ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ARTIFICIAL INTELLIGENCE

MASTER THESIS

in

Generative Deep Models

COMPARISON OF LATENT-SPACE GENERATIVE MODELS THROUGH STATISTICS AND MAPPING

CANDIDATE
Valerio Tonelli

SUPERVISOR
Prof. Andrea Asperti

Academic year 2021-2022

Session 3rd

Abstract

Although data generation is a task with broad and exciting applications, samples created by generative models often fall victim to reduced variability and biases which, when coupled with the lack of explainability common to all neural networks, makes the evaluation of issues and limitations of these systems challenging. Much effort has been devoted to the exploration of the latent spaces of generative models in order to find more controllable editing directions and to the idea that better models would produce more disentangled representations. In this thesis we present a detailed and comparative analysis of latent-space generative models, beginning from their theoretical foundation and up to a number of statistical and empirical findings. We show that the original data is the sole factor truly impacting how different generative models learn, more than one may imagine: under the same dataset, even very different architectures distribute their latent spaces in essentially the same way. These results suggest new directions of research for representation learning, with the potential to transfer acquired knowledge between models and to understand the common mechanisms behind learning as a whole.

Parts of the topics discussed in this thesis are a joint work, particularly those related to the mappings between models; they have already seen publication as a paper [9].

To my mom, who never stops believing in me,

To my dad, who I know would be proud of me,

To my sister and her son, who constantly warm my heart,

To my grandma, who in her pain always finds a smile for me.

To Sara, may our relationship blossom like our friendship has,

To Zanna, Eric, Alice, Samu, for the wonderful times together,

Last but not least, Michele and Caterina, here once again.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The prospect of algorithmically generating new data is one that has always exerted a strong fascination, as it carries the potential to attain an endless stream of information, to streamline complex creative tasks and to reach new boundaries in our understanding of inductive processes. Recent advances in the field have made names for themselves: neural architectures like Dall-E [72] and ChatGPT [45] have achieved stunning results, to the point that some are questioning the morality behind the usage of these systems [75] as the distinction line between user-generated and AI-generated content becomes ever more blurry.

It is clear that the effectiveness of generative techniques crucially depends on the quality of the training data, which should be prototypical of the underlying population manifold; in addition, it is key to have a well-behaved representation of this information within the generative model, where an internal encoding having more or less entangled combinations of the different explanatory factors of variation behind the data is produced [12, 50]. Unfortunately, and in spite of the huge amount of work devoted to the engineering, exploration and analysis of generative models, it is not obvious in which directions these models should be systematically pushed to attain a better representation of the data manifold and of its characteristics. Furthermore, the link between factors of variations and the semantics of expected features in generated data is not entirely clear, and issues have been raised on the concept of disentanglement as a whole [63]. Other approaches have focused on finding trajectories in the encoded space of a model in order to produce desired alterations in output, but these methods only work locally, and thus do not provide a comprehensive understanding of the knowledge acquired by a generative system.

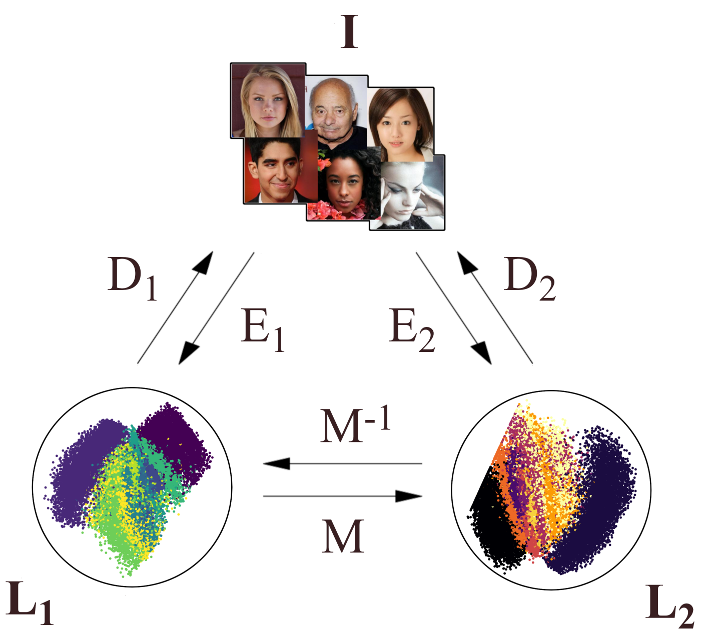
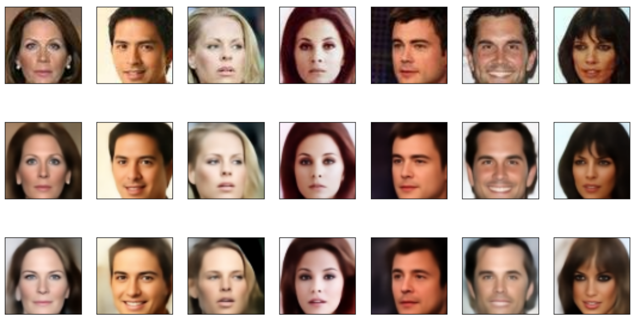


Figure 1.1: A latent-space generative model uses an encoder E_1 to compress information about data I, such as a dataset of faces, into points with lower dimensionality than that of the original data, which we name *latent* vectors (even GANs can have an encoder, see Section 2.1.2). From arbitrary points of the corresponding latent space L_1 we can generate new data samples through a decoder D_1. This latent space is semantically obfuscated but can be investigated for many properties, among which adherence to a certain shape and distances between meaningful vectors (see Section 4.2). Furthermore, given a second generative model trained on the same data and its latent space L_2, we can attempt to correlate the two spaces through a mapping M; this works surprisingly well in preserving information, as it can be seen in Figure 1.2.

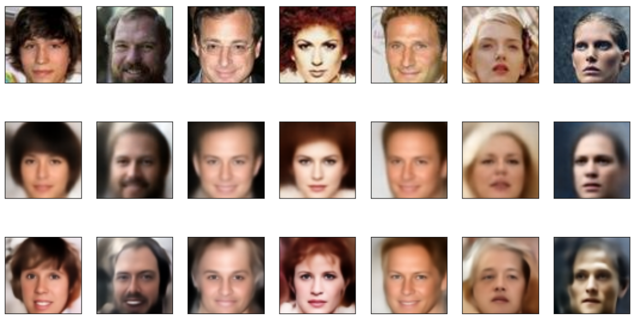
One style of analysis which, in our opinion, has seen relatively little attention is the *comparative* study of generative architectures, especially when it comes to the spaces of learned encodings. This thesis provides such an investigation: we will look at the most popular latent-space generative models, their strengths, their limitations and, most importantly, their differences and similarities. We have confined the investigation to the familiar and well-explored data manifold of human faces, and the questions we have sought to answer are the following:

- Do different generative architectures share a common understanding of data, and therefore are their latent spaces substantially equivalent up to, say, linear transformations?
- How is the latent space shaped by the neural structure and learning objectives of the model?
- How do regularization, conditioning and other secondary mechanisms influence the latent spaces?

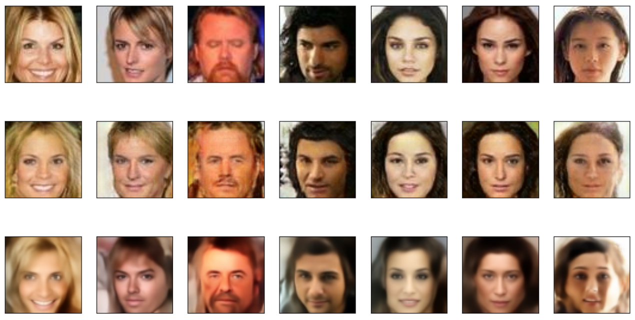
We find that different models share a strong empirical baseline under statistical measures and that the relationships between data points, such as neighbors, can be used to predict some key characteristics of the latent spaces, including the presence or lack of semantic features. Even more surprisingly, it seems to be possible to pass from the latent space of one model to that of another by means of a simple linear mapping M which preserves most of the information, so long as an encoding-decoding pair is available for both generative models; the schema in use for this experimentation is shown in Figure 1.1. While details may slightly differ, Figure 1.2 markedly shows that the overall appearance of data is mostly preserved, even when mapping between dummy and state-of-the-art architectures. Considering the complexity of any generative process, this result is quite marvelous: pairs of points related by a simple linear mapping from two different generative models are decoded by the corresponding decoders into strongly correlated—in some cases almost identical—images!



(a) Mapping between two instances of the same generative model SplitVAE [6].



(b) Mapping between two architectural variants, from VAE [53] to SplitVAE.



(c) Mapping between two entirely different models, from GAN [35] to SplitVAE.

Figure 1.2: Examples of mappings between pairs (A, B) of generative models. For all three mappings, the first row contains original samples from CelebA [62], and the second row their reconstructions from A through an encoding-decoding cycle; the third row shows the decoding from B after a linear mapping from A, as defined in Section 4.3. Results are strongly correlated for all mappings and in some cases even identifiable as the same person, especially in (a). It is also noteworthy that details are sometimes guessed or even reconstructed, particularly when going towards a higher-quality model as in (b). Unsurprisingly, the further away the two models are, the more the face structure and the details tend to be distorted, which is clearly the case in (c); nonetheless, images still remarkably similar to the originals.

Chapter 2

Background

This chapter presents the foundational knowledge upon which the thesis is then developed. In particular, it explains the ideas and current techniques behind generative modelling, as well as an overview on representation learning.

2.1 Generative Modelling

Generative modelling describes a data manifold as a probabilistic distribution, such that by sampling from this distribution we can generate new data which believably belongs to the manifold. This task implies a synthesis of information, from that of the discrete data points to a smooth probability curve, hence a generative model must gain an understanding, whether implicit or explicit, of the higher-level features that generalize a dataset. Unfortunately, this understanding cannot be trivially extracted, explored and tagged with a human-meaningful semantic under all models.

This is especially the case for the very successful field [67, 78] of Deep Generative Modelling, where part or all of the components used for the task are Deep Neural Networks (DNNs) [33, 77]. The main techniques by which DNNs have been applied to generation tasks, and particularly image generation, are as follows:

- Auto-Regressive modelling [85, 21] factorizes the distribution of the dataset over its dimensions through chain-rule and then approximates each factor with some parametric function;

- Flow modelling [55, 88, 56] such as RealNVP [28] decomposes a complex data distribution through a sequence of small, invertible functions, ultimately reshaping it into a known parametric distribution;
- Energy-based modelling [40, 82, 41] such as diffusion modelling [29] interprets the entropy in information as the energy of a physical system, and applies energy distribution laws to express the likelihood of its possible states;
- Latent-Variable modelling such as Generative Adversarial Networks (GANs) [16, 46, 70] and Variational AutoEncoders (VAEs) [54, 74, 17] learns to compress and decompress information through a latent space with lower dimensionality than that of the data and with a known distribution for its points.

