deep-fakes. However, controlling these models remains an open question. In this work, we propose three approaches that address this problem. One is based on the exploration of the latent representation of these models. It has been published at ICLR 2020. Our second work aims to combine two kinds of generative models while keeping their respective strengths: GAN (Generative Adversarial Network) and VAE (Variational Auto-Encoder). VAEs are better understood, their training is usually easier than GANs, and their auto-encoder nature is useful to control their generative process. This second work has been accepted at ICPR 2020. Finally, our last work presents a framework to generate scenes containing multiple objects with explicit control of objects' geometric properties without labeled data.

Keywords: Control, generative models, image generation, adversarial learning, VAE, interpretability

List of Tables

List of Figures

Contents

Chapter 1

Introduction

1.1 Artificial intelligence

Artificial intelligence is a relatively old domain dating back from 1936 when Alan Turing presented the abstract system today known as a “Turing Machine” [Turing, 1936] showing that it is possible to create machines capable of doing any calculation and thus capable of simulating decisions or perceptions. Such a machine could in theory simulate a mind, while the program to do so is yet to be determined. As early as 1943, Warren S. McCulloch and Walter Pitts published a paper [McCulloch and Pitts, 1990] where they describe a model of artificial neurons, creating a parallel view between the brain and computers and posing the fundamental question posed by Turing in 1948 of "whether it is possible for machinery to show intelligent behavior" [Turing, 2004].

With the increasing computing power, the availability of massive data and advances in both mathematics and computer science, computers have shown their capacity to perform well in a lot of applications often surpassing humans on some tasks. For instance, in the AI community, challenging AI to human players in strategic games is a common practice to see the evolution of the field. On 11 May 1997 Deep Blue, a computer won against the world chess champion, Garry Kasparov and more recently AlphaGo [Silver et al., 2016] beaten Lee Sedol one of the world’s top players in the game of Go. Another example of a machine surpassing a human is Watson an expert system that in 2011, defeated the two greatest Jeopardy! Champions, Brad Rutter and Ken Jennings. Today, these achievements seem natural as we have accepted that computers are much more capable than humans when the task involves computations or memory and algorithms like Dijkstra, MinMax or A* are well understood. However, despite the success of computers, some aspects of human intelligence seemed to resist machines. For instance, perception, imagination, intuition or body control are problems where writing a good algorithm to solve them is often challenging and manually written algorithms’ performance on modern computers often falls well behind human performance. Paradoxically, tasks on which human often struggle (e.g. calculus, planning, memorization, ...) is what computers are the best at,

while what seems natural and easy to us humans (e.g. perception, intuition, body control, ...) is what computers seem to struggle with. Hence, these fields are now at the heart of AI Research with computer vision, speech recognition systems, generative models (e.g.

generating text or images), and agent behavior in an environment. Today most of these fields have seen rapid progress with the use of Machine Learning.

1.1.1 Machine learning

This difficulty of finding the right algorithm to solve some specific task has motivated the development of *Machine Learning* or ML [Samuel, 1959]. Machine learning is the domain interested in making programs that can adapt themselves to solve specific tasks by being exposed to data (see Fig 1.1). It contrasts with classical programming in that the logic of the program is not written directly but learned during a *training* phase. The role of the ML practitioner often consists in defining a family of models before using data to select a model inside this family which performs well on the target task. A wide variety of families have been designed together with techniques to choose the right model for these families. Often we have a parametrized model whose parameters are yet to be chosen (e.g. an artificial neural network). This final choice of parameters is done using optimization techniques or analytic solutions to minimize some errors of the model on some training data with the hope that the model performance will still be acceptable on data outside the data used for training. These approaches have been used for example to build classifiers or other systems where hand-tuning model parameters is difficult. Formally, we define a function f_θ parametrized by θ a set of parameters this function is called the *model*. We also need some training dataset $\mathcal{D} = x^1, ...x^N$ containing a number N of examples x^i that will help us determine a 'good' θ. Here we voluntarily do not specify the domains of the function f_θ nor what kind of object are the examples x^i as the framework of machine learning is aimed to be very general. To find a 'good' θ we first have to specify a way to describe what we mean by 'good'. This is the role of the loss function $\mathcal{L}(\theta, \mathcal{D})$ which encodes the desirable properties of our model. Choosing this function is often a critical task for the ML practitioner. We then use an optimization algorithm to find a θ which minimizes the loss this step is referred to as the *training phase*. Often such optimization algorithms rely on gradient descent, but other techniques can be used like reinforcement learning or genetic algorithms. Once a 'good' θ is found the model can be used with new inputs as a regular function. One key assumption - that can be experimentally verified to some extent - is that the model will generalize well, i.e. will perform well on new samples not present in the training dataset.

1.1.2 Deep learning

Deep learning is a branch of machine learning. Deep learning designates machine learning methods that use deep models, that is to say, models that perform a long/deep sequence of simple parametrizable computations. Deep models are often artificial neural networks, sequences of parametrized linear transformations and nonlinear transformations that can approximate a wide variety of functions if big enough. However, training artificial neural network models is a challenging problem. Indeed, today we know that they often require large amounts of training data and powerful hardware to achieve the desired performance. More importantly, training a neural network is not trivial and many important ideas were necessary to efficiently train such models. We can cite [Hinton et al., 2006] which introduced an initialization procedure to allow the training of deep models by progressively training layers

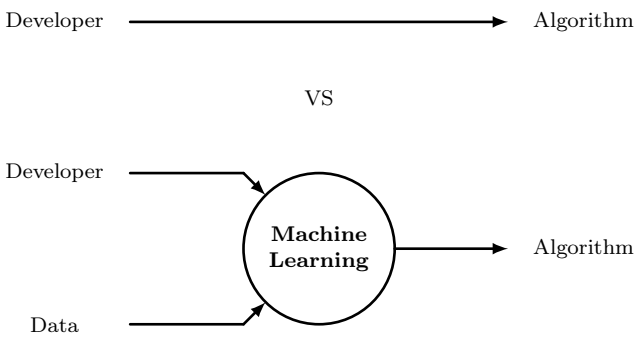

Figure 1.1: Comparison of classical programming versus machine learning. Machine learning leverages large amounts of data to produce an algorithm that solve a problem while classically only the skills of the programmer are used to design the algorithm

of the model, [Ioffe and Szegedy, 2015] which introduced batch normalization: a method to solve initialization issues and speed up training. Through the years new optimization techniques [Zeiler, 2012; Tieleman et al., 2012; Kingma and Ba, 2014], regularization methods [Srivastava et al., 2014; Foret et al., 2020] and neural network modules like adaptive instance normalization [Huang and Belongie, 2017] or LSTM (Long Short Term Memory) [Hochreiter and Schmidhuber, 1997] have been proposed to facilitate training or improve performance of deep models. Deep models have been performing surprisingly well, such over-parametrized models were expected to generalize poorly to examples unseen during training. With the development of the field, we have seen neural networks reach and even surpass the classical approaches and human performance. One famous milestone was in 2012, when what is known today as a deep model won the ILSVRC challenge Russakovsky et al. [2015] (a competition of methods to classify images automatically) by a wide margin, shifting further the interest of the computer vision community on this new field. Indeed, an increase in data availability and advance in GPU technology as well as better understanding has allowed the training of large models capable of learning complex tasks. Soon after the interest in deep learning extended to natural language processing, audio analysis, and decision-making to cite only a few. Today the field is expanding rapidly thanks to a wide community of researchers as the potential applications are numerous, and the entry cost is much more affordable than in other domains of research such as physics or robotics. The frontier of what is possible with deep learning is pushed back regularly. The technology while relatively young begins to be sufficiently mature to appear in an increasing number of industries. Here are some examples to only cite a few: Autonomous driving cars (e.g. Tesla and other brands), video-games (e.g. NVIDIA DLSS, AI Sensei), in healthcare (computer vision, imaging), productivity tools (e.g. ChatGPT, GitHub copilot) and even artistic creation with (e.g. stable diffusion Rombach et al. [2021], Midjourney or Aiva).

1.1.3 Image generation

Among all the applications that deep learning has, image generation is a particularly surprising and successful one. Image generation (or synthesis) is the process of artificially generating random images following a chosen distribution. Such distributions can be varied from faces,

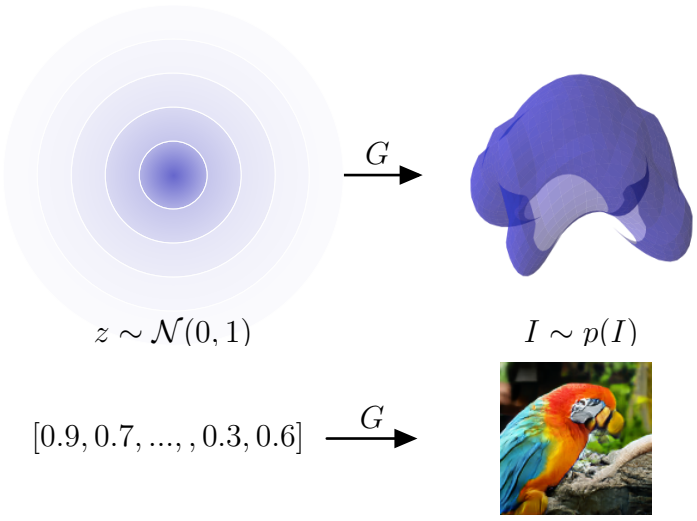

Figure 1.2: This figure shows the process of generation. In the upper-left corner, the latent space is depicted. In the top-right corner, the target distribution is represented. In the bottom-left corner is an example of latent code sampled from the latent space and in the bottom-right corner is the generated image obtained from this latent code. The generator is depicted with the letter G.

and artworks, or can even be conditioned like in super-resolution where we aim at generating high-resolution images given an input low-resolution image. Deep generative models are a kind of model used to generate data. A trained generative model acts as a random number generator but for different kinds of objects like images, text, or audio samples: A user calls the model, and it outputs a random sample. The goal of the generator is to mimic the dataset's underlying distribution. It works by learning to map vectors sampled from a simple random distribution to the desired space. The space of sampled vectors is often called "latent space" and vectors in this space "latent codes" (see Fig 1.1.3). We will use this terminology in this document. Once the model is trained, generating a new sample is as simple as sampling a vector from our simple random distribution and using it as an input for our generator model. Generative models could be used by artists, content creators, video games for asset generation or super-resolution and already used in industries (e.g. NVIDIA DLSS, stable diffusion Rombach et al. [2021] or Midjourney).

However, learning an arbitrary distribution is a challenging task. Distributions of natural images are especially complex as pixels are correlated in subtle ways. Indeed, close pixels have often similar values but can also exhibit correlations with far-away pixels. For instance, the two eyes of most people are of the same color. One solution to this problem is to use rendering using 3d models and light simulation with random scenes. However, such an approach needs a lot of work to design 3d models and choose the distribution of the scene parameters, and even after this work is done, the result may not be realistic enough. The field of differentiable rendering [Kato et al., 2020] could be promising to bridge the gap between classical rendering algorithms and deep learning, however, may not be suited to all the applications of image generative models. Thus, we can understand how automated

approaches that can learn to reproduce a distribution of images can be of interest. Although the task can seem complex, early deep image generation models were able to learn complex image distributions [Kingma and Welling, 2014]. Nevertheless, they often lacked the realism and sharpness of natural images. The development of GANs (Generative Adversarial Networks) [Goodfellow et al., 2014] enabled generative models to achieve such a performance that it is hard for a human being to distinguish a real image from a fake generated one. GANs [Goodfellow et al., 2014] are trained using two models one generator and one critic, the critic learns to distinguish real from fake images and "teaches" the generator model to produce more realistic images (see Fig 1.3). The whole training is thus done without the supervision of a human being but converges to a solution that can fool the human eye. Initially, such models were difficult to train and rather limited to low-resolution images but research to understand the training behavior and evolution in artificial neural network architectures have led to substantial improvements: Radford et al. [2015] used convolutional neural network in a GAN the structure of convolutional networks proved to be a powerful prior for learning to model image distribution leading to better results and easier convergence, Karras et al. [2017] proposed to increase image resolution progressively during training, Karras et al. [2018] proposed a novel architecture that improved both quality and controllability of GANs, Donahue et al. [2016] shows that given sufficient computational resources it is possible to generate high-resolution images of 1000 classes. More recently, Ramesh et al. [2021] pushed the limit further by training a very large model (12 billion parameters) to generate images from arbitrary text queries. One of the key strengths of generative models is their ability to learn from not annotated data. Indeed, by contrast to other deep learning fields like image recognition which require both images and labels, generative models only need images to be trained. Today, generative models will play an increasing role as their potential applications are far-reaching. Generative models have already made their way in the industry, the art world (e.g. Portrait of Edmond Bellamy, Memories of Passersby) and the news. For example, deep-fake (see [Tolosana et al., 2020] for a survey on the topic) technology uses generative models to copy-paste the face of a person on another one realistically. NVIDIA DLSS technology reduces scene rendering computational cost by generating some pixels instead of rendering them. However, as with numerous other technologies, generative models are a double-edged sword. Indeed, powerful models have the potential to fool human beings. As a consequence, they are a weapon of choice to produce convincing bots, fake documents or images to support ideological claims (e.g. article denouncing the use of fake images to propagate fake news) or to write large numbers of ideological articles (e.g. article denouncing the use of text generation models to write articles). That makes the problem of trust in our digital world increasingly concerning.

1.2 Controlling image generation

Recent deep generative models can produce photo-realistic images, sounds, or texts useful to address various tasks in computer vision or other domains. Their usefulness is nevertheless often limited by the lack of control over the generative process caused by the opacity of the learned representation. Indeed, as we have seen, a new sample is generated from a randomly sampled vector and the semantic mapping between vectors and images is not known a priori.

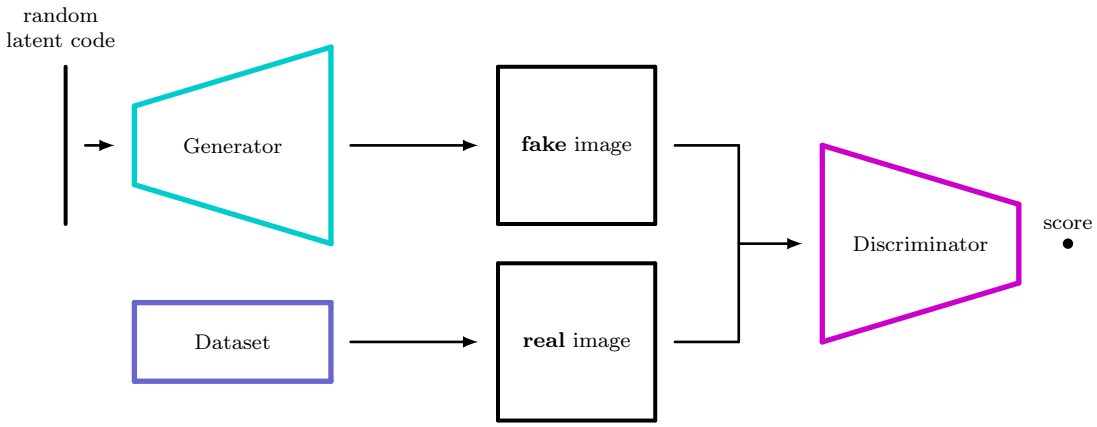

Figure 1.3: Illustration of the GAN framework. The generator produces images from latent codes sampled randomly from a known distribution and the discriminator tries to discriminate real images sampled from the dataset from fake ones generated by the generator.

A first approach would be to add supervision to the generative model to create a chosen correspondence between its inputs and some characteristics of its outputs. This idea has been explored in [Odena et al., 2017] where a model is trained to generate images whose class is conditioned by a user-inputted value. Conditional generation is also achieved in [Ramesh et al., 2021] by training a model on a multi-modal distribution containing images associated with descriptions. More exotic image generation models can also be designed with control as a primary feature, [Tolosana et al., 2020; Hao et al., 2021; Zhang et al., 2016a], but they often need some supervision.

Control over the generative process is in general desirable. Indeed, such models could be used by artists who may not be satisfied with a randomly sampled result and do not want to sample randomly thousands of images to find the right one. It could also be used to generate images from labels to generate labeled synthetic datasets. One way to bring control is to give the user the ability to move through the manifold of an image distribution with some guidance. This manifold may be of high dimensionality but finding a semantic structure in it is the key to choosing properties of generated samples and exploring efficiently possible variations. Being able to control a generative model can also give insights into how the model can generate a wide variety of images and can be helpful to get information on the dataset distribution. Indeed, given the structure of the latent space, it is possible to estimate the distribution of some properties of the image in the training dataset as we have chosen and therefore know the sampling distribution of the latent space that leads to reproducing the dataset distribution. Pushing control even further we could imagine creating animated images by moving in the latent space to make an avatar speak for instance. Control may also be useful to combine generative models with other methods. For example, generative models can be combined with rendering methods to produce more realistic scenes [Hao et al., 2021]. They can also be used to edit captured images or videos [Tolosana et al., 2020; Yeh et al., 2017] or enhance low-resolution images to high resolution by generating details [Wang et al., 2020]. Super-resolution is already used in the industry (e.g. NVIDIA DLSS) and image generation models will likely find an increasing number of applications as their quality and

Figure 1.4: *Edmond de Belamy*, a generative adversarial network portrait painting. The signature on the bottom right of the painting is the formula of the GAN loss used to train the model.

controllability will move forward.

However, when annotated data is not available as in most cases (as they are hard to obtain) such methods can often not be used. Combining supervision with generative models thus comes at a compromise as we lose one key advantage of generative models. Indeed, as researchers we have access to large annotated databases that can benefit the whole community, for example, Imagenet [Russakovsky et al., 2015] and CelebAHQ [Karras et al., 2018] to cite a few. These datasets are great to test ideas and to compare the performance of different approaches on a common stage. However, in the real world, engineers are confronted with specific problems with no publicly available dataset which fits their needs. Collecting data and annotating them can be a tedious task that they have to go through to achieve the desired performance on their target problem.

In our work, we propose to tackle this issue by exploring ways to bring more control to image generation models without using labeled data. We aim to generate high-quality images while improving the controllability of the generation process but without relying on often expensive labeled data. We show that some degree of control can be achieved through different complementary means. The methods that we propose are for the most part centered around image generation, however, some ideas are more general and may be extended to other kinds of data. We distinguish three promising approaches by which we can go through to achieve our goals.

1. A mean by which a generative model can be controlled is by inverting the mapping from the image space to the latent space. Indeed, being able to project an image in the latent space could be used to produce variants of the original image by moving in the latent space around this projection. Knowing this inverse mapping could also be used to search for regions corresponding to some attributes. Say we want to generate images

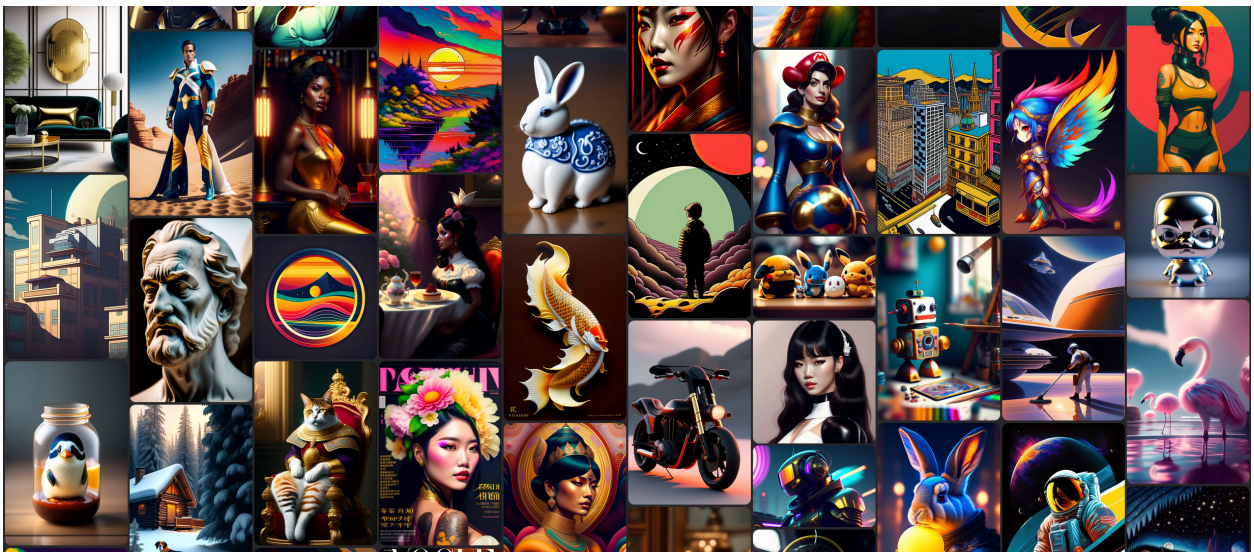

Figure 1.5: Some examples from lexica.art, a website grouping samples generated by users using text-guided latent diffusion models. (see the website for examples of text image pairs and to generate your own images.)

of the face of a desired person, we could use the inverse mapping of the generator to know which parts of the latent space correspond to this person by observing where photos of this person are projected in the latent space. Without knowing the inverse mapping of the generator, this task can be near impossible to do manually as latent spaces are in general of very high dimension (often hundreds). Among the wide variety of image generative models, two models stand out: Variational Auto Encoders (VAE) [Kingma and Welling, 2014] and Generative Adversarial Networks (GAN) [Goodfellow et al., 2014]. GANs can produce realistic images, but they suffer from mode collapse, a pathological event during training that causes the model to only produce a bunch of different images of often low quality and does not provide simple ways to get the latent representation of an image. On the other hand, VAEs do not have these problems, but they often produce less realistic results than GANs. VAEs are thus providing an inverse mapping that brings a form of control but at the price of image realism. Furthermore, VAEs have been shown to exhibit interesting disentanglement properties of their latent space [Chen et al., 2018; Higgins et al., 2017]. Disentanglement is a property of the latent space that is achieved when moving along some dimensions of the latent space induces a modification of a property of the generated image without affecting the other properties of this image. Disentanglement is desirable because it potentially improves the interpretability and controllability of the model. In the case of models containing an encoder from image to latent, the encoder can also be used to produce compact representations that can be used for downstream tasks without the need for supervision. The question is thus: **How to design a model combining both the controllability of VAEs and the quality of GANs ?**

2. Generative models are essentially an approximate mapping between the latent space and the data manifold. Thus, very recent works have shown an interest in studying the semantics of the latent space of generative models [Goetschalckx et al., 2019b; Jahanian

et al., 2020]. Indeed, in the absence of supervision to structure it, the latent space is not directly interpretable: a user is not able to predict what properties the generated image will have until the generation is performed. However, it has been found that trained generative models can exhibit a semantic organization of their latent space [Radford et al., 2015] with a vector space structure that matches the semantic structure. For instance, given a model which generates faces, if we found three vectors corresponding to the generation of the faces of a man, a woman and a man wearing glasses, performing the operation "man with glasses - man + woman" will give us a vector corresponding to a woman wearing glasses (See Fig 1.6). With such a vector space structure, we can hope to find axes in the latent space that are describing a property of the image (for example the axis: wearing glasses $\leftrightarrow$ not wearing glasses). Some model architectures like the one proposed in [Karras et al., 2018] even show out of the box some level of latent space controllability by separating it into parts, each assigned to the generation of details of a particular scale (see Fig 1.7). Understanding this organization in detail could allow the user to choose vectors in this space that lead to the desired semantic attributes of the generated samples. The question is thus **how to uncover this structure or part of this structure in an automated way without the use of annotated data**.

3. Another approach that can be explored is the design of explicitly controllable models. By carefully choosing the model architecture and training procedure, a model can be forced to "act" in an interpretable and controllable way. A particularly good example of such an approach can be found in the voice conversion literature [Chou et al., 2019] where the authors designed a VAE architecture that acts on voice segments. The framework contains two encoders and a unique decoder. However, the encoders' architectures differ: one is better fitted to capture local information while the other is better fitted to capture global information. The decoder takes the outputs of the two encoders to reconstruct the voice segment by using the outputs of the encoders as local and global information. By choosing this architecture the authors were able to separate the information of the content of an utterance with the identity of the speaker allowing control of the speaker's identity independently of the content of the utterance all without the need for annotated data. We can thus ask the following questions: **Could Similar approaches be used for image generation? Is a careful choice of model architecture can force the model into producing an interpretable and thus controllable representation of an image ?**

1.3 Contributions

In this work, we propose some advances in these three directions. In Chapter 3, we first present our contribution to the ability to have a latent code like in the VAE framework [Kingma and Welling, 2014] for models generating images of similar quality to ones produced by a model trained with the GAN framework [Goodfellow et al., 2014]. This achievement was obtained by designing a framework that includes a mapping from image space to latent space while preserving the quality of GAN based model. In particular, we explain why this lack

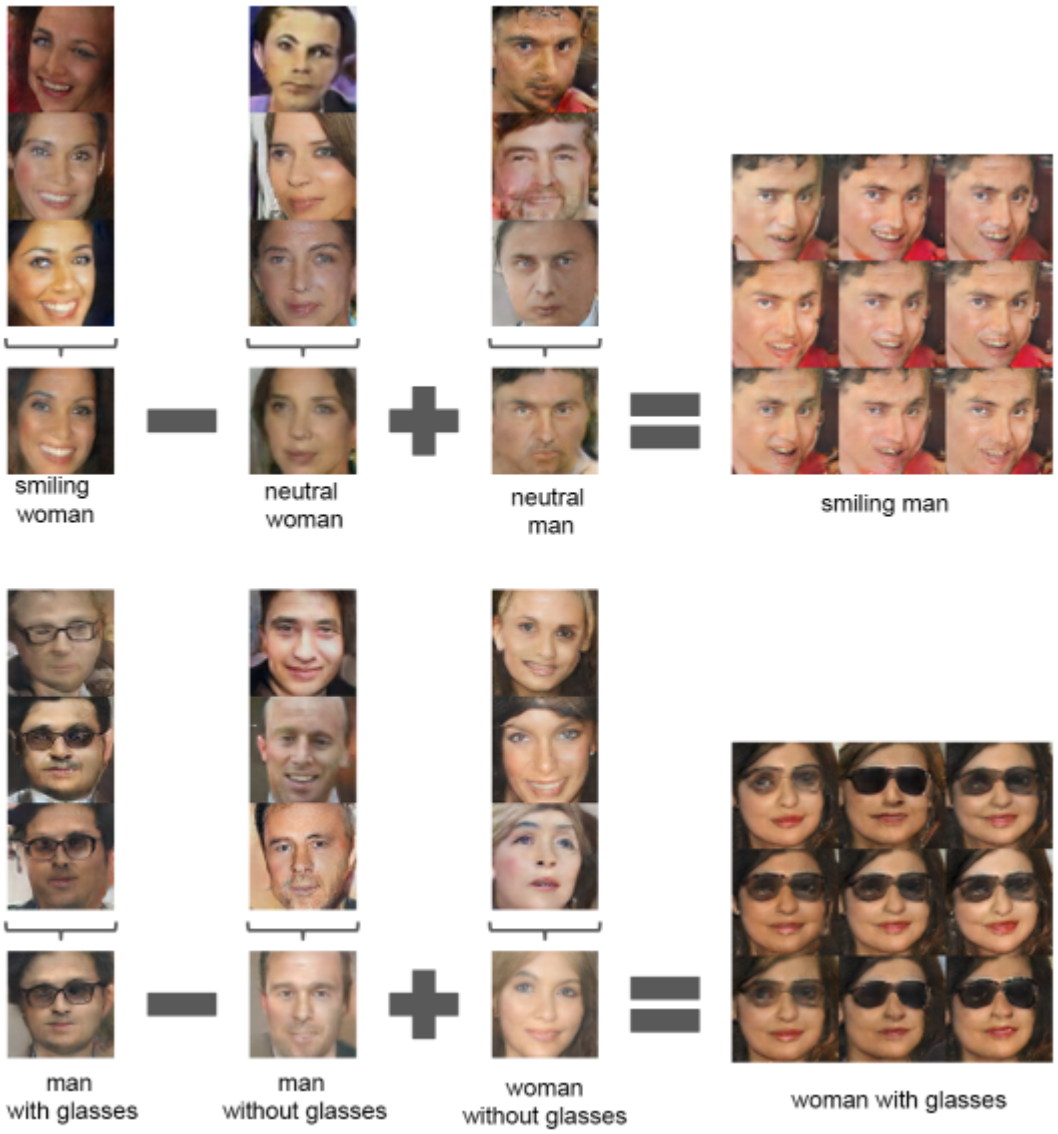

Figure 1.6: Figure and caption taken from Radford et al. [2015]. Vector arithmetic for visual concepts. For each column, the latent codes of samples are averaged. Arithmetic was then performed on the mean vectors creating a new vector Y. The center sample on the right-hand side is produced by feeding Y as input to the generator. To demonstrate the interpolation capabilities of the generator, uniform noise sampled with scale ± 0.25 was added to Y to produce the 8 other samples.

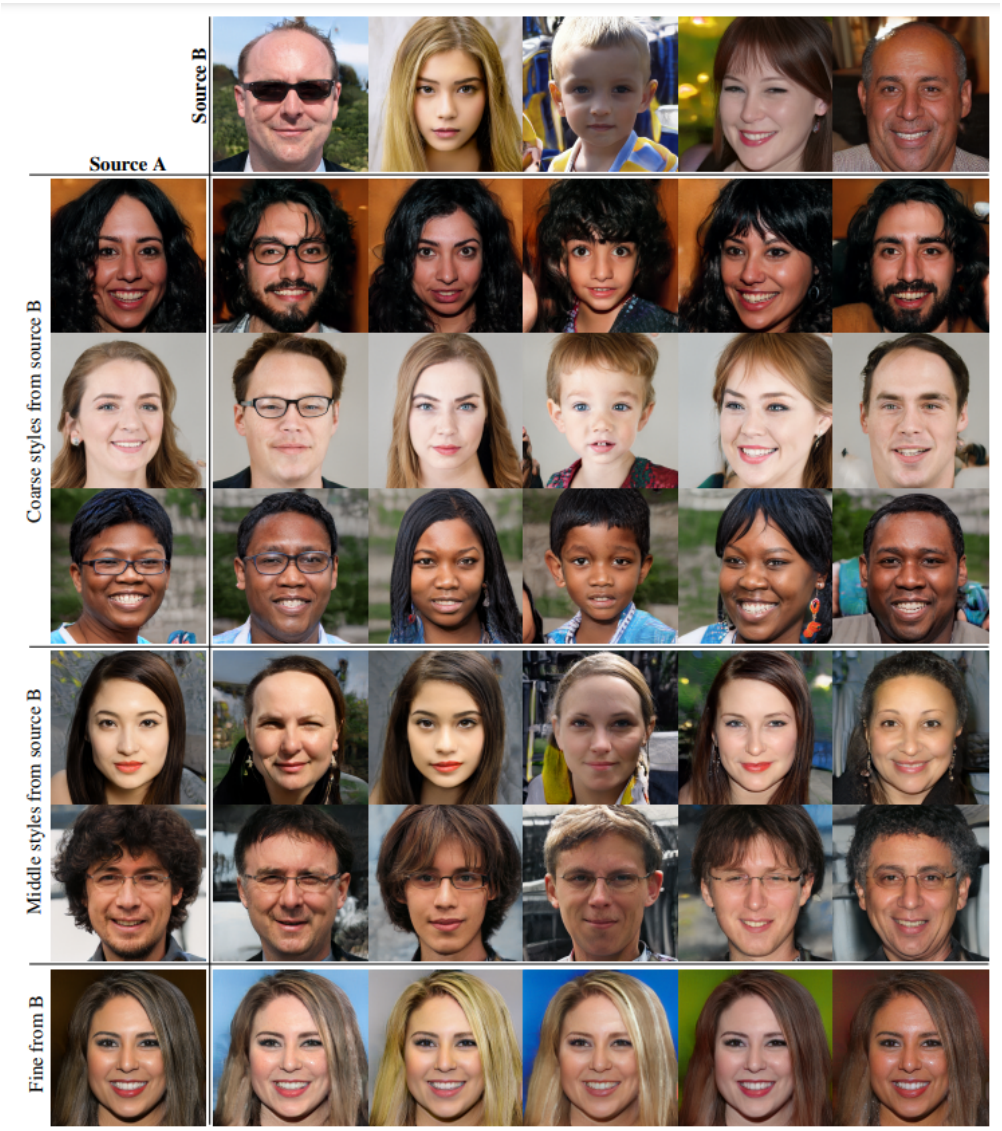

Figure 1.7: Figure and caption taken from Karras et al. [2018]: Two sets of images were generated from their respective latent codes (sources A and B); the rest of the images were generated by copying a specified subset of styles from source B and taking the rest from source A. Copying the styles corresponding to coarse spatial resolutions (4^2 – 8^2) brings high-level aspects such as pose, general hairstyle, face shape, and eyeglasses from source B, while all colors (eyes, hair, lighting) and finer facial features resemble A. If we instead copy the styles of middle resolutions (16^2 – 32^2) from B, we inherit smaller scale facial features, hairstyle, amd eyes open/closed from B, while the pose, general face shape, and eyeglasses from A are preserved. Finally, copying the fine styles (64^2 – 1024^2) from B brings mainly the color scheme and microstructure.

of realism of VAEs is partially due to a common underestimation of the natural image manifold dimensionality. Indeed, while the natural image manifold can often be approximated with a low dimensionality manifold, this approximation does not take into account complex textures which are described by many independent variables and thus are hard to encode in a latent space of relatively low dimensionality. We show that this underestimation of the manifold dimensionality combined with pixel-wise reconstruction errors leads to texture-less reconstruction explaining the low quality of VAE-generated images. From this observation, we introduce a new framework that combines VAE and GAN in a novel and complementary way to produce an auto-encoding model that keeps VAE properties while generating samples of GAN quality. We theoretically motivated our approach and validated it on several datasets. Our proposed framework not only provides an encoder, but it is also trained using ingredients from the VAE framework that make it benefits from the disentanglement properties of the latter.

