Latent Space Optimization

Solving Discrete, High-dimensional, and Expensive-to-evaluate Black-box Problems via the Latent Space of Deep Generative Models

Bachelor thesis by Mark Rothermel

Date of submission: October 16, 2022

1. Review: Prof. Dr. Marc Pfetsch
2. Review:

Darmstadt

TECHNISCHE UNIVERSITÄT DARMSTADT

Mathematics Department

Research Group Optimization

Erklärung zur Abschlussarbeit gemäß § 22 Abs. 7 APB TU Darmstadt

Hiermit versichere ich, Mark Rothermel, die vorliegende Bachelorarbeit gemäß § 22 Abs. 7 APB der TU Darmstadt ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Falle eines Plagiats (§ 38 Abs. 2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei einer Thesis des Fachbereichs Architektur entspricht die eingereichte elektronische Fassung dem vorgestellten Modell und den vorgelegten Plänen.

Darmstadt, 16. Oktober 2022

[illegible]

M. Rothermel

Contents

Abstract

Discrete, high-dimensional optimization problems that have an expensive-to-evaluate black-box objective are challenging to solve. Some examples are protein design, chemical design, and neural architecture search. One heuristic approach to solve this class of problems is Latent Space Optimization (LSO). LSO is an emerging ML-based optimization method, recently applied even beyond the said design tasks. The core idea of LSO is (1) to fit a Deep Generative Model (DGM) to the distribution of the solutions, yielding a low-dimensional, continuous representation space—the latent space—and (2) to perform Bayesian optimization in that latent space using a cheap-to-evaluate surrogate function replacing the original objective.

This work gives a structured introduction to LSO and one of its extensions: weighted retraining. To this end, the underlying concepts are defined rigorously, supplemented with insightful examples to aid intuition. Moreover, the state of the art of theoretical and applied science on DGMs and LSO is reviewed. Throughout the work, the Generative Adversarial Network (GAN) serves as an exemplary DGM instance. Besides, this work completes the proof of Goodfellow et al. (2014) on the optimal GAN discriminator and points out crucial shortcomings of current GAN theory. Finally, this work examines one critical failure mode of LSO, addresses it, highlights other limitations, and discusses its broader impact.

Acknowledgements

I thank the Friedrich Naumann Foundation (FNF) for supporting me throughout my course of study with a generous scholarship. Moreover, I also thank Niklas Conen for the numerous talkative nights we've had, supporting my inspiration for this work. Besides, many thanks go to the German Academic Exchange Service (DAAD) which funded my exchange year in Illinois with a remarkable scholarship. Finally, special thanks go to the student councils of both, Mathematics and Computer Science (D120), who invest their outstanding engagement to enable a better life for all students in the respective departments—I had a wonderful time being part of their teams.

1 Introduction

In a world with exponential growth of data and computational power (as by Moore's Law), the possibilities to tackle complex real-world problems expand. As a result, since the last decade, the fields of Artificial Intelligence (AI) and Machine Learning (ML) emerged, giving birth to a powerful set of methods and tools, creating unprecedented capabilities for solving difficult tasks of a huge variety.

AI models can be roughly divided into two groups: generative models and discriminative models. The former explicitly or implicitly learn the "distribution of some entity." For example, as successfully demonstrated in previous works, generative models are capable to approximate the distribution of chemical molecules (Gómez-Bombarelli et al., 2016; Jin et al., 2018), proteins (Castro et al., 2022), fashion and virtual gaming worlds (*AI Index Report* 2022), human face images (Karras et al., 2020), etc. One of the perhaps most prominent examples is *DALL-E* (Ramesh et al., 2022), a generator that takes a textual description and turns it into an image.

A generative model can be seen as a function that maps from some lower-dimensional, continuous space into the actual data space. The smaller space is called latent space. Its advantageous properties can be utilized to heuristically solve optimization tasks that, otherwise, would have been intractable. So does Latent Space Optimization (LSO), which is going to be presented in this work. To this end, existing research on the theory and application of LSO is reviewed and a rigorous introduction to all concepts involved in LSO is given.

The rest of this chapter states basic notation definitions and specifies the fundamental type of optimization problems that we want to solve. Afterward, Chapter 2 presents Deep Generative Models (DGMs), including their status quo in current research, before going deeper into the theory of one specific, popular DGM representative: the Generative Adversarial Network (GAN). Understanding DGMs generally and GANs specifically will help to form an intuition for the concept of latent space, covered in Chapter 3. Equipped with all the required knowledge, Chapter 4–the core chapter of this work–presents the algorithm for performing Latent Space Optimization (LSO). Issues and potential broad-scale consequences of LSO will be discussed in Chapter 5 before Chapter 6 concludes this work. Besides, this work features an acronym glossary which can be found at the end.

A clear mathematical exposition of LSO in Chapter 4 doesn't necessarily require all the mathematical concepts covered in the previous two chapters, Chapters 2 and 3. Nevertheless, they aim to provide the reader with an intuitive grasp of the concept of latent space including its (qualitative) properties. Thus, DGM connoisseurs may head directly to Chapter 4 after finishing Chapter 1. However, readers with little or no knowledge about DGMs or latent space are advised to also read Chapters 2 and 3.