Some hybrid composition of these architectures have also been attempted, with varying degrees of success [60, 68, 43].

The previous decade has seen a surge in popularity and effectiveness [8, 44] for latent-variable models—unsurprisingly, they were the dominant topic of studies in the field. This thesis will follow suit, but it is nonetheless worth mentioning that the other techniques have shown the potential to achieve similar, if not equivalent, performances [13]; diffusion models have even arguably surpassed them [27, 83], reaching peak popularity for text-to-image applications [71].

Considering latent-variable architectures allows for a variety of approaches to their analysis. Indeed, these models expose an explicit space representing their summary of the input features, for a given data element, and their understanding of general characteristics over the entire dataset; by analyzing this space we may therefore be able to decode what the model has learned about the dataset and how it summarizes its information content. This analysis can be done, as with any vector space, through a variety of tools and techniques, for instance density analysis, dimensionality reduction, or by performing comparisons among different spaces.

More in-detail, latent-variable models assume that a given dataset D can be represented as a distribution of points $p(x)$ from which we can sample inputs $x \sim p(x)$, and that this distribution is dependant on a vector z of *latent variables*. We can then reconstruct $p(x)$ via marginalization:

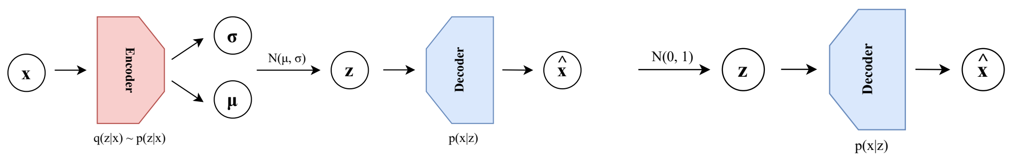
$$p(x) = \int_z p(x|z)p(z)dz = \mathbb{E}_{p(z)}[p(x|z)] \tag{2.1}$$

where z we call the latent encoding of x, which we assume is distributed with a known and parametrizable distribution $p(z)$, named the *prior distribution*. The distribution $p(x|z)$ is the likelihood, which can be approximated through a Deep Neural Network. Once learned, it can be used to generate new, original samples by approximating the expected value in Equation 2.1 with a single point, that is, we first sample from the parametric, prior distribution $z \sim p(z)$ and we then sample $\hat{x} \sim p(x|z)$.

It should be noted that with this setting, while it is simple enough to generate new samples, learning the *posterior* $p(z|x)$, in other words the distribution of the encodings of data points, usually becomes intractable. Indeed, from Bayes' Theorem:

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)} \tag{2.2}$$

But the denominator $p(x)$ is non-trivial to compute, as it is obtained through marginalization over *all* latent vectors, as per Equation 2.1.



(a) Training (b) Generation

Figure 2.1: (a) A Variational AutoEncoder is composed of an encoder and a decoder. The encoder takes in a data element x and outputs the parameters of a known distribution $q(z|x)$ such as a Gaussian, while the decoder is fed a sample taken from this distribution z and tries to reconstruct the original data $\hat{x}$. The loss is a (weighted) sum of the mean squared error between the original sample x and the reconstructed output $\hat{x}$, and of the Kullback-Leibler divergence between the distribution outputted by the encoder $q(z|x)$ and a known prior distribution for the encodings $p(z)$, usually chosen to be a standard Gaussian. (b) During generation, we directly sample from the prior $p(z)$ and pass this sample z to the decoder, skipping the encoder altogether.

2.1.1 Variational AutoEncoders

An autoencoder [33, 92] is a neural network which learns to efficiently compress inputs by learning to regenerate them from their encoding, where the rate of compression is statically chosen by fixing the dimensionality of the encodings with respect to that of the inputs. It is composed of an *encoder* producing a latent vector z from an input x and of a *decoder* which reconstructs the input $\hat{x}$ from a latent code; the resulting network is trained to regenerate inputs through a reconstruction loss $\mid x - \hat{x} \mid_2$.

A Variational AutoEncoder (VAE) [53] is a modification of an autoencoder to support generative tasks. To do so, we require the additional condition that the distribution of the encodings $p(z|x)$ is tractable. This constraint allows for meaningful generation of new samples from previously unseen encodings; however, since the likelihood $p(x|z)$ must also be learned for proper decoding, at least one of these two distributions cannot be tractable due to Equation 2.2. With some degree of approximation, the posterior $p(z|x)$ can become tractable using techniques such as Variational Inference [31]: we choose a tractable distribution $q(z|x)$, typically a Gaussian, which is then pushed towards the true posterior $p(z|x)$ via Kullback-Leibler divergence minimization (a measure of distance between distributions). The mathematical derivation of this optimization problem results in the addition of a Kullback-Leibler divergence term $KL(q(z|x) \parallel p(z))$ to the reconstruction loss for the learning process, where the true prior is the desired distribution for the latent codes, usually a standard Gaussian $\mathcal{N}(0, 1)$.

This KL term can also be seen as a regularization of the latent space of a classical autoencoder towards a chosen prior: it ensures that the data points are well dispersed over the latent space; in other words, it allows for valid sampling not only close to the encodings of the original data, but also in-between them.

While VAEs have pioneered the field of latent-space generative modelling, they tend to produce blurry results [5] as a consequence of the Gaussian assumptions [8]. A partial mitigation of this problem is possible by quantization of the latent codes (VQ-VAE [86]), and further improvements can be achieved by a careful balancing of the two loss components during learning, typically done via dynamic or learned weighting [10, 39, 19].

Layering and compounding of these models, potentially with different architectures, can also improve results [25, 36, 55], albeit at the expense of longer training and execution times. A naive yet functional approach in this direction is known as SplitVAE (or SVAE) [6]: two VAEs are run in parallel and their outputs are averaged through a learned weight map σ. This procedure serves well to reduce correlation for nearby pixels by taking information from two different sources, hence producing a more crisp image.

2.1.2 Generative Adversarial Networks

A Generative Adversarial Network (GAN) [35, 34, 70] tackles the generative task through a *generator* which, as the name suggests, directly learns how to

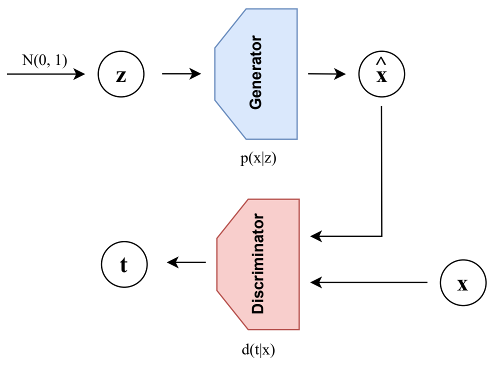


Figure 2.2: A GAN is composed of a generator and a discriminator. The generator takes in a random seed z and learns to directly create a new data sample $\hat{x}$. At the same time, the discriminator is trained to distinguish real samples x from generated ones $\hat{x}$ by outputting a classification t. The generator is trained to fool the discriminator, therefore it is encouraged to produce believable data.

produce outputs following the desired likelihood distribution $p(x|z)$. Once trained, this model can produce new data samples by taking in input a seed z sampled from a known distribution, usually a standard Gaussian $\mathcal{N}(0, 1)$.

The adjective "adversarial" comes from the presence of an opponent to the generator, called the *discriminator*, whose role is to distinguish real data samples from generated ones, generally as a classification task in a continuous range $[0; 1]$. With a parallel to game theory, the loss function is seen as the game payoff: the two components are jointly trained and they each try to push it towards opposing values. This loss function is defined as the sum between the expected value for the discriminator over true data samples $E_x[log(D(x))]$, and the expected value of one minus the discriminator prediction over generated samples $E_z[log(1 - D(G(z))]$; the discriminator attempts to maximize both terms, which corresponds to the correct labeling of samples, while the generator opposes this process by minimizing the latter term.

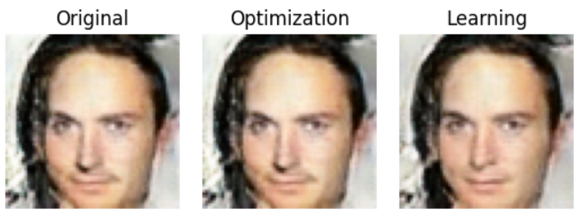
The reasoning behind this game setting is to interpret the characteristics of the data manifold as strategies that the generator can use against the discriminator, so long as it learns them. Ideally, the game ends in the generator's win: once all useful strategies are found, the generator has effectively learned the data manifold and the discriminator cannot distinguish samples anymore.

GANs have become widespread since they are simple to implement and understand [44], produce quite believable results and do not produce blurry samples, unlike VAEs [13]. Nonetheless, their training is delicate and unstable: rather than optimizing against a static solution space, the generator must find a point of equilibrium in a dynamic, ever-changing system (the one that the discriminator sets). This leads to issues of non-convergence, oscillatory patterns and early divergence in the level of game play between the two components [91, 73]. Another known issue comes in the form of mode collapse [91, 34], where the generator simplifies its strategy to a very small set of samples which work to fool the discriminator. Finally, since the loss function is not relative to the dataset but to the difference in play between the two components, it is not a good metric for the model's performance [14].

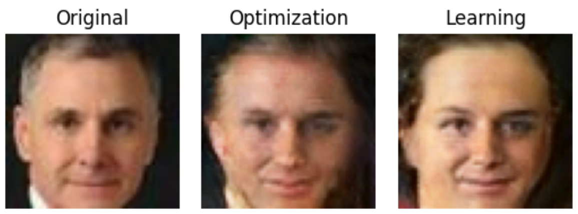
Multiple proposals have attempted to replace the standard loss [64] , since the objective function has a critical role in determining training effectiveness. These include the Wasserstein loss [4], the least squares loss [65], the unrolling of the discriminator loss over k steps [66] and the introduction of a penalty term for the discriminator [57]. As for modifications on the components and their interaction, some notable examples include GANs with attention layers [93], techniques to improve large GAN generation [16] and the cyclic conditioning of pairs of GANs on each other [96].