In Chapter 4, we make an advance on the second direction which is the interpretability of the latent space structure in particular on finding a semantic cartography of the latent space. We provide ways to bring more control over the properties of generated images. We introduce a new method to find meaningful directions in the latent space of any generative model along which we can move to control precisely specific properties of the generated image like the position or scale of the object in it. Our method is centered around what we call "continuous factors of variations", properties that can be gradually modified. It also differs from preceding works in that it does not require human annotations and is particularly well suited for the search of directions encoding simple transformations of the generated image, such as translation, zoom, or color variations. Overall, we show that while some degree of control can be achieved, the more disentangled the latent space is the better our method works.

In Chapter 5, we finally tackle the possibility of creating image generation models both interpretable and controllable by design. In the continuity of existing works, we propose a multi-object deep model which uses explicit objects' geometric properties allowing direct control over these aspects of the image. The model uses geometric transformation operators to extract objects, uses an auto-encoder on these objects and projects their reconstruction back. The model thus learns to decompose a scene and reconstructs it. Our method aims to improve the performance of existing approaches by proposing a hybridization between sequential and parallel object handling. It also brings the generation of realistic objects' geometric properties something that was missing in previously proposed approaches. One important aspect of this work is that here too, no supervision is needed. The model learns by itself how to decompose the scene into multiple objects. This is achieved by a combination of losses and architecture choices which encourages such decomposition without providing labels for the geometric properties of the objects.

While our three contributions can be taken individually, they tackle complementary aspects of image generation control. Bringing encoder to GANs, interpreting the latent space and explicitly describing geometric properties of a scene all without the need for labeled data. All our contributions could be used in a single model in principle. Indeed, if we take the model from our last contribution, non-geometric aspects of the objects are not explicitly encoded, but objects are reconstructed with a VAE. Here, our second model combining GAN and VAE could be used instead of the simple VAE to increase the realism of generated

samples even if it was not needed in our experiments as our dataset was relatively simple and texture-less. We could then use the method described in our first contribution to gain insights into the structure of the objects' latent space to find how color or angle is encoded in the latent space of the object VAE. Ultimately such a model could be used to generate realistic complex scenes containing multiple objects while being able to control the position, scale, rotation and colors of the different objects without the need for any annotated data.

Chapter 2

State-of-the-Art

2.1 Deep image generative models

2.1.1 General framework

The problem of image generation is of particular interest in the deep-learning research community. Researchers have proposed several frameworks and models to solve it. The variational auto-encoder framework [Kingma and Welling, 2014], introduced an efficient way to train models capable of mimicking image distributions. It uses an auto-encoder model with a loss that pushes the distribution of vectors in the intermediate representation to fit a known distribution. An auto-encoder is a model constituted of two parts: An encoder which encodes the signal for instance an image into a compact representation and a decoder which reconstruct the original image from the compact representation, hence the name auto-encoder. Despite their success, VAEs are nevertheless unable to produce realistic samples on a challenging dataset. In the same year, generative adversarial network [Goodfellow et al., 2014] was introduced. They use a different approach: a generator model learns the data distribution concurrently another model called the discriminator tries to predict the probability that a given image originates from the generator or the dataset. The discriminator is used to supervise the generator. This process usually leads to very realistic images. However, GANs are hard to train. The community put a lot of effort to stabilize its training and improve its performance. Techniques and guidelines have been introduced to increase training stability and sample quality over the years [Radford et al., 2015; Salimans et al., 2016; Arjovsky et al., 2017; Miyato et al., 2018; Karras et al., 2017; Zhang et al., 2019] and very successful models have been developed [Brock et al., 2018; Karras et al., 2018, 2019]. Autoregressive models are another approach to the image generation problem. These models are generating images progressively by predicting pixels in a sequence based on the values of the pixels generated previously. Despite receiving less attention due to their longer inference time because of their sequential nature which is difficult to parallelize, they achieve impressive performance [Oord et al., 2016; van den Oord et al., 2016, 2017; Razavi et al., 2019]. One last category of generative models that we want to evoke here is normalizing flow models. They consist of invertible models that map data space to a latent space. The inverse model is then used to perform generation [Dinh et al., 2014, 2016; Kingma and Dhariwal, 2018; Grathwohl et al., 2018]. In this work, we decided to focus mainly on generative adversarial networks and vari-

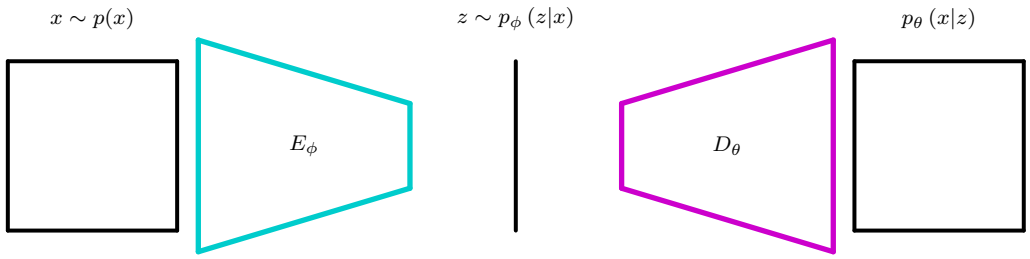

Figure 2.1: Variational Auto Encoder architecture. From left to write, in cyan the encoder model and in magenta the decoder model, in black: the input image, the latent representation and the image reconstructed from the latent code.

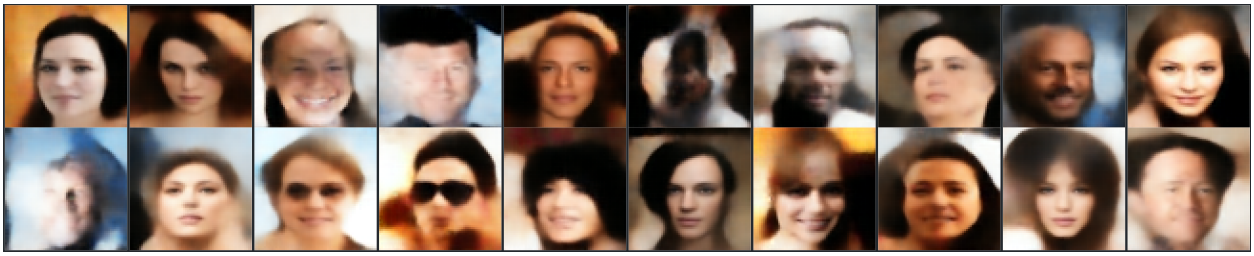

Figure 2.2: Some samples from a small VAE trained on CelebA [Liu et al., 2015] dataset. The images are often blurry for the reasons evoked here.

ational auto-encoder. This choice was motivated by the fact that these models have received the most attention from the community. In our work, we focus on the generative application of such models. We aim to find ways to make these models more controllable while allowing little to no compromises on what makes their strength: the unlabeled nature of training data and the realism of the generated images. In the next to sections we will present the VAE and GAN frameworks in more detail.

2.1.2 Variational Auto-encoders

Variational Auto-Encoder or VAE for short Kingma and Welling [2014] (Figure 2.1) is a framework to learn deep latent variable models. It assumes that observed data X result from random variables $z \sim p(\mathbf{z})$ in a latent space $\mathcal{Z}$ such that it exists a deterministic function $f : (\mathbf{z}, \varepsilon) \to x$, ε being a stochastic noise. The probability of observing x knowing $\mathbf{z}$ is estimated by a *decoder* model $p_{\theta_d} : \mathbf{z} \mapsto p_{\theta_d}(x|\mathbf{z})$ parametrized by θ_d and on the contrary, the probability that $\mathbf{z}$ is the latent source of x is estimated by an *encoder* model $q_{\theta_e} : x \mapsto q_{\theta_e}(\mathbf{z}|x)$ parametrized by θ_e. To estimate the parameters of the generative model of the data $X = (x^{(1)}, ..., x^{(N)})$ with N the number of observed samples, we maximize the log-likelihood of the observations: $\log p_{\theta_d}\left(x^{(i)}\right) = \log \int_{\mathcal{Z}} p_{\theta_d}\left(x^{(i)} \middle| \mathbf{z}\right) p(\mathbf{z}) d\mathbf{z}$. Computing $\log p_{\theta_d}\left(x^{(i)}\right)$ is nevertheless intractable in practice, thus Kingma and Welling [2014] proposes to maximize a tractable lower bound, leading to the following loss to train the VAE:

$$\mathcal{L}_{\text{VAE}}(\theta_e, \theta_d; x) = \underbrace{\mathbb{E}_{q_{\theta_e}(\mathbf{z}|x)}\left[-\log p_{\theta_d}(x|\mathbf{z})\right]}_{\mathcal{L}_{\mathcal{R}}} + \underbrace{\text{KL}\left(q_{\theta_e}(\mathbf{z}|x) \| p(\mathbf{z})\right)}_{\mathcal{L}_{\mathcal{P}}} \tag{2.1}$$

With p_{θ_d} usually chosen as a Gaussian distribution $\mathcal{N}(x; \mu_{\theta_d}(\mathbf{z}), Id)$ and KL the Kullback-

Leibler divergence. Hence, $\mathcal{L}_{\mathcal{R}} = \mathbb{E}_{q_{\theta_e}(\mathbf{z}|x)}\left[-\log p_{\theta_d}(x|\mathbf{z})\right] = \mathbb{E}_{q_{\theta_e}(\mathbf{z}|x)}\left[\frac{1}{2}\left\|\mu_{\theta_d}(\mathbf{z}) - x\right\|^2\right]$ is similar to a mean square error and thus is often interpreted as a *reconstruction* error and is estimated by Monte-Carlo method (usually with a single sample), and the term $\mathcal{L}_{\mathcal{P}} = \mathrm{KL}\left(q_{\theta_e}(\mathbf{z}|x)\|\, p(\mathbf{z})\right)$ forces the distribution of the latent space at the output of the encoder model to match the *prior* $p(\mathbf{z})$. Indeed, if this distribution matches the prior it should be possible to sample vectors in the latent space using the prior distribution to randomly generate samples once the training is done. Usually, $p(\mathbf{z})$ is a standard Gaussian distribution $\mathcal{N}(\mathbf{z}; 0, Id)$). It is also important to notice that the $\mathcal{L}_{\mathcal{P}}$ term acts as an information bottleneck on the latent produced by the encoder. Indeed:

$$
\begin{aligned}
\mathbb{E}\left[\mathcal{L}_{\mathcal{P}}\right] &= \mathbb{E}\left[\mathrm{KL}\left(q_{\theta_e}(\mathbf{z}|x)\,\|p(\mathbf{z})\right)\right] \\
&= \sum_{i=1}^{N} p(x^{(i)}) \int_{\mathcal{Z}} q_{\theta_e}(\mathbf{z}|x^{(i)}) \log \frac{q_{\theta_e}(\mathbf{z}|x^{(i)})}{p(\mathbf{z})} d\mathbf{z} \\
&= \sum_{i=1}^{N} \int_{\mathcal{Z}} p_{\theta_e}(\mathbf{z}, x^{(i)}) \log \frac{p_{\theta_e}(\mathbf{z}, x^{(i)})}{p(x^{(i)})p(\mathbf{z})} d\mathbf{z} \\
&= \sum_{i=1}^{N} \int_{\mathcal{Z}} p_{\theta_e}(\mathbf{z}, x^{(i)}) \log \frac{p_{\theta_e}(\mathbf{z}, x^{(i)})p_{\theta_e}(\mathbf{z})}{p(x^{(i)})p(\mathbf{z})p_{\theta_e}(\mathbf{z})} d\mathbf{z} \\
&= \sum_{i=1}^{N} \int_{\mathcal{Z}} p_{\theta_e}(\mathbf{z}, x^{(i)}) \log \frac{p_{\theta_e}(\mathbf{z}, x^{(i)})}{p(x^{(i)})p_{\theta_e}(\mathbf{z})} d\mathbf{z} \\
&\quad + \int_{\mathcal{Z}} p_{\theta_e}(\mathbf{z}) \log \frac{p_{\theta_e}(\mathbf{z})}{p(\mathbf{z})} d\mathbf{z} \\
&= I_\theta(x; \mathbf{z}) + \mathrm{KL}(p_{\theta_e}(\mathbf{z})||p(\mathbf{z}))
\end{aligned}
\tag{2.2}
$$

With $I_\theta(x; \mathbf{z})$ the mutual information between x and $\mathbf{z}$. This term when minimized thus limits the amount of information about the original image that goes through the latent code and pushes the distribution of the latent code produced by the encoder to match the prior latent code distribution. The framework thus makes a compromise between reconstruction accuracy measured with the MSE and the information that goes through the latent code. This compromise is determined by the choice of p_{θ_d} which is usually chosen to be a Gaussian distribution $\mathcal{N}(x; \mu_{\theta_d}(\mathbf{z}), Id)$. However, changing the variance from Id to $\sigma^2 Id$ still is a valid design choice which has the consequence of weighting the $\mathcal{L}_{\mathcal{R}}$ loss by a factor $1/\sigma^2$, thus changing the balance between reconstruction accuracy and information going through the latent space as well as constraining the latent distribution to match prior distribution. Once the model is trained we can sample images from the distribution $p_{\theta_d}(x) = \int_{\mathcal{Z}} p_{\theta_d}(x|\,\mathbf{z})\, p(\mathbf{z}) d\mathbf{z}$ by sampling first $\mathbf{z} \sim p(z)$ and then $x \sim p_{\theta_d}(x|\,\mathbf{z})$ as an approximation of sampling from the real data distribution $p(x)$.

2.1.3 Generative Adversarial Networks

Generative adversarial network or GAN for short Goodfellow et al. [2014] (Figure 2.3) globally consists in training two neural networks with adversarial objectives to generate samples indistinguishable from the samples taken from the dataset. The *generator* network

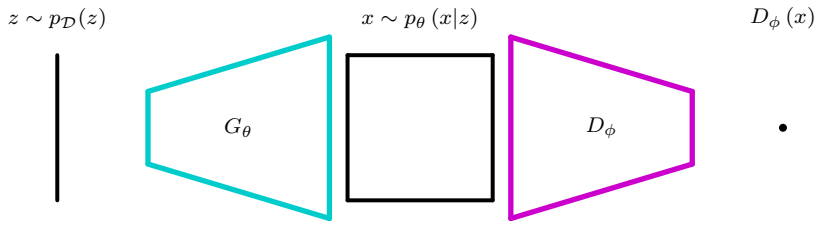

Figure 2.3: Generative Adversarial Network architecture. From left to write, in cyan the generator and in magenta the discriminator, in black: the latent code, the generated image and the score given by the discriminator.

Figure 2.4: Some samples from a small GAN trained on CelebA [Liu et al., 2015] dataset.

parametrized by θ_g is trained to map a random vector to the data space. It is thus very similar to the *decoder* of the VAE framework. The *discriminator* or *critic* network parametrized by θ_c is a classifier that is trained to distinguish real samples from generated ones. The key point is that the generator does not have access to real data and can only improve its parameters through its ability to fool the discriminator. The objective of the critic is:

$$\begin{aligned}\mathcal{O}_C(\theta_c) = &\mathbb{E}_{x\sim p(x)}\left[\log\left(1 - C_{\theta_c}(x)\right)\right] \\ &+ \mathbb{E}_{x\sim p_{\theta_g}(x|\mathbf{z})}\left[\log C_{\theta_c}(x)\right]\end{aligned} \tag{2.3}$$

while the generator tries to fool the critic by minimizing:

$$\mathcal{O}_G(\theta_g) = \mathbb{E}_{x\sim p_{\theta_g}(x|\mathbf{z})}\left[\log\left(1 - C_{\theta_c}(x)\right)\right] \tag{2.4}$$

The loss of the critic is a simple classifier loss, and we will not discuss it a lot. In the loss of the generator, however, it must be noted that it involves the *critic*, the generator is trained to produce adversarial examples for the critic and have access to some supervision of the critic during joint training. Indeed, the gradient of the critic for the image proposed by the generator will indicate what changes in the image will make it more realistic to the eyes of the discriminator in that sense, the gradient of the critic acts as a sort of teacher for the generator. While simple in its principle, adversarial training is often difficult to perform correctly as the whole system can fall into pathological configurations from which it is near impossible to escape and as we have seen the community has put a lot of effort to make it work consistently by avoiding such configurations.

2.1.4 Worth mentioning other image generation models

We mention here some other important kinds of generative models that are present in the literature as they look promising and offer some advantages over VAEs and GANs

Normalizing flow models

Normalizing flow models e.g. [Dinh et al., 2014; Kingma and Dhariwal, 2018] are invertible models build using a sequence of invertible operations. Such models f maps a latent space $\mathcal{Z}$ to data space $\mathcal{X}$ by making sure that if $f(\mathbf{z}) = x$, $p(x) \approx p(\mathbf{z})$ with $p(x)$ the data distribution and $p(\mathbf{z})$ a fixed chosen distribution. Such a model can be thought of as an auto-encoder where the decoder is the exact inverse of the encoder. It has the advantage of being an exact inverse and of not suffering from some limitations of VAEs or GANs. It does not involve an information bottleneck as it is a deterministic function and is much more stable to train than GANs. However, this lack of flexibility is a double-edged sword as such models are often less expressive than GANs and VAEs.

Autoregressive models

Autoregressive e.g. [Oord et al., 2016] models are generative models that construct images sequentially by predicting pixels or patches sequentially with a predictive model. Such models have achieved impressive results but are slower than convolutional models because of their sequential nature involving many calls to the model which predict the value of the next pixel. However, their lack of a usable latent space makes them less attractive to solve the problems raised in this thesis.

Diffusion models

Diffusion models Sohl-Dickstein et al. [2015] is a generative model approach that emerged relatively recently. It has proven very effective for image generation tasks Ho et al. [2020]; Song et al. [2020b]. It works by using a denoiser model to progressively denoise an image until the image is fully denoised. This approach produces samples of quality on par with GAN while being easier to train as it does not suffer from the instabilities of two player game. In addition, It does not suffer from mode collapse and scales well to large models and datasets. The main downside of diffusion models is the time needed for inference as the models need to be called 10 - 50 times to progressively denoise images. Recent works show that reducing this cost is possible using distillation Salimans and Ho [2022] or by performing the costly inverse diffusion process in a latent space of much lower dimensionality than the image space Rombach et al. [2022]. Diffusion models are an active research subject and open-source projects like "stable-diffusion" have democratized the use of such models for artists. The nature of the denoising process allows unique ways to control the generative process making the same model able to perform image generation, in-painting, out-painting, image-to-image translation and more. (See for example https://github.com/AUTOMATIC1111/stable-diffusion-webui an interface to interact with stable-diffusion a latent-diffusion model). Diffusion models that outperform GANs are relatively new Dhariwal and Nichol [2021]. As a consequence, we

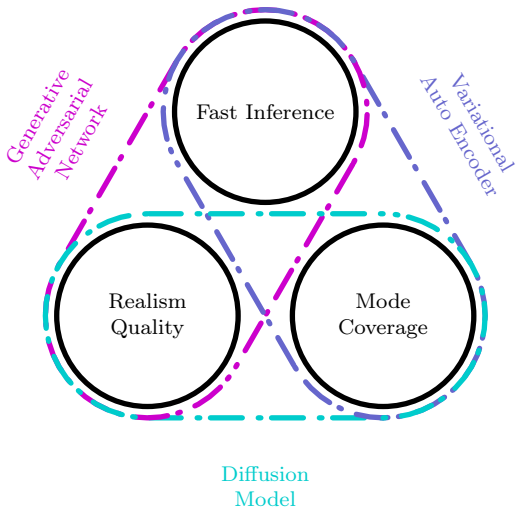

Figure 2.5: Strengths and weaknesses of main generative models.

did not explore their control in this document even if some methods we present in the next chapters may apply to this class of models.

Wrap-up

To wrap up there exist different kinds of generative models for image generation, mainly VAEs, GANs and diffusion models. They have different strengths and weaknesses summarized in Figure 2.5. In addition, VAEs and deterministic diffusion models [Song et al., 2020a] have access to a mapping from image to latent space (in the case of deterministic diffusion models this mapping is the inverse diffusion process). It is also worth noting that VAEs have a usually more disentangled latent space compared to GANs. Concerning diffusion models, to the best of our knowledge, the structure of their latent space has not been studied in detail yet.

2.2 Interpretability and control of the generative process

2.2.1 The latent space

One common attribute of all generative models is the use of latent space. The latent space is a space of usually high dimension in which vectors are sampled using a simple known distribution (e.g. Gaussian distribution). The sampled vectors are then used as the input of the generative model which transforms them into realistic images (See Figure 1.1.3). Ultimately, it serves the purpose of a source of randomness for the generative model. Generative models map the latent space to the data manifold. However, it must be noted that while in most cases the latent space is the input of the generator network and is the space in which we know how to sample, it is possible to consider intermediate representations as the latent space. For instance in [Karras et al., 2018, 2019] a so-called *mapping-network* is first used to map vectors sampled from a Gaussian distribution to a space that can be called the latent

space before the generative model. Moreover, in the case of hierarchical models, the model could take vectors from multiple latent spaces as input. While this diversity of latent space structure exists, we will consider the simplest case of one latent space as it is always possible to concatenate to vectors. In the case of auto-encoders and normalizing flows, the model also learns the inverse mapping from image space to the latent space. Understanding how this latent space is structured semantically is key to being able to sample from generative models in an informed way.

2.2.2 Exploring the latent space of generative models

Getting the latent code of an image

One way to probe the latent space to uncover its structure is to get the latent codes of specific images with properties of interest. However, while it is straightforward for auto-encoders based architectures such as the VAE that possesses an encoder model that does just that, this problem is much harder when using GAN. Indeed, while GAN is arguably the most popular framework for image generation, it does not include an encoder model.

Let's describe more precisely what we mean by "finding the latent code of an image". Given an image $I \in \mathcal{I}$, a generator model G and a latent space $\mathcal{Z}$, the problem is to find its approximate latent code. Indeed, in a real situation, we cannot expect the generator to be able to produce all the images possible and in particular for I. We want to find the approximate latent code $\hat{\mathbf{z}}$ that minimizes a reconstruction error $\mathcal{L}$ between I and $\hat{I} = G(\hat{\mathbf{z}})$ ($\hat{I}$ can be seen as the projection of I on $G(\mathcal{Z})$) i.e. $\hat{\mathbf{z}} = \text{argmin}_{\mathbf{z} \in \mathcal{Z}} \mathcal{L}(I, G(\mathbf{z}))$. Finding such latent code is difficult in practice as the generator network is often highly non-linear and consequently the optimization landscape. Several previous works have studied how to invert the generator of a GAN to find the latent code of an image. Creswell and Bharath [2016] showed on simple datasets (MNIST [Lecun et al., 1998] and Omniglot [Lake et al., 2015]) that this inversion process can be achieved by optimizing the latent code to minimize the reconstruction error between the generated image and the target image. Lipton and Tripathi [2017] introduced tricks to improve the results on a more challenging dataset (CelebA [Liu et al., 2015]). However, we observed that these methods fail when applied to a more complex dataset (ILSVRC [Russakovsky et al., 2015]).

Inverting the generator can have interesting control applications. For instance, [Zhu et al., 2016a] by training a model to predict z from I and then use z as a starting point for the optimization procedure the authors managed to create an application that allows complex editing of images with the help of a GAN. By first finding an approximate latent code of the image to edit and then moving through this latent space while following approximately the human editing while staying on the manifold of natural images. An alternative to learning an approximate inverse mapping of the latent space a posteriori is to learn it jointly. This approach is proposed in [Donahue et al., 2016] which shows that the GAN framework can be applied to both sides by having a classical generator and a second generator that consider images as latent codes and latent codes as samples to generate. More recently, other refinements were proposed to this learning-based approach, including a gain of efficiency to find the latent code thanks to a finer initialization and an iterative process [Guan et al., 2020].

However, other authors showed that a more direct optimization-based approach can be used in practice. In particular, [Creswell and Bharath, 2019] proposed such an optimization-based approach consisting in finding the best possible latent code $z^\star$ that corresponds to an image I by maximizing $\mathbb{E}_x \log(G(z))$ over the latent space with gradient descent. Such an approach goes against the assumption of [Zhu et al., 2016b] who estimated it was not feasible due to the highly non-convex nature of this optimization problem. A reason for the success of [Creswell and Bharath, 2019] may be due to a specific process on batch normalization in the context of GAN inversion, as well as the addition of a kind of regularization on the loss to better take into account the Gaussian nature of the latent space. More recently, Some approaches focus on a specific GAN architecture. It is the case of Image2StyleGAN [Abdal et al., 2019a,b] that aims at finding the latent code in the latent space $\mathcal{W}$ of StyleGAN [Karras et al., 2018]. Starting from a suitable initialization, they look for the latent code that maximizes the similarity between the given image and that generated by the synthesis network. It thus consists in minimizing a loss defined as a weighted combination of the VGG-16 perceptual loss computed by comparing the activation of a VGG network for the two images, and a pixel-wise mean square error. The "suitable initialization" can be a uniform sampling or the mean vector of the latent space, this last corresponding to the region producing images of high quality. More recent works like [Wang et al., 2021] push the idea even further by not only inverting the model to map images to latent space but also making use of annotated data to learn a second mapping between text and the latent space. Ultimately, it allows the user to submit text queries to the image generator.

Uncovering the semantic structure of the latent space

Research has also been conducted to make the latent space of generative models more interpretable. From a model architecture standpoint, conditional GAN [Odena et al., 2017] allows the user to choose the category or some specific properties of the generated images by conditioning the generated classes to some predefined parts of the latent code. While successful, this approach requires a labeled dataset and cannot be used to control a pre-trained model. Following this idea of mapping interpretable inputs (classes in the case of conditional GAN) [Wang et al., 2021] who learn a mapping between text and latent space. This second approach also requires annotated data. However, in this case, the data consists of images associated with generic text rather than fixed classes which can be argued to be easier to collect online.

More generally [Radford et al., 2015] has shown that one can modify a specific attribute of a generated image by adding or subtracting a learned vector on a latent code with the same vector usable on any latent code. This result is of particular interest as it shows that generative models exhibit a semantic vector space structure to some degree. In an ideal case, this structure allows using vector arithmetic on the features of interest, similar to what has been observed with word embeddings in natural language processing: "woman with glasses" = "man with glasses" - "man" + "woman". Understanding this structure could allow very intuitive control of the properties of generated images by finding a "semantic basis" of the latent space.

Based on this observation, several works have focused on approaches that do not require a labeled dataset to bring control to generative models. Combining the latent code of two

images [Karras et al., 2018], showed an impressive ability to mix portrait images (e.g. creating an image that exhibits fine details of a first image but the overall structure of another one) thanks to a hierarchical latent space. Recent works [Goetschalckx et al., 2019a; Jahanian et al., 2019] try to find interpretable axes in the latent space of the BigGAN model [Brock et al., 2018]. Indeed, identifying "semantic" axes allows us to determine some sub-spaces that may correspond to features of interest for the user. With the knowledge of these sub-spaces, the model can be used as if it was a conditional GAN but with the advantage of being unsupervised and thus not needing labeled data.

The importance of disentangled latent space

These approaches often suppose that the latent space is "disentangled" [Bengio et al., 2013] while the degree of disentanglement is delicate to evaluate it is related to the idea that factors of variations that explain an image should be encoded independently of each other in the latent space. For instance, in the case of a latent space embedded in a vector space, we should be able to find a vector basis with coordinates over each axis encoding a specific factor of variation of the final image.

While disentanglement is a desirable property of the latent space, it can be somewhat tricky to guarantee. Indeed, in the universal function approximation, it is trivial to notice that models with entangled latent space could in theory produce images as realistic as models with a disentangled latent space. One could argue that maybe when model capacity is limited, the optimal solution may be the one with a disentangled latent space, this could explain why in our experiments and, in general GANs tends to exhibit fair disentanglement properties even if the loss function does not encourage disentanglement. However, it seems that VAEs tend to exhibit even better disentanglement of their latent space. Consequently, a lot of work has been conducted on the latent representation of VAEs [Higgins et al., 2017; Kim and Mnih, 2018; Burgess et al., 2018; Mathieu et al., 2018] to understand why they exhibit such disentanglement properties and to make them as disentangled as possible. For VAEs, one obvious source of disentanglement is found in the loss and more precisely in the KL divergence term that penalizes the latent space distribution toward matching a fixed one and in the reparameterization trick that forces the mapping between latent and image space to be smooth by introducing a specific noise in the encoder output. However, generative models must also rely on entangled factors of variation to produce realistic images. For example in a face generation application, we expect the ensemble of male characteristics to be highly correlated between them but negatively correlated with female traits as a result, a direction in the latent space will likely define a continuum between male and female faces, but that finer details will be entangled. More generally if two factors of variations are correlated, the directions encoding them in a Gaussian-distributed latent space will not be orthogonal to reflect their correlation and thus will be entangled. To circumvent this problem, [Karras et al., 2019] introduced a secondary latent space which is a non-linear mapping of the latent space in which we randomly sample vectors following a normal distribution. They show that this new latent space is more disentangled than the one from which vectors are sampled from a Gaussian distribution (according to a disentanglement metric that they introduce in the paper).

More disentangled representations are the promise of more interpretable and controllable

generative models. Also, having factors of variations encoded in a factorized way makes it more usable for downstream tasks. Some works also try to disentangle the latent space of GANs. For instance, InfoGan, Chen et al. [2016] shows that adding a secondary latent code to the input of the GAN generator and optimizing with an appropriate regularization term leads to a better disentanglement of the latent space and makes it possible to find a posteriori meaningfully directions. Regarding VAE, Engel et al. [2018] identified that this kind of model exhibits a trade-off between reconstruction accuracy and sample plausibility and proposed to identify regions of the latent space that correspond to plausible samples to improve reconstruction accuracy. They also use conditional reconstruction to control the generative process.

Wrap-up

The latent space plays a fundamental role in deep generative models as it determines which image will be generated. Methods have been developed to invert generative models to get a mapping from image space to latent space. Others have attempted to understand the semantic organization of this space. Researchers also tried to understand how it is structured through the notion of disentanglement, a property of the latent that makes orthogonal directions in the latent space encode independent factors of variations.

2.3 Supervised, Unsupervised and self-supervised control of deep image generation models

Now that we have described the key concepts, behind generative models and their control, we will present the methods that have been proposed by the research community to solve this problem. We distinguish four types of approaches, namely, Supervised, Unsupervised and Self-Supervised approaches and Closed-form solutions. Supervised control methods require the use of annotated data, unsupervised and self-supervised do not.

2.3.1 Supervised control

Early works on GANs explored their latent space representations to better understand their structure and to be able to manipulate the generator. Bau et al. [2018] observed that the generator seemed to learn specific object representations for major scene components. They trained simple models to detect some of these objects from the feature activations and showed that a simple manipulation consisting in setting specific feature maps to zero removes the said object from the generated images. Inversely, by setting the good feature maps to a high value they can force the corresponding object to be present in the generated image. In the paper describing DCGAN (Deep Convolutional GAN) [Radford et al., 2015] the authors showed that the latent space of DCGAN supports semantic vector arithmetic on faces samples, similar to that was found on textual representation a couple of years earlier [Mikolov et al., 2013]. However, to get significant results, it required to average several (3) latent codes. Moreover, if the principle of vector arithmetic was shown on some examples, it did not propose any method that allowed actual control of the generated images. Following

their method to find the latent code of a real image, [Zhu et al., 2016b] proposed a method to allow a naive user to edit the color or some details of an image, at a local scale while staying on the manifold of realistic images. The basic idea is to find the latent code of an image and to move in the latent space in the right direction to produce a chosen local change in the output image. This process produces a sequence of images between the original one and the fully constrained one, that may nevertheless exhibit a degraded quality compared to the original image. Hence, inspired by optical flow methods, they estimate the color and shape changes in the sequence and apply them to the original image to produce a new photo-realistic image with the desired modifications.

In the same vein, [Abdal et al., 2019a] rely on their optimization-based approach that determines the latent code of an image for a StyleGAN model. They define a new latent space $W+$ that is built from the concatenation of many latent vectors of the original W one. They show that within this new latent space, one can perform a linear interpolation between two images (morphing) as well as a transfer from the style of one image to another with better qualitative results than previous methods. More interestingly in terms of control, they also show how to transfer the facial expression of one face to another. For example, given an image of a neutral and smiling face of a person, the method can generate a smiling and angry version of any other face we have with a neutral expression. They extend their approach in [Abdal et al., 2019a] by introducing a "noise space" that allows a better rendering of small details (high frequencies). They also proposed an extension of their $W+$ space that allows local changes of the generated images. A different approach was adopted by [Shen et al., 2020; Shen et al., 2020] who proposed to learn SVMs (Support Vector Machines) in the latent spaces to determine the direction of interest. The method consists in annotating a set of generated images automatically with a set of pre-trained binary attribute predictors. Then, linear SVMs are trained with the latent codes used to generate the images for each attribute, and the found direction is given by the classification hyperplanes' normal vectors. Such a method can thus be potentially applied to any attribute of interest, although it is mainly tested on faces, to identify attributes such as age, gender, presence of eyeglasses or smiling.

Built-in control

Sometimes, control can be directly baked into the model. The most typical example of such a model is conditional GAN [Odena et al., 2017] that we already encountered. But this category is much richer, we list here some examples:

- Super-resolution models which generate high-resolution images from low-resolution images
- Style transfer model allows generating images with a learned style from base images
- Neural rendering also allows the generation of images with constraints on their content.
- Diffusion models that produce spectacular text-to-image results that have recently emerged as a competitor to GANs and autoregressive models.

2.3.2 Closed-form solutions for control

Recent approaches also propose some closed-form solutions to some directions of interest in the latent space, thus allowing some control over the generated images. Härkönen et al. [2020] proposed to apply Principal Component Analysis either in latent space or feature space and showed that the resulting principal vectors correspond to interpretable directions. Hence, moving in the latent space along these directions results in some changes of interest in the generated images. To avoid entanglement on StyleGAN, the authors proposed a layer-wise edition, that is they apply the perturbation to the latent code for some chosen layers only. In the same vein, Shen and Zhou [2021] consider a small change of a latent code, that may correspond to an edition of interest for the generated image, and the resulting output of the first layer of the generator of a CNN-based GAN. Then, they propose an optimization problem that reflects the directions that can cause large variations after the projection of this layer and derive a closed-form solution of this last that results in the wanted direction. They thus propose an unsupervised approach that is independent of data sampling and model training. Spingarn-Eliezer et al. [2020] proposed a method that finds directions in the latent space similar to those of Jahanian et al. [2020] but using a closed-form solution rather than an optimization process. It leads to better-defined directions, that provide images of better quality for the larger geometric transforms.

2.3.3 Unsupervised control and Self-supervised control

To avoid the need for annotated data, Jahanian et al. [2020] proposed a self-supervised scheme that determines directions in the latent space that correspond to features of interest in the generated images. They consider an image and a transformation of it, this last being e.g. a translation, a zoom or a change in luminosity. While Jahanian et al. [2020] directly minimizes an L2 loss to estimate the direction.

Voynov and Babenko [2020] proposed to apply a shift to the latent code by multiplying it by a matrix A, that defines as many potential directions of interest. Then, they learn a "reconstructor" between both images to identify the direction that is easiest to identify. While no semantic a priori is included in the process, the identified direction, or at least some of them, reflects some transformations that are obvious to interpret semantically. An extension of this idea has also been proposed by Cherepkov et al. [2021], where the directions are found in the space of the generator parameters instead of the latent space. They also propose an unsupervised method that finds directions using the singular values of the Hessian matrix of an LPIPS model computed with respect to the parameters of the generator.

Hybrid VAE/GAN approaches

To get the advantages of both GANs and VAEs, Boesen Lindbo Larsen et al. [2015] proposed the VAE/GAN architecture that combines them. They added a discriminator to push the reconstructions from the VAE toward more realism. Their framework resembles the VAE framework in which they incorporated a perceptual similarity metric based on the filters learned by the discriminator instead of the standard reconstruction error. In theory, this approach is problematic because the discriminator's only objective is to predict whether an

image is a real one or a fake one. Thus, the features extracted from it may not be descriptive of image content, which makes them a disputable choice as a base for a similarity metric. Moreover, we noticed that VAE/GAN sometimes fails to reconstruct precisely skin color on the CelebA dataset as this information is probably useless to encode precisely as the discriminator only needs to know if the skin color is in some particular range and thus the discriminator may be blind to some details. If carefully tuned, this approach tends to work well in practice and allows sharper reconstructions. However, Dumoulin et al. [2016] pointed out that it also tends to be a compromise between VAE and GAN and produces less realistic samples than GAN. They propose the BiGAN architecture [Donahue et al., 2016] which is composed of an encoder that transforms real images into latent codes, a generator that translates latent codes sampled randomly into images, and a discriminator which tries to guess the origin of an (image, latent) tuple. While this approach is elegant and produces samples of the same quality as GANs, it put an accent on finding good feature representations in an unsupervised way and often fails to produce very accurate reconstructions. Chen et al. [2017] and Li et al. [2017], propose variations of the BiGAN framework and additional theoretical insights about the latter. They indeed produce more accurate reconstructions in terms of MSE. However, these reconstructions are blurry (no hair texture when trained on face images). That blurriness is precisely one of the issues that we try to tackle in this document as it represents the main part of the quality gap that we observe between images generated by GANs and VAEs. In [Brock et al., 2017], the authors propose a variation of the VAE/GAN framework where the encoder and the discriminator network are combined in a unique model. While it is not clear why this choice is a good one or not, the model reconstruction loss is the combination between a pixel-wise error and the VAE/GAN reconstruction loss. Using a pixel-wise reconstruction loss unfortunately introduces a compromise between the blurriness of the reconstructions and the features' reconstruction fidelity. In the same vein, Huang et al. [2018] has proposed an elegant framework where discrimination is performed at the latent space level but still uses the MSE as a reconstruction error. More recently Pidhorskyi et al. [2020] proposes to keep the GAN structure but to cut both the generator and discriminator in two. The first half of the discriminator outputs a prediction of the intermediate representation located between the two parts of the generator. Concurrently to our work, Lee et al. [2020] have used distillation to transfer the latent space of a VAE to a GAN.

The work presented in Chapter 3 consists of combining the GAN and the VAE frameworks to use the latent space of the VAE as a part of the GAN latent space. Indeed, VAE are known for being able to produce more disentangled latent spaces. Such representations are usually more interpretable. This disentanglement property is also related to our first work as the performance of our method to control GAN generation benefits from more disentangled representations. We focus on providing a framework that allows this transfer of latent space without compromising the quality of samples generated. While Lee et al. [2020] idea and architecture are very similar to ours, we focus mainly on ensuring that we do not introduce compromise between reconstruction accuracy and realism. Our loss differs in this sense. Indeed, they primarily aimed their attention at the disentangled properties of the latent space. Moreover, we justify theoretically why our model can produce realistic images and show results on a different choice of datasets.

Exploring the latent space

Chapter 4 aim is to find interpretable directions in the latent space of any generative models with a focus on GANs while remaining generic as GANs represent the hardest case as the framework does not include an encoder model to map images to latent codes and the disentanglement properties of its latent space are less understood than for VAE latent space. One of our contributions and a part of our global method is a procedure to find the latent representation of an image when an encoder is not available. As we have discussed above, several previous works have studied how to invert the generator of a GAN to find the latent code of an image. Creswell and Bharath [2016] showed on simple datasets (MNIST [Lecun et al., 1998] and Omniglot [Lake et al., 2015]) that this inversion is possible by optimizing the latent code to minimize the reconstruction error between the generated image and the target image. Lipton and Tripathi [2017] introduced tricks to improve the results on a more challenging dataset (CelebA [Liu et al., 2015]). However, we observed that these methods fail when applied to more complex datasets (ILSVRC [Russakovsky et al., 2015]). The reconstruction loss that we introduce is adapted to this particular problem and improves the quality of reconstructions significantly. Plus, we propose a theoretical explanation of why inverting a generative model by optimization is hard compared to other optimization problems. Additionally, in the context of vector space arithmetic in a latent space, White [2016] argues that replacing a linear interpolation with a spherical one reduces the blurriness as well. This work also proposes an algorithmic data augmentation, named "synthetic attributes" to generate images with less noticeable blur with a VAE. Instead, we act directly on the loss. If the method of two concurrent works [Goetschalckx et al., 2019a; Jahanian et al., 2019] published around the same time as our work exhibits similarities with ours (use of transformation, linear trajectories in the latent space), it also differs on several points. From a technical point of view, our training procedure differs in the sense that we first generate a dataset of interesting trajectories and then train our model, while they proceed with training directly. Our evaluation procedure is also more general as we use a saliency model instead of a MobileNet-SSD v1 Liu et al. [2016] trained on specific categories of the ILSVRC dataset. That allowed us to measure performance on a broader set of classes. We also provide additional insight into how this method applies to auto-encoder models, the impact of disentangled representations on the control, and on the structure of the latent space of BigGAN[Brock et al., 2018]. We also propose an alternative reconstruction error to invert generators. However, the main difference we identify between the two works is how the latent space is modeled. We find that our model choice allows for more precise control over the generative process and is more general.

Scene decomposition models

Recently, some approaches have been proposed to tackle the representation of multiple objects scenes Eslami et al. [2016]; Crawford and Pineau [2019]; Burgess et al. [2019]; Greff et al. [2019]; Lin et al. [2020]; Engelcke et al. [2019]. These methods propose two different ways to decompose the scene into a set of objects: Burgess et al. [2019]; Greff et al. [2019]; Engelcke et al. [2019] create a partition of the image using masks. On the other hand Eslami et al. [2016]; Crawford and Pineau [2019]; Lin et al. [2020] uses a region proposal network

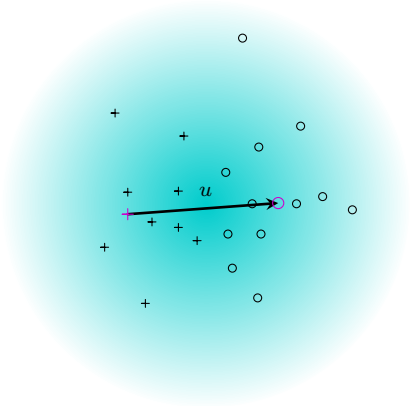

Figure 2.6: A simplified representation of the latent space. In cyan the likelihood of sampling distribution.+ and ∘ symbols represent latent codes for dataset samples with different labels. We can find directions represented by the arrow along which we can move to translate an image of a particular class to another one.

that produces bounding boxes for each object. All these methods except for Crawford and Pineau [2019]; Lin et al. [2020] are using a recurrent neural network (RNN) to produce masks or bounding boxes. However, Crawford and Pineau [2019] has shown that these RNNs do not scale well when the scene contains a lot of objects. To solve this issue, they propose to use a YOLO Redmon et al. [2015] type network, which decomposes the image into a grid of cells, each of them proposing a bounding box. With this approach Crawford and Pineau [2019] and Lin et al. [2020] can tackle scenes with a high number of objects. In Lin et al. [2020] however, the authors explain that SPAIR Crawford and Pineau [2019] needs to process the cells sequentially which hurts performance and scalability, that is why they proposed SPACE Lin et al. [2020] a fully parallel architecture that scales well to scenes with a lot of objects and which proposes to use a separate module for representing the background where SPAIR put this problem aside by testing their model on "scenes" with uniform background.

While this approach is effective it still demands a considerable amount of resources to be trained. Indeed, to be able to consistently find all objects in the image, one may have to divide the image into a lot of cells to have at most one object on each cell as each cell is only able to describe one object. This necessity for a lot of cells is usually not an issue when training for object detection as in this case processing one cell is a low-cost operation. However, in the case of Crawford and Pineau [2019] and Lin et al. [2020], at training time each proposed region has to be passed into a variational auto-encoder Kingma and Welling [2014] which can be very costly (for a 16x16 grid and a batch size of 16 it represents 4096 calls to the VAE !). In chapter 5 we propose to reduce this problem by making each cell capable of proposing multiple regions with the help of an RNN and show that with four times less decomposition capacity (the maximum number of objects that can be proposed by the model) we can still tackle scenes with a lot of objects, and we show that the number of VAE calls can be reduced further.

Moreover, the SPAIR and SPACE frameworks incorporate priors on the objects' geometric properties distributions (position, scale, depth). This is not ideal for several reasons.

First, we show that it introduces unnecessary hyperparameters which are problem dependent as the geometric properties of the objects can vary from one task to another and thus must be tuned manually for each new task. Second, and more importantly, the scale prior can make the model unable to tackle scenes containing objects of various sizes. This problem is mentioned in the official SPACE GitHub repository: "Learning is difficult when object sizes vary a lot. In SPACE, we need to set a proper prior for object sizes manually and that turns out to be a crucial hyperparameter. For example, for the 10 Atari games we tested, objects are small and roughly of the same size. When object sizes vary a lot, SPACE may fail.". In chapter 5 we propose an effective way to solve this problem and show that our model performs well on CLEVR, a dataset with a lot of variations in object sizes.

We also propose to push further the explicit discovery of objects' geometric properties by taking into account rotation in the object proposal mechanism. Indeed, the spatial transformers Jaderberg et al. [2015] used to extract objects from the image before passing it to the object VAE supports rotation. To this end, we propose a loss designed to push the model to align the bounding box with some axis of the object.

Wrap-up

Several methods have been designed to improve the controllability of deep generative networks. Some of them use annotated data, others are using closed-form solutions and some of them are unsupervised or self-supervised. In this work, we focused on unsupervised or self-supervised methods and identify several axes of work. The first axis is to combine VAE and GAN to benefit from the superior controllability of VAE while keeping the quality of GANs, the second axis is the exploration of the latent space to better understand how to choose a latent code to get an image with the desired properties, and the third is the design of models with control explicitly baked-in through the use of models which automatically learn to decompose scenes in an interpretable way without annotated data.

Chapter 3

Combining VAE and GAN

This chapter contains our work on combining VAE and GAN to bring some advantages of VAE to the GAN framework (namely encoder network and disentangled latent space) without sacrificing the quality of generated samples.

3.1 Introduction

Since the original GAN paper Goodfellow et al. [2014], generative models have successfully leveraged the power of deep learning to generate complex image distributions with increasing fidelity. Generative models are now used for a wide variety of tasks, including notably sample generation but also photo manipulation Brock et al. [2017], style transfer Zhu et al. [2017], pre-processing for face recognition Huang et al. [2017], text-to-image translation Zhang et al. [2016b] and controlled image generation Plumerault et al. [2020] as we will see in 4.

In the literature, two families of generative models stand out for image data: Variational Auto Encoders (VAE) Kingma and Welling [2014] and Generative Adversarial Networks (GAN) Goodfellow et al. [2014], each exhibiting respective advantages and limitations. The main limitation of GANs is poor mode coverage of the target distribution, while the main issue with VAE is the lack of realism of samples it generates. Some previous works Boesen Lindbo Larsen et al. [2015]; Donahue et al. [2016]; Dumoulin et al. [2016] have proposed solutions to solve these problems by combining or modifying these two frameworks. However, these methods systematically exhibit a trade-off between the realism of the generated images and the fidelity of the reconstructions. In this work, we show that GANs and VAEs can be complementary in the sense that we can derive two losses from them that tackle orthogonal objectives and can be combined effectively. From this observation, we propose the AVAE (for Adversarial Variational Auto Encoder) model which is a VAE-style model to produce samples of comparable quality as those generated by a GAN while allowing high-fidelity reconstructions when used as an auto-encoder. In comparison to Boesen Lindbo Larsen et al. [2015] who first introduces the idea of a combination of the two frameworks, we provide theoretical insights to show the pertinence of our approach, and we address the problem of the trade-off between realism and reconstruction accuracy.

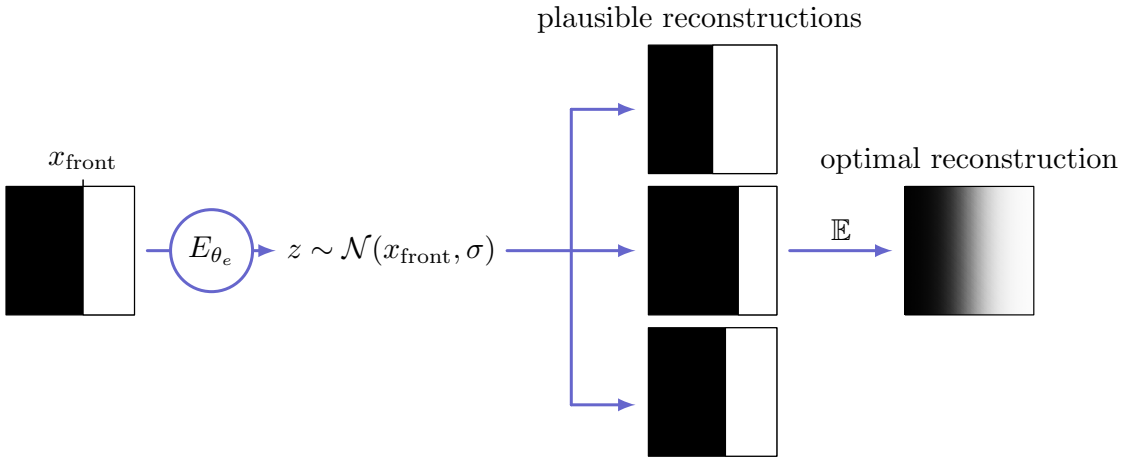

Figure 3.1: Illustration on why VAEs produce blurry reconstructions. Consider the example of a binary frontier in an image i and a latent code $\mathbf{z}$ which encodes the position of the frontier x_{front} such that $q_{\theta_e}(\mathbf{z}_0|i) = \mathcal{N}(x_{\text{front}}, \sigma)$ then $p_{\theta_e}(x_{\text{front}}|\mathbf{z}) = \mathcal{N}(\mathbf{z}_0, \sigma)$ and the optimal reconstruction of the pixel at position x is $\mathbb{E}\left[pixel(x)|\mathbf{z}\right] = 1 \times \mathbb{P}_{\theta_e}(x > \mathbf{z}_0) + 0 \times \mathbb{P}_{\theta_e}(x < \mathbf{z}_0) = \frac{1}{2}\left(1 + \text{erf}(\frac{x-\mathbf{z}_0}{\sqrt{2}\sigma})\right)$ which is a smooth transition between black and white instead of a sharp transition as in the dataset images.

The paper is organized as follows. We begin with a reminder of GAN and VAE frameworks and explain their limitations. Then we investigate how they can be combined effectively. We thus propose an effective approach to do so, named AVAE (for Adversarial Variational Auto Encoder). At last, we present a qualitative and quantitative evaluation of the performance of our model on a variety of image datasets comparing it with the state of the art. We also show that our method scales well to high-resolution images.

3.2 Background

3.2.1 Variational Auto Encoders limitations

Understanding and improving VAE are active subjects of research. Some works have focused on reducing the gap in quality which often exists between reconstructions produced by VAEs and images sampled with them Dai and Wipf [2019]. Others have aimed at learning more interpretable latent space structure Klushyn et al. [2019]. While dealing with interesting issues, these papers are not in line with the problem tackled in our work related to the lack of realism and the blurry aspect of images generated or reconstructed with VAEs.

As explained in Equation 2.2, $\mathcal{L}_{\mathcal{P}}$ acts as an information bottleneck that limits the information about the original image x that passes through the latent code. This creates an uncertainty on the attributes of the original image x when trying to reconstruct it. This uncertainty combined with the use of the mean square error as a reconstruction error causes the generated images to be blurry. The issue lies in the hypothesis we used when designing the loss. We supposed that given a latent code, each pixel is sampled from independent Gaussian distributions with a common variance. We can see why it is an issue when considering the case where the real distribution of a pixel is multi-modal. Indeed, in this case, our model

is likely to assign a high likelihood to pixel values close to the mean of the real distribution but which in reality have a very low likelihood (and are thus unrealistic). Furthermore, our hypothesis also ignores relations between pixels that are not encoded in the latent code. In short, in those circumstances, the optimal value for each pixel of the reconstructed image is its expected value given the information available in the latent code Mathieu et al. [2015] (We present an illustration of this problem in Figure 3.1). If we refer to the theoretical aspect of the VAE the output of the VAE is not an image but a distribution which under commonly used choices (MSE or L1 reconstruction error) is far from matching the true data distribution as it relies on false assumption namely independence of pixels given the latent code and unimodal distribution.

The second aspect that prevents VAE and AE in general to produce realistic samples is the use of a pixel-wise reconstruction error combined with the high dimensionality of the natural image manifold. Indeed, it is often assumed that natural images lie on a low dimensional manifold of image space, in particular, because of a strong redundancy at a local scale Kretzmer [1952]. This point is globally asserted by empirical evidence Ruderman [1994] but can be mitigated concerning textures. We argue that textures like wood, hair or waves of the ocean are living in a much higher dimensional manifold. This manifold thus cannot be captured in the low dimensional latent space of generative models even in the absence of an explicit information bottleneck. Indeed, it would require a network with a very high capacity to map the low-dimensional latent into a high-dimensional manifold. One can convince himself of this fact by considering that hair configuration is the product of the configuration of each strand of hair. GANs can partially overcome this problem with mode collapse on textures by generating only a subset of this manifold which is enough to fool the discriminator network. Mode collapse is usually viewed as a problem for GANs However, we argue that it is necessary to have some mode collapse to produce realistic samples. However, the use of a powerful pixel-wise reconstruction error in the case of VAE prevents the decoder from using this strategy leading to unrealistic results. It should also be noted that some works [Makhzani et al., 2015; Ghosh et al., 2019] have removed the information bottleneck from the VAE framework and still exhibit this problem of blurriness on the level of textures. This support further our explanation.

3.2.2 Generative Adversarial Networks limitations

GANs have proven to be very successful for generation tasks but suffer from two major limitations in comparison to VAEs: training difficulties due to mode collapse (see Arjovsky and Bottou [2017] for a detailed explanation of the causes of this behavior) and the absence of an encoder network. *Mode collapse* occurs when, at each step, the generator is able to only produce a few different samples. In its extreme case, the generator only produces one type of sample, that is thus easily recognized by the discriminator. In return, the discriminator does not need real data to train and its feedback to the generator through back-propagation does no longer contain useful information. More commonly, the generator produces a limited number of samples and interpolation of them. When mode collapse occurs, it is very difficult for the GAN to escape this pathological state, making further training near impossible. This phenomenon makes the training of GANs unstable and thus hard to conduct successfully.

Even when a GAN appears to have attained a good solution, *mode collapse* may have

occurred slightly and some modes of the data distribution may be missed by the generator. *Mode collapse* also raises the question of the existence of an acceptable pseudo-inverse mapping of the generator defined on the entire dataset space.

The second issue is that the GAN framework does not provide an explicit model to find the latent codes of samples as it does not have an encoder.

3.3 The AVAE framework

3.3.1 Complementarity between VAE and GAN

Despite their differences, we will show that VAE and GAN exhibit some form of complementarity and that we can build a hybrid approach that solves several problems listed above. One naive hybridization could be to train a VAE with an additional adversarial loss term to push reconstructions toward more realism. However, as we have seen, optimal reconstructions are not always realistic. This approach would lead to choosing a trade-off between reconstruction accuracy and realism as both have conflicting objectives. One of the contributions of this work is to show that we can derive two complementary losses from the VAE and GAN frameworks which share an optimal solution allowing accurate and realistic reconstructions.

In the GAN framework, we can derive a *manifold* loss $\mathcal{L}_{\mathcal{M}}$ from the discriminator network which judges the realism of a given sample. This loss can be interpreted as a "distance" between the data manifold and a sample generated or not as described in [Zhao et al., 2019]. In the VAE framework, we train an encoder that maps data in a latent space $\mathcal{Z}$. This latent space can be identified as the data manifold as they exist a correspondence between points in the latent space and points on the data manifold this correspondence being encoded in the generator network. Distances in the latent space can be interpreted as a distance between two points of the data manifold. This loss is noted $\mathcal{L}_{\mathcal{Z}}$. Our intuition, depicted by Figure 3.2, is that these two losses can be used in conjunction to train a model which produces realistic images while keeping approximately the latent space organization of a VAE.

We give here further explanation on why the VAE framework fails to produce realistic images and what conditions a reconstruction error should satisfy to achieve accurate and realistic reconstructions. Let us consider an auto-encoder that uses a reconstruction error of the form $\mathcal{L}(x, y) = \|x - y\|^2$. Let us note x the input, $\mathbf{z}$ the output of the encoder E_{θ_e} and $\hat{x}$ the output of the decoder D_{θ_d}. With the parameters of the encoder fixed, the optimal reconstruction should minimize the expected cost over the potential images $\tilde{x}$ that could have produced the observed $\mathbf{z}$. i.e.

$$\hat{x}^*(\mathbf{z}) \in \underset{\hat{x}}{\operatorname{argmin}}\, \mathbb{E}_{\tilde{x}\sim p_{\theta_e}(\tilde{x}|\mathbf{z})}\left[\|\tilde{x} - \hat{x}\|^2\right] \tag{3.1}$$

Thus, the optimal solution is given by $\hat{x}^*(\mathbf{z}) = \mathbb{E}_{\tilde{x}\sim p_{\theta_e}(\tilde{x}|\mathbf{z})}[\tilde{x}]$.

The problem is that in this case the optimal reconstruction $\hat{x}^*$ is the expected value of all the possible reconstructions given the knowledge of the latent code. It leads to a blurry reconstruction, quite unlikely under the data distribution $p_{\mathcal{D}}$ (i.e. $p_{\mathcal{D}}(\hat{x}^*)$ is small).

Taking the L1 loss instead of the MSE will lead to $x^*(z)$ being the geometric median of the distribution $p_{\theta_e}(\tilde{x}|z)$ which in the case of a one-dimensional distribution is the median. This explains why L1 loss tends to produce sharper edges (See Figure 3.1 by replacing the

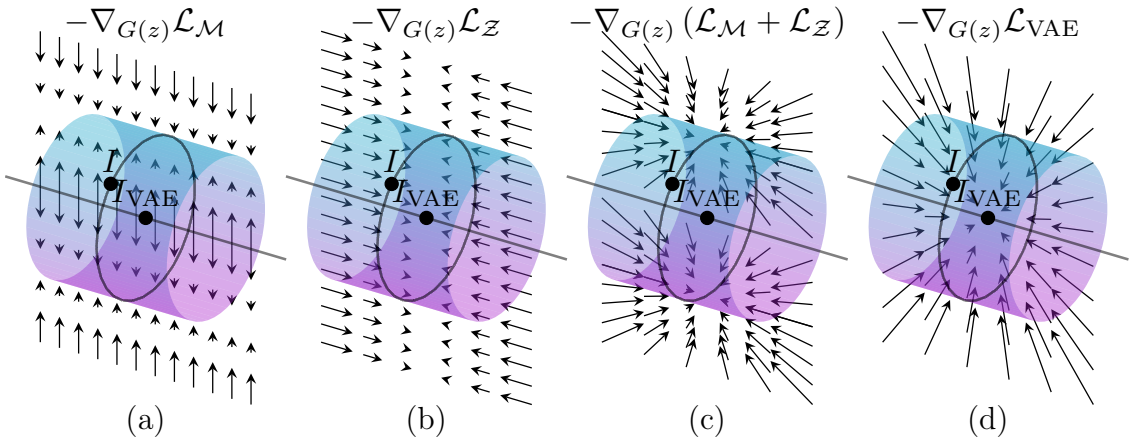

Figure 3.2: The figure depicts a small portion of data space. The cylinders symbolize the real data high-dimensional manifold, the black line represents the low-dimensional manifold on which the reconstructions of VAEs are restricted. The images on the black circle where the image I is located are all mapped to the same latent code by the encoder network. Thus, they share a common reconstruction: I_{VAE}. This reconstruction is outside the data manifold as it is the expected value of the original image given the latent code computed by the encoder which is blurry. The arrows represent the gradient of different losses (w.r.t the reconstruction) that are minimized during training: (a) the loss derived from the GAN framework that pushes the reconstructions toward the data manifold, (b) the loss derived from the VAE framework that pushes the reconstructions toward a region where images are mapped to the same latent code by the encoder, (c) their combination and (d) the VAE reconstruction loss i.e. the mean square error. (Note that gradients are represented on a single plane, while there is a radial symmetry around the black line)

mean with the median to see it) but still fails to produce sharp results on more difficult challenges like textures.

In a more general setting, we can consider objectives of the form: $L(x, a) = ||f(x) - a||^2$ where f is an arbitrary differentiable function and a, is a random variable. In this case, the optimal solution verifies:

$$f(\hat{x}^*(\mathbf{z})) = \mathbb{E}_{a \sim p_{\theta_e}(a|\mathbf{z})}[a] \tag{3.2}$$

This objective has a common optimum with the GAN objective, if and only if we have $p(f(x^*(\mathbf{z}))) = p(f(x))$ for $\mathbf{z} \sim p(\mathbf{z})$ and $x \sim p_{\mathcal{D}}(x)$. Indeed, if this equation is verified, it means that the distribution of f applied to generated images is indistinguishable from the distribution of f applied to the real images. Thus, at the optimum of $\mathcal{L}_{\mathcal{Z}}$, we can also have $\nabla\mathcal{L}_{\mathcal{M}} = 0$ and thus the two loss can attain their optimum simultaneously, there is no trade-off between them. However, to be what we can call a good reconstruction error, $f(x)$ should also carry the maximum of information about x. Indeed, we could otherwise choose $f = a$, but it would be useless as a reconstruction error as it would always be minimal.