The flexibility in modelling and expressive power of these architectures has become especially clear with the generation of state-of-the-art, high-quality results from the application of style transfer concepts to GANs (StyleGAN and its successors [48, 49, 47]).

As a final remark for this overview, do note that GANs only learn the likelihood $p(x|z)$, therefore they do not need to tackle an intractable distribution, as VAEs must. However, this also implies that GANs cannot natively encode data points. Therefore, if, as is the case for our study, both generative and encoding processes are needed, further structure must be added to a GAN. This



(a) Example of GAN inversion using an image generated by the GAN from a randomly sampled $z \sim p(z)$. The learning-based technique achieves a quite good reconstruction of the original image, albeit it struggles with some of the finer details (e.g. lips, hair). On the other hand, the optimization-based reconstruction is virtually identical to the original image, at the cost of being much slower to compute.



(b) Example of GAN inversion using an image taken from CelebA, the same dataset on which the model was trained. Even the optimization-based method shows significant difficulty in reconstruction, demonstrating that general inversion is a non-trivial problem: the way that the GAN has incorporated the characteristics of the training dataset into its latent space is a key factor in determining the reconstruction quality.

Figure 2.3: Examples of GAN Inversions using both a Learning-based approach (a simple neural network as defined in Section 3.1.1) and an Optimization-based approach (using the BFGS algorithm [32]).

is a problem known as *GAN Inversion*.

GAN Inversion

The generator of a GAN typically uses random inputs $z \sim \mathcal{N}(0, 1)$, which can be interpreted as latent vectors being fed to a decoder-like structure. However, GANs lack an explicit encoding process for the original input sample, which is instead only seen by the discriminator. To allow encoding, an additional component has to be added to a GAN, either embedded within the architecture and jointly trained, or as an a-posteriori wrapper.

We can identify two main classes of techniques for GAN inversion. One is optimization-based approaches, where the latent code is retrieved per-image and the process is therefore targeted, but slow. Any function minimization technique can be applied here (e.g. stochastic gradient descent [23], BFGS [32], ADAM [52]), where the objective function is the distance between an original image I and its reconstruction $G(\hat{z})$. On the other end of the spectrum we have learning-based approaches, where we attempt to generalize the inversion process on a per-model basis. For instance, a neural network could be trained on images generated by a GAN $G(z)$ by tasking it to reconstruct codes $\hat{z}$ using a mean squared error loss $\mid z - \hat{z} \mid_2$, with the advantage that over-fitting is never an issue since the training set is a continuous distribution $z \sim p(z)$.

Broadly speaking, there is a quality-time trade off between these two modalities: learning-based methods are faster to compute, but may lack the finesse obtained through an image-specific optimization process. Hybrid inversion techniques can be employed to play to the strengths of both, by using the learning-based output as the initialization vector for the optimization algorithm [95, 94]. A visual comparison of typical inversion results on a vanilla GAN model can be seen in Figure 2.3a.

The quality of inversion also crucially depends on two factors: biases in the training dataset and the characteristics of the latent space. Indeed, images which are completely outside of the generative range or poorly approximated by a model will hardly be invertible. This is exemplified in Figure 2.3b.

While in principle these techniques are model-agnostic, recent works have greatly focused on the inversion of the popular StyleGAN [1, 22, 2, 69, 3], due to the potential that inversion has for editing applications.

2.2 Semantic Interpretation of Latent Spaces

During training, a generative latent-space model learns to encode data into codes, which it can then read to revert the process back to the original samples. Since different codes can produce outputs with varying characteristics, and since the decoding process is deterministic, there must be some amount of sample information hidden within the code vector. However, this data is generally obfuscated due to the nature of neural networks [33], as in, we cannot link it to a human-meaningful semantic such as symbols, attributes or labels. At the same time, knowledge of this mapping could be applied to rigorously improve the generation quality of models, for instance by modifying their loss function towards a more statistically regular encoding of their latent spaces, as well as provide a foundation to neural network explainability.

The semantic interpretation of latent spaces is therefore a problem of great interest in the field of generative modelling. Several directions of research have already been investigated, of which we present the main findings.

2.2.1 Exploration Research

One possibility to explore the space of a model is to introduce small nudges to latent vectors, based on the principle that they should correspond to small changes in the corresponding generated data and, thus, be somewhat controllable. Albeit empirical, this line of work is supported by claims of regularity (by construction for VAEs) and quasi-linearity of the latent space manifold over a single feature in GANs [61]. As we iterate this shifting process several times and we classify differences in the outputs, directions corresponding to semantically meaningful features may be found (e.g. color, pose, shape). This can be especially useful for image editing, as once such a direction is found, it can be further traveled to tweak an image, such as to add, remove or modify the degree to which a feature is present in a sample without the need to condition the generative model on labels or attributes.

InterFaceGAN [79], for instance, supposes that for any given feature taking values in $(-\infty; \infty)$ there exists an hyperplane in the latent space which is invariant for that feature. Therefore, the corresponding normal vector—which can be found e.g. via an SVM [15]—allows for a gradual modification of the feature, and further tweaking to the direction can be made to ensure it is orthogonal to other features we desire to be left untouched as much as possible. Similarly, the APCR measure [61] can find these directions with the guidance of a semantic classifier by performing an average of the direction over many different samples, either as an iterative or an optimization problem.

A different, more systemic approach to the problem is to construct these directions progressively: if a single layer i of a neural network is considered, a closed-form equation [80] can be derived to systematically find an editing direction n_i. Through their composition, the overall feature direction n can then be obtained. Other approaches of the same "arithmetic" flavor tends to be limited to smaller datasets. For instance, a non-linear variant of PCA [81] could be used to determine the most important hidden features of a latent space [89], however the semantic of these features would still be heavily reliant on human interpretation.

2.2.2 Disentanglement Research

The broad topic of disentanglement in latent-vector models is based on the concept that a more effective representation of real-world data should be summarized by a few, well-separated factors of variations [51, 87, 84, 18]. At the same time, it is argued that such a representation would also be more explainable, with roots in psychology [59] and empirical analogies with commonly seen real-world data characteristics [11], such as function smoothness, natural manifolds, sparsity of data and analytical dependencies between factors which appear in many laws of physics, biology and other fields.

Several quantitative measures for disentanglement have been proposed over the last few years [20, 30, 58], and several architectural modifications and strategies based on the idea of disentanglement have been attempted [87, 26, 90], generally producing improved results with respect to their base architectures and better downstream performances [87].

However, there is currently no commonly agreed definition of disentanglement, nor an objective measure for it. There is also research suggesting that the topic should be re-evaluated at its core: a large-scale analysis on more than ten thousand generative models [63] reported that, while different disentanglement strategies do push some level of improved generation quality when they are applied during training, visual inspection is still imperative to discern results, and typical disentanglement metrics exhibit systematic differences and inconsistencies in evaluation. The authors further challenge the idea that a more disentangled representation truly leads to better performances in downstream tasks and, most crucially, they argue that biases in data play a fundamental role in disentangled representations, rather than being simply issues to be fixed.

Chapter 3

Development

This chapter discusses the main engineering steps that have been implemented to define the generative models and prepare the data for our experiments.

3.1 Models

For our generative models, we have chosen a small set of latent-space architectures among the two most common models of this variety: VAEs and GANs.

Two of the models are simply their vanilla implementations [35, 53]. Their structures have been made as similar as possible, an intentional choice made in order to evaluate the impact of the objective function independently from the network architecture. Multiple instances of these models were trained using Adam optimization [52] on the CelebA dataset [62], after a pre-processing phase detailed in Section 3.2. The Frechét Inception Distance (FID) was used on a validation split of the dataset to determine when model improvements saturated; since this metric is typically computed on a high number of samples ($> 50k$) and the validation set was much smaller than this quantity (~ 2000), a generous delta threshold and epochs patience number were yielded to the setup.

The other two model under our consideration are state-of-the-art, pre-trained architectures of a VAE and a GAN, named SplitVAE [6] and StyleGAN

[48] respectively. These models have been chosen due to the interesting challenges in understanding the behavior of a more complex architecture. The surprising results, reported all over Section 4, demonstrate that their understanding of the original dataset is closer to that of the vanilla models than we may imagine.

3.1.1 Vanilla GAN

Our vanilla GANs [35, 24] use the standard mini-max loss function with joint training of the discriminator and generator.

The structure of the **Discriminator** is as follows:

1. A Convolutional layer going from an input of size $(64, 64, 3)$ with stride $s = 2$, "same" padding, ReLU activation, kernel size $k = 4$ and 128 channels, followed by a Leaky ReLU layer with $\alpha = 0.2$ for regularization;
2. A Convolutional layer as in $1.$ but with 256 channels, followed by another leaky ReLU;
3. A Convolutional layer as in $1.$ but with 512 channels, followed by another leaky ReLU;
4. A Dropout layer with $\alpha = 0.2$ for regularization;
5. A Dense layer outputting a single value, which is the confidence the discriminator has that its input image is real.

The structure of the **Generator** is instead the following:

1. A Dense layer going from L to size $(8, 8, 16)$;
2. A Transposed Convolutional layer with stride $s = 2$, "same" padding, ReLU activation, kernel size $k = 4$ and 128 channels, followed by a Leaky ReLU layer with $\alpha = 0.2$ for regularization;
3. A Transposed Convolutional layer as in $1.$ but with 256 channels, followed by another leaky ReLU;

4. A Transposed Convolutional layer as in $1.$ but with 512 channels, followed by another leaky ReLU;
5. A Convolutional layer with 3 channels, kernel size $k = 5$, sigmoid activation and same padding, thus producing a $(64, 64, 3)$ output.

We have also implemented a **Recoder** for GAN inversion using a neural network to reconstruct latent codes from images, which is what we detail in Section 2.1.2 as the learning-based approach to reconstruction. The network is built using the same structure as that of the generator, simply swapping inputs and outputs in order to reverse the image generation process.

3.1.2 Vanilla VAE

Our vanilla VAEs [54] use a monotonically decreasing γ factor to balance the two loss components (the KL divergence and the reconstruction error), in order to reduce blurriness and improve variability [10]. This coefficient is computed as the minimum between the current and previous estimation of the reconstruction error at each minibatch.