3.3.2 Architecture

Similarly to VAE, the proposed AVAE framework is based on an encoder E_{θ_e} and a decoder D_{θ_d}. We add two additional models: a generator G_{θ_g} and a critic C_{θ_c}. The role of the generator is to produce realistic samples from latent codes.

The VAE part of our framework is similar to a classical VAE: it is a parametrized model $q_{\theta_e}(\mathbf{z}|x) = \mathcal{N}\left(\mathbf{z}; \mu_{\theta_e}(x), \Sigma_{\theta_e}\right)$ with Σ_{θ_e} a diagonal matrix of the form $\text{diag}(\sigma^2_{\theta_e})$ (note that it does not depend on the input image x). The prior distribution of the latent codes is $p(\mathbf{z}) = \mathcal{N}\left(\mathbf{z}; 0, Id\right)$ and $p_{\theta_d}(x|\mathbf{z}) = \mathcal{N}\left(x; \mu_{\theta_d}(\mathbf{z}), Id\right)$. With such choices, $\mathcal{O}_{\text{VAE}}(\theta_e, \theta_d; x)$ can be estimated by a Monte-Carlo method. Indeed, the Kullback-Leibler divergence term of the loss $\text{KL}\left(q_{\theta_e}\left(\mathbf{z}|x\right) \| p(\mathbf{z})\right)$ is equal to:

$$\frac{1}{2} \sum_{j=1}^{\dim(\mathcal{Z})} \sigma^2_{\theta_e j} + \mu^2_{\theta_e}(x)_j - 1 - \log \sigma^2_{\theta_e j} \tag{3.3}$$

The reconstruction term of the loss $\mathbb{E}_{q_{\theta_e}(\mathbf{z}|x)}\left[\log p_{\theta_d}(x|\mathbf{z})\right]$ can be estimated by Monte-Carlo, sampling $\mathbf{z}$ from $q_{\theta_e}(\mathbf{z}|x)$ and noting that:

$$\log p_{\theta_d}(x|z) = -\frac{\dim(x)}{2} \log 2\pi - \frac{1}{2} \left\| \mu_{\theta_d}(z) - x \right\|^2 \tag{3.4}$$

$\mathbf{z}$ being sampled from $q_{\theta_e}\left(\mathbf{z}|x\right)$, the loss of the VAE for one sample is the following (without constant terms):

$$\begin{aligned} \mathcal{L}_{\text{VAE}}(\theta_e, \theta_d; x) = & \frac{1}{2} \left\| \mu_{\theta_d}(\mathbf{z}) - x \right\|^2 \\ & + \frac{1}{2} \sum_{j=1}^{\dim(\mathcal{Z})} \sigma^2_{\theta_e j} + \mu^2_{\theta_e}(x)_j - \log \sigma^2_{\theta_e j} \end{aligned} \tag{3.5}$$

For the generator part, when we want to use it for reconstruction, we build its input by concatenating $\mathbf{z}$ the latent code produced by the encoder with a random vector ξ sampled from $\mathcal{N}(0, Id)$ to form the latent code for our generator. $\mathbf{z}$ encodes the information captured by the encoder while ξ encode the variation not captured by it. With these choices, we sample from $p_{\theta_g}(x|\mathbf{z})$ by taking $x = G_{\theta_g}(\mathbf{z}, \xi)$. It is a conditional GAN conditioned by z. Note that ξ can be removed if we consider that for a given $\mathbf{z}$ there is only one possible reconstruction, but we present here the general setting. To sample a random image from the generator we simply sample $\mathbf{z}$ from the prior distribution defined in the VAE part and ξ from $\mathcal{N}(0, Id)$. Ideally, the generator should invert the encoder and thus $p_{\theta_g}(x|\mathbf{z})$ should be as close as possible as $p_{\theta_e}(x|\mathbf{z})$. This consideration leads us to minimize the following negative log-likelihood with $z \sim \mathcal{N}(0, Id)$ and $x \sim p_{\theta_g}(x|\mathbf{z})$:

$$\begin{aligned} \mathcal{L}_G(\theta_g) &= \mathbb{E}\left[-\log p_{\theta_e}(x|\mathbf{z})\right] \\ &= \mathbb{E}\left[-\log p_{\theta_e}(\mathbf{z}|x) p(x)\right] + C \\ &= \mathbb{E}\left[-\log p_{\theta_e}(\mathbf{z}|x)\right] + \mathbb{E}\left[\log \frac{p_{\theta_g}(x)}{p(x) p_{\theta_g}(x)}\right] + C \\ &= \mathbb{E}\left[-\log p_{\theta_e}(\mathbf{z}|x)\right] + \text{KL}(p_{\theta_g}(x) \| p(x)) + H_{\theta_g} + C \end{aligned} \tag{3.6}$$

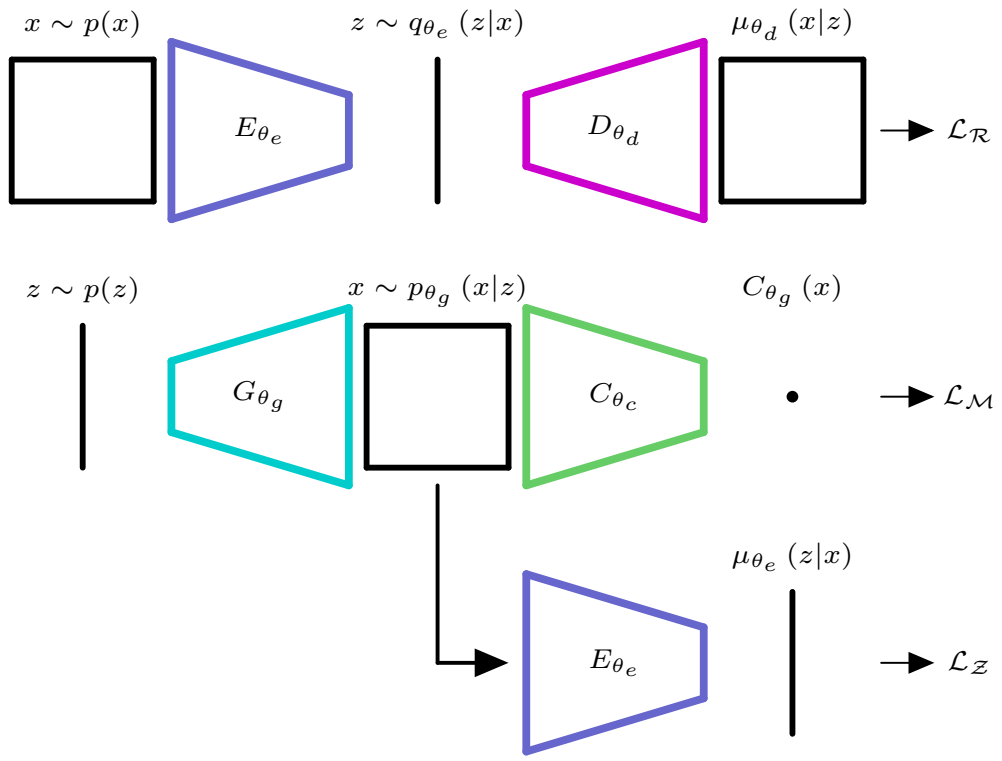

Figure 3.3: Summary of our Adversarial Variational Auto Encoder framework. E_{θ_e}, D_{θ_d}, G_{θ_g} and C_{θ_c} are respectively the encoder, decoder, generator and critic (discriminator). Note that the weights of the encoder E_{θ_e} are shared between the two architectures. Images are denoted by the letter x and latent codes by the letter $\mathbf{z}$.

With H_{θ_g} the differential entropy of the distribution $p_{\theta_g}(x)$. The term $\log p_{\theta_e}(\mathbf{z}|x)$ can be computed directly:

$$\begin{aligned} \log p_{\theta_e}(\mathbf{z}|x) =& \log \mathcal{N}(\mathbf{z}; \mu_{\theta_e}(x), \Sigma_{\theta_e}) \\ =& -\frac{\dim(\mathcal{Z})}{2}\log 2\pi - \frac{1}{2}\log|\Sigma_{\theta_e}| \\ & -\frac{1}{2}\left\|\frac{\mu_{\theta_e}(x)-\mathbf{z}}{\sigma_{\theta_e}}\right\|^2 \end{aligned} \tag{3.7}$$

We define the reconstruction loss $\mathcal{L}_{\mathcal{Z}}$ by removing constant terms in Equation 3.7:

$$\mathcal{L}^a_{\mathcal{Z}}(\theta_g; z, \theta_e) = \frac{1}{2}\left\|\frac{\mu_{\theta_e}(x)-\mathbf{z}}{\sigma_{\theta_e}}\right\|^2 \tag{3.8}$$

We can estimate the second term by training a classifier C that discriminates generated images from real ones by minimizing the cross-entropy:

$$\begin{aligned} \mathcal{L}_C(\theta_c) =& -\mathbb{E}_{x\sim p(x)}\left[\log\left(1 - C_{\theta_c}(x)\right)\right] \\ & -\mathbb{E}_{x\sim p_{\theta_g}(x|\mathbf{z})}\left[\log C_{\theta_c}(x)\right] \end{aligned} \tag{3.9}$$

Under this loss, the optimal solution for C is:

Initialize parameters of the models: θ_e, θ_d, θ_g, θ_c
while training **do**
{Forward pass.}
$x^{\text{real}} \leftarrow$ batch of images sampled from the dataset.
$z_\mu^{\text{real}}, z_\sigma^{\text{real}} \leftarrow E_{\theta_e}(x^{real})$
$z^{\text{real}} \leftarrow z_\mu^{\text{real}} + \epsilon z_\sigma^{\text{real}}$ with $\epsilon \sim \mathcal{N}(0, Id)$
$\mu^{\text{real}} \leftarrow D_{\theta_d}(\mathbf{z}^{\text{real}})$
$x^{\text{fake}} \leftarrow G_{\theta_g}(\mathbf{z}^{\text{fake}}, \xi)$ with $z^{\text{fake}}, \xi \sim \mathcal{N}(0, Id)$
$z_\mu^{\text{fake}}, z_\sigma^{\text{fake}} \leftarrow E_{\theta_e}(x^{\text{fake}})$
$C^{\text{real}}, C^{\text{fake}} \leftarrow C_{\theta_c}(x^{\text{real}}), C_{\theta_c}(x^{\text{fake}})$
{Compute losses gradients and update parameters.}
$\theta_e \overleftarrow{-} \nabla_{\theta_e} \mathcal{L}_{\text{VAE}}(\theta_e, \theta_d)$; $\theta_g \overleftarrow{-} \nabla_{\theta_g} \mathcal{L}_G(\theta_g)$
$\theta_d \overleftarrow{-} \nabla_{\theta_d} \mathcal{L}_{\text{VAE}}(\theta_e, \theta_d)$; $\theta_c \overleftarrow{-} \nabla_{\theta_c} \mathcal{L}_C(\theta_c)$
end while

Figure 3.4: Algorithm to train the Adversarial Variational Auto Encoder.

$$C^* : x \rightarrow \frac{p_{\theta_g}(x)}{p(x) + p_{\theta_g}(x)} \tag{3.10}$$

$\mathcal{L}_\mathcal{M}$ is then defined by:

$$\mathcal{L}_\mathcal{M}(\theta_g; x, \theta_e) = \operatorname{logit} C \tag{3.11}$$

With x sampled from $p_{\theta_g}(x)$. This is motivated by the fact that $\operatorname{logit} C \approx \operatorname{logit} C^* = \log\left(\frac{p_{\theta_g}(x)}{p(x)}\right)$ which is an unbiased estimator of the Kullback-Leibler divergence term. Concerning the third term, minimizing the differential entropy H_{θ_g} of the distribution $p_{\theta_g}(x)$ will push it to be as peaked as possible and is not data dependent. Moreover, this term is intractable. It pushes the entropy of $p_{\theta_g}(x)$ to be small, forcing the generator to produce a limited diversity of images. Hence, as a form of regularization, we remove it. One problem still remains. Indeed, the optimal reconstruction for $\mathcal{L}_\mathcal{Z}^a$ verifies the following equation: $\mu_{\theta_e}(\hat{x}^*(\mathbf{z})) = z$ and thus $p(\mu_{\theta_e}(\hat{x}^*(\mathbf{z}))) = \mathcal{N}(\mu_{\theta_e}(\hat{x}^*(\mathbf{z})); 0, I)$ while $p(\mu_{\theta_e}(x)) = \mathcal{N}(\mu_{\theta_e}(x); 0, I - \Sigma)$. The condition for the existence of a common optimum with the realism objective presented in Section 3.3.1 is not verified as $p(\mu_{\theta_e}(x^*(z))) \neq p(\mu_{\theta_e(x)})$.

$$\mathcal{L}_\mathcal{Z}^b(\theta_g; z, \theta_e) = \frac{1}{2} \left\| \frac{\mu_{\theta_e}(x) - \sqrt{1 - \sigma_{\theta_e}^2}\mathbf{z}}{\sigma_{\theta_e}} \right\|^2 \tag{3.12}$$

With this loss, we can see that the optimal solution $\hat{x}^*(\mathbf{z})$ verifies $\mu_{\theta_e}(\hat{x}^*(\mathbf{z})) = \sqrt{1 - \sigma_{\theta_e}^2} z$ thus $p(\mu_{\theta_e}(\hat{x}^*(\mathbf{z}))) = \mathcal{N}(\mu_{\theta_e}(x); 0, I - \Sigma) = p(\mu_{\theta_e}(x))$ as we have seen, it ensures that this loss has a common optimum with the GAN objective. This new loss takes into account the fact that when σ_{θ_e} is large, the observed $\mathbf{z}$ is mostly noise and $\mu_{\theta_e}(x)$ is close to zero. The loss resulting from these considerations is $\mathcal{L}_G = \mathcal{L}_\mathcal{Z}^b + \mathcal{L}_\mathcal{M}$. It combines a GAN type loss $\mathcal{L}_\mathcal{M}$ and a reconstruction loss on the latent codes $\mathcal{L}_\mathcal{Z}$ which is similar to that described in Section 3.3.1. The AVAE framework is globally presented in Figure 3.3, with the relations

GENERATOR
Dense
units: $w * 128$,
Reshape
new size: $(4, 4, w * 8)$
Batch normalization
ReLU
Up-block
channels: $w * 4$
Up-block
channels: $w * 2$
Up-block
channels: w
Transposed convolution
channels: 3, stride: 2
Tanh

DOWN-BLOCK
Convolution
stride: 2, no bias
Batch normalization
ReLU

CRITIC
Convolution
channels: w,
stride: 2
Leaky-ReLU
Down-block
channels: $w * 2$
Down-block
channels: $w * 4$
Down-block
channels: $w * 8$
Dense
units: 1
Sigmoid

Figure 3.5: Generator and critic architectures. The decoder architecture is identical to the generator architecture and the encoder architecture differs from the critic architecture by the number of units in the last layer and by the absence of a Sigmoid activation at the end. Up-block is similar to Down-block but with transposed convolutions instead of convolutions and ReLUs instead of leaky-ReLUs. All convolutions and transposed convolutions share the same filter size (5) and use 'same' padding. σ_z is chosen independent of x and is learned directly. w is a width multiplier (we typically use $w = 128$). For the BiGAN implementation, we use a two-hidden-layer MLP for the latent code inputs and a critic-style architecture for the image inputs. The two outputs representations are then concatenated and used as input of a two-hidden-layers MLP.

between its components, and Figure 3.3.2 gives the algorithm to train it. From a GAN perspective, the method can be viewed as constraining the latent space organization of the generator with the encoder model. The method also limits to some point the problem of mode collapse as the reconstruction error on the latent code prevents the generator to produce similar samples. As a consequence, it counteracts the mechanism pointed out by Metz et al. [2016] to explain mode collapse by pushing generated samples apart from each other. The proposed architecture differs from VAE/GAN in several important aspects. The decoder and generator are separated in our work and our reconstruction error is based on the encoder model and not on the discriminator as in VAE/GAN to ensure that the error is informative about the image content.

3.4 Experimental results

Datasets: We evaluate the models on six image datasets: LSUN bedroom [Yu et al., 2015] (64x64 images of bedrooms), CelebA [Liu et al., 2015] (64x64 faces cropped images), FFHQ dataset (256x256 faces) [Karras et al., 2018], CIFAR10, CIFAR100 [Krizhevsky, 2009] (32x32 images of 10 and 100 categories) and SVHN [Netzer et al., 2011] (32x32 images of house

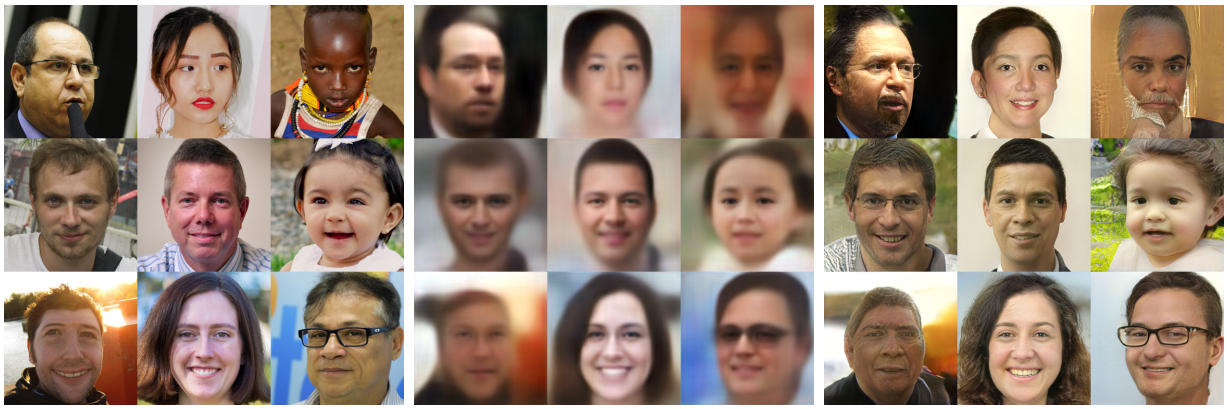

Figure 3.6: Qualitative results on high-resolution images. Left to right: original images, reconstructions with the VAE decoder, and reconstructions with the generator. This figure shows that with a limited amount of information, the decoder fails to produce realistic reconstruction while our generator is capable of it. Note that the fidelity of the reconstruction is ultimately limited by the information contained in the latent code produced by the encoder.

numbers images). Images are resized to the sizes mentioned above and CelebA images are center-cropped at 70%.

Implementation details: all the low-resolution experiments have been conducted with Tensorflow 2.0 [Abadi et al., 2015] on an NVIDIA GTX 1080 Ti GPU with 11Go of memory. The full code will be available on GitHub. All models share similar architecture blocks, inspired by [Radford et al., 2015], to allow a fair comparison. Architecture details are presented in Figure 3.5. Each model is trained with hyperparameters recommended in [Radford et al., 2015] for $5e4$ iterations with a batch size of 64. Because the reconstruction loss of the VAE part of VAE/GAN is a perceptual loss that differs from the MSE used in our model and in the classical VAE, the balance between the Kullback-Leibler divergence term and the reconstruction term in the VAE loss is not the same between models. In all our experiments, the latent dimension is 100. For experiments using auxiliary latent ξ, both z and ξ have 100 dimensions. We observed that the Kullback-Leibler divergence term is usually much higher for the VAE/GAN model which indicates that it conveys much more information in its latent code and thus introduces a bias in the reconstruction performance comparison between models. To solve this problem, we introduced a hyperparameter β to weight the Kullback-Leibler divergence in the encoder loss as in [Higgins et al., 2017] in order to get similar Kullback-Leibler divergences. This hyperparameter search leads us to the following ($\beta_{\text{LSUN bedroom}} = 4$, $\beta_{\text{celeba}} = 5$, $\beta_{\text{CIFAR10}} = 10$, $\beta_{\text{CIFAR100}} = 10$, $\beta_{\text{SVHN}} = 20$). The high-resolution experiment was conducted with a network architecture derived from the StyleGAN V2 architecture [Karras et al., 2019] trained on 8 NVIDIA Quadro P5000 GPUs.

3.4.1 Toy dataset

We begin by testing our approach on a toy dataset to validate the theory and illustrate it. The dataset is composed of 2D points generated from two generative factors z_1 and z_2. The data generation procedure is the following: $z_1, z_2, \epsilon \sim \mathcal{N}(0, 1)$ and $x = f(\mathbf{z}_1, z_2, \epsilon) = (3z_1 + 0.1\epsilon, cos(3z_1 + \tanh(3z_2)) + 0.1\epsilon)$ (See Figure 3.7). For the model, we use a latent

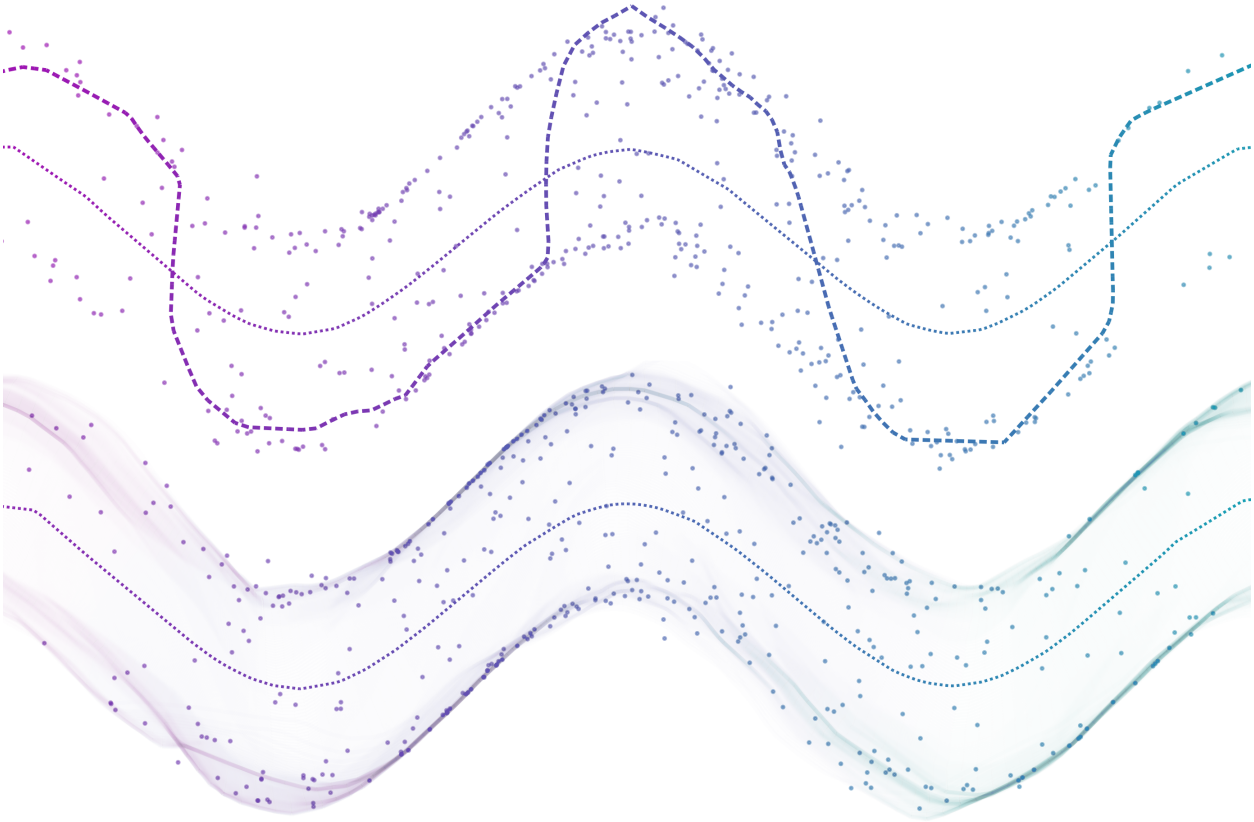

Figure 3.7: Illustration of a toy example with two-dimensional data and a one-dimensional latent space. Points: data, dotted line: manifold of reconstructions from VAE, dashed line/density: manifold of reconstruction with our model. Color encodes the position in the one-dimensional latent space of the model. Top: with a deterministic generator. Bottom: with a probabilistic generator. (best seen with zoom and color)

space of dimension one to simulate the problem of the low dimensionality of the latent space compared to the high dimensionality of the data manifold. Models for this experiment are two hidden-layer perceptrons with 128 units. We then draw the manifold of the generated points to see how the model behaves compared to a VAE. Results of this experiment can be seen in Figure 3.7 where we can see that reconstructions from the VAE are in a region where the likelihood of the data distribution is low while AVAE reconstructions follow the shape of the VAE manifold while covering regions of higher likelihood. It shows that our model can produce realistic reconstructions even when the latent code does not contain all the information needed to reconstruct the original image perfectly. Here there is an ambiguity as we do not know if the original sample is from the top distribution or the bottom one given there is a unique latent code for the two. To produce a realistic result the generator has to make an arbitrary choice that is some form of mode collapse. Our approach allows the generator to make such a choice while the decoder from the VAE outputs the average of possible choices resulting in an unlikely/unrealistic reconstruction. In the same figure we can see that when using a stochastic generator with additional latent variables, it learns to generate missing regions of the data distribution while keeping the VAE latent space

Table 3.1: Reconstruction errors (MSE and LPIPS Zhang et al. [2018]) and FID Heusel et al. [2017] of generated images for different models. Lower values are better for all metrics. Reported results are the average and standard deviation over five runs.

		Bedroom	CelebA	CIFAR10	CIFAR100	SVHN
VAE	MSE	0.06 ± 0.00	0.03 ± 0.00	0.05 ± 0.00	0.05 ± 0.00	0.02 ± 0.00
	LPIPS	0.58 ± 0.00	0.18 ± 0.00	0.26 ± 0.00	0.25 ± 0.00	0.08 ± 0.00
	FID	229.75 ± 1.45	60.04 ± 0.47	136.75 ± 0.57	129.71 ± 1.01	68.16 ± 2.10
GAN	FID	110.59 ± 19.55	14.54 ± 0.41	32.01 ± 0.41	34.51 ± 0.59	23.83 ± 3.99
VAE/GAN	MSE	0.18 ± 0.01	0.07 ± 0.00	0.14 ± 0.02	0.15 ± 0.02	0.06 ± 0.02
	LPIPS	0.26 ± 0.01	0.09 ± 0.00	0.08 ± 0.01	0.08 ± 0.01	0.08 ± 0.02
	FID	60.02 ± 2.36	26.45 ± 4.66	39.04 ± 2.42	40.03 ± 0.71	17.02 ± 2.58
BiGAN	MSE	0.42 ± 0.05	0.18 ± 0.01	0.31 ± 0.02	0.33 ± 0.01	0.12 ± 0.01
	LPIPS	0.44 ± 0.02	0.16 ± 0.00	0.14 ± 0.00	0.16 ± 0.00	0.12 ± 0.01
	FID	91.72 ± 18.10	18.49 ± 5.06	34.61 ± 1.29	35.40 ± 1.23	27.77 ± 2.96
Ours with ξ with $\mathcal{L}_{\mathcal{Z}}^{a}$	MSE	0.12 ± 0.00	0.05 ± 0.00	0.09 ± 0.00	0.09 ± 0.00	0.04 ± 0.00
	LPIPS	0.36 ± 0.00	0.11 ± 0.00	0.10 ± 0.00	0.11 ± 0.00	0.10 ± 0.00
	FID	85.11 ± 2.87	16.99 ± 0.58	33.65 ± 0.28	39.81 ± 0.60	27.64 ± 2.41
Ours without ξ with $\mathcal{L}_{\mathcal{Z}}^{a}$	MSE	0.12 ± 0.00	0.05 ± 0.00	0.09 ± 0.00	0.09 ± 0.00	0.04 ± 0.00
	LPIPS	0.35 ± 0.00	0.11 ± 0.00	0.10 ± 0.00	0.11 ± 0.00	0.09 ± 0.00
	FID	84.29 ± 5.28	16.23 ± 0.50	33.49 ± 0.50	38.69 ± 0.62	28.47 ± 8.24
Ours without ξ with $\mathcal{L}_{\mathcal{Z}}^{b}$	MSE	0.12 ± 0.00	0.05 ± 0.00	0.09 ± 0.00	0.09 ± 0.00	0.04 ± 0.00
	LPIPS	0.35 ± 0.00	0.11 ± 0.00	0.10 ± 0.00	0.11 ± 0.00	0.08 ± 0.00
	FID	80.99 ± 1.82	15.01 ± 0.82	33.67 ± 0.61	38.35 ± 0.57	21.11 ± 0.42

structure represented by the color.

3.4.2 Qualitative results

Here, we present some qualitative results on the CelebA SVHN and LSUN bedroom datasets. A comparison of sample reconstruction between our model and other models is presented in Figure 3.8. We also present a visual comparison of samples generated by our model and other generative models in Figure 3.9. Additional qualitative results will be available on GitHub. We can see from these figures that generated images are of comparable quality to GAN-generated images for both generation and reconstruction. VAE reconstructions and generated samples look blurry, BiGAN-generated images are of good quality, but reconstructions are not accurate. VAE/GAN produces both good reconstructions and generated samples. However, while our judgment is subjective, we find that reconstructions produced by VAE/GAN are less accurate than ours and images are less realistic than with GAN, BiGAN or our approach.

One may notice that for the LSUN bedroom dataset, reconstructions produced by our model are not convincing. However, we can explain this by the very poor performance of the VAE suggesting that not enough information passes through the latent code to create a reconstruction visually close to the original image. However, even here, our model still produces sharp images close to the target in terms of MSE showing that our model follows

the latent structure of the VAE trained with the MSE as a reconstruction error.

We also experimented with higher-resolution images (256x256 FFHQ face images) to see if our method can be scaled to high-resolution images. To conduct this experiment, we made straightforward modifications to the style GAN V2 [Karras et al., 2019] using the approach proposed here. The results of these experiments are presented in Figure 3.6. These results confirm the scalability of the proposed approach to bigger architectures.

3.4.3 Quantitative results

To quantitatively evaluate the performance of our method, we selected several metrics. The quality of the reconstructed images is evaluated by the Mean Squared Error or MSE and the LPIPS [Zhang et al., 2018]. We use the FID [Heusel et al., 2017] to measure the realism of generated images. A comparison between VAE, GAN, BiGAN [Dumoulin et al., 2016; Donahue et al., 2016], VAE/GAN [Boesen Lindbo Larsen et al., 2015], and our model is presented in Table 3.1. Reconstructions errors are computed on validation images not used during training, namely the *test* or *validation* splits of TensorFlow datasets. FID is computed over 50000 randomly generated samples and compared to training data samples as FID requires a lot of samples to be calibrated. It must be noted that some metrics are biased toward some architectures: the MSE is favorable to the VAE model because it is the loss used to train it. It is also the case for our approach, as information contained in the latent code is optimized to produce accurate reconstructions in terms of MSE. VAE/GAN is also advantaged in terms of LPIPS and FID as this model uses a perceptual similarity metric based on a classifier as a reconstruction error and the FID and LPIPS are also based on deep features. Globally, our model exhibits a good compromise between accurate reconstructions (MSE and LPIPS) and realism (FID), thus combining the best of VAE and GAN.

3.5 Conclusion

We proposed a novel way to train a generative adversarial network that mimics the latent space of a variational auto-encoder. Our main concern was to design a method that does not bias the adversarial network toward blurry, unrealistic results. To do so we specifically designed a loss that pushes the latent space of the generator to match the latent space of a VAE, while ensuring that the optimization of this loss optimum does not conflict with the optimization of the adversarial loss. We also investigated why VAEs are in general not able to produce realistic results and identified that the dimension of their latent space could be a limiting factor and how GANs circumvent this problem. Using our approach we show it yields good results in terms of reconstruction fidelity when used with the VAE encoder as well as producing realistic images. The approach is useful in scenarios where we usually use a combination of pixel-wise reconstruction losses and adversarial training to get auto-encoders that yields good result for instance in recently introduced latent diffusion models Rombach et al. [2021].