The structure of the **Encoder** is as follows:

1. A Convolutional layer going from an input of size $(64, 64, 3)$ with stride $s = 2$, ”same” padding, ReLU activation, kernel size $k = 4$ and 128 channels, followed by a Leaky ReLU layer with $\alpha = 0.2$ for regularization;
2. A Convolutional layer as in $1.$ but with 256 channels, followed by another leaky ReLU;
3. A Convolutional layer as in $1.$ but with 512 channels, followed by another leaky ReLU;
4. A Dropout layer with $\alpha = 0.2$ for GAN regularization;

5. Two separate Dense layers corresponding to the mean and variance vectors of the inference distribution $q(z \mid x)$ with size L, plus a third non-trainable layer which performs the sampling.

The structure of the **Decoder**, instead, is the same as the structure of the GAN generator from Section 3.1.1.

3.1.3 StyleGAN

StyleGAN [48] is a variant of a classical GAN for enabling high-quality, high-dimensional image generation. The key idea is to attempt to separate the information of an image at the large scale (e.g. pose, facial features) from the smaller, pixel-wise details (e.g. freckles, hair). To achieve this, StyleGAN first computes a style vector $w \in W$ from a seed $z \in Z$, where $p(z)$ is the classical prior and the distribution of $p(w)$ is learned by a fully connected neural network called the *Mapping* network. This first step supposedly brings the prior distribution closer to the distribution of the original data.

We therefore have two latent spaces: the prior space Z of arbitrary distribution, typically a standard Gaussian $N(0, 1)$, and the style space W. As it is customary for many inversion studies on this architecture [1, 2], we will work with the W space; among the reasons for this choice is that we presume, by construction, that it would be hopeless to map the dense *Mapping* network to a convolutional architecture.

This style vector w is then passed through several learned affine transformations (Blocks A), each of which is used to perform style transfer through Adaptive Instance Normalization (AdaIN) [42] onto a layer of the convolutional *Synthesis* network, which is the generator proper. As the initial input to this network, a simple learned constant is passed. Figure 3.1 shows the overall generative architecture of StyleGAN; the discriminator is not included in this picture since it is unvaried with respect to that of a classical GAN.

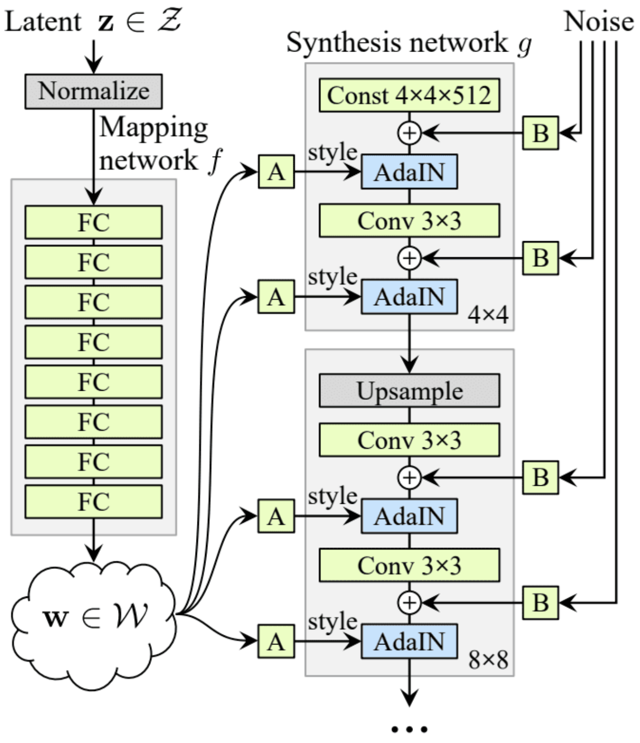


Figure 3.1: Structure of the StyleGAN generative network (picture taken from [48]). A latent vector $z \in Z$ is first passed through a dense *Mapping* network f to produce a style vector w/inW; this vector is then passed through learned affine transformations (Blocks A) to extract style information, which are fed via a style-transfer operator (Adaptive Instance Normalization, or AdaIN [42]) to each progressive convolution level of the $Synthesis$ network g. Do also notice the introduction of Noise as an additional source of randomness, which is added to the convolutions through learned scaling layers (Blocks B).

In order to enable high-quality results, training of the convolutional layers is done progressively [46]: the architecture is trained starting from down-sampled images at very low resolution, and at each progression step the input size is increased while additional layers are introduced to both StyleGAN and the discriminator. This method allows the model to focus on different levels of detail at each stage of training, similarly to an attention mechanism. Due to this multi-stage process, training of StyleGAN is unfeasible unless high-end resources are available, therefore we will use the pre-trained model available from the original work [48].

As for the recoder component that we will use for StyleGAN inversion, it will have no differences with that of the vanilla GAN recoder described in 3.1.1, except for working with a higher number of dimensions and going towards the W space, rather than the Z space.

3.1.4 SplitVAE

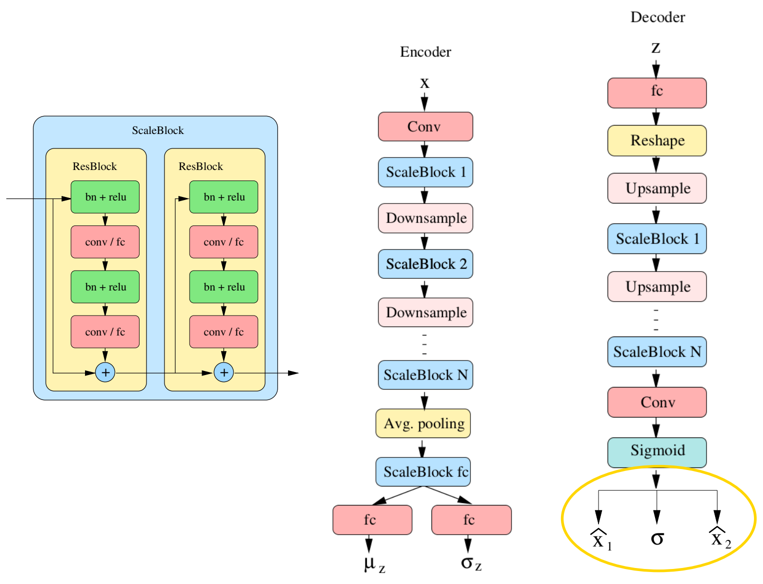
SplitVAE (SVAE) [6] is a simple extension of a traditional VAE where the output $\hat{x}$ is computed as a weighted sum

$$\hat{x} = \sigma \odot \hat{x_1} + (1 - \sigma) \odot \hat{x_2}$$

of two generated images $\hat{x_1}, \hat{x_2}$ coming from the same decoder, and a learned compositional map σ. This split, at no additional training cost, produces sharper images, with a better FID in comparison to analogous architectures and in contrast with the low-pass tendency of classical VAEs [5]. The authors attribute this improvement to the discrete nature that the split forces, causing the model to "make choices" and, hence, reduce correlation. A diverging phenomena in learning is also typically observed, where the model seems to split either the syntactical or (the more interesting case) semantical features in the data.

To improve results even further and obtain state-of-the-art performances, it is suggested to use a ResNet-like architecture derived from [25] for the implementation of both encoder and decoder components. This structure alternates the typical convolutional layers with Scale-blocks, which in turn are composed of Batch Normalization layers, non-linear units and convolutions.

The model in all of its components can be seen in Figure 4.1. For our experiments, we will use pre-trained models on CelebA provided by the original authors of SplitVAE [6].



(a) Scale-block (b) Encoder (c) Decoder

Figure 3.2: SplitVAE structure. (a) A Scale-block is a sequence of Residual-blocks intertwined with residual connections. Each Residual-block alternates Batch Normalization layers, non-linear units and convolutions. (b) The ResNet-like structure of the Encoder is similar to a typical convolutional network, with a series of down-sampling convolutional layers and a global average pooling layer at the end extracting features that are further processed via dense layers to compute mean and variance for latent variables; the only difference is the introduction of Scale-blocks between convolutions. (c) The Decoder is essentially symmetric with the encoder up to the final layer (circled in the picture) where, instead of directly producing $\hat{x}$, a Split-VAE produces two images $\hat{x_1}$ and $\hat{x_2}$ and a compositional map σ, defining $\hat{x} = \sigma \odot \hat{x_1} + (1 - \sigma) \odot \hat{x_2}$. Pictures taken from [6].

3.2 Datasets

We confined our work to the largely investigated and well-known data manifold of human faces. CelebA [62] is the dataset used in all of our experiments, including its higher-quality version CelebA-HQ [46].

In order to provide optimal training data for our models, the CelebA dataset has been pre-processed as follows: images have been aligned as per their paper [62] and then cropped to size 128×128 with a y offset of 45 and an x offset of 25 so that as much background as possible was removed while preserving face information. This crop is then downsampled to size 64×64 (with bilinear interpolation), producing the final images. The vanilla VAE and GAN models have been trained on this pre-processed dataset using a typical train/val/test split, and the SplitVAE pre-trained weights also came from this dataset.

CelebA-HQ is, on the other hand, a dataset of 30K images at resolution 1024×1024. These were obtained from a subset of CelebA with a complex methodology explained in Appendix C of Karras & Al. [46], comprising a sophisticated preprocessing phase, super-resolution techniques, and selection of best quality samples. This dataset was used as a reference for the StyleGAN pre-trained model.

Chapter 4

Experiments

This chapter presents the final experimental setup and its main results.

4.1 Experimental Setup

For our experiments we took into considerations 4 different types of models, two GANs and two VAEs; in each class, we investigated a basic, average quality ”vanilla” version and a more sophisticated, state-of-the-art model. To be precise, we have investigated the following architectures:

1. Vanilla VAE [53] using γ balancing [10] with a latent dimension Z = 64 trained on the cropped CelebA;
2. Vanilla GAN [35] with a latent dimension Z = 64 trained on the cropped CelebA;
3. SplitVAE [6] pre-trained [1] with a latent dimension Z = 150 trained on the cropped CelebA;
4. StyleGAN [48] pre-trained[2] on CelebA-HQ, which has a latent dimension Z of size 512 and a style-vector latent dimension W of the same

[1]The code for loading these models is in the repo of the paper derived from this thesis at ”We Love Latent Space” GitHub

[2]We have used the original model weights at StyleGAN-Repo and the Tensorflow 2.0 conversion repository StyleGAN-TF2-Repo.