Our approach is however more complicated than training a unique VAE or GAN model which can limit its adoption. But as it stabilizes GAN training by limiting mode collapse we believe it may be worth trying. It should also be noted that while we presented a joint

training procedure, the training of the VAE part could in principle be decoupled from the training of the GAN part to save precious VRAM. However, we did not investigate how it would impact convergence, and we leave this as a possible future work.

The proposed framework can be used to generate images from a pre-trained representation. Thus, it is not a feature learning method and only features learned by the VAE are described by the representation. However, while we focused on a VAE architecture to produce the latent representation, our approach can be further extended. Indeed, one could for example train a classifier while constraining its last feature layer in the same way the latent code is constrained in the VAE and use it as a latent code in our method to focus on different features of the image. One could even concatenate several of these representations to train a model which fits their needs. We keep this extension as a potential future work.

Figure 3.8: Qualitative comparison of the quality of reconstructions between several frameworks namely VAE, VAE/GAN, BiGAN and our model on three datasets: CelebA, SVHN and LSUN bedroom.

Figure 3.9: Generated images for randomly sampled latent codes for CelebA, SVHN and LSUN bedroom.

Chapter 4

Controlling generative models with continuous factors of variations

This chapter contains our work on exploring the latent space of generative models to find directions that correspond to the variation of continuous factors of variation such as the position or scale of the main subject of a scene.

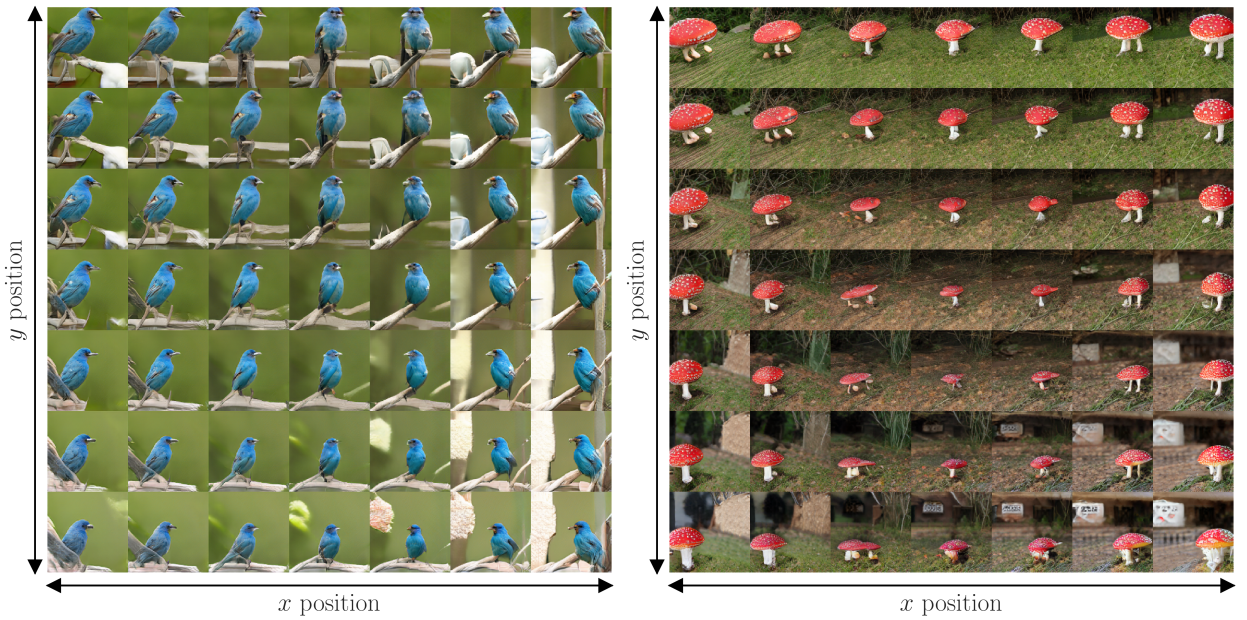

Figure 4.1: Images generated with our approach and a BigGAN model [Brock et al., 2018], showing that the position of the object can be controlled within the image.

4.1 Introduction

With the success of recent generative models to produce high-resolution photo-realistic images [Karras et al., 2018; Brock et al., 2018; Razavi et al., 2019], an increasing number of

concrete applications are emerging, such as image in-painting, dataset-synthesis, and deepfakes. However, the usability of generative models is often limited by the lack of control that the user of the system can have over the generated images. More control could be used to improve existing approaches that aim at generating new training examples [Bowles et al., 2018] by allowing the user to choose more precisely the properties of the generated images.

First attempts in this direction showed that one can act on the latent code of a generated image to produce a predictable change in the image. For instance to modify an attribute of a generated image by adding a learned vector on its latent code [Radford et al., 2015] or by stitching parts of several latent codes to produce an image that shares some properties of various images as demonstrated in [Karras et al., 2018]. Moreover, the study of the latent space of generative models provides insights into its structure. This is of particular interest as generative models are also powerful tools to learn unsupervised data representations. This motivates trying to make these representations more interpretable.

One of the assumptions often used in generative models is that images result from underlying *factors of variation* such as the presence of objects, their relative positions or the lighting of the scene (See latent variable models). These factors of variation can be seen as knobs or switches that can be tuned to produce a desired scene. Not unlike character editing menus from video games that allow the user to choose physical traits of their avatar by manipulating sliders and check-boxes. We distinguish two categories of factors of variations. *Modal factors of variation* (e.g. check-boxes) are discrete values that correspond to isolated clusters in the data distribution, such as the category of the generated object. On the other hand, the size of an object or its position is described by *Continuous factors of variations* (e.g. sliders), expressed in a range of possible values. As humans, we naturally describe images by using *factors of variations* suggesting that they are an efficient representation of natural images. For example, to describe a scene, one likely enumerates the objects seen, their relative positions and relations and their characteristics [Berg et al., 2012]. This way of characterizing images is also described in [Krishna et al., 2016]. Because this notion seems so useful, explaining the latent space of generative models through the lens of factors of variation is promising. However, the control over image generation is often limited to discrete factors and requires both labels and an encoder model. Moreover, for continuous factors of variations described by a real parameter t, previous works do not provide a way to get precise control over t.

In this chapter, we propose a method to find meaningful directions in the latent space of generative models that can be used to control precisely specific continuous factors of variations while the literature has mainly tackled semantic labeled attributes like gender, emotion or object category [Radford et al., 2015; Odena et al., 2017]. We tested our method quantitatively on image generative models for three factors of variation of an object in an image: vertical position, horizontal position and scale. We also tested other factors of variation qualitatively. Our method has the advantage of not requiring a labeled dataset or a model with an encoder. It could be adapted to other factors of variations such as rotations, changes of brightness, contrast, color or more sophisticated transformations like local deformations. However, we focused on the position and scale as these are quantities that can be evaluated quantitatively with good precision, allowing us to measure effectively the effectiveness of our method. We demonstrate both qualitatively and quantitatively that such directions can be used to control precisely the generative process and show that our

method can reveal interesting insights about the structure of the latent space. In this section:

- We propose a method to find interpretable directions in the latent space of generative models, corresponding to parameterizable continuous *factors of variations* of the generated image.
- We show that properties of generated images can be controlled precisely by sampling latent representations along linear directions.
- We propose a novel reconstruction loss for inverting generative models with gradient descent.
- We give insights into why inverting generative models with optimization can be difficult by reasoning about the geometry of the natural image manifold. And propose a method to overcome this problem for our specific use case.
- We study the impacts of disentanglement on the ability to control the generative models with our method.

4.2 Latent space directions of a factor of variation

We argue that it is often easier to modify a property of an image than to obtain a label describing that property. For example, it is easier to translate an image than to determine the position of an object within said image. Hence, if we can determine the latent code of a transformed image, we can compute its difference with the latent code of the original image to find the direction in the latent space which corresponds to this specific transformation as in [Radford et al., 2015].

Let us consider a generative model $G : \mathbf{z} \in \mathcal{Z} \to \mathcal{I}$, with $\mathcal{Z}$ its latent space of dimension d and $\mathcal{I}$ the space of images. Let's also introduce a transformation $\mathcal{T}_t : \mathcal{I} \to \mathcal{I}$ characterized by a continuous parameter t. For example, if $\mathcal{T}$ is a rotation, then t could be the angle of this rotation, and if $\mathcal{T}$ is a translation, then t could be a component of the vector of the translation in an arbitrary frame of reference. Let $\mathbf{z}_0$ be a vector of $\mathcal{Z}$ and $I = G(\mathbf{z}_0)$ a generated image. Given a transformation $\mathcal{T}_T$, we aim at finding $\mathbf{z}_T$ such that $G(\mathbf{z}_T) \approx \mathcal{T}_T(I)$ to then use the difference between $\mathbf{z}_0$ and $\mathbf{z}_T$ as an estimate of the direction encoding the factor of variation described by $\mathcal{T}$.

4.2.1 Latent space trajectories of an image transformation

Given an image: $I \in \mathcal{I}$, we want to determine its latent code. When no encoder is available we can search an approximate latent code $\hat{\mathbf{z}}$ that minimizes a reconstruction error $\mathcal{L}$ between I and another image $\hat{I} = G(\hat{\mathbf{z}})$ ($\hat{I}$ can be seen as the projection of I on $G(\mathcal{Z})$) i.e. we search $\hat{\mathbf{z}}$ defined by:

$$\hat{\mathbf{z}} = \underset{\mathbf{z} \in \mathcal{Z}}{\operatorname{argmin}} \, \mathcal{L}(I, G(\mathbf{z})) \tag{4.1}$$

Solving this problem by optimization often leads to solutions located in regions where the likelihood of the distribution of latent codes used during training is low. It causes the reconstructed image $\hat{I} = G(\hat{\mathbf{z}})$ to look unrealistic[1]. Since $\mathbf{z}$ follows a normal distribution $\mathcal{N}(0, \mathbf{I_d})$ in a d-dimensional space, we have $||\mathbf{z}|| \sim \chi_d$. Thus, $\lim_{d\to+\infty} \mathbb{E}\left[||\mathbf{z}||\right] = \sqrt{d - \frac{1}{2}}$ and $\lim_{d\to+\infty} \mathrm{Var}\left(||\mathbf{z}||\right) = \frac{1}{2}$. Hence, when d is large, the norm of $\mathbf{z}$ is approximately equal to $\sqrt{d}$. This can be used to regularize the optimization by constraining $\mathbf{z}$ into a ball of radius $\sqrt{d}$ i.e. forcing $||\mathbf{z}|| \leq \sqrt{d}$:

$$\hat{\mathbf{z}} = \underset{\mathbf{z}\in\mathcal{Z}, ||\mathbf{z}||\leq\sqrt{d}}{\operatorname{argmin}} \mathcal{L}(I, G(\mathbf{z})) \tag{4.2}$$

4.2.2 Choice of the reconstruction error $\mathcal{L}$

One of the important choices regarding this optimization problem is that of $\mathcal{L}$. In the literature, the most commonly used are the pixel-wise Mean Squared Error (MSE) and the pixel-wise cross-entropy as in Lipton and Tripathi [2017] and Creswell and Bharath [2016]. However, in practice, pixel-wise losses are known to produce blurry images for reasons we explored in Chapter 3. To address this issue, other works have proposed alternative reconstruction errors. However, they are based on an alternative neural network [Boesen Lindbo Larsen et al., 2015; Johnson et al., 2016] making them computationally expensive.

The explanation usually given for the poor performance of pixel-wise mean square error is as we have seen in Chapter 3 that it favors the solution which is the expected value of all the possibilities [Mathieu et al., 2015][2]. We propose to go deeper into this explanation here by studying the effect of the MSE on images in the frequency domain. In particular, we hypothesize that due to its limited capacity and the low dimension of its latent space, the generator can not produce arbitrary texture patterns as the manifold of textures is very high-dimensional. This uncertainty over texture configurations explains why textures are reconstructed as uniform regions when using pixel-wise errors.

In Section 4.2.1, we consider a target image $I \in \mathcal{I}$ and a generated image $\hat{I} = G(\hat{z})$ to be determined according to a reconstruction loss $\mathcal{L}$ (4.1). Let us note $\mathcal{F}\{\cdot\}$ the Fourier transform. If $\mathcal{L}$ is the usual MSE, from the Plancherel theorem, we have $||\hat{I} - I||^2 = ||\mathcal{F}\{\hat{I}\} - \mathcal{F}\{I\}||^2$. Let us consider a particular frequency ω in the Fourier space and compute its contribution to the loss. The Fourier transform of I (resp. $\hat{I}$) having a magnitude r (resp. $\hat{r}$) and a phase θ (resp. $\hat{\theta}$) at ω, we have:

[1] We could have used an L^2 penalty on the norm of $\mathbf{z}$ to encode a centered Gaussian prior on the distribution of $\mathbf{z}$. However, the L^2 penalty requires an additional hyperparameter β we can get rid of.

[2] Indeed if we model the value of a pixel by a random variable x then $\operatorname{argmin}_x \mathbb{E}\left[(x - \mathrm{x})^2\right] = \mathbb{E}\left[\mathrm{x}\right]$. This problem can easily be generalized at every pixel-wise loss if we assume that nearby pixels follow approximately the same distribution as $\operatorname{argmin}_x \mathbb{E}\left[\mathcal{L}(x, \mathrm{x})\right]$ will have the same value for nearby pixels thus blurring the image. Take for example a texture. All the pixels of the texture follow the same distribution, but it's their correlations that make the singularity of this particular texture. As a pixel-wise loss ignore these correlations, it is inappropriate to reconstruct such texture and will push the system to replace it with a homogeneous area

$$
\begin{aligned}
|\mathcal{F}\{\hat{I}\}(\omega) - \mathcal{F}\{I\}(\omega)|^2 &= |\hat{r}e^{i\hat{\theta}} - re^{i\theta}|^2 \\
&= (\hat{r}cos(\hat{\theta}) - rcos(\theta))^2 + (\hat{r}sin(\hat{\theta}) - rsin(\theta))^2 \\
&= \hat{r}^2 + r^2 - 2\hat{r}r\Big(cos(\hat{\theta})cos(\theta) + sin(\hat{\theta})sin(\theta)\Big) \\
&= \hat{r}^2 + r^2 - 2\hat{r}r\Big(cos(\hat{\theta})cos(\theta) + sin(\hat{\theta})sin(\theta)\Big)
\end{aligned}
\tag{4.3}
$$

If we model the disability of the generator to model every high-frequency pattern as uncertainty on the phase of high-frequency of the generated image, i.e. by posing $\hat{\theta} \sim \mathcal{U}([0, 2\pi])$, the expected value of the high-frequency contributions to the loss is equal to:

$$
\begin{aligned}
\mathbb{E}\left[|\mathcal{F}\{\hat{I}\}(\omega) - \mathcal{F}\{I\}(\omega)|^2\right] &= \hat{r}^2 + r^2 - 2\hat{r}r\left(\underbrace{\mathbb{E}\left[cos(\hat{\theta})\right]}_{=0}cos(\theta) + \underbrace{\mathbb{E}\left[sin(\hat{\theta})\right]}_{=0}sin(\theta)\right) \\
&= \hat{r}^2 + r^2
\end{aligned}
\tag{4.4}
$$

The term r^2 is a constant w.r.t the optimization of $\mathcal{L}$ and can thus be ignored. The contribution to the total loss $\mathcal{L}$ thus directly depends on $\hat{r}^2$. While minimizing $\mathcal{L}$, the optimization process tends to favor images $\hat{I} = G(\hat{z})$ with smaller magnitudes in the high frequencies, that is to say, smoother images, with less high frequencies.

To get sharper results we therefore propose to reduce the weight of high frequencies into the penalization of errors with the following loss:

$$
\mathcal{L}(I_1, I_2) = ||\mathcal{F}\{I_1 - I_2\}\mathcal{F}\{\sigma\}||^2 = ||(I_1 - I_2) * \sigma||^2 \tag{4.5}
$$

Where $\mathcal{F}$ is the Fourier transform, $*$ is the convolution operator and σ is a Gaussian kernel. This loss can be thought of as the mean square error between the two blurred images. The blurring causes two images with similar but out-of-phase textures to be close according to this loss. With reduced importance given to the high frequencies to determine $\hat{\mathbf{z}}$ when one uses this loss in (4.2), it allows benefitting from a larger range of possibilities for $G(\mathbf{z})$ as it does not push $G(\mathbf{z})$ to be blurry and thus allows images with more details (i.e. with more high frequencies) and appropriate texture to get more realistic generated images.

Figure 4.2: Reconstruction results with different σ values. We typically used a standard deviation of 3 pixels for the kernel.

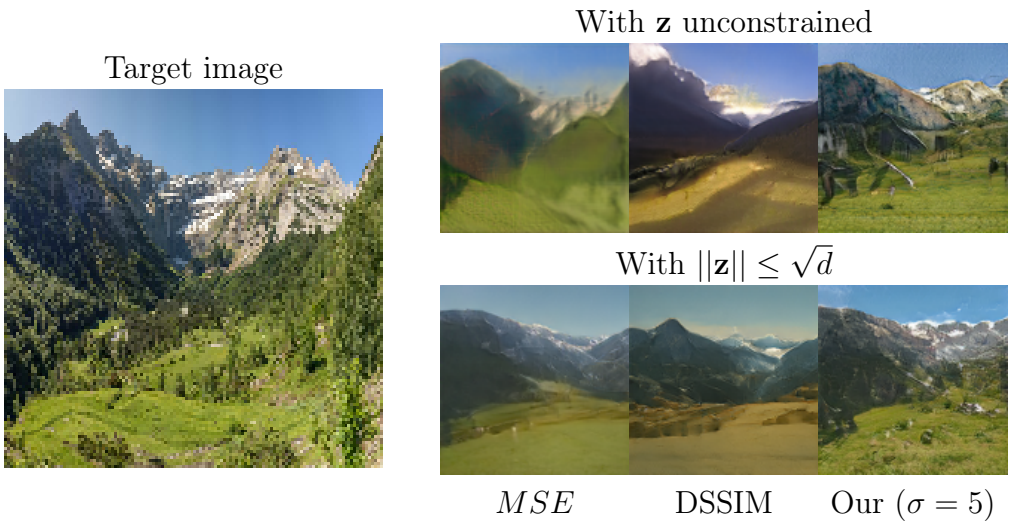

Figure 4.3: Reconstruction results obtained with different reconstruction errors: MSE, DSSIM [Zhou Wang et al., 2004] and our loss. With or without the constraint on $||\mathbf{z}||$. Note the artifacts when using our loss without constraining $\mathbf{z}$ (best seen with zoom).

In Fig. 4.2 we show qualitative reconstruction results with our method (4.5) for several values of σ. In this representative example, we observe quite good results with $\sigma = 3$ and $\sigma = 5$. Higher values of σ tend to reduce too much the penalization of lower frequencies errors leading to less accurate reconstruction.

We also illustrate in Fig. 4.3 a comparison of our approach to two others, namely classical Mean Square Error (MSE) and Structural dissimilarity (DSSIM) proposed by Zhou Wang et al. [2004]. Results are also presented with an unconstrained latent code during optimization (4.1) and the approach proposed (4.2). This example shows the accuracy of the reconstruction obtained with our approach, as well as the fact that the restriction of z to a ball of radius $\sqrt{d}$ avoids the presence of artifacts.

We also performed a quantitative evaluation of the performance of our approach is also available. We randomly selected one image for each of the 1000 categories of the ILSVRC dataset and reconstructed it with our method with a budget of 3000 iterations. Then we computed the Learned Perceptual Image Patch Similarity (LPIPS), proposed by Zhang et al. [2018], between the final reconstruction and the target image. We used the official implementation of the LPIPS paper with default parameters. Results are reported in Table 4.1. It suggests that images reconstructed using our reconstruction error are perceptually closer to the target image than those obtained with MSE or DSSIM. The higher standard deviation for the MSE reconstructed image LPIPS suggests that some images are downgraded in terms of perception. It can be the case for the textured ones in particular, for the reasons explained in this section.

Reconstruction error	mean LPIPS	std LPIPS
MSE	0.57	0.14
DSSIM	0.58	0.12
Our ($\sigma = 3$)	**0.52**	**0.12**

Table 4.1: Perceptual similarity measurements between an image and its reconstruction for different reconstruction errors.

It should be noted that while this loss does not penalize out-of-phase textures, it is also insensible to texture content and small details. As a result, it cannot be used alone as a reconstruction error but can be useful in some specific situations or can be combined with other losses.

4.2.3 Recursive Estimation of the Trajectory

Summarizing what we have described up until now, using Equation 4.2 and Equation 4.5, our problem of finding $\mathbf{z}_T$ such that $G(\mathbf{z}_T) \approx \mathcal{T}_T(I)$, given transformation $\mathcal{T}_T$, can be solved through the following optimization problem:

$$\mathbf{z}_T = \underset{\mathbf{z}\in\mathcal{Z}, ||\mathbf{z}||\leq\sqrt{d}}{\operatorname{argmin}} \mathcal{L}(G(\mathbf{z}), \mathcal{T}_T(I)) \tag{4.6}$$

However, in practice, this problem is difficult to solve and an "unlucky" initialization of $\mathbf{z}$ can lead to a very slow convergence. Zhu et al. [2016b] proposed to use an auxiliary network to estimate $\mathbf{z}_T$ and use it as initialization. Training a specific network to initialize this problem is nevertheless costly. Here we propose an explanation for the slow convergence of the optimization problem of Equation 4.6 and a way to circumvent it. We begin with the observation that a linear combination of natural images is usually not a natural image itself, this fact highlights the highly curved nature of the manifold of natural images in pixel space. Indeed, in flat Euclidean space, all segments between two points lie in the space but, for example, the points of the segment that join two points on a sphere do not fall on the sphere giving a hint about the curved nature of the sphere. The same reasoning applies when considering the manifold of natural images. In practice, the trajectories corresponding to most transforms in pixel space may imply small gradients of the loss when projected o the manifold of natural images.

Indeed, the curvature of the natural image manifold makes the optimization problem of Equation 4.2 difficult to solve. This is especially true for factors of variation which correspond to curved walks in pixel space (for example translation or rotation by opposition to brightness or contrast changes which are linear in pixel space).

We illustrate in Figure 4.4 that the trajectory described by an image undergoing common transformations is curved in pixel space. We consider three types of transformations, namely translation, rotation and scaling, and the images from the dSprites [Matthey et al., 2017] dataset which correspond to these progressive transforms for an image. To visualize, we compute the PCA of all these images and project the trajectories on the two first principal axes.

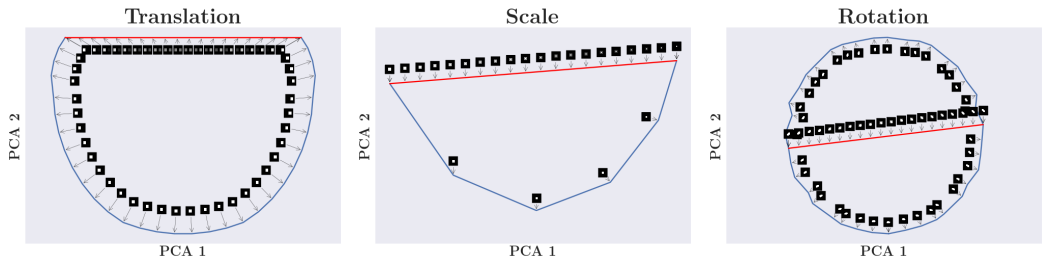

Figure 4.4: Two trajectories are shown in the pixel space, between an image and its transformed version, for three types of transformations: translation, scale and orientation. Red: shortest path (interpolation) between the two extremes of the trajectory. Blue: the trajectory of the actual transformation. At each position along the trajectories, we report the corresponding image (best seen with zoom).

In this figure, we can see that for large translations, the direction of the shortest path between two images in pixel space is near orthogonal to the manifold. The same problem occurs for rotation and, to a smaller extent, for scale. This is problematic during optimization of the latent code because the gradient of the reconstruction loss with respect to the generated image is tangent to this direction. Thus, when we are in the case of near orthogonality, the magnitude of the gradient of the error with respect to the latent code is close to zero.

Indeed, let us consider an ideal case where G is a bijection between $\mathcal{Z}$ and the manifold of natural images. For $\mathbf{z} \in \mathcal{Z}$, a basis of vectors tangent to the manifold at point $G(\mathbf{z})$ is given by $\left(\frac{\partial G(\mathbf{z})}{\partial \mathbf{z}_1}, ..., \frac{\partial G(\mathbf{z})}{\partial \mathbf{z}_d}\right)$.

If $\nabla_{G(\mathbf{z})}\mathcal{L}(G(\mathbf{z}), I_{\text{target}})$ is near orthogonal to the manifold then:

$$\forall i \in 1, ..., d : \langle \nabla_{G(\mathbf{z})}\mathcal{L}(G(\mathbf{z}), I_{\text{target}}), \frac{\partial G(\mathbf{z})}{\partial \mathbf{z}_i} \rangle \triangleq \epsilon_i \quad \text{with} \quad \epsilon_i \approx 0 \tag{4.7}$$

Thus,

$$\|\nabla_z \mathcal{L}(G(\mathbf{z}), I_{\text{target}})\| = \left\| \frac{\partial G(\mathbf{z})}{\partial \mathbf{z}}^* \nabla_{G(\mathbf{z})}\mathcal{L}(G(\mathbf{z}), I_{\text{target}}) \right\| = \sqrt{\sum_{i=1}^{d} \epsilon_i^2} \approx 0 \tag{4.8}$$

This phenomenon is also depicted in Figure 4.5. It shows that when the direction of descent in pixel space is near orthogonal to the manifold described by the generative model, optimization gets slowed down and can stop if the gradient of the loss with respect to the generated image is orthogonal to the manifold.

For example, let's assume we have an ideal GAN which generates a small white circle on a black background, with a latent space of dimension 2 that encodes the position of the circle. Let's consider a generated image with the circle on the left of the image, and we want to move it to the right. We thus have $\nabla_{\mathbf{z}}||G(\mathbf{z}) - \mathcal{T}_T(G(\mathbf{z}_1))||^2 = 0$ if the intersection of the two circles is empty (see Figure 4.4) since a small translation of the object does not change the reconstruction error.

To address this, we guide the optimization on the manifold by decomposing the transformation $\mathcal{T}_T$ into smaller transformations $[\mathcal{T}_{\delta t_0}, \ldots, \mathcal{T}_{\delta t_N}]$ such that $\mathcal{T}_{\delta t_0 = 0} = Id$ and $\delta t_N = T$ and solve sequentially:

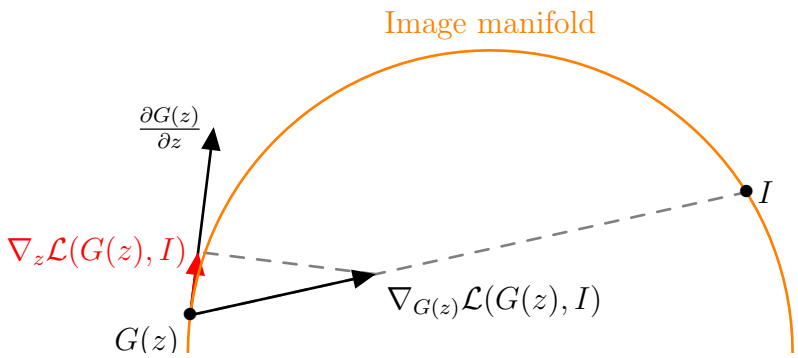

Figure 4.5: Illustration of the gradient vanishing problem encountered when doing optimization on a curved manifold.

$$\mathbf{z}_n = \underset{\mathbf{z} \in \mathcal{Z}}{\operatorname{argmin}} \mathcal{L}\left(G\left(\mathbf{z}; \mathbf{z}_{init} = \mathbf{z}_{n-1}\right), \mathcal{T}_{\delta t_n}\left(G\left(\mathbf{z}_0\right)\right)\right) \quad \text{for } n = 1, \ldots, N \tag{4.9}$$

Each time initializing $\mathbf{z}$ with the result of the previous optimization. In comparison to Zhu et al. [2016b], our approach does not require extra training and can thus be used straight ahead directly without training a new model. We compare qualitatively our method to a naive optimization in Figure 4.6.

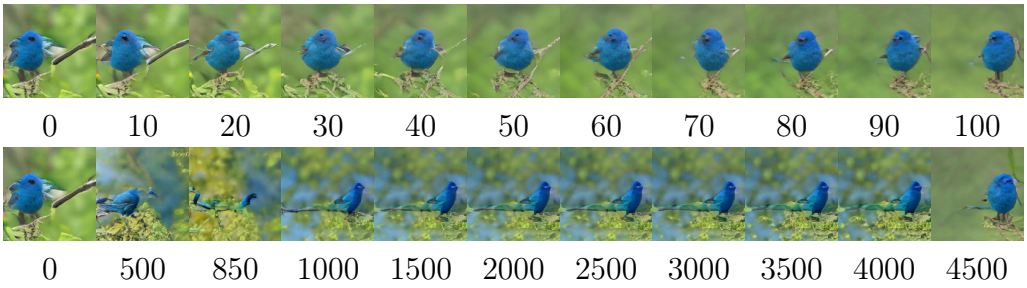

Figure 4.6: Comparison of the speed of convergence on a single example for our method (top) given by (4.9) and a naive approach (bottom) given by (4.6). The numbers indicate the step of optimization. Both experiences have been conducted with Adam optimizer with a learning rate of 1e−1.

Finally, we noted that a transformation on an image usually leads to undefined regions in the new image (for instance, for a translation to the right, the left-hand side is undefined). Hence, we ignore the value of the undefined regions of the image when computing $\mathcal{L}$. Another difficulty is that often the generative model cannot produce arbitrary images. For example, a generative model trained on a given dataset is not expected to be able to produce images where the object shape position is outside the distribution of object shape positions in the dataset. This is an issue when applying our method because as we generate images from a random start point, we have no guarantee that the transformed images are still on the data manifold. To reduce the impact of such outliers, we discard latent codes that give a reconstruction error above a threshold in the generated trajectories. In practice, we remove one-tenth of the latent codes which leads to the worst reconstruction errors. It

Algorithm 1 Create a dataset of trajectories in the latent space which corresponds to a transformation $\mathcal{T}$ in the pixel space. The transformation is parametrized by a parameter δt which controls a degree of transformation. We typically use $N = 10$ with $(\delta t_n)_{(0 \leq n \leq N)}$ distributed regularly on the interval $[0, T]$. Note that z_0 and δt_n are retained in D at each step to train the model of Section 4.3.

Inputs: number of trajectories S, generator G, transformation function $\mathcal{T}$, trajectories length N, threshold Θ.
Output: dataset of trajectories D
Procedure:
$D \leftarrow \{\}$
for $i \in [\![1, S]\!]$ **do**
  $\mathbf{z}_0 \sim \mathcal{N}(0, \mathbf{I})$
  $I_0 \leftarrow G(\mathbf{z}_0)$
  $\mathbf{z}_{\delta t} \leftarrow \mathbf{z}_0$
  for $n \in [1, N]$ **do**
    $\mathbf{z}_{\delta t} \leftarrow \text{argmin}_{\mathbf{z}} \mathcal{L}(G(\mathbf{z}), \mathcal{T}\delta t_n(I_0))$
    if $\mathcal{L}(G(\mathbf{z}), \mathcal{T}_{\delta t_n}(I_0)) < \Theta$ **then**
      $D \leftarrow D \cup \{(\mathbf{z}_0, \mathbf{z}_{\delta t}, \delta t_n)\}$
    end if
  end for
end for

finally results in Algorithm 1 to generate trajectories in the latent space that corresponds to transformations in image space.

4.3 Encoding Model of the Factor of Variation in the Latent Space.

After generating trajectories with Algorithm 1, we need to define a model which describes how factors of variations are encoded in the latent space. To this end, we make the core hypothesis that the parameter t of a specific factor of variations can be predicted from the coordinate of the latent code along an axis $\mathbf{u}$, thus we pose a model $f : \mathcal{Z} \to \mathbb{R}$ of the form $t = f(\mathbf{z}) = g(\langle \mathbf{z}, \mathbf{u} \rangle)$, with $g : \mathbb{R} \to \mathbb{R}$ and $\langle \cdot, \cdot \rangle$ the euclidean scalar product in $\mathbb{R}^d$.

If g is monotonic. We can without loss of generality, suppose that $|u| = 1$ and that g is an increasing function. If g is also differentiable, the density distribution of $t = g(\langle \mathbf{z}, \mathbf{u} \rangle)$ when $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$ is given by $\varphi : \mathbb{R} \to \mathbb{R}_+$:

$$\varphi(t) = \mathcal{N}(g^{-1}(t); 0, 1) \frac{dg^{-1}}{dt}(t) \tag{4.10}$$

For example, consider the dSprites dataset [Matthey et al., 2017] and the factor corresponding to the horizontal position of an object x in an image, we have x that follows a uniform distribution $\mathcal{U}([-0.5, 0.5])$ in the dataset while the projection of $\mathbf{z}$ onto an axis $\mathbf{u}$ follows a normal distribution $\mathcal{N}(0, 1)$. Thus, it is natural to adopt $g : \mathbb{R} \to [-0.5, 0.5]$ and

for $x = g(\langle \mathbf{z}, \mathbf{u} \rangle)$:

$$\begin{aligned}
&\varphi(x) = \mathcal{U}\left(x, [-0.5, 0.5]\right) = \mathcal{N}(g^{-1}(x); 0, 1)\frac{dg^{-1}}{dx}(x) \quad \iff \\
&1 = \mathcal{N}(\langle \mathbf{z}, \mathbf{u} \rangle; 0, 1)\frac{dg^{-1}}{dx}(g(\langle \mathbf{z}, \mathbf{u} \rangle)) \quad \iff \\
&\frac{1}{\frac{dg^{-1}}{dx}\left(g\left(\langle \mathbf{z}, \mathbf{u} \rangle\right)\right)} = \frac{dg}{dx}\left(\langle \mathbf{z}, \mathbf{u} \rangle\right) = \mathcal{N}(\langle \mathbf{z}, \mathbf{u} \rangle; 0, 1) \quad \iff \\
&g(\langle \mathbf{z}, \mathbf{u} \rangle) = \frac{1}{2}\,\mathrm{erf}\left(\frac{\langle \mathbf{z}, \mathbf{u} \rangle}{\sqrt{2}}\right)
\end{aligned} \tag{4.11}$$

Where $\frac{df}{dx}$ is the derivative of f with respect to its argument. However, in general, the distribution of the parameter t is not known. Thus, we cannot derive the proper form of g as trivially as above. One can nevertheless adopt a more general parametrized model g_θ of the form:

$$t = f_{(\theta,\mathbf{u})}(\mathbf{z}) = g_\theta\left(\langle \mathbf{u}, \mathbf{z} \rangle\right) \quad \text{with} \quad ||\mathbf{u}|| = 1 \tag{4.12}$$

With $g_\theta : \mathbb{R} \to \mathbb{R}$ and (θ, $\mathbf{u}$) trainable parameters of the model. We typically used piece-wise linear functions for g_θ.

However, even there, this model cannot be trained directly as we do not have access to t (in the case of horizontal translation the x-coordinate for example) but only to the difference $\delta t = t_{G(\mathbf{z}_{\delta t})} - t_{G(\mathbf{z}_0)}$ between an image $G(\mathbf{z}_0)$ and its transformation $G(\mathbf{z}_{\delta t})$ (δx or δy in the case of translation). We solve this issue by modeling δt instead of t:

$$\delta t = f_{(\theta,\mathbf{u})}(\mathbf{z}_{\delta t}) - f_{(\theta,\mathbf{u})}(\mathbf{z}_0) \quad \text{with} \quad ||\mathbf{u}|| = 1 \quad \text{and} \quad g_\theta(0) = 0 \tag{4.13}$$

Hence, $\mathbf{u}$ and θ are estimated by training $f_{(\theta,\mathbf{u})}$ to minimize the MSE between δ_t and $f_{(\theta,\mathbf{u})}(\mathbf{z}_{\delta t}) - f_{(\theta,\mathbf{u})}(\mathbf{z}_0)$ with gradient descent on a dataset produced by Algorithm 1 for a given transformation.

An interesting side application of this method is that it allows the estimation of the distribution of the images generated by G by using (4.10). With the knowledge of the learned g_θ, we can also choose how to sample images. For instance, let say that we want to have $t \sim \phi(t)$, with $\phi : \mathbb{R} \to \mathbb{R}_+$ an arbitrary distribution, we can simply transform $z \sim \mathcal{N}(0, 1)$ as follows:

$$\mathbf{z} \leftarrow \mathbf{z} - \langle \mathbf{z}, \mathbf{u} \rangle\, \mathbf{u} + (h_\phi \circ \psi)(\langle \mathbf{z}, \mathbf{u} \rangle)\mathbf{u} \tag{4.14}$$

with $h_\phi : [0, 1] \to \mathbb{R}$ and ψ such that:

$$\psi(x) = \int_{-\infty}^{x} \mathcal{N}(t; 0, 1)dt \quad ; \quad h_\phi^{-1}(x) = \int_{-\infty}^{x} \phi(g_\theta(t))\frac{d}{dt}g_\theta(t)dt \tag{4.15}$$

These results are interesting to bring control not only on a single output of a generative model but also on the distribution of its outputs. Moreover, since generative models reflect the datasets on which they have been trained, the knowledge of these distributions could be applied to the training dataset to reveal potential bias in the distribution of some factors of variation.

Encoder
Convolution + ReLU
filters=32 size=4 stride=2 pad=SAME
Convolution + ReLU
filters=32 size=4 stride=2 pad=SAME
Convolution + ReLU
filters=32 size=4 stride=2 pad=SAME
Convolution + ReLU
filters=32 size=4 stride=2 pad=SAME
Dense + ReLU
units=256
Dense + ReLU
units=256
μ: **Dense + Identity**
σ: **Dense + Exponential**
units=10

Decoder
Dense + ReLU
units=256
Dense + ReLU
units=256
Reshape
shape=4x4x32
Transposed Convolution + ReLU
filters=32 size=4 stride=2 pad=SAME
Transposed Convolution + ReLU
filters=32 size=4 stride=2 pad=SAME
Transposed Convolution + ReLU
filters=32 size=4 stride=2 pad=SAME
Transposed Convolution + Sigmoid
filters=1 size=4 stride=2 pad=SAME

Table 4.2: β-VAE architecture used during experiments with the dSprites dataset.

4.4 Experiments

Datasets: We performed experiments on two datasets. The first one is dSprites [Matthey et al., 2017], composed of 737280 binary 64×64 images containing a white shape on a dark background. Shapes can vary in position, scale and orientation making it ideal to study disentanglement. The second dataset is ILSVRC [Russakovsky et al., 2015], containing $1.2M$ natural images from one thousand different categories.

Implementation details: All our experiments have been implemented with TensorFlow 2.0 [Abadi et al., 2015] and the corresponding code is available on GitHub `here`[3]. We used a BigGAN model [Brock et al., 2018] whose weights are taken from TensorFlow-Hub allowing easy reproduction of our results. The BigGAN model takes two vectors as inputs: a latent vector $\mathbf{z} \in \mathbb{R}^{128}$ and a one-hot vector to condition the model to generate images from one category. The latent vector $\mathbf{z}$ is then split into six parts which are inputted at different scale levels in the generator. The first part is injected at the bottom layer while the next parts are used to modify the *style* of the generated image thanks to Conditional Batch Normalization layers [de Vries et al., 2017]. We also trained several β-VAEs [Higgins et al., 2017] to study the importance of disentanglement for the success of our method.

The β-VAE framework was introduced by Higgins et al. [2017] to discover interpretable factorized latent representations for images without supervision. For our experiments, we designed a simple convolutional VAE architecture to generate images of size 64x64, the decoder network is the opposite of the encoder with transposed convolutions. The exact β-VAE architecture used is given in Table 4.2.

[3]Source code available at: https://github.com/AntoinePlumerault/Controlling-generative-models-with-continuous-factors-of-variations

The models were trained on dSprites [Matthey et al., 2017] with an Adam optimizer during 1e5 steps with a batch size of 128 images and a learning rate of 5e−4.

4.4.1 Quantitative evaluation method

Evaluating quantitatively the effectiveness of our method on complex datasets is intrinsically difficult as it is not always trivial to measure a factor of variation directly. We focused our analysis on two factors of variations: position and scale. On simple datasets such as dSprites, the position of the object can be estimated effectively by computing the barycenter of white pixels. However, for natural images sampled with the BigGAN model, this problem is much harder. We decided to use first, saliency detection on the generated image to produce a binary image from which we can extract the barycenter. Saliency detection is an application of deep learning which aims at producing a binary mask that indicates the most salient region of an image given an input image. For saliency detection, we used the model provided by Hou et al. [2016] which is implemented in the PyTorch framework [Paszke et al., 2019]. The scale is evaluated by the proportion of salient pixels in the total image. The evaluation procedure is the following:

1. Get the direction $\mathbf{u}$ which should describe the chosen factor of variation with our method.
2. Sample latent codes $\mathbf{z}$ from a standard normal distribution.
3. Generate images with latent code $\mathbf{z} - \langle \mathbf{z}, \mathbf{u} \rangle \, \mathbf{u} + t\mathbf{u}$ with $t \in [-T, T]$.
4. Estimate the real value of the factor of variation for all the generated images with the saliency detection.
5. Measure the standard deviation of this value with respect to t.

Jahanian et al. [2019] proposed an alternative method for quantitative evaluation that relies on an object detector. Similarly to us, it allows an evaluation for x and y shift as well as scale but is restricted to image categories that can be recognized by a detector trained on some categories of ILSVRC. Our proposed approach is thus more generic.

4.4.2 Results on BigGAN

We performed quantitative analysis on ten chosen categories of objects of ILSVRC, avoiding non-actual objects such as “beach” or “cliff”.

Results are presented in Figure 4.7 (top). We observe that for the chosen categories of ILSVRC, we can control the position and scale of the object relatively precisely by moving along the directions of the latent space found by our method.

However, one can still wonder whether the directions found in the latent space are independent of the category of interest. To answer this question, we merged all the datasets of trajectories into one and learned a common direction with the resulting datasets. Results for the ten test categories are shown in Figure 4.7 (bottom). This figure shows that the

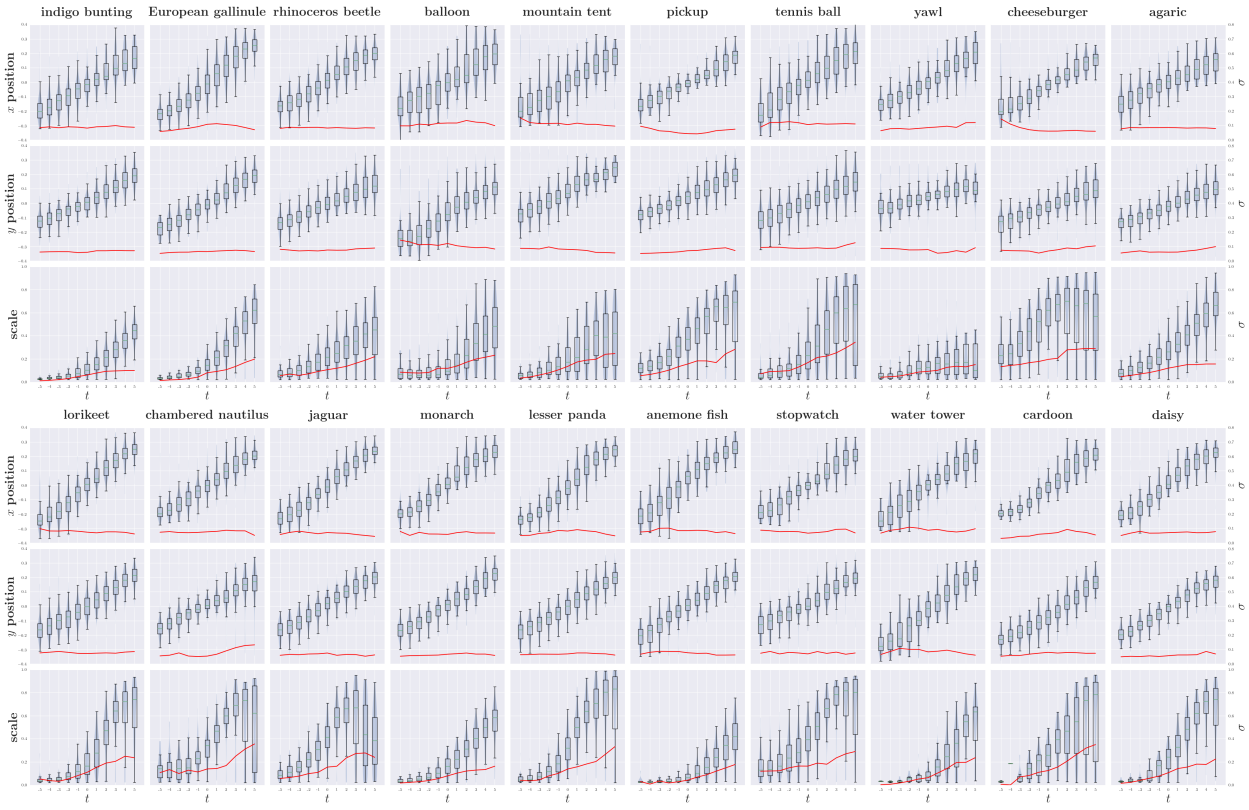

Figure 4.7: Quantitative results on the ten categories of the ILSVRC dataset used for training (Top) and on ten other categories used for validation (Bottom) for three geometric transformations: horizontal and vertical translations and scaling. In blue: the distribution of the measured transformation parameter and in red: the standard deviation of the distribution with respect to t. Note that for large scales the algorithm seems to fail. However, this phenomenon is very likely due to the poor performance of the saliency model when the object of interest covers almost the entire image (scale ≈ 1.0). (best seen with zoom)

directions which correspond to some factors of variations are indeed shared between all the categories. Qualitative results are also presented in Figure 4.9 & Figure 4.8 for illustrative purposes.

Figure 4.8: Qualitative results for 10 categories of ILSVRC dataset for three geometric transformations (horizontal and vertical translations and scaling) and for brightness.

We show qualitative examples of images generated with the BigGAN model for position, scale and brightness. The images' latent codes are sampled in the following way: $\mathbf{z} - \langle \mathbf{z}, \mathbf{u} \rangle \mathbf{u} + \alpha \mathbf{u}$ with $\alpha \in [-3, 3]$ and $\mathbf{u}$ the learned direction. We have chosen the categories to produce interesting results. For position and scale, categories are objects. For brightness, categories are likely to be seen in a bright or dark environment. Notice that for some chosen categories, we failed to control the brightness of the image. It is likely due to the absence of dark images for these categories in the training data. For position and scale, the direction is learned for

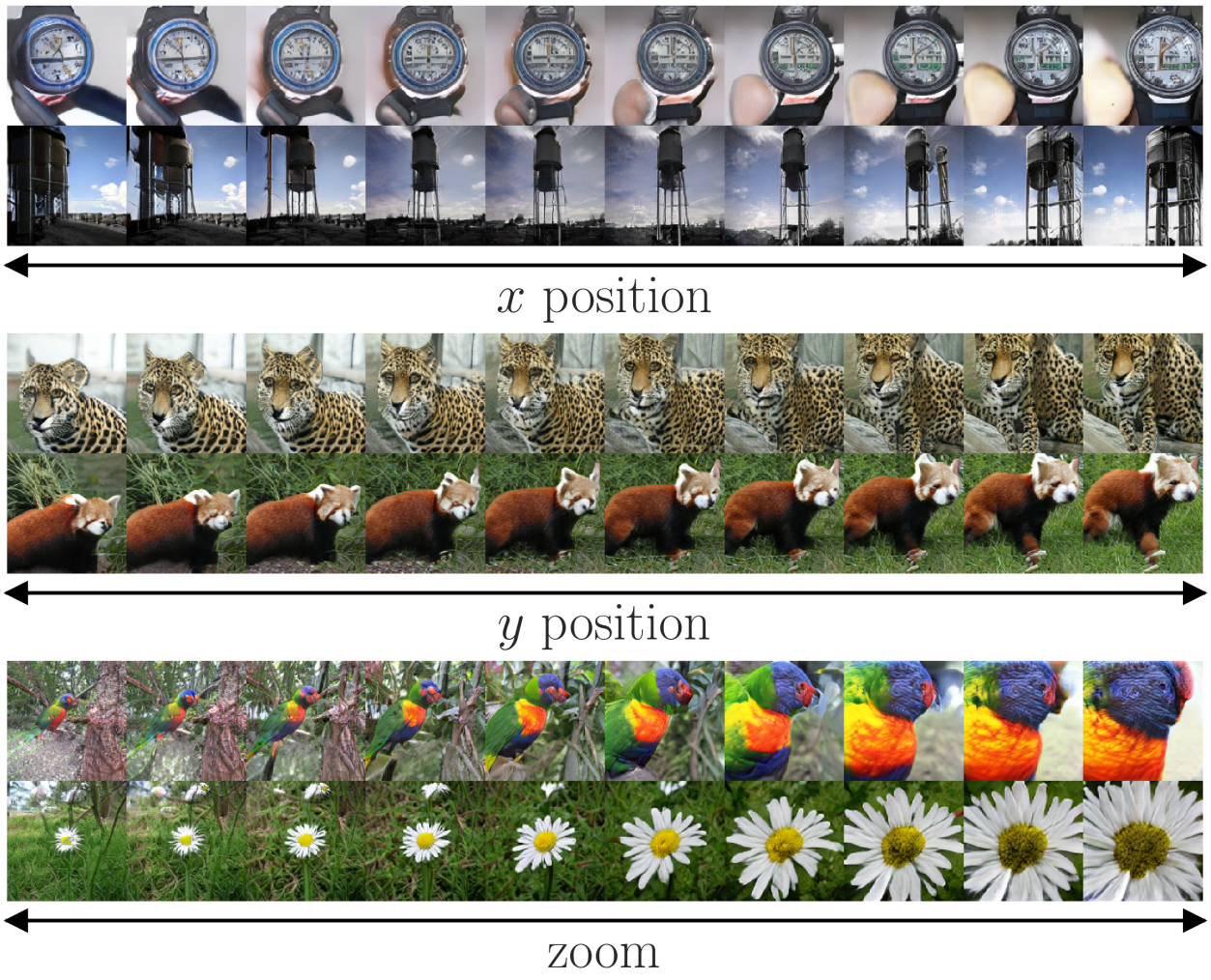

Figure 4.9: Qualitative results for some categories of ILSVRC dataset for three geometric transformations: horizontal and vertical translations and scaling.

the ten categories presented here while for brightness only the five top categories are used.

We also checked which parts of the latent code are used to encode position and scale. Indeed, BigGAN uses hierarchical latent code which means that the latent code is split into six parts which are injected at different levels of the generator. We wanted to see by which part of the latent code these directions are encoded. The squared norm of each part of the latent code is reported in Figure 4.10 for horizontal position, vertical position and scale. This figure shows that the directions corresponding to *spatial factors of variations* are mainly encoded in the first part of the latent code. It is coherent with our earlier observation that the direction found by our method is independent of the category of the object. However, for the y position, the contribution of latent code level 5 is higher than for the x position and the scale. We suspect that this anomaly is due to correlations between the vertical position of the object in the image and its background. Indeed, when shifting our image along the vertical axis, we also move the horizon. That we introduced by transforming the objects as

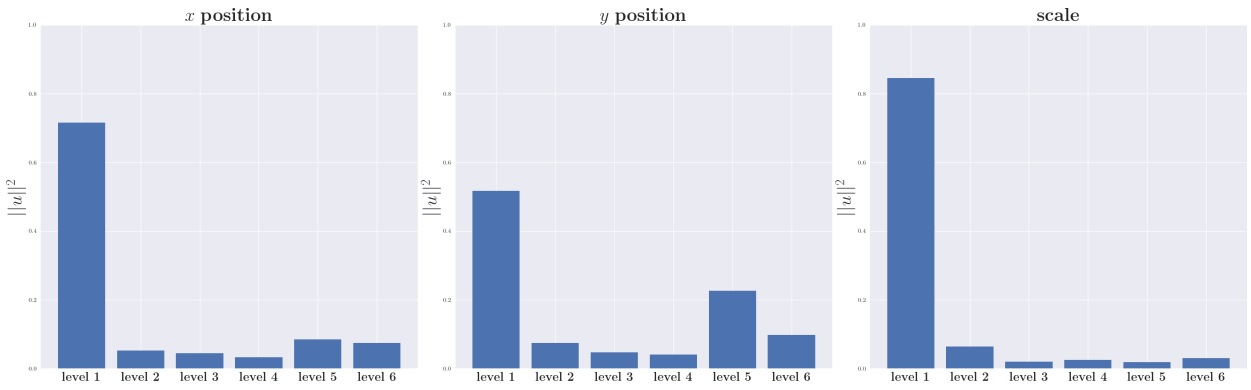

Figure 4.10: Squared norm of each part of the latent code for horizontal position, vertical position and scale.

the background is not invariant by vertical translation because of the horizon and that the position of an object in an image is mostly due to variation in the angle at which the camera is held.

4.4.3 The importance of disentangled representations

To test the effect of disentanglement on the performance of our method, we trained several β-VAE [Higgins et al., 2017] on dSprites [Matthey et al., 2017], with different β values. Indeed, β-VAEs are known for having more disentangled latent spaces as the regularization parameter β increases. β-VAEs are VAEs in which the KL term in the loss is weighted by a parameter β. The KL term pushes the VAE latent space to be more disentangled but penalizes the information that passes through the latent space. Hence, it usually degrades the reconstruction accuracy.

Results can be seen in Figure 4.11. The figure shows that it is possible to control the position of the object on the image by moving in the latent space along the direction found with our method. As expected, the effectiveness of the method depends on the degree of disentanglement of the latent space since the results are better with a larger β. Indeed, we can see in Figure 4.11 that as β increases, the standard deviation decreases (red curve), allowing more precise control of the position of the generated images. This observation motivates further the interest in disentangled representations for control of the generative process.

4.5 Conclusion

Generative models are increasingly more powerful but suffer from little control over the generative process and the lack of interpretability in their latent representations. In this context, we propose a method to extract meaningful directions in the latent space of such models and use them to control precisely some properties of the generated images. We show that a linear subspace of the latent space of BigGAN can be interpreted in terms of

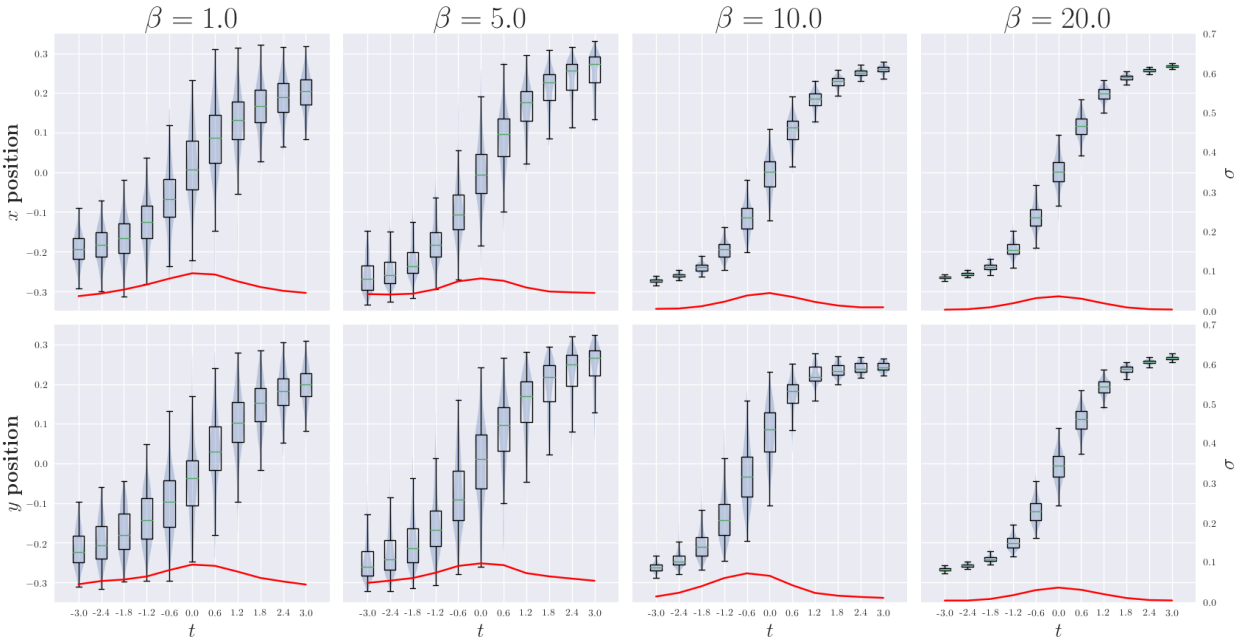

Figure 4.11: Results of our evaluation procedure with four β-VAE for $\beta = 1, 5, 10, 20$. Note the *erf* shape of the results which indicates that the distribution of the shape positions has been correctly learned by the VAE. See Figure 4.7 for additional information on how to read this figure.

intuitive factors of variation (namely translation and scale). It is an important step toward the understanding of the representations learned by generative models. For our method to work, we had to design a novel method to invert the generator of a GAN. Our method is based on optimizing a latent code to match a target image. We showed that this is a challenging problem because of the curved nature of the image manifold. To perform this task we also proposed a loss that fights the general problem of blurred reconstruction that is generally encountered when using pixel-wise errors.

Our method can be extended to other factors of variation but is fundamentally limited to the requirement that we must be able to perform this transformation approximately automatically. Fortunately, a lot of useful factors of variations can be approximated easily, for example, we could automatically deform a face image using a face landmark detector to find directions in the latent space associated with expressions or facial features. As such we believe that extending this work to local deformations guided by some key points of interest could be a promising future application of the work presented in this chapter.

Chapter 5

Generative model with off-the-shelf geometric control

This chapter contains our work on designing an efficient image generation model which decomposes a scene into background and objects. We propose techniques to improve on existing approaches in both the number of objects that can be processed and generation quality.

5.1 Introduction

Generating images of complex scenes containing several objects is a challenging problem. Such scenes are nevertheless common in our everyday lives. To describe them, we often decompose them into independent "objects" that we describe separately and in relation to each other. Using computers to describe such scenes similarly is thus a promising approach. Such representation could then be used for downstream tasks like RL (reinforcement learning) or VQA (visual question answering) reducing training complexity for specific problems. This problem can be tackled with supervision with object detection approaches like SSD Liu et al. [2016] or YOLO Redmon et al. [2015]. However, one needs to train a new model with new data if he wants to detect a new kind of object. Being able to perform such decomposition more systematically without supervision is thus crucial as problems faced by the industry are diverse and the need for annotated data can be a limiting factor for small and medium businesses.

While a lot of effort has been made to provide useful unsupervised single-object representations that can be used for downstream tasks like classification or face recognition, tackling scenes composed of multiple objects remains a key problem to address more complex RL or VQA tasks. Being able to decompose a scene into objects and parts and providing a representation for each object and part individually could potentially help downstream models to discover relationships between components of the scene, focus on specific objects relevant to the task, track an object in a video or predict the dynamics of the scene for instance.

In recent years some progress has been made on multi-object scene decomposition and in this chapter we propose a novel effective approach that solves some problems encountered

by previous methods. Our method achieves a significant training performance boost by combining a parallel architecture with a recurrent object proposal mechanism, we use fewer hyperparameters than similar works (namely SPACE Lin et al. [2020] and SPAIR Crawford and Pineau [2019]) which work by predicting bounding boxes whose properties are predicted by an encoder network trained with a variational framework with object properties priors by removing these priors in the problem formulation. We show that our method can be used when the objects in the scene are of various sizes, and we provide a method to take into account object rotation in our scene decomposition.

Our main contributions are the following:

- We propose a novel architecture that proposes multiple objects per cell and shows that it can reduce the training cost of the model.
- The model we propose requires fewer task-dependent hyperparameters making it more general.
- It can be used for generation and can produce sensible generated images.
- We tackle the problem of various sizes of objects in a scene.
- We propose to explicitly take into account object rotation in the object proposal mechanism.

5.2 SPACE model

5.2.1 Architecture

The architecture of the model is based on SPACE Lin et al. [2020]. Let's briefly review the SPACE architecture. SPACE is composed of two modules:

- A *background module* based on GENESIS Engelcke et al. [2019] which decomposes the background into parts defined by predicted masks. Similar to Burgess et al. [2019]. While important we do not focus on this aspect of the model as we are more interested on the objects which are part of the foreground.
- A *foreground module* which decomposes the foreground into objects using a bounding box proposal mechanism. It works as follows:
    1. The input image is given to an object detector which outputs grid of "cells" each containing a bounding box properties (position & size) and the probability of presence of object in the cell. This grid approach is similar to the YOLO Redmon et al. [2015] object detector. It is fast as objects are predicted in parallel in opposition to other approaches such as Eslami et al. [2016]; Burgess et al. [2019]; Greff et al. [2019]; Engelcke et al. [2019] which uses a sequential approach. This is motivated by the authors observation that sequential approaches do not scale well for a large number of objects.
    2. The bounding boxes are then used to extract crops/glimpses of the image.

3. Objects are then encoded and decoded by a classical VAE named the object VAE. (the decoder also outputs a binary mask in addition to the object reconstruction)
4. Objects reconstructions are then projected back onto the full image to get the foreground reconstruction

5.2.2 Limitations

While SPACE adresses some limitations of SPAIR (background generation, parallel generation), it presents some limitations that we wanted to address in this work:

1. **Expensive training:** Indeed to be trained, the object VAE needs to be called one time for each cell on every image i.e. 64 times for a 8x8 grid or 256 times for a 16x16 grid. This represents an important computational cost.
2. **Poor generation capabilities:** Indeed, SPACE requires the choice of a prior on the distribution of the geometric properties of objects. In SPACE the chosen prior is unrealistic which leads to poor generation performance.

In our architecture, we keep the SPACE background module as is. However, we propose a novel foreground module that adresses some important issues of SPACE. A visual representation of the architecture can be found in Figure 5.1. Like SPACE, it has two modules: a foreground module which is responsible for the objects and a background module. The background module is a GENESIS model. Our model architecture differs from SPACE in its foreground module. This difference and its motivation are explained in sections 5.3 & 5.4.

5.3 Better performance with multiple objects' cells

We explain here the bounding boxes' proposal mechanism. First a CNN (Convolutional Neural Network) produces a latent representation $\mathbf{z}_{i,j}^{\text{cell}}$ of dimension d_{cell}. Then for each cell, a shared LSTM (Long Short Term Memory) RNN take this latent representation as inputs and outputs two latent representation $\mathbf{z}_{i,j,k}^{\text{box}}$ and $\mathbf{z}_{i,j,k}^{\text{pres}}$. Then a shared decoder takes $\mathbf{z}_{i,j,k}^{\text{box}}$ as input and outputs the position, scale, (angle) and depth of the boxes encoded **explicitly** in $\mathbf{w}^{\text{where}}$ and $\mathbf{w}^{\text{depth}}$. Another decoder takes as input $\mathbf{z}_{i,j,k}^{\text{pres}}$ and outputs the probability of the presence of an object in the boxes $\mathbf{w}^{\text{pres}}$. When used, the angle is obtained by computing the angle of a two-dimensional vector at the last layer of the box decoder. We also tried using a linear layer, but the model then failed to produce sensible box angles.