Model	Instances	Latent dim	Resolution	FID	I-MSE
GAN	5	64	64×64	56.86	0.1971
VAE	5	64	64×64	60.78	*0.0*
SplitVAE	3	150	64×64	35.12	*0.0*
StyleGAN	1	512	1024×1024	*5.06*	0.0183

Table 4.1: Characteristics of the different models, including Dimension of the Latent Space, Image Resolution, Fréchet Inception Distance (FID) [38] and the Inverter Mean Squared Error (I-MSE), which is the error between the original and recoded latent vectors. The FID for StyleGAN is measured w.r.t. CelebA-HQ and it is not directly comparable with the others. The I-MSE for GAN architectures is that of their recoder component, while for VAE models it is technically zero since their encoding is native; in all cases, it can be disregarded, since it is a quite low value.

size.

For each one of the previous models, apart StyleGAN where we only had at our disposal a single set of pre-trained parameters, we trained and tested multiple instances. When reporting statistics for a model type, if not differently stated, they are to be understood as the average over the different trainings, or instances, of that type.

A summarizing Table 4.1 of the model types and their characteristics is provided. We note, besides the different latent dimensionality and image resolution of the models, the clear difference in Fréchet Inception Distance (FID) [38] between the vanilla and state-of-the-art models. Also, do note that the re-encoding loss (I-MSE) is not always zero, particularly for GAN architectures (see Section 2.1.2). Nonetheless, the actual values are quite low in all cases and we disregard them; this is a necessary supposition for some of our analyses, particularly those related to mappings.

4.2 Statistical Analyses

We begin our investigations by attempting to shed some light on properties of the latent spaces, such as whether they respect a Gaussian prior, the distances between pairs of significant codes and the properties of the corresponding images.

4.2.1 Normality Tests

Given that all prior distributions are typically assumed to be standard Gaussians $N(0, 1)$, it is interesting to see the degree to which the latent spaces of our models follow this distribution in practice. We can test our latent spaces for this property through the Henze-Zirkler statistic (HZ) [37], a multivariate normality test which measures the distance between the points of any distribution with respect to those of a normal distribution. The resulting test statistic should be log-normally distributed if and only if the given distribution is also normal.

We will simplify our analysis to the HZ values we obtain by considering a single set of discrete points for each test. If we attempt to sample points from the prior distribution $p(z)$ and cycle them through a decoding-encoding iteration, the resulting distribution scores a HZ close to 1.0, regardless of the underlying model. This demonstrates the efficacy of the GAN recoder and of the auto-encoding behavior of VAEs when working within the prior sampling regime.

Figure 4.1 shows the application of the statistic to the set of latent codes obtained by encoding test samples with different generative models; histograms of the first dimension for these same codes show how these values visually reflect in the density of the latent space. The latent space of the vanilla VAE is effectively a Gaussian, which is precisely the behavior enforced through the Kullback-Leibler loss. Similarly, we expect the vanilla GAN to have no such shape due to the lack of such a regularization term. However, it is surprising to note that the SplitVAE has almost the same distribution distance as the GAN, even if the SplitVAE is trained using the same loss as the VAE. A suggestive theory is that the two may be conforming to common characteristics of the dataset, hence why they can both produce sharp samples (although with different levels of quality). As for StyleGAN, it is the furthest away from a Gaussian. This also comes as a confirmation of proper training, given that the W space should be conforming to a distribution with lower dimensionality but of similar shape of the original data.

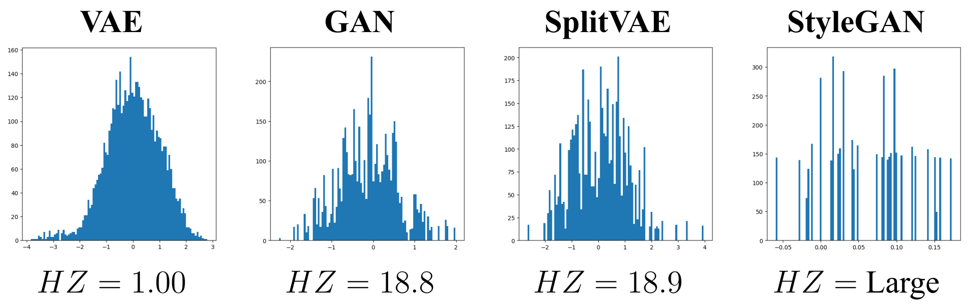


Figure 4.1: Henze-Zirkler (HZ) statistic applied to our models on a set of ~ 5000 points encoded from the test set. Accompanying plots are histogram bins of the first latent dimension. Lower values of the statistic imply a more Gaussian-shaped distribution of points, which is also reflected in each latent dimension: notice how the vanilla VAE is essentially a Gaussian, due to the KL regularization term. Interestingly, it seems that both the vanilla GAN and SplitVAE reach a similar HZ distance for the encodings of test images, suggesting a possible common property of their latent spaces. StyleGAN, on the other hand, is much less predictable, since the W space is pushed towards the original data distribution, rather than towards a chosen prior.

4.2.2 Distance Analysis

Our next step of analysis consists in comparing distant and close points of the various high-dimensional spaces we are working with and finding parallels between data and latent properties.

It is well-known that CelebA suffers from a variety of biases, such as its uneven distribution of features [76], the sampling from only a small subset in the population of human faces—indeed, those of celebrities—and the fact that the mean image of the dataset is, in itself, a face. These biases are reflected in generative models, as they learn the underlying statistics of the distribution, but are also dependant on which statistics the model is more reliant on.

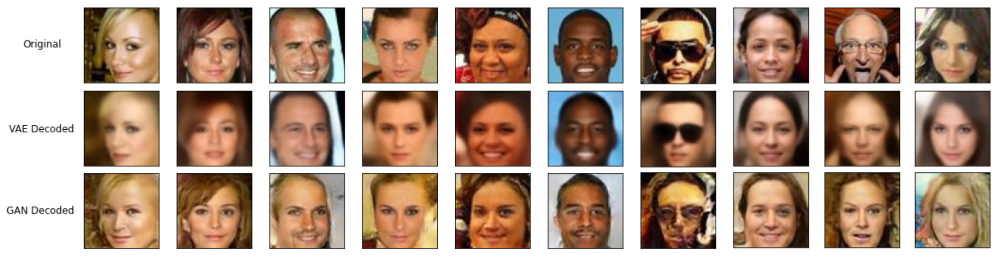


Figure 4.2: Reconstruction of images taken from CelebA, as they are encoded through a GAN or VAE into codes and then transformed back into images. The images are all clearly correlated, but while the VAE maintains all kinds of details (e.g. glasses) so long as they are macroscopic, the GAN loses such features and even invents new ones when it receives unexpected shapes, creating its characteristic aberrations.

To better see this, we can encode CelebA faces through our generative models and see how they get reconstructed. Looking at Figure 4.2, it is clear that GANs and VAEs are generally able to correlate faces with the inner understanding they have of them. For instance, VAEs focus mostly on an average and can therefore reproduce most features (but not all!), provided their scale are sufficiently large; on the other hand, GANs are strongly inclined to see only learned biased traits, to the point of replacing unknown features and traits with new data consonant to their knowledge, creating quite typical aberrations. From this point of view, these two models can be seen as opposites in a bias-variance trade-off.

Nonetheless, both models struggle significantly more on certain "hard" features which are under-represented in the population, such as hair details, glasses and hats. We ask ourselves if there is some natural, intuitive way to relate this to the characteristics of the training data. A preliminary study on the CelebA dataset reveals that, if we attempt to find the nearest neighbor of any given sample, some surprising behaviors emerge, similar to those of generative models: smaller variability than that of the original data and lack or sparsity of certain features (see Figure 4.3). If we treat the nearest neighbor algorithm akin to an encoding-decoding cycle of a generative model, it becomes suggestive to think of deeper, model-agnostic generative properties at play here. It also explains the training efficacy of data augmentation techniques, as they fill-in more of the missing distribution and allows for a better “fit” (less duplicates) by the nearest neighbors method.

To conclude this section, some numerical analyses of all spaces under consideration are provided in Table 4.2. There, we consider the average distances between random sample points and compare them with those induced by a small quantity of noise ($\sim 10\%$), and those between neighbors. We show that all generative models can well distinguish between correlated and uncorrelated points, producing the highest distance between random images and the lowest for noisy variations.

However, vanilla models do struggle significantly with noise, which can account for up to 60% of the distance between points. State-of-the-art models are shown to be much more resistant to this effect; StyleGAN, especially, is an order of magnitude more resistant than even SplitVAE, regardless of its higher latent dimensionality. Inadvertently, the model may be trained to have a low-pass behavior through the presence of noise blocks in the architecture, which are injected at each convolution step, but only at a scale smaller than the human perception of details.

Finally, we notice that none of the model performs particularly well on neighbors, if we compare these distances to those between random points. This may be a property of the data, as the Image space behaves similarly.

4.3 Mapping Experiments

Our work so far has highlighted the possibility of fundamental, common mechanisms in how models learn and encode data. One of the most interesting endeavours we propose in this work is to reach a greater level of comparison by searching for a mapping between the latent spaces of different models. Defining a map is, in itself, actually quite simple: given two models A and B trained on the same dataset, a mapper between them can be defined as a single-layer

	Random			**+Noise**			**Near**		
	64	**150**	**512**	**64**	**150**	**512**	**64**	**150**	**512**
Image (PCA)	34.34	35.65	38.11	3.70	6.00	6.15	27.59	28.72	29.91
VAE	11.12			6.52			9.61		
GAN	8.33			3.29			7.12		
SplitVAE		16.97			0.74			14.47	
StyleGAN			2.98			0.073			2.34

Table 4.2: Average Euclidean norms between pairs of points in the CelebA Image space and the latent spaces of our models. Distances are computed for a number of dimensions L (either 64, 150, or 512), in accordance to our models latent dimensions; the Image space is first reduced through PCA to L dimensions and distances are then computed, to obtain comparable measurements. Given a point randomly sampled from the dataset, the second point is either: also sampled from the same dataset (Random), the same as the first point with additive uniform noise in range $[-0.05; 0.05]$ (+Noise), or its approximate nearest neighbor in the dataset (Near).

neural network with linear activation function and no bias. This mapper can be trained, given images x from the common dataset, by passing as inputs the encodings from the first model $z_A(x)$ and by teaching it to minimize the mean squared error between the encodings from the second model $z_B(x)$ and its own output $\hat{z}_B(x)$.

Since we are working with a single-layer network, we may only require a small set of samples from the common dataset to work with, so long as it provides enough data variability. We name it our Support Set. There exists several possibilities to define this set, from variance analysis to a model-based selection of outliers [9]; we will work with the latter. If we cannot use a support set, we may of course directly work with the entire visible domain (or the subset of the image domain common to the two datasets), sampling minibatches in it. Yet another option is to disregard data entirely and work with generated samples: sampling z from the prior distribution of the first model, the corresponding generated image $x_A(z)$ can be fed to the encoder of the second model to obtain the expected output $z_B(x_a(z))$ that the mapper should get close to.

Albeit conceptually simple, mapping between different models can have

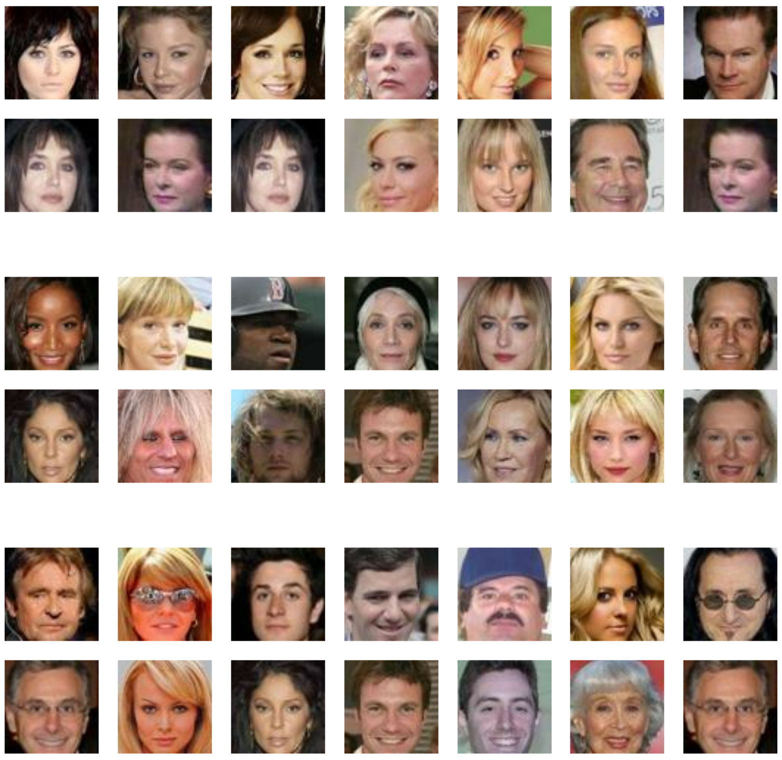


Figure 4.3: Approximate nearest neighbors between points in the CelebA dataset. First rows are random samples from the dataset, following rows are their nearest neighbors. Notice how, even on this small selection of points, some neighbors tend to be repeated multiple times, suggesting the dataset has somewhat innate clustering centroids. Notice also the distinctive lack of certain hard-to-generate features (e.g. hats) on neighbors: it seems that these features are easily lost as we move within the data space.

a lot of additional issues. Firstly, the two latent spaces may have sensibly different dimensions, e.g. 512 for StyleGAN, 150 for the SplitVAE and 64 for the vanilla models; secondly, they may work at different resolutions, for instance 1024×1024 for StyleGAN versus 64×64 for the other models.

Furthermore, given our setup, the two generative models may have been trained on the two different datasets CelebA and CelebA-HQ which, albeit similar, have different distributions of data and, most importantly, different crops. For our experiments, this is the case when considering StyleGAN as one of the models. To ensure we can still properly train the mapper in such cases, from CelebA-HQ we take a simplified crop of dimension 880×880 with an height offset of 20 and a width offset of 60, which is then downsampled to size 64×64 with bilinear interpolation, so as to make it compatible with the corresponding (and expected) CelebA image size.

4.3.1 Mapping Results

Let us come to the resulting transformations between our latent spaces. We consider mappings between all pairs of model types; for all of them we build a set of input-output pairs by encoding from generated samples (for mappings involving StyleGAN) or from the support set [9] (for all other mappings) into the two latent spaces. Then, we directly build a linear map by linear regression, minimizing the mean squared error between target and predicted latent vectors.

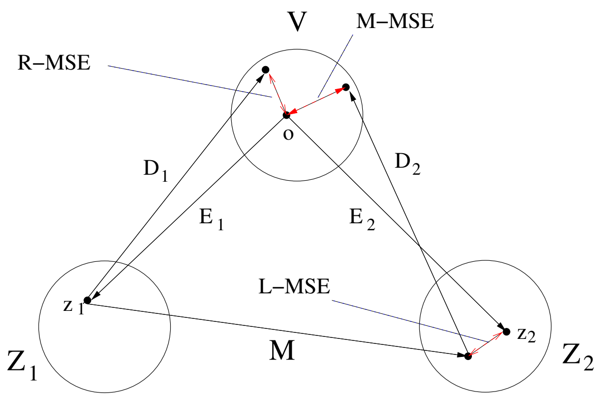


Figure 4.4: Visual explanation of the types of errors computed in Table 4.3. An image o is encoded through the source model, obtaining $z_1 = E_1(o)$. If we decode this vector back into the image space we get an approximation $\hat{o} = D_1(E_1(o))$; the error between o and $\hat{o}$ is the Reconstruction MSE (R-MSE). The encoded source latent vector can also be mapped through a learned M to obtain a predicted target vector $\hat{z}_2$: the error w.r.t. the actual target latent vector $z_2 = E_2(o)$ is the Loss MSE (L-MSE), since it is effectively the loss of the mapping M. Finally, the error between the (ideal) decoding of these two vectors is the Mapping MSE (M-MSE). Note that the I-MSE provided in Table 4.1, which would be the difference between any source latent vector z'_1 and $\hat{z}'_1 = E_1(D_1(z'_1))$, is not taken into consideration here.

To our surprise, as we pass from one latent space into another we can produce results with quite the fidelity: our mappings effectively work! Figure 1.2

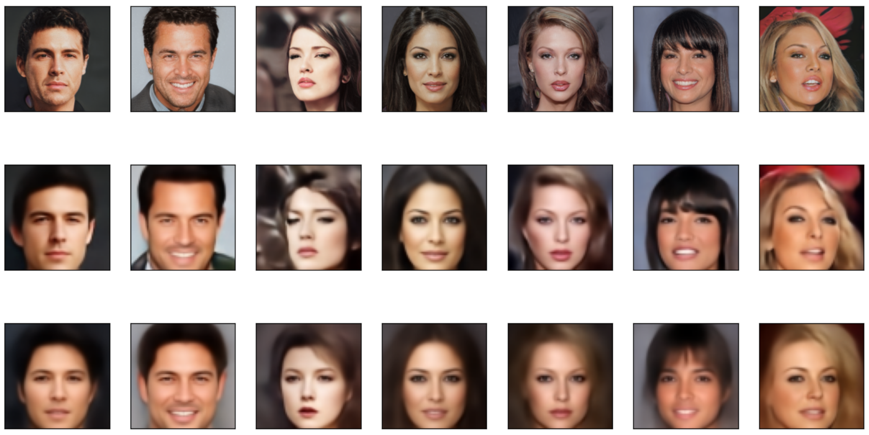
shows the comparison between an original image, its re-generation within a model (encoding-decoding loop) and the mapping onto another model, whether it is another instance of the same model (e.g. GAN to GAN), a variant architecture of the same type (e.g. VAE to SplitVAE) or two entirely different architectures (e.g. GAN to SplitVAE). Although results are not perfect, especially as we map between a state-of-the-art architecture and a vanilla one, samples are always strongly correlated and visually alike, in some cases even identifiable as the same face. Figure 4.5 shows the same process applied between the StyleGAN and SplitVAE architectures, proving that these concepts are also applicable to state-of-the-art, complex architectures.

From	**To**	**L-MSE**	**R-MSE**	**M-MSE**
VAE	VAE	0.03	0.0073	0.0103
VAE	SplitVAE	0.72		0.0105
VAE	GAN	0.49		0.0339
GAN	GAN	0.43	0.0284	0.0335
GAN	VAE	0.50		0.0254
GAN	SplitVAE	0.86		0.0275
SplitVAE	SplitVAE	0.20	0.0035	0.0067
SplitVAE	VAE	0.20		0.0125
SplitVAE	GAN	0.63		0.0388
SplitVAE	StyleGAN	0.029		0.076
StyleGAN	SplitVAE	0.45		0.014

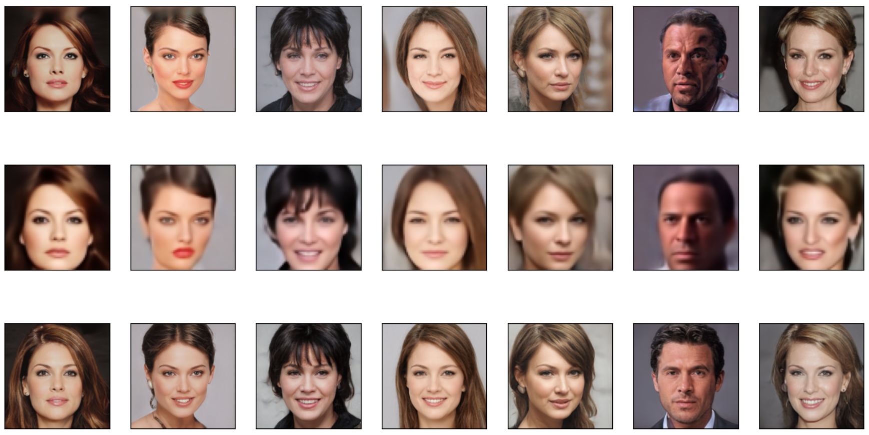
Table 4.3: Different types of errors (MSE) for all considered mapping between a source and target model. R-MSE is the error between an image from CelebA and its reconstruction for the source model; L-MSE is the error between the source and mapped latent vectors, while M-MSE is the error of their corresponding decoded images. Do refer to Figure 4.4 for a visual comparison of the three type of errors. Mappings involving StyleGAN are not directly comparable with the others, since they involve a different dataset (CelebA-HQ) and are trained on generated samples.

We provide more precise measurements in Table 4.3. As we may expect, the Mapping Mean Squared Error (M-MSE) tends to be higher the more the two models differ from each other, however there is no otherwise predictable pattern: two GANs have a mapping error very close to that from a VAE to a GAN, and swapping places between any two models can produce significant deviations in M-MSE; these findings may be partially explained by training instability for GANs. Nonetheless, all mappings are shown to be successful: for all mappings the error is always below 0.039, whereas the mean squared error between CelebA images is 0.116.