Why proposing multiple objects for each cell is more effective? To answer this question, let us consider a dataset composed of scenes containing multiple objects. We model this situation by assuming that each pixel has a fixed probability p of being the center of an object. Thus, for scenes with an average N objects distributed uniformly and P pixels, $p = N/P$. Let's divide the image into G cells, the number of objects in one cell is given by the binomial law $\mathcal{B}(P/G, p)$. In the limit of an infinite amount of pixels, this law can be approximated by a Poisson law of parameter $\lambda = N/G$. Within this approximation, the probability of finding more than K objects in one of the cells is $1 - (Q(K+1, \lambda))^G$ (with Q the cumulative distribution function of the Poisson law) thus to avoid collisions (i.e. having

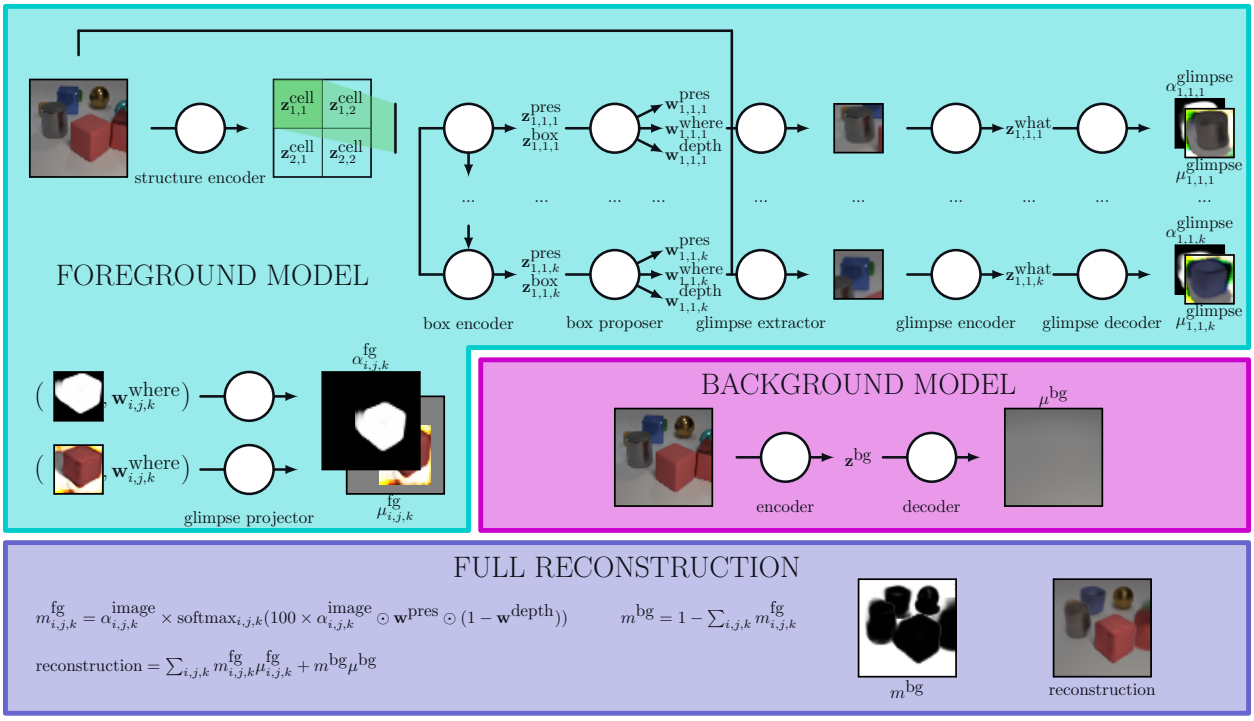

Figure 5.1: Architecture of the model. The background module (in purple) is the same as SPACE (it is a GENESIS Model). The foreground module (in cyan) begins with a *structure encoder* responsible of producing a grid of latents encoding implicitly for each cell the position, scale, depth & presence of objects. It is followed by the *box encoder* module: a RNN which takes a cell latent as inputs and produces a sequence of bounding box latents $\mathbf{z}^{\text{box}}$ and object presence probability latents $\mathbf{z}^{\text{pres}}$. These latents are then decoded by a *box proposer* module into explicit properties (i.e. probability of presence of an object in the box, its depth and its position, size, angle encoded **explicitly** in $\mathbf{w}^{\text{where}}$). Bounding boxes glimpses are then extracted from the image using a differentiable *glimpse extractor*. These glimpses are then encoded and decoded by a *glimpse encoder* and *glimpse decoder* modules. A binary mask of the object is also produced by the glimpse decoder. Finally these reconstruction are projected back on the full image by a differentiable *glimpse projector* to get the final foreground mask and reconstruction. The background and foreground can then be combined using the equations described in the blue box (More details in section 5.5).

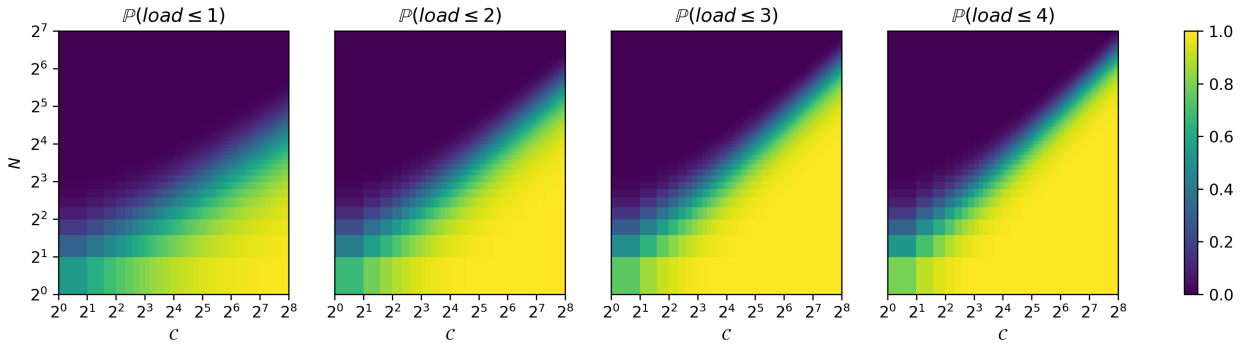

Figure 5.2: Probability that the load of one cell is less than its capacity K, $\mathcal{C} = G \times K$ is the maximum number of objects that the model can manage. N is the expected number of objects in an image of the dataset. With a constant number of regions proposed a model with a load per cell of four is less likely to get one of its cells saturated under our modelization of the object distribution. This modelization seems to describe accurately the reality considering that SPACE uses an 8×8 grid (64) cells for a dataset composed of images containing 4-8 objects and a 16x16 grid (256 cells) for a dataset with images containing 18-24 objects. Having a number of cells an order of magnitude higher than the number of objects encountered on average represents an important computational cost during training.

more than K objects in a single cell), $(Q(K+1, \lambda))^G$ should be maximized while keeping the decomposition capacity $\mathcal{C} = G \times K$ small as ultimately each bounding box content must be encoded into a latent $\mathbf{z}_{\text{what}}$ and decoded using an *object-VAE*. The number $\mathcal{C}$ is thus directly related to training cost. In Figure 5.2 we can see that the risk of overloading a cell decrease drastically for a constant $\mathcal{C}$ when K increase.

This approach can have another performance benefit. Indeed, during training the full parallel approach have to process each proposed region. For example, if an object is in one region where the network did not detect it, the gradient of the loss function should allow the network to correct its response by allowing the object-VAE to try to reconstruct the content of the box. If the object-VAE succeeds in this task, the region proposal network gradient will push it to increase the estimated probability of the presence of an object in this box. With our approach the RNN can output a variable number of regions, to allow this RNN to train, we only need to process one more region than the proposed ones to allow the network to add an object to its prediction if needed. This means that if we group 4 cells into one with an RNN capable of proposing 4 regions, with our method we only need one pass through the object-VAE instead of four for the fully parallelized method. In practice, the boxes outputted by the RNN are reordered in decreasing $\mathbf{w}^{\text{pres}}$ order. We then discard all boxes with index i, j, k such that $\mathbf{w}^{\text{pres}}_{i,j,k-1} < 0.01$ and do not run the object-VAE on these regions.

5.4 Better generation with learned geometric priors

SPACE is built around the VAE framework, where the latent is decomposed into geometric aspects (i.e. bounding box position, scale & depth) and semantic aspects (background latent & objects VAE latents). However, to perform generation, the VAE framework imposes a choice of prior distribution over the latents. SPACE chooses to use Gaussian prior for semantic latents as well as for geometric properties. This is problematic as generated images will exhibit objects whose position, size, depth will be distributed according to independent normal distribution which is in general unrealistic.

To avoid this issue we propose to learn a rich prior which can capture some aspects of the distribution of objects geometric properties. To do this we train a RNN (Recurrent Neural Network) shared among cells. The RNN takes as input the position of the cell and the latent of the background and produces a sequence of normal distributions parameters from which box latents $\mathbf{z}^{\text{box}}$, presence latents $\mathbf{z}^{\text{pres}}$ and object latents $\mathbf{z}^{\text{what}}$ are sampled and used as input for the next RNN stage. The latent $\mathbf{z}^{\text{box}}$ are then decoded into explicit geometric properties using a small neural network named the *box proposer* and $\mathbf{z}^{\text{what}}$ is decoded using the *object VAE decoder*. Using this approach it is possible to capture rich distributions of object geometric properties than can:

- take into account the relationship between the background and the foreground.
- take into account the relationship between the objects in the same cell.
- produce geometric properties that are not necessarily distributed normally.

5.5 Training framework

5.5.1 Variational framework

The model is based on the variational auto-encoder framework Kingma and Welling [2014]. We model the image as a mixture between a foreground component and a background component. We present next the losses used during training.

Reconstruction Loss

Given our modeling of the scene between several independent components, the negative log-likelihood of the original image given its reconstruction by the model is:

$$L_{\text{rec}} = -\log\left(\sum_{p}^{P} m_p^{\text{bg}}\mathcal{L}^{\text{bg}}(x_p|\pi_p, \hat{x}_p^{\text{bg}}) + (1-m_p^{\text{bg}})\mathcal{L}^{\text{fg}}(x_p|\hat{x}_p^{\text{fg}})\right) \tag{5.1}$$

With P the number of pixels, m_p^{bg} the estimated probability of a pixel to be a part of the background, $\mathcal{L}^{\text{bg}}$ and $\mathcal{L}^{\text{fg}}$ are likelihoods of real pixel values, given the model estimation, usually modeled as $\mathcal{L}(x|y) = \mathcal{N}(x; y, \sigma^2 I)$ with σ fixed. As in Lin et al. [2020]; Engelcke et al. [2019], the background model is based on Burgess et al. [2019], which models the background as a mixture of components. The $\mathcal{L}^{\text{bg}}$ has the following form:

$$\mathcal{L}^{\text{bg}}(x_p|\pi_p, \hat{x}_p^{\text{bg}}) = \sum_{k=1}^{K} (\pi_p)_k \mathcal{L}(x_p|(\hat{x}_p^{\text{bg}})_k) \tag{5.2}$$

With K the number of background components, $(\pi_p)_k$ the mask of the k$^{\text{th}}$ component of the background (i.e. the estimated probability of a pixel to be a part of the k$^{\text{th}}$ component of the background given it is a part of the background), $\mathcal{L}$ is the likelihood of the real pixel value given the pixel value estimated by the model.

Kullback-Leibler divergence loss

A key difference between our work and Lin et al. [2020] is that we do not impose a fixed prior on $\mathbf{z}^{\text{depth}}$ (the estimated depth of objects), $\mathbf{z}^{\text{where}}$ (the estimated position of objects) and $\mathbf{z}^{\text{pres}}$ (the estimated probability of the presence of an object) as their distribution is problem dependent and the priors can be hard to tune manually (see Lin et al. [2020] hyperparameters tuning). Moreover, in SPACE priors chosen for $\mathbf{z}^{\text{depth}}$, $\mathbf{z}^{\text{where}}$ and $\mathbf{z}^{\text{pres}}$ do not always reflect the real distribution of the dataset. For example, $\mathbf{z}^{\text{pres}}$ is chosen to be small to ensure that the model does not produce too many bounding boxes and $\mathbf{z}^{\text{where}}$ reflects only poorly the real position distribution of the objects in the images of the dataset. In addition, sampling independently objects and properties cannot reflect the real distribution of objects (for instance position in the image and depth are related: objects on top of the images are behind objects at the bottom of the images in CLEVR and Object 3D due to the view angle of the scene). These choices are making SPACE perform very poorly when used as a generative model although the VAE formulation is designed as such. To address these problems and to allow our model to be used as a generative model, we instead use a recurrent network to learn the prior of $\mathbf{z}^{\text{box}}$ (a latent used to generate $\mathbf{w}^{\text{depth}}$, $\mathbf{w}^{\text{where}}$) and $\mathbf{z}^{\text{pres}}$ (a latent used to generate $\mathbf{w}^{\text{pres}}$ the estimated probability of the presence of an object). This network can then be used as a generator, where our model outperforms SPACE which does not produce realistic images when used as a generator. This learned prior is similar to the network used in GENESIS however it is there used to predict bounding boxes for objects and not masks for "components". We model the foreground with the following probabilistic model:

$$q(\mathbf{z}^{\text{what}}, \mathbf{z}^{\text{pres}}, \mathbf{z}^{\text{box}}|x) = \prod_{i,j,k} q(\mathbf{z}_{i,j,k}^{\text{pres}}|x) \left(q(\mathbf{z}_{i,j,k}^{\text{box}}|x) q(\mathbf{z}_{i,j,k}^{\text{what}}|x)\right)^{\mathbf{w}_{i,j,k}^{\text{pres}}} \tag{5.3}$$

With i, j the index of the cells in the 2D grid and k the index of the object in the cell. Let's introduce the following notation for clarity $\text{KL}_{\cdots} \triangleq D_{\text{KL}}\left(p(\mathbf{z}_{i,j,k}^{\cdots}|x)||p(\mathbf{z}^{\cdots})\right)$. The Kullback-Leibler divergence term for the foreground is thus given by:

$$L_{\text{KL}}^{\text{fg}} = \sum_{i,j,k} \mathbf{w}_{i,j,k}^{\text{pres}} \left[\beta_{box}\text{KL}_{\text{box}} + \beta_{\text{what}}\text{KL}_{\text{what}}\right] + \beta_{\text{pres}}\text{KL}_{\text{pres}} \tag{5.4}$$

For the background, it is the term used in GENESIS and SPACE applied on the concatenation of the latent codes of each component. Each component is decoded from a latent $\mathbf{z}^{\text{comp}}$ and its mask is decoded from a latent $\mathbf{z}^{\text{mask}}$ resulting in a Kullback-Leibler divergence loss of the form:

$$L_{\mathrm{KL}}^{\mathrm{bg}} = \sum_{i,j,k} \beta_{\mathrm{mask}} \mathrm{KL}_{\mathrm{mask}} + \beta_{\mathrm{comp}} \mathrm{KL}_{\mathrm{comp}} \tag{5.5}$$

Where $\beta_{\mathrm{pres}}, \beta_{\mathrm{box}}, \beta_{\mathrm{what}}, \beta_{\mathrm{mask}}, \beta_{\mathrm{comp}}$ are weights introduced to be able to balance the different parts of the total loss as in Higgins et al. [2017].

5.5.2 Object-prior loss

The Kullback-Leibler term in the loss limits the amount of information carried by $\mathbf{z}^{\mathrm{box}}$ and $\mathbf{z}^{\mathrm{what}}$ together. This can be problematic as it does not encourage the model to decompose the scene into objects. This can be seen by noticing that given a "budget" of information one could use $\mathbf{z}^{\mathrm{box}}$ to encode geometric properties but can also not use $\mathbf{z}^{\mathrm{box}}$ and encode these properties in $\mathbf{z}^{\mathrm{what}}$. These two strategies will be rewarded with the same loss. As a consequence, the model could behave in pathological ways by for example only producing regions of the size of the image to reconstruct each object separately. We believe that SPACE Lin et al. [2020] circumvents this problem with the use of a small $\mathbf{w}^{\mathrm{pres}}$ prior, forcing the network to output small bounding boxes and thus preventing the issue. However, as they mentioned this prior introduces another problem, the partition of objects into multiple regions. They solve this problem by introducing a "boundary loss" specifically designed to reduce this issue.

We see multiple solutions to face this problem by encouraging the model to propose regions that match actual objects sizes:

- Give weights to the different terms of the Kullback-Leibler divergence loss to favor the use of $\mathbf{z}^{\mathrm{box}}$ when possible as $\mathbf{z}^{\mathrm{box}}$ can only act on geometric properties of the scene.
- Design an additional loss that encodes more directly what is an object and forces the model to act as desired.

The second approach is more promising for multiple reasons that we list here: first, we can design this loss precisely to encode a prior about what an object is and to guide the model more directly toward our goals. Moreover, using small weights on $\mathbf{z}^{\mathrm{box}}$ KL term could potentially harm the generation performance of the model as the distribution of $\mathbf{z}^{\mathrm{box}}$ during training is less constrained to be close to the prior Gaussian distribution used for generation. In addition, increasing the KL loss on $\mathbf{z}^{\mathrm{what}}$ could lead to less precise reconstruction due to an increased information bottleneck Higgins et al. [2017].

A similar issue arises when considering the distinction between foreground and background. However, in this case, the problem is solved by incorporating priors over the background and foreground models directly inside our choice of architectures. Here our background model is more suited to reconstruct large regions with little variations as opposed to our foreground model which can deal more easily with small detailed objects.

Object-prior-loss formulation

We think of objects as spatial entities. As such their appearance on an image depends on their position in space. In particular, their scale and position should be normalized inside

the regions proposed by the model such that the size of the bounding box is proportional to the size of the object and such that the object is centered inside the bounding box. To enforce this normalization property, we designed a loss term computed as follows:

1. The barycenter **shift** and the standard deviation **scale** of the alpha channel $\alpha^{\text{glimpse}}_{i,j,k}$ of the decoded glimpses are computed.

2. By assuming that $\alpha^{\text{glimpse}}_{i,j,k}$ is part of a bigger "$\alpha^{\text{image}}_{i,j,k}$", only defined by the parameters of the bounding box $\mathbf{w}^{\text{shift}}$ and $\mathbf{w}^{\text{scale}}$, we compute what value of $\mathbf{w}^{\text{shift}}$ and $\mathbf{w}^{\text{scale}}$ would produce a centered and normalized $\alpha^{\text{glimpse}}_{i,j,k}$. These are computed using the following formula:
$$\begin{aligned} \hat{\mathbf{w}}^{\text{shift}} &= \mathbf{w}^{\text{shift}} + \mathbf{shift} * \mathbf{w}^{\text{scale}} \\ \hat{\mathbf{w}}^{\text{scale}} &= \mathbf{w}^{\text{scale}} \times \mathbf{scale}/\sigma \end{aligned} \tag{5.6}$$
note that if $\mathbf{w}^{\text{scale}} \neq 0$, $\hat{\mathbf{w}}^{\text{shift}} = \mathbf{w}^{\text{shift}} \iff \mathbf{shift} = 0$ and $\hat{\mathbf{w}}^{\text{scale}} = \mathbf{w}^{\text{scale}} \iff \mathbf{scale} = \sigma$.

3. We then use $D_{\text{KL}}(\mathcal{N}(\mathbf{w}^{\text{shift}}, \text{diag}(\mathbf{w}^{\text{scale}})^2)||\mathcal{N}(\hat{\mathbf{w}}^{\text{shift}}, \text{diag}(\hat{\mathbf{w}}^{\text{scale}})^2))$ as a loss term to push the proposed regions to wrap nicely the objects while stopping the gradient from flowing through $\hat{\mathbf{w}}^{\text{shift}}$ and $\hat{\mathbf{w}}^{\text{scale}}$.

This loss does not make any assumptions on the distributions of scale, size and position of the objects and thus can be used without problem-specific tuning. Thus, it addresses one of the main limitations of the SPACE framework. Indeed, in the official GitHub implementation of SPACE the authors acknowledge that "Learning is difficult when object sizes vary a lot. In SPACE, we need to set a proper prior for object sizes manually and that turns out to be a crucial hyperparameter. For example, for the 10 Atari games we tested, objects are small and roughly of the same size. When object sizes vary a lot, SPACE may fail.". Our loss address this problem and we demonstrate that our method does not suffer from such drawback on the CLEVR dataset where the size of the objects varies a lot and where there are large occlusions.

Rotation loss

To add angle in the region proposal mechanism we added a loss designed to align the bounding box axes with the main axes of the objects predicted masks. This loss is computed by first computing a histogram of "white" pixels of the object mask for each angle. We then compute a discrete cosine transform of the histogram to make this histogram differentiable. We can then use this function of angle as a loss to make the model choose to align the axes of the bounding box with some geometric properties of the object.

5.5.3 Initialization and early training

We found that initialization of the model plays an important role in its behavior, in particular, we found that initializing the bias and standard deviations of the final layer of the box decoder to have near zero $L^{\text{fg}}_{\text{KL}}$ was beneficial. We also found out that beginning the training

of the different parts of the models at different moments was also beneficial. We first start the training of the background module then the training of the object encoder and decoder and ultimately the training of the box encoder, box decoder and box prior networks. By doing this, the model first learns a gross approximation of the image with the background model, then the object auto-encoder improve the reconstruction on areas that are not well captured by the background model and finally, the box proposal mechanism improves the boxes positions to minimize the number of boxes, to satisfy the Object-prior-loss and to learn a generative model of the distribution of boxes.

5.6 Duplicate detection and bounding-boxes refinement

In SPAIR, the author has used lateral connections between each cell to avoid duplicate detection. In an ablation study, they showed that these lateral connections are crucial for performance. On the other side, SPACE has not used such connections and has obtained comparable performance with SPAIR with lateral connections. They argue that these are not needed because the architecture already shares information between cells. However, in our case, we noticed that our model often produces duplicate detection at the frontier between our cells. The parallel nature of the architecture prevents the use of explicit lateral connections. We thus perform a refinement of the proposed boxes by removing boxes that have IoU (Intersection over Union) over a fixed threshold.

The method does not force the bounding boxes to be tight because of the loss we used (see Section 5.5.2). Thus, to measure the quality of the image segmentation we choose to evaluate with bounding boxes which wrap tightly the object mask found in each glimpse instead of the bounding box produced directly by the model.

5.7 Qualitative results

In Figure 5.3 and Figure 5.4 we present some qualitative results on Object3D small, Object3D large, dSprite and CLEVR datasets. Our model successfully manages to discover objects and decompose the scene in the desired way. Gray images in the "foreground reconstructed objects" correspond to patches not processed by the object-VAE saving computational resources. This is especially clear in the first column of Figure 5.3 where we can see that the model is only evaluating the object VAE for a small fraction of the proposed regions. Figure 5.4 is showing results on the dSprites dataset where we incorporated rotations in the region proposal mechanism.

5.8 Quantitative results

We compare our method against SPACE and SPAIR with usual metrics in Table 5.1. With a lower computational cost, our method performs similarly on IoU metrics, The difference is in part due to our object loss which does not encourage the bounding boxes to wrap tightly around the objects. Our method performs poorly on 3D-Room in terms of object count error rate (the percentage of times the number of predicted bounding boxes is different from the

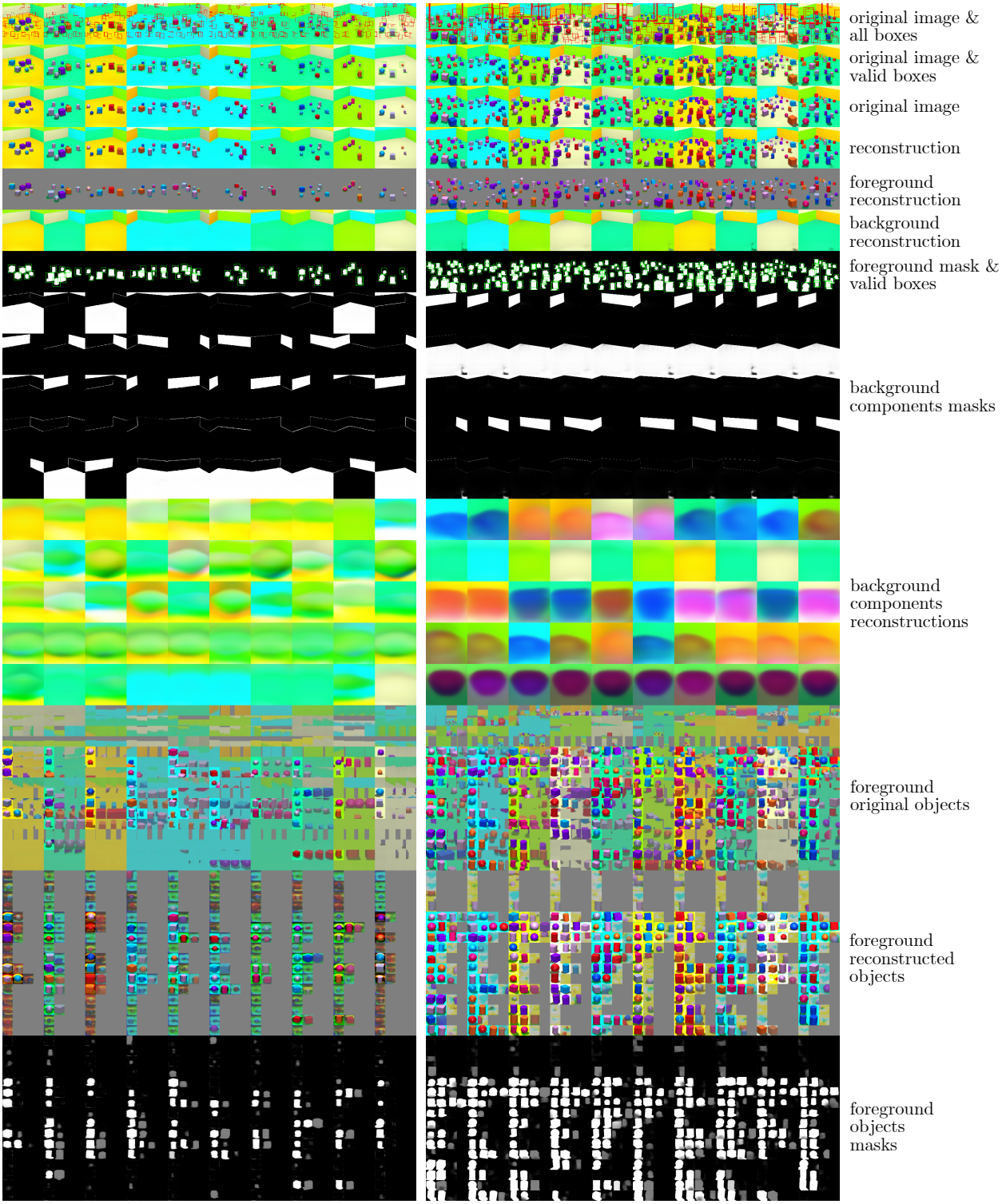

Figure 5.3: Results on 3D-Room S and 3D-Room L datasets. In gray/black skipped regions, low contrast: regions where the probability of containing an object is low (but not skipped). The model can decompose the scene into objects and reconstructs it accurately.

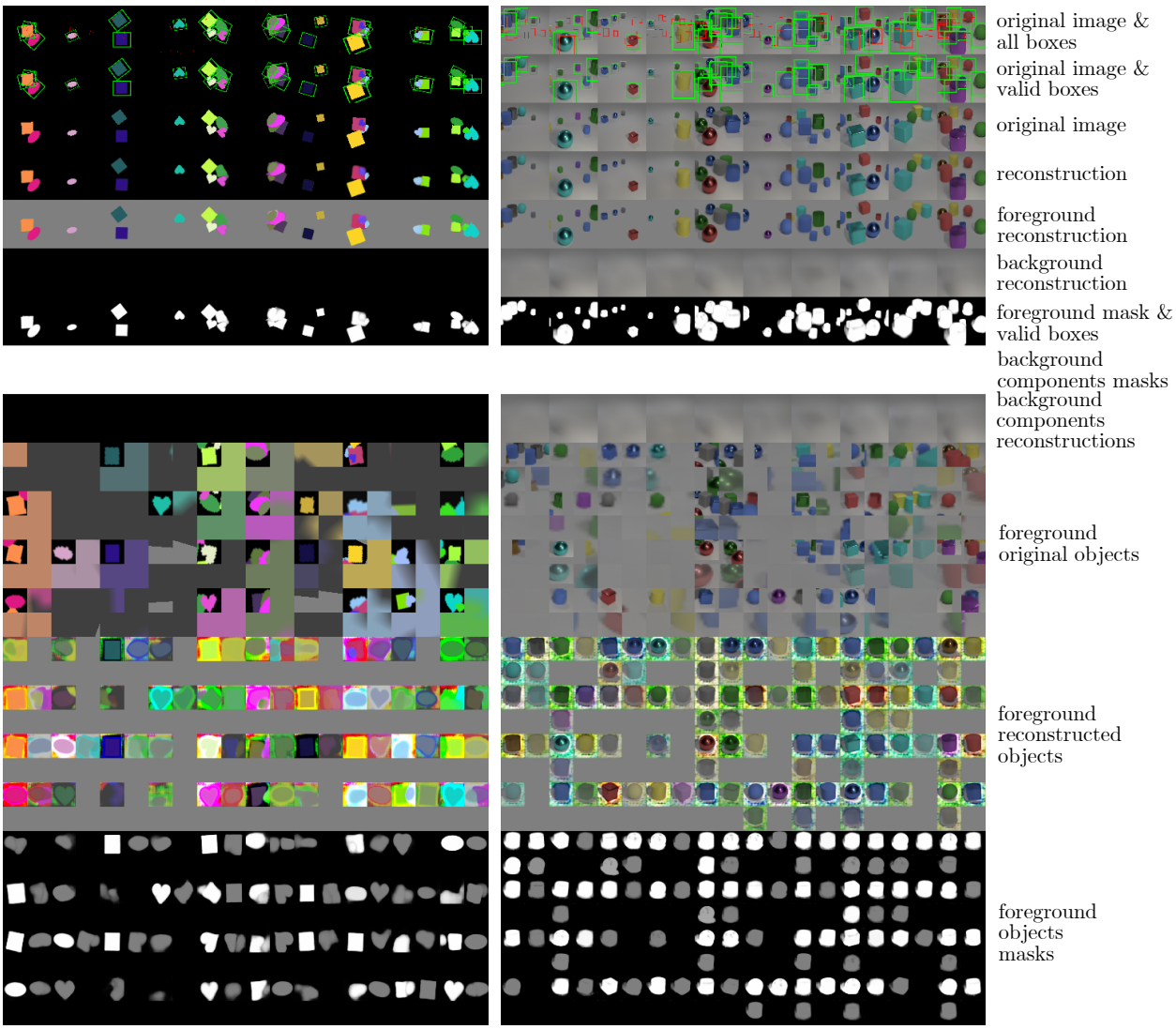

Figure 5.4: Results on dSprite and CLEVR datasets. In gray/black skipped regions, low contrast: regions where the probability of containing an object is low (but not skipped). The model can decompose the scene into objects and reconstructs it accurately.

number of objects in the scene). This is explained by the fact that our model tends to use two separate bounding boxes for an object, and its shadow.

Model	Dataset	Avg. Precision IoU T = 0.5	Avg. Precision IoU T $\in [0.5 : 0.05 : 0.95]$	Object Count Error Rate
Our ($4 \times 4 \times 4$)	3D-Room L	0.816	0.349	0.701
SPACE (16×16)	3D-Room L	0.894	0.461	0.051
SPAIR (16×16)	3D-Room L	0.908	0.445	0.043
Our ($4 \times 4 \times 4$)	3D-Room S	0.845	0.317	0.075
SPACE (16×16)	3D-Room S	0.905	0.506	0.057
SPAIR (16×16)	3D-Room S	0.904	0.488	0.026

Table 5.1: Quantitative evaluation of the model compared to results reported in Lin et al. [2020]. We adapted the code provided in the official SPACE GitHub implementation to ensure that the comparison is done on the same ground. Our model performs poorly on object count error rate as it occasionally dissociates an object, and its shadow into two objects.