These results demonstrate a few important and general properties about learned latent embeddings. First, that all models trained on the same dataset (or arguably even similarly-natured datasets) are well-suited for comparisons among each other, even if the architectures wildly differ; in other words, there is a fundamental mechanism of learning common to all generative models. Secondly, since these mappings are linear functions, the latent spaces at their two ends must be (almost) equivalent, as one can be transformed into the other through linear operations such as rotations and scaling.



(a) Mappings from StyleGAN to SplitVAE. In the first row we have faces generated from StyleGAN by sampling vectors from a standard Gaussian $N(0, 1)$ and passing them through the dense network to obtain style vectors w, while in the second row we have the reconstruction of these images from SplitVAE after cropping and scaling and through an encoding-decoding cycle. The third row shows the mapping proper, that is, the SplitVAE decoding of the StyleGAN w vector after mapping onto the SplitVAE space.



(b) Mappings from SplitVAE to StyleGAN. Similarly to (a), the first row shows original images from StyleGAN and the second the SplitVAE reconstructions. The images from the first row are encoded onto the SplitVAE space after cropping and scaling; these z vectors are then mapped to w vectors for StyleGAN, and their decodings are shown in the third row.

Figure 4.5: Mappings between StyleGAN (W space) and SplitVAE. Original images are generated in both cases from StyleGAN since they are the training set for both mappers.

Chapter 5

Conclusions

This thesis has delved into the fascinating world of generative latent spaces, which may be described as the conceptual manifold that a generative model has of the differences and similarities of the objects it is capable of understanding, and therefore creating.

We began our discussion on the topic by providing an overview of the field of generative modelling as a whole: its opportunities for a deeper understanding of data, its exploration and disentanglement challenges, as well as the main known approaches to the problem. We further inquired about the most popular latent-space techniques by which the task can be tackled, namely Variational AutoEncoders (VAEs) and Generative Adversarial Networks (GANs). For each, we presented the base architecture and a state-of-the-art variant—SplitVAE and StyleGAN, respectively. We highlighted strengths and weaknesses for each model, both theoretically and experimentally. We formulated and implemented these four model types for generation via training on CelebA, a dataset of human faces.

We then conducted a series of experiments to test hypotheses on both the dataset and the latent spaces learned by the models. We have found that while assumptions regarding the normality of latent spaces depend on the regularization of a model, they may depend even more strongly on the data itself. These findings challenge whether a Gaussian may be the best choice for the prior distribution: StyleGAN, with its initial 8 dense layers, is explicitly engineered [48] to overcome this limitation, which may be one of the contributing factors to its high-quality results.

Further analyses were done to understand why certain features are easily lost by generative architectures, as it can be seen e.g. by passing an image through an encoding-decoding cycle, and why generated samples show less variety than the original data, even in the case of StyleGAN [9]. A key point we have raised is that if we apply a nearest neighbor algorithm to faces in the data, we find that neighbors tend to be somewhat repeated, and they are also less likely to display the same features that generative models miss. We can therefore argue that the correlation between nearby pairs of points in the dataset is what models truly learn, besides—and, possibly, even more than—the specifics of each sample. Indeed, the main difference among all tested generative models seems to lie in how they handle noise added to an image, with state-of-the-art architecture keeping the encoding of these modified samples much closer to their encodings of the originals, whereas the latent distances of random points or nearby points are approximately proportional among all models. This explanation would also provide support to the usage of data augmentation techniques, as they fill-in the neighboring space of the original points and therefore reduce this phenomenon.

These preliminary experiments are suggestive of a common, foundational mechanism by which all generative models understand their (narrow) world. One way to test for this claim is to attempt a direct map between one latent space and another via a linear transformation M, which works remarkably and surprisingly well. Since we were able to successfully find such transformations, we proved it possible to pass between two latent spaces while preserving most of the information, or in some cases even regaining it. Even more so, we showed that latent spaces of different models are essentially all the same and all seem to naturally organize themselves in a way that is merely dependent from the data manifold, so long as they are not artificially constrained to behave differently. Hence, it also follows that the organization of a latent space is largely independent from the network architecture, the training processes or the loss function. Of course, we expect this "natural" structure to be partially lost when employing strong regularization factors or when artificially reshaping the latent space (as in conditioning).

Albeit largely unexpected, our observations do agree with the recent observations [79, 61] that variations over a single semantical feature is a quasi-linear manifold in the latent space of generative models, and that biases may be an essential property of data, rather than an issue [63].

Our results are full of exciting implications for the fields of disentanglement and latent space explainability. Indeed, the fact that the latent space has a sort of inherent shape raises promising expectations about the possibility to "port" disentangled features between different spaces, and may even bring closer the comprehension of a general, shared intelligence model through an understanding of the data in a network-agnostic framework.

As for future works, they may space into many different directions and are really up to one's imagination. For instance, similar findings to ours are to be confirmed when training models on different datasets, and it may be interesting to extend this line of work into correlating the latent spaces of similar datasets (e.g. other datasets on faces). We might go even further in this direction by applying modifications to a dataset in order to understand how they affect the corresponding latent space. Another direction of research would be to attempt to empower the linear mapping with a residual layer, to "fix" deviations from the natural organization that specific models have, and based on this information to try to define a "standard" latent space for a given dataset.

We also plan to extend our findings to more types of generative models: a recent work has focused on finding embeddings for the latent space of diffusion models [7], providing a foundation from which it is theoretically possible to apply a deterministic encoding-decoding process to this class of models and, thus, study them under experiments like ours.

Data Availability

The training datasets can be found at CelebA-dataset and CelebAHQ-dataset. The code relative to this work is available on Github in the following repository: https://github.com/asperti/We_love_latent_space. We also provide pretrained weights for the models we have tested there.

Publication Disclaimer

Parts of the contents and results of this thesis have been published as a paper [9] in the journal **Neural Computing and Applications** with doi `10.1007/s00521-022-07890-2`. Credits go to the corresponding authors.

Bibliography

[1] R. Abdal, Y. Qin, and P. Wonka. Image2stylegan: how to embed images into the stylegan latent space? In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 4431–4440. IEEE, 2019. URL: `https://doi.org/10.1109/ICCV.2019.00453`.

[2] R. Abdal, Y. Qin, and P. Wonka. Image2stylegan++: how to edit the embedded images? In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8296–8305, 2020.

[3] Y. Alaluf, O. Tov, R. Mokady, R. Gal, and A. Bermano. Hyperstyle: stylegan inversion with hypernetworks for real image editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18511–18521, 2022.

[4] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223. PMLR, 2017. URL: `http://proceedings.mlr.press/v70/arjovsky17a.html`.

[5] A. Asperti. Variance loss in variational autoencoders. In *Machine Learning, Optimization, and Data Science: 6th International Conference, LOD 2020, Siena, Italy, July 19–23, 2020, Revised Selected Papers, Part I 6*, pages 297–308. Springer, 2020.

[6] A. Asperti, L. Bugo, and D. Filippini. Enhancing variational generation through self-decomposition. *IEEE Access*, 10:67510–67520, 2022. DOI: `10.1109/ACCESS.2022.3185654`. URL: `https://doi.org/10.1109/ACCESS.2022.3185654`.

[7] A. Asperti, D. Evangelista, S. Marro, and F. Merizzi. Image embedding for denoising generative models. *arXiv preprint arXiv:2301.07485*, 2022.

[8] A. Asperti, D. Evangelista, and E. L. Piccolomini. A survey on variational autoencoders from a green AI perspective. *SN Comput. Sci.*, 2(4):301, 2021. DOI: `10.1007/s42979-021-00702-9`. URL: `https://doi.org/10.1007/s42979-021-00702-9`.

[9] A. Asperti and V. Tonelli. Comparing the latent space of generative models. *Neural Computing and Applications*:1–18, 2022.

[10] A. Asperti and M. Trentin. Balancing reconstruction error and kullback-leibler divergence in variational autoencoders. *IEEE Access*, 8:199440–199448, 2020. DOI: `10.1109/ACCESS.2020.3034828`.

[11] 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.

[12] Y. Bengio, A. C. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828, 2013. DOI: `10.1109/TPAMI.2013.50`. URL: `https://doi.org/10.1109/TPAMI.2013.50`.

[13] S. Bond-Taylor, A. Leach, Y. Long, and C. G. Willcocks. Deep generative modelling: a comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models. *arXiv preprint arXiv:2103.04922*, 2021.

[14] A. Borji. Pros and cons of gan evaluation measures. *Computer Vision and Image Understanding*, 179:41–65, 2019.

[15] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.

[16] A. Brock, J. Donahue, and K. Simonyan. Large scale gan training for high fidelity natural image synthesis, arXiv, 2018. DOI: `10.48550/ARXIV.1809.11096`.

[17] Y. Burda, R. B. Grosse, and R. Salakhutdinov. Importance weighted autoencoders. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL: `http://arxiv.org/abs/1509.00519`.

[18] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner. Understanding disentangling in β-vae. *arXiv preprint arXiv:1804.03599*, 2018.

[19] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner. Understanding disentangling in β-vae. *CoRR*, abs/1804.03599, 2018. arXiv: `1804.03599`. URL: `http://arxiv.org/abs/1804.03599`.

[20] R. T. Chen, X. Li, R. Grosse, and D. Duvenaud. Isolating sources of disentanglement in vaes. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 2615–2625.

[21] X. Chen, N. Mishra, M. Rohaninejad, and P. Abbeel. Pixelsnail: an improved autoregressive generative model. In *International Conference on Machine Learning*, pages 864–872. PMLR, 2018.

[22] E. Collins, R. Bala, B. Price, and S. Susstrunk. Editing in style: uncovering the local semantics of gans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5771–5780, 2020.

[23] A. Creswell and A. A. Bharath. Inverting the generator of a generative adversarial network. *IEEE Trans. Neural Networks Learn. Syst.*, 30(7):1967–1974, 2019. URL: `https://doi.org/10.1109/TNNLS.2018.2875194`.

[24] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath. Generative adversarial networks: an overview. *IEEE Signal Processing Magazine*. IEEE, 35(1):53–65, 2018.

[25] B. Dai and D. P. Wipf. Diagnosing and enhancing vae models. In *Seventh International Conference on Learning Representations (ICLR 2019), May 6-9, New Orleans*, 2019.

[26] E. L. Denton et al. Unsupervised learning of disentangled representations from video. *Advances in neural information processing systems*, 30, 2017.

[27] P. Dhariwal and A. Nichol. Diffusion models beat gans on image synthesis. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 8780–8794. Curran Associates, Inc., 2021.