5.9 Generation

In Figure 5.5 we demonstrate how our model is capable of generating scenes with sensible object distribution thanks to our learned geometric prior. SPACE was not designed with generation in mind even if it used the VAE framework to be trained and the authors fixed unrealistic (namely Gaussian) priors on the position and scale of the object, worse their prior on the scale of objects is willingly unrealistically small to help the network in producing small bounding boxes. By learning the prior our model is even capable to learn the relationship between objects and their properties for example objects at the top of the image are generally smaller and at the back of the scene this is captured by our prior.

5.10 Conclusion

Generative models are often black boxes containing millions of parameters, making the generation process quite difficult to understand and control. By making the model explicitly encode some properties of the scenes, it gains in terms of controllability and transparency over how the scene is generated. In this respect, models that decompose scenes into independent objects are a promising approach. However, such previously proposed methods have an important training cost when the amount of objects the model is capable of dealing with is high. Here we proposed to reduce this cost with a novel architecture that finds a compromise between sequential a parallel approaches. We also fill the gap between reconstruction and generation observed in some models by making the priors over geometric properties of the scene learnable. Finally, we propose new loss terms to ease the tuning of the framework to specific tasks and incorporate rotation in the geometric properties explicitly encoded by the model.

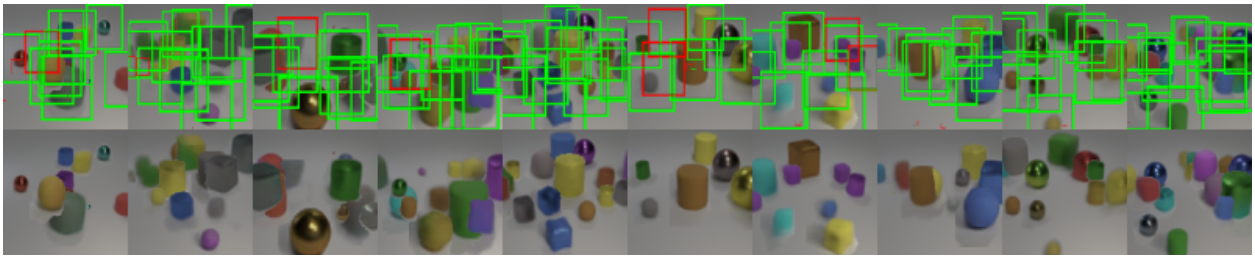

Figure 5.5: Generated samples from our model. Top row: generated images with bounding boxes Bottom row: generated images. Learning the prior of the box distribution helps to produce a sensible reconstruction in terms of objects position, scale and depth. SPACE is not designed with generation in mind, their choice of prior would produce objects with normally distributed position and scale and depth. We do not show SPACE results here as this model was not designed to perform generation even if the VAE framework used in SPACE should in theory allow us to use it as a generator.

The model presented here is based on the variational auto-encoder framework. As such it can be used to generate scenes but can also be used to encode scenes, extracting objects from the background and encoding each object separately. This decomposition of the scene is learned in an unsupervised fashion and could be used as a preprocessing for tasks where such an intermediate representation could be beneficial.

Conclusion and Perspective

Contributions summary

The three approaches proposed in this document are designed to provide greater control over the output of generative models, without the need for costly annotated data. The first approach combines VAE (variational auto-encoder) with GAN (generative adversarial network) to benefit from the controllability of VAE while maintaining the high quality of GAN-generated samples. In addition to bringing more control, our approach helps against the issue of mode collapse. The second approach involves understanding the latent space semantic structure of a generic trained image generation model, allowing for more precise control over the generated output. The last approach introduces a model architecture and training procedure that makes the model explicitly consider certain geometric properties of the scene like the position and scale of objects. By treating these properties explicitly, it is not only bringing interpretability but also controllability by making geometric properties of the scene objects trivially controllable. The two approaches presented in the first chapters have been presented in papers published at the ICPR'20 and ICLR'20 conferences.

In Chapter 3, we introduce a method for bringing the controllability of VAEs (variational auto-encoders) to GANs (generative adversarial networks). This is achieved by using the VAE latent space structure with a GAN model, while still maintaining the realism of the generated samples. The main downside of this approach is that in opposition to the work presented in Chapter 4 it requires the training of a new model and therefore cannot be applied to already existing models that have not been trained in this way.

The approach proposed in Chapter 4 is designed to identify explainable structures in the latent space of any generative model. One of the main advantages of this model is its ability to handle continuous factors of variation, allowing for precise control over these factors. However, it is somewhat limited in its ability to control geometric or global factors of variation and is therefore primarily applicable to scenarios involving a single object. Despite this limitation, the approach proposed in Chapter 4 can still be a useful complement to text-driven control methods. Indeed, the ability to control continuous factors of variation is especially important as recent text-to-image models have demonstrated the ability to efficiently control discrete factors of variation using text prompts. Controlling continuous factors of variation allows for more fine-grained control over the generated output.

Extending the work presented in Chapter 4 to scenes containing several objects can be challenging. In Chapter 5 we propose an efficient model capable of generating scenes containing numerous objects of various sizes while encoding these geometric aspects explicitly to allow more control. As for the two preceding chapters, we aimed to achieve these results

in an unsupervised way as it is a fundamental strength of generative models. Several recent works have been conducted on this topic like Lin et al. [2020] which proposes a VAE framework to tackle scenes with a lot of objects but neglects the generative aspect of the VAE framework. Our third work aimed at using a similar approach i.e. using a bounding-boxes proposal mechanism to decompose the scene but with reduced training computational cost and a more general and better probabilistic model to allow plausible scene generation. We also proposed a method to explicitly handle the angle of objects which was not present in previous works.

Contributions perspectives

In the longer term, we could try to adapt the AVAE framework to other types of representations e.g. intermediate representation of classification models. Indeed, The proposed framework is flexible enough to be used with representations that are not produced by auto-encoder models. VAEs use a pixel-wise reconstruction loss but if we are not interested in the reconstruction of pixel values but in higher-level features, like category, we can use different "reconstruction" losses (e.g. a classifier objective). Our framework remains valid in this case allowing different types of guidance for the generator. Another extension could be to use hybrid latent codes as in [Odena et al., 2017] with an interpretable part of the latent and a classic latent applying the loss $\mathcal{L}_z$ only on the interpretable part. We did not explore these possibilities during this thesis, and it may be a possible follow-up work. However, we must keep in mind that using directly interpretable representations necessitate the use of annotated data (to train the classifier or an image-to-text model like CLIP [Radford et al., 2021] for instance). On another aspect, the StyleGAN-V2 [Karras et al., 2019] hierarchical and high dimensional latent space structure could solve the issue of underestimating the image manifold dimensionality we talked about in our second work. Adapting it to an auto-encoder architecture is thus a promising perspective that we identified as without information bottleneck and a sufficiently high dimensional latent space we expect the reconstructions to be more realistic and diverse. Moreover, it would be the occasion to test our theory on a larger scale. On this topic, one can note that diffusion models' initial noise map can be interpreted as a high dimensional latent code when using deterministic samplers like DDIM Song et al. [2020a], it may be a part of the reasons for the recent success of these models. Moreover, diffusion models are trained by minimizing a variational lower-bound [Ho et al., 2020] as in the [Kingma and Welling, 2014] framework showing that it is effectively possible to get high-quality results without adversarial training. We could thus try to investigate to what extent the success of diffusion models is due to a much higher dimensionality of the latent space compared to VAEs.

The work presented in this chapter was one of the first dedicated to the interpretability of GAN latent space. Since its publication at ICLR'20 it received some interest with more than 100 citations, with notable follow-up papers like [Härkönen et al., 2020; Shen and Zhou, 2021; Voynov and Babenko, 2020]. One possible extension of the work presented in Chapter 4 could consist in finding other transformation that could be found by the method automatically. It is also interesting to discuss what "useful" directions could be found. One example that we have in mind is to use an existing face key-points detection model and a face

generation model we could apply elastic deformation to faces using key-points information to find directions in the latent space related to geometric aspects of faces allowing a finer control of the geometry of faces for a character design program.

Diffusion models

Recently, several impressive works have emerged with diffusion models Ho et al. [2020] and general text conditioning, like stable-diffusion Rombach et al. [2021] an open-source diffusion model which can generate images with text prompts as input. These works represent an important step in terms of sample quality and controllability of generative models. Indeed, they do not suffer from mode collapse and their training is stable while allowing similar or superior quality to GAN-based approaches. The sequential denoising nature of their generative process is a key advantage in terms of controllability as it offers a space where it is possible to tinker to bend the generation process toward one's goal. Their main downside resides in the cost of inference they imply due to their sequential nature.

We believe our work while developed with VAEs and GANs in mind can also benefit these models. For example, latent-diffusion models Rombach et al. [2021] could benefit from the AVAE framework for their encoder-decoder part. Indeed, latent diffusion models are a type of diffusion models that reduces the cost of training and inference of classical diffusion models by performing the inverse diffusion process in the latent space of a VAE instead of the image space. This latent space is the latent space of a VAE trained with a combination of losses which as we discussed in chapter 3 could lead to a trade-off between realism and reconstruction accuracy and we suspect that in the case of stable diffusion, the image generated could have better textures if the VAE model was trained using the method we proposed in the chapter 3 that does not suffer from this trade-off.

Techniques from Chapter 4 could be adapted to work with diffusion models. For instance, one could explore text embeddings or VAE latent to find directions that allow control of continuous factors of variation like zoom, position or luminosity in a way similar to what we did with VAEs and GANs in Chapter 4.

General perspectives on controlling generative models

One aspect of the interpretability and controllability of generative models is the disentanglement of their latent space. It is still an open question why this property manifests itself to some extent in deep generative models and how to improve the disentanglement of latent space. We believe that advancing on this subject is a promising direction.

We could also explore other ways to control diffusion models indeed Song et al. [2020a] showed that when choosing a deterministic sampling procedure, the initial noise map of diffusion models can be interpreted as a latent code. Understanding How information is encoded in this latent could allow finer control of these kinds of models and a better understanding of the reverse diffusion process in the deterministic case.

In addition, diffusion models' progressive sampling procedure nature could open new ways to control the generation, for instance, Lugmayr et al. [2022] showed a procedure to transform a generic diffusion model into an in-painting model by tweaking the sampling

procedure. In Balaji et al. [2022], the authors proposed a model that allows the user to "paint with words" conditioning the model with different prompts for different parts of the image allowing finer control of the composition. Text-conditioned generative models will certainly benefit from recent advances in language models that will allow easier and finer control with more natural text. However, controlling continuous factors of variation for this kind of model is still an open problem. We could imagine ways to weigh the importance of some words in the input text to control in a continuous fashion one aspect of the generated image. Exploring the text embedding space could also be a promising research direction. Text-to-image models could be a way to continuously explore this space structure.

The recent breakthroughs in generative artificial intelligence notably for images and text have demonstrated the fundamental importance of this aspect of deep learning. The field has progressed at an impressive pace partly thanks to the open nature of deep learning research. It is thus more crucial than ever to bring understandability and control to this class of models. We hope that this work is bringing some interesting ideas that can help to face these challenges.

Bibliography

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

R. Abdal, Y. Qin, and P. Wonka. Image2StyleGAN: How to Embed Images Into the StyleGAN Latent Space? *arXiv e-prints*, art. arXiv:1904.03189, Apr. 2019a.

R. Abdal, Y. Qin, and P. Wonka. Image2StyleGAN++: How to Edit the Embedded Images? *arXiv e-prints*, art. arXiv:1911.11544, Nov. 2019b.

M. Arjovsky and L. Bottou. Towards Principled Methods for Training Generative Adversarial Networks. *arXiv e-prints*, art. arXiv:1701.04862, Jan. 2017.

M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein GAN. *arXiv e-prints*, art. arXiv:1701.07875, Jan. 2017.

Y. Balaji, S. Nah, X. Huang, A. Vahdat, J. Song, K. Kreis, M. Aittala, T. Aila, S. Laine, B. Catanzaro, T. Karras, and M.-Y. Liu. eDiff-I: Text-to-Image Diffusion Models with an Ensemble of Expert Denoisers. *arXiv e-prints*, art. arXiv:2211.01324, Nov. 2022. doi: 10.48550/arXiv.2211.01324.

D. Bau, J.-Y. Zhu, H. Strobelt, B. Zhou, J. B. Tenenbaum, W. T. Freeman, and A. Torralba. GAN Dissection: Visualizing and Understanding Generative Adversarial Networks. *arXiv e-prints*, art. arXiv:1811.10597, 11 2018.

Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013. doi: 10.1109/TPAMI.2013.50.

A. C. Berg, T. L. Berg, H. Daumé, J. Dodge, A. Goyal, X. Han, A. Mensch, M. Mitchell, A. Sood, K. Stratos, and K. Yamaguchi. Understanding and predicting importance in images. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3562–3569, June 2012.

A. Boesen Lindbo Larsen, S. Kaae Sønderby, H. Larochelle, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. *arXiv e-prints*, art. arXiv:1512.09300, Dec 2015.

C. Bowles, L. Chen, R. Guerrero, P. Bentley, R. Gunn, A. Hammers, D. A. Dickie, M. Valdés Hernández, J. Wardlaw, and D. Rueckert. GAN Augmentation: Augmenting Training Data using Generative Adversarial Networks. *arXiv e-prints*, art. arXiv:1810.10863, Oct 2018.

A. Brock, T. Lim, J. M. Ritchie, and N. Weston. Neural Photo Editing with Introspective Adversarial Networks. In *International Conference on Learning Representations*, Apr 2017.

A. Brock, J. Donahue, and K. Simonyan. Large scale GAN training for high fidelity natural image synthesis. *CoRR*, abs/1809.11096, 2018. URL `http://arxiv.org/abs/1809.11096`.

C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner. Understanding disentangling in β-VAE. *arXiv e-prints*, art. arXiv:1804.03599, Apr 2018.

C. P. Burgess, L. Matthey, N. Watters, R. Kabra, I. Higgins, M. Botvinick, and A. Lerchner. MONet: Unsupervised Scene Decomposition and Representation. *arXiv e-prints*, art. arXiv:1901.11390, Jan. 2019.

L. Chen, S. Dai, Y. Pu, C. Li, Q. Su, and L. Carin. Symmetric Variational Autoencoder and Connections to Adversarial Learning. *arXiv e-prints*, art. arXiv:1709.01846, Sep 2017.

R. T. Chen, X. Li, R. Grosse, and D. Duvenaud. Isolating sources of disentanglement in variational autoencoders. *arXiv preprint arXiv:1802.04942*, 2018.

X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. *arXiv e-prints*, art. arXiv:1606.03657, 06 2016.

A. Cherepkov, A. Voynov, and A. Babenko. Navigating the gan parameter space for semantic image editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3670–3679, 06 2021. doi: 10.1109/CVPR46437.2021.00367.

J.-c. Chou, C.-c. Yeh, and H.-y. Lee. One-shot Voice Conversion by Separating Speaker and Content Representations with Instance Normalization. *arXiv e-prints*, art. arXiv:1904.05742, Apr. 2019.

E. Crawford and J. Pineau. Spatially invariant unsupervised object detection with convolutional neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33: 3412–3420, 07 2019. doi: 10.1609/aaai.v33i01.33013412.

A. Creswell and A. A. Bharath. Inverting The Generator Of A Generative Adversarial Network. *arXiv e-prints*, art. arXiv:1611.05644, 11 2016.

A. Creswell and A. A. Bharath. Inverting the generator of a generative adversarial network. *IEEE Transactions on Neural Networks and Learning Systems*, 30(7):1967–1974, 2019. doi: 10.1109/TNNLS.2018.2875194.

B. Dai and D. Wipf. Diagnosing and enhancing VAE models. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=B1e0X3C9tQ`.

H. de Vries, F. Strub, J. Mary, H. Larochelle, O. Pietquin, and A. Courville. Modulating early visual processing by language. *arXiv e-prints*, art. arXiv:1707.00683, Jul 2017.

P. Dhariwal and A. Nichol. Diffusion Models Beat GANs on Image Synthesis. *arXiv e-prints*, art. arXiv:2105.05233, May 2021.

L. Dinh, D. Krueger, and Y. Bengio. NICE: Non-linear Independent Components Estimation. *arXiv e-prints*, art. arXiv:1410.8516, Oct. 2014.

L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using Real NVP. *arXiv e-prints*, art. arXiv:1605.08803, May 2016.

J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial Feature Learning. *arXiv e-prints*, art. arXiv:1605.09782, May 2016.

V. Dumoulin, I. Belghazi, B. Poole, O. Mastropietro, A. Lamb, M. Arjovsky, and A. Courville. Adversarially Learned Inference. *arXiv e-prints*, art. arXiv:1606.00704, Jun 2016.

J. Engel, M. Hoffman, and A. Roberts. Latent constraints: Learning to generate conditionally from unconditional generative models. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=Sy8XvGb0-`.

M. Engelcke, A. R. Kosiorek, O. Parker Jones, and I. Posner. GENESIS: Generative Scene Inference and Sampling with Object-Centric Latent Representations. *arXiv e-prints*, art. arXiv:1907.13052, July 2019.

S. M. A. Eslami, N. Heess, T. Weber, Y. Tassa, D. Szepesvari, K. Kavukcuoglu, and G. E. Hinton. Attend, Infer, Repeat: Fast Scene Understanding with Generative Models. *arXiv e-prints*, art. arXiv:1603.08575, Mar. 2016.

P. Foret, A. Kleiner, H. Mobahi, and B. Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

P. Ghosh, M. S. M. Sajjadi, A. Vergari, M. Black, and B. Schölkopf. From Variational to Deterministic Autoencoders. *arXiv e-prints*, art. arXiv:1903.12436, Mar. 2019.

L. Goetschalckx, A. Andonian, A. Oliva, and P. Isola. GANalyze: Toward Visual Definitions of Cognitive Image Properties. *arXiv e-prints*, art. arXiv:1906.10112, Jun 2019a.

L. Goetschalckx, A. Andonian, A. Oliva, and P. Isola. GANalyze: Toward Visual Definitions of Cognitive Image Properties. *arXiv e-prints*, art. arXiv:1906.10112, Jun 2019b.

I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Networks. *arXiv e-prints*, art. arXiv:1406.2661, Jun 2014.

W. Grathwohl, R. T. Q. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud. FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models. *arXiv e-prints*, art. arXiv:1810.01367, Oct. 2018.

K. Greff, R. Lopez Kaufman, R. Kabra, N. Watters, C. Burgess, D. Zoran, L. Matthey, M. Botvinick, and A. Lerchner. Multi-Object Representation Learning with Iterative Variational Inference. *arXiv e-prints*, art. arXiv:1903.00450, Mar. 2019.

S. Guan, Y. Tai, B. Ni, F. Zhu, F. Huang, and X. Yang. Collaborative Learning for Faster StyleGAN Embedding. *arXiv e-prints*, art. arXiv:2007.01758, July 2020.

Z. Hao, A. Mallya, S. Belongie, and M.-Y. Liu. GANcraft: Unsupervised 3D Neural Rendering of Minecraft Worlds. *arXiv e-prints*, art. arXiv:2104.07659, Apr. 2021.

E. Härkönen, A. Hertzmann, J. Lehtinen, and S. Paris. GANSpace: Discovering Interpretable GAN Controls. *arXiv e-prints*, art. arXiv:2004.02546, Apr. 2020.

M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. *arXiv e-prints*, art. arXiv:1706.08500, Jun 2017.

I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. M. Botvinick, S. Mohamed, and A. Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017.

G. E. Hinton, S. Osindero, and Y.-W. Teh. A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation*, 18(7):1527–1554, 07 2006. ISSN 0899-7667. doi: 10.1162/neco.2006.18.7.1527. URL `https://doi.org/10.1162/neco.2006.18.7.1527`.

J. Ho, A. Jain, and P. Abbeel. Denoising Diffusion Probabilistic Models. *arXiv e-prints*, art. arXiv:2006.11239, June 2020.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Q. Hou, M.-M. Cheng, X.-W. Hu, A. Borji, Z. Tu, and P. Torr. Deeply supervised salient object detection with short connections. *arXiv e-prints*, art. arXiv:1611.04849, 11 2016.

H. Huang, Z. Li, R. He, Z. Sun, and T. Tan. IntroVAE: Introspective Variational Autoencoders for Photographic Image Synthesis. *arXiv e-prints*, art. arXiv:1807.06358, July 2018.

R. Huang, S. Zhang, T. Li, and R. He. Beyond Face Rotation: Global and Local Perception GAN for Photorealistic and Identity Preserving Frontal View Synthesis. *arXiv e-prints*, art. arXiv:1704.04086, Apr 2017.

X. Huang and S. Belongie. Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization. *arXiv e-prints*, art. arXiv:1703.06868, Mar 2017.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial Transformer Networks. *arXiv e-prints*, art. arXiv:1506.02025, June 2015.

A. Jahanian, L. Chai, and P. Isola. On the "steerability" of generative adversarial networks. *arXiv e-prints*, art. arXiv:1907.07171, Jul 2019.

A. Jahanian, L. Chai, and P. Isola. On the "steerability" of generative adversarial networks. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=HylsTT4FvB`.

J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. *arXiv e-prints*, art. arXiv:1603.08155, Mar 2016.

T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive Growing of GANs for Improved Quality, Stability, and Variation. *arXiv e-prints*, art. arXiv:1710.10196, Oct. 2017.

T. Karras, S. Laine, and T. Aila. A Style-Based Generator Architecture for Generative Adversarial Networks. *arXiv e-prints*, art. arXiv:1812.04948, Dec 2018.

T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila. Analyzing and Improving the Image Quality of StyleGAN. *arXiv e-prints*, art. arXiv:1912.04958, Dec. 2019.

H. Kato, D. Beker, M. Morariu, T. Ando, T. Matsuoka, W. Kehl, and A. Gaidon. Differentiable Rendering: A Survey. *arXiv e-prints*, art. arXiv:2006.12057, June 2020.

H. Kim and A. Mnih. Disentangling by Factorising. *arXiv e-prints*, art. arXiv:1802.05983, 02 2018.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

D. P. Kingma and P. Dhariwal. Glow: Generative Flow with Invertible 1x1 Convolutions. *arXiv e-prints*, art. arXiv:1807.03039, July 2018.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. In Y. Bengio and Y. LeCun, editors, *ICLR*, 2014. URL `http://dblp.uni-trier.de/db/conf/iclr/iclr2014.html#KingmaW13`.

A. Klushyn, N. Chen, R. Kurle, B. Cseke, and P. van der Smagt. Learning hierarchical priors in vaes. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*,

pages 2870–2879. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/8553-learning-hierarchical-priors-in-vaes.pdf.

E. R. Kretzmer. Statistics of television signals. *Bell System Technical Journal*, 31(4):751–763, 1952. doi: 10.1002/j.1538-7305.1952.tb01404.x.

R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, M. Bernstein, and L. Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. In *arXiv preprint arXiv:1602.07332, 2016.*, 2016.

A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015. ISSN 0036-8075. doi: 10.1126/science.aab3050. URL https://science.sciencemag.org/content/350/6266/1332.

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

W. Lee, D. Kim, S. Hong, and H. Lee. High-Fidelity Synthesis with Disentangled Representation. *arXiv e-prints*, art. arXiv:2001.04296, Jan. 2020.

C. Li, H. Liu, C. Chen, Y. Pu, L. Chen, R. Henao, and L. Carin. ALICE: Towards Understanding Adversarial Learning for Joint Distribution Matching. *arXiv e-prints*, art. arXiv:1709.01215, Sep 2017.

Z. Lin, Y.-F. Wu, S. Vishwanath Peri, W. Sun, G. Singh, F. Deng, J. Jiang, and S. Ahn. SPACE: Unsupervised Object-Oriented Scene Representation via Spatial Attention and Decomposition. *arXiv e-prints*, art. arXiv:2001.02407, Jan. 2020.

Z. C. Lipton and S. Tripathi. Precise Recovery of Latent Vectors from Generative Adversarial Networks. In *ICLR workshop*, 2017.

W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

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

A. Lugmayr, M. Danelljan, A. Romero, F. Yu, R. Timofte, and L. Van Gool. RePaint: Inpainting using Denoising Diffusion Probabilistic Models. *arXiv e-prints*, art. arXiv:2201.09865, Jan. 2022. doi: 10.48550/arXiv.2201.09865.

A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey. Adversarial Autoencoders. *arXiv e-prints*, art. arXiv:1511.05644, Nov. 2015.

E. Mathieu, T. Rainforth, N. Siddharth, and Y. Whye Teh. Disentangling Disentanglement in Variational Autoencoders. *arXiv e-prints*, art. arXiv:1812.02833, Dec. 2018.

M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv e-prints*, art. arXiv:1511.05440, Nov 2015.

L. Matthey, I. Higgins, D. Hassabis, and A. Lerchner. dsprites: Disentanglement testing sprites dataset. https://github.com/deepmind/dsprites-dataset/, 2017.

W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, 52(1):99–115, 1990. ISSN 0092-8240. doi: https://doi.org/10.1016/S0092-8240(05)80006-0. URL `https://www.sciencedirect.com/science/article/pii/S0092824005800060`.

L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein. Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163*, 2016.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient Estimation of Word Representations in Vector Space. *arXiv e-prints*, art. arXiv:1301.3781, Jan. 2013.

T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral Normalization for Generative Adversarial Networks. *arXiv e-prints*, art. arXiv:1802.05957, Feb. 2018.

Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Ng. Reading digits in natural images with unsupervised feature learning. *NIPS*, 01 2011.

A. Odena, C. Olah, and J. Shlens. Conditional image synthesis with auxiliary classifier GANs. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2642–2651. PMLR, 06–11 Aug 2017. URL `https://proceedings.mlr.press/v70/odena17a.html`.

A. V. Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. In M. F. Balcan and K. Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1747–1756, New York, New York, USA, 20–22 Jun 2016. PMLR. URL `http://proceedings.mlr.press/v48/oord16.html`.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

S. Pidhorskyi, D. Adjeroh, and G. Doretto. Adversarial Latent Autoencoders. *arXiv e-prints*, art. arXiv:2004.04467, Apr. 2020.

A. Plumerault, H. L. Borgne, and C. Hudelot. Controlling generative models with continuous factors of variations. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1laeJrKDB.

A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2015. URL http://arxiv.org/abs/1511.06434. cite arxiv:1511.06434Comment: Under review as a conference paper at ICLR 2016.

A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning Transferable Visual Models From Natural Language Supervision. *arXiv e-prints*, art. arXiv:2103.00020, Feb. 2021. doi: 10.48550/arXiv.2103.00020.

A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever. Zero-shot text-to-image generation. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8821–8831. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/ramesh21a.html.

A. Razavi, A. van den Oord, and O. Vinyals. Generating Diverse High-Fidelity Images with VQ-VAE-2. *arXiv e-prints*, art. arXiv:1906.00446, Jun 2019.

J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You Only Look Once: Unified, Real-Time Object Detection. *arXiv e-prints*, art. arXiv:1506.02640, June 2015.

R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-Resolution Image Synthesis with Latent Diffusion Models. *arXiv e-prints*, art. arXiv:2112.10752, Dec. 2021.

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

D. L. Ruderman. The statistics of natural images. *Network: Computation in Neural Systems*, 5(4):517–548, 1994. doi: 10.1088/0954-898X_5_4_006.

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

T. Salimans and J. Ho. Progressive Distillation for Fast Sampling of Diffusion Models. *arXiv e-prints*, art. arXiv:2202.00512, Feb. 2022.

T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved Techniques for Training GANs. *arXiv e-prints*, art. arXiv:1606.03498, June 2016.

A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, 1959. doi: 10.1147/rd.33.0210.

Y. Shen and B. Zhou. Closed-form factorization of latent semantics in gans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1532–1540, June 2021.

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

Y. Shen, C. Yang, X. Tang, and B. Zhou. InterFaceGAN: Interpreting the Disentangled Face Representation Learned by GANs. *arXiv e-prints*, art. arXiv:2005.09635, May 2020.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. *arXiv e-prints*, art. arXiv:1503.03585, Mar. 2015.

J. Song, C. Meng, and S. Ermon. Denoising Diffusion Implicit Models. *arXiv e-prints*, art. arXiv:2010.02502, Oct. 2020a. doi: 10.48550/arXiv.2010.02502.

Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-Based Generative Modeling through Stochastic Differential Equations. *arXiv e-prints*, art. arXiv:2011.13456, Nov. 2020b.

N. Spingarn-Eliezer, R. Banner, and T. Michaeli. GAN "Steerability" without optimization. *arXiv e-prints*, art. arXiv:2012.05328, Dec. 2020.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

T. Tieleman, G. Hinton, et al. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

R. Tolosana, R. Vera-Rodriguez, J. Fierrez, A. Morales, and J. Ortega-Garcia. DeepFakes and Beyond: A Survey of Face Manipulation and Fake Detection. *arXiv e-prints*, art. arXiv:2001.00179, Jan. 2020.

A. Turing. *The essential Turing : seminal writings in computing, logic, philosophy, artificial intelligence, and artificial life, plus the secrets of Enigma*. Clarendon Press Oxford University Press, Oxford New York, 2004. ISBN 9780198250807.

A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Journal of the London Mathematical Society*, 42:230–265, 1936. ISSN 0024-6115 (print), 1460-244X (electronic). This is the paper that introduced what is now called the *Universal Turing Machine*.

A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel Recurrent Neural Networks. *arXiv e-prints*, art. arXiv:1601.06759, Jan. 2016.

A. van den Oord, O. Vinyals, and K. Kavukcuoglu. Neural Discrete Representation Learning. *arXiv e-prints*, art. arXiv:1711.00937, Nov. 2017.

A. Voynov and A. Babenko. Unsupervised Discovery of Interpretable Directions in the GAN Latent Space. *arXiv e-prints*, art. arXiv:2002.03754, Feb. 2020.

H. Wang, G. Lin, S. C. H. Hoi, and C. Miao. Cycle-Consistent Inverse GAN for Text-to-Image Synthesis. *arXiv e-prints*, art. arXiv:2108.01361, Aug. 2021.

Z. Wang, J. Chen, and S. C. Hoi. Deep learning for image super-resolution: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 2020.

T. White. Sampling generative networks: Notes on a few effective techniques. *CoRR*, abs/1609.04468, 2016. URL `http://arxiv.org/abs/1609.04468`.

R. A. Yeh, C. Chen, T. Yian Lim, A. G. Schwing, M. Hasegawa-Johnson, and M. N. Do. Semantic image inpainting with deep generative models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *CoRR*, abs/1506.03365, 2015. URL `http://dblp.uni-trier.de/db/journals/corr/corr1506.html#YuZSSX15`.

M. D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. Metaxas. StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks. *arXiv e-prints*, art. arXiv:1612.03242, Dec. 2016a.

H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. Metaxas. StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks. *arXiv e-prints*, art. arXiv:1612.03242, Dec 2016b.

H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena. Self-attention generative adversarial networks. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7354–7363, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL `http://proceedings.mlr.press/v97/zhang19d.html`.

R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. *arXiv e-prints*, art. arXiv:1801.03924, Jan 2018.

J. Zhao, M. Mathieu, and Y. LeCun. Energy-based generative adversarial networks. In *5th International Conference on Learning Representations (ICLR 2017)*, 2019.

Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13 (4):600–612, April 2004. doi: 10.1109/TIP.2003.819861.

J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros. Generative Visual Manipulation on the Natural Image Manifold. *arXiv e-prints*, art. arXiv:1609.03552, Sept. 2016a.

J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros. Generative Visual Manipulation on the Natural Image Manifold. *arXiv e-prints*, art. arXiv:1609.03552, 09 2016b.

J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. *arXiv e-prints*, art. arXiv:1703.10593, Mar 2017.