[28] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real NVP. *arXiv preprint arXiv:1605.08803*, 2016.

[29] Y. Du and I. Mordatch. Implicit generation and generalization in energy-based models. *arXiv preprint arXiv:1903.08689*, 2019.

[30] C. Eastwood and C. K. Williams. A framework for the quantitative evaluation of disentangled representations. In *International Conference on Learning Representations*, 2018.

[31] A. Ganguly and S. W. Earp. An introduction to variational inference. *arXiv preprint arXiv:2108.13083*, 2021.

[32] P. E. Gill, W. Murray, and M. H. Wright. *Practical optimization*. SIAM, 2019, pages 116–127.

[33] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. 2016. `http://www.deeplearningbook.org`.

[34] I. J. Goodfellow. NIPS 2016 tutorial: generative adversarial networks. *CoRR*, abs/1701.00160, 2017. arXiv: `1701.00160`. URL: `http://arxiv.org/abs/1701.00160`.

[35] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2672–2680, 2014.

[36] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra. DRAW: A recurrent neural network for image generation. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1462–1471. JMLR.org, 2015.

[37] N. Henze and B. Zirkler. A class of invariant consistent tests for multivariate normality. *Communications in statistics-Theory and Methods*, 19(10):3595–3617, 1990.

[38] 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, 2017. DOI: `10.48550/ARXIV.1706.08500`. URL: `https://arxiv.org/abs/1706.08500`.

[39] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. Beta-vae: learning basic visual concepts with a constrained variational framework. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL: `https://openreview.net/forum?id=Sy2fzU9gl`.

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

[41] E. Hoogeboom, J. Heek, and T. Salimans. Simple diffusion: end-to-end diffusion for high resolution images. *arXiv preprint arXiv:2301.11093*, 2023.

[42] X. Huang and S. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE international conference on computer vision*, pages 1501–1510, 2017.

[43] S. M. Iranmanesh and N. M. Nasrabadi. Hgan: hybrid generative adversarial network. *Journal of Intelligent & Fuzzy Systems*, 40(5):8927–8938, 2021.

[44] A. Jabbar, X. Li, and B. Omar. A survey on generative adversarial networks: variants, applications, and training, arXiv, 2020. DOI: `10.48550/ARXIV.2006.05132`. URL: `https://arxiv.org/abs/2006.05132`.

[45] S. John, B. Zoph, C. Kim, J. Hilton, et al. Chatgpt: optimizing language models for dialogue. 2022. URL: `https://openai.com/blog/chatgpt/` (visited on 02/20/2023).

[46] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL: `https://openreview.net/forum?id=Hk99zCeAb`.

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

[48] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

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

[50] H. Kim and A. Mnih. Disentangling by factorising. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 2654–2663, 2018. URL: `http://proceedings.mlr.press/v80/kim18b.html`.

[51] H. Kim and A. Mnih. Disentangling by factorising. In *International Conference on Machine Learning*, pages 2649–2658. PMLR, 2018.

[52] D. P. Kingma and J. Ba. Adam: a method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[53] D. P. Kingma and M. Welling. An introduction to variational autoencoders. *Found. Trends Mach. Learn.*, 12(4):307–392, 2019. DOI: `10.1561/2200000056`. URL: `https://doi.org/10.1561/2200000056`.

[54] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[55] D. P. Kingma and P. Dhariwal. Glow: generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018.

[56] I. Kobyzev, S. J. Prince, and M. A. Brubaker. Normalizing flows: an introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11):3964–3979, 2021. DOI: `10.1109/tpami.2020.2992934`.

[57] N. Kodali, J. Abernethy, J. Hays, and Z. Kira. On convergence and stability of gans. *arXiv preprint arXiv:1705.07215*, 2017. URL: `https://arxiv.org/abs/1705.07215`.

[58] A. Kumar, P. Sattigeri, and A. Balakrishnan. Variational inference of disentangled latent concepts from unlabeled observations. *arXiv preprint arXiv:1711.00848*, 2017.

[59] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40, 2017.

[60] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. In *International conference on machine learning*, pages 1558–1566. PMLR, 2016.

[61] Z. Li, R. Tao, J. Wang, F. Li, H. Niu, M. Yue, and B. Li. Interpreting the latent space of gans via measuring decoupling. *IEEE transactions on artificial intelligence*, 2(1):58–70, 2021.

[62] 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)*, pages 3730–3738, 2015.

[63] F. Locatello, S. Bauer, M. Lucic, G. Rätsch, S. Gelly, B. Schölkopf, and O. Bachem. A sober look at the unsupervised learning of disentangled representations and their evaluation. *arXiv preprint arXiv:2010.14766*, 2020.

[64] M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet. Are gans created equal? A large-scale study. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 698–707, 2018.

[65] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, Z. Wang, and S. P. Smolley. Least squares generative adversarial networks. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2813–2821. IEEE Computer Society, 2017. URL: `https://doi.org/10.1109/ICCV.2017.304`.

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

[67] A. Oussidi and A. Elhassouny. Deep generative models: survey. In *2018 International Conference on Intelligent Systems and Computer Vision (ISCV)*, pages 1–8, 2018. DOI: `10.1109/ISACV.2018.8354080`.

[68] K. Pandey, A. Mukherjee, P. Rai, and A. Kumar. Diffusevae: efficient, controllable and high-fidelity generation from low-dimensional latents. *arXiv preprint arXiv:2201.00308*, 2022.

[69] Y. Poirier-Ginter, A. Lessard, R. Smith, and J.-F. Lalonde. Overparameterization improves stylegan inversion. *arXiv preprint arXiv:2205.06304*, 2022. URL: `https://arxiv.org/abs/2205.06304`.

[70] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL: `http://arxiv.org/abs/1511.06434`.

[71] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[72] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.

[73] L. J. Ratliff, S. A. Burden, and S. S. Sastry. Characterization and computation of local nash equilibria in continuous games. In *2013 51st Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 917–924. IEEE, 2013.

[74] A. Razavi, A. Van den Oord, and O. Vinyals. Generating diverse high-fidelity images with VQ-VAE-2. *Advances in neural information processing systems*, 32, 2019.

[75] E. Ruane, A. Birhane, and A. Ventresque. Conversational ai: social and ethical considerations. In *AICS*, pages 104–115, 2019.

[76] E. M. Rudd, M. Günther, and T. E. Boult. Moon: a mixed objective optimization network for the recognition of facial attributes. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part V 14*, pages 19–35. Springer, 2016.

[77] S. J. Russell. *Artificial intelligence a modern approach*. Pearson Education, Inc., 2010, pages 727–737.

[78] T. J. Sejnowski. The unreasonable effectiveness of deep learning in artificial intelligence. *Proceedings of the National Academy of Sciences*, 117(48):30033–30038, 2020.

[79] Y. Shen, C. Yang, X. Tang, and B. Zhou. Interfacegan: interpreting the disentangled face representation learned by gans. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(4):2004–2018, 2022. DOI: `10.1109/TPAMI.2020.3034267`. URL: `https://doi.org/10.1109/TPAMI.2020.3034267`.

[80] Y. Shen and B. Zhou. Closed-form factorization of latent semantics in gans. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 1532–1540, 2021. URL: `https://openaccess.thecvf.com/content/CVPR2021/html/Shen%5C_Closed-Form%5C_Factorization%5C_of%5C_Latent%5C_Semantics%5C_in%5C_GANs%5C_CVPR%5C_2021%5C_paper.html`.

[81] J. Shlens. A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100*, 2014.

[82] Y. Song and D. P. Kingma. How to train your energy-based models. *CoRR*, abs/2101.03288, 2021. arXiv: `2101.03288`. URL: `https://arxiv.org/abs/2101.03288`.

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

[84] R. Suter, D. Miladinovic, B. Schölkopf, and S. Bauer. Robustly disentangled causal mechanisms: validating deep representations for interventional robustness. In *International Conference on Machine Learning*, pages 6056–6065. PMLR, 2019.

[85] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1747–1756. JMLR.org, 2016. URL: `http://proceedings.mlr.press/v48/oord16.html`.

[86] A. van den Oord, O. Vinyals, and K. Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing*

Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, pages 6306–6315, 2017.

[87] S. Van Steenkiste, F. Locatello, J. Schmidhuber, and O. Bachem. Are disentangled representations helpful for abstract visual reasoning? *Advances in Neural Information Processing Systems*, 32, 2019.

[88] V. Voleti, C. Finlay, A. Oberman, and C. Pal. Multi-resolution continuous normalizing flows. *arXiv preprint arXiv:2106.08462*, 2021.

[89] D. Winant, J. Schreurs, and J. A. K. Suykens. Latent space exploration using generative kernel PCA. *CoRR*, abs/2105.13949, 2021. arXiv: `2105.13949`. URL: `https://arxiv.org/abs/2105.13949`.

[90] J. Yang, S. E. Reed, M.-H. Yang, and H. Lee. Weakly-supervised disentangling with recurrent transformations for 3d view synthesis. *Advances in neural information processing systems*, 28, 2015.

[91] J. Yoo, J. Park, A. Wang, D. Mohaisen, and J. Kim. On the performance of generative adversarial network (gan) variants: a clinical data study. In *2020 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 100–104. IEEE, 2020.

[92] J. Zhai, S. Zhang, J. Chen, and Q. He. Autoencoder and its various variants. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 415–419, 2018. DOI: `10.1109/SMC.2018.00080`.

[93] H. Zhang, I. J. Goodfellow, D. N. Metaxas, and A. Odena. Self-attention generative adversarial networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 7354–7363. PMLR, 2019. URL: `http://proceedings.mlr.press/v97/zhang19d.html`.

[94] J. Zhu, Y. Shen, D. Zhao, and B. Zhou. In-domain GAN inversion for real image editing. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XVII*, volume 12362 of *Lecture Notes in Computer Science*, pages 592–608. Springer, 2020. URL: `https://doi.org/10.1007/978-3-030-58520-4%5C_35`.

[95] J. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros. Generative visual manipulation on the natural image manifold. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part V*, volume 9909 of *Lecture Notes in Computer Science*, pages 597–613. Springer, 2016. URL: `https://doi.org/10.1007/978-3-319-46454-1%5C_36`.

[96] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

Acknowledgements

I would like to thank prof. Andrea Asperti for his collaboration: this thesis would not have been possible without his knowledge, contribution, expertise and guidance. I would also like to thank Fabio Merizzi for many interesting discussions on the subject of this thesis.