1.1 Basic Notation

Throughout this work, we'll use the same notation consistently, defined as follows.

Definition 1.1.1 (Data space, data points, continuous, discrete). The set $\mathcal{X}$ is called *data space* if it is a non-empty topological space. Moreover:

(1) The elements of $\mathcal{X}$ are called *data points*.

(2) We say $\mathcal{X}$ is *continuous* if $\mathcal{X} = \mathbb{R}^n$ for some $n \in \mathbb{N}$.

(3) We say $\mathcal{X}$ is *discrete* if the topological space $\mathcal{X}$ is a discrete.

Definition 1.1.2 (Objective function, value). Let $\mathcal{X}$ be some data space. The function $f : \mathcal{X} \to \mathbb{R}$ which assigns a real value to each data point $\mathbf{x} \in \mathcal{X}$ is called *objective function*. Furthermore, we call $f(x)$ the *value of* $\mathbf{x} \in \mathcal{X}$.

1.2 Data Generation Process (DGP) & True Distribution

We start with an example. Consider the data space $\mathcal{X} := [0, 255]^{6000\times4000\times3}$ of RGB images with 6000×4000 pixels. If we take a camera that has a sensor with the said resolution, go to some nice viewpoint and take a picture from the landscape, we receive an image $\mathbf{x} \in \mathcal{X}$. This situation is an instance of a *Data Generation Process (DGP)* (King, 2020) where the image $\mathbf{x}$ is a result of the DGP, see also Figure 1.1. We use $\mathscr{D}$ to denote a DGP. In short, a

DGP, as the term suggests, is an abstract concept that randomly produces/generates data points (that are elements of $\mathcal{X}$).

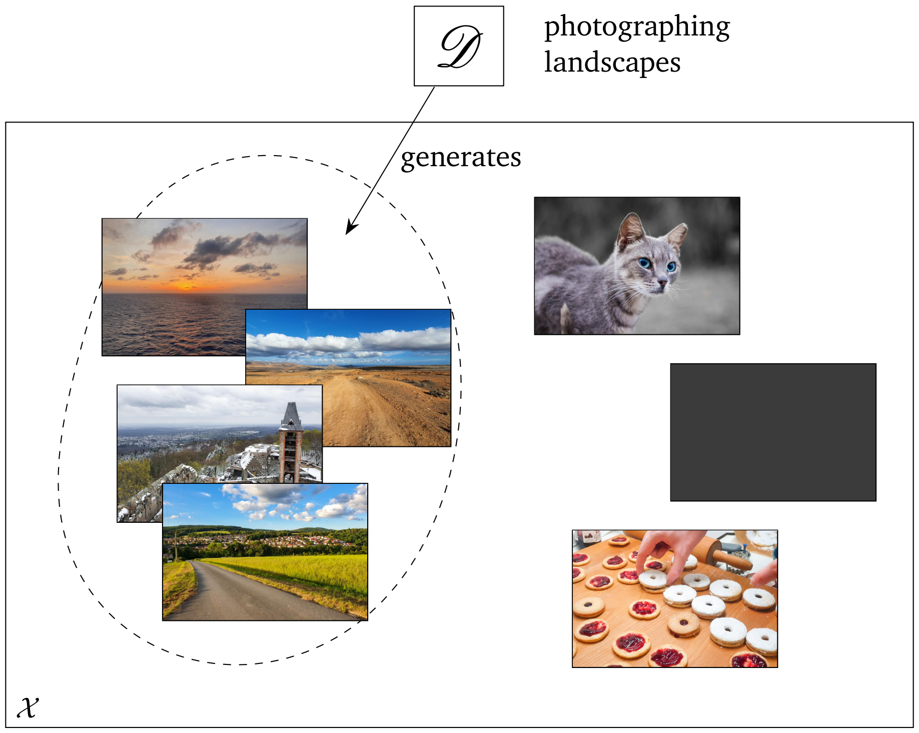


Figure 1.1: The DGP $\mathscr{D}$ of photographing landscapes conceptually visualized. The dashed lines encircle the region of valid samples, i.e., supp$(P_{\mathscr{D}})$. The three images outside that boundary on the right are examples of invalid data points.

For simplicity, assume that $\mathcal{X}$ is discrete. Each DGP that produces points in $\mathcal{X}$ corresponds to a probability distribution $P_{\mathscr{D}}$ where, for $\mathbf{x} \in \mathcal{X}$, $P_{\mathscr{D}}(\mathbf{x})$ is the probability for the event that generating an output using $\mathscr{D}$ indeed results in $\mathbf{x}$. In particular, generating a data point $\mathbf{x}$ using the DGP can be considered equivalent to sampling $\mathbf{x}$ from $P_{\mathscr{D}}$ in the stochastic sense. In the following, we are going to state a clear probabilistic definition of that $P_{\mathscr{D}}$ with all necessary conditions.

In order to define a probability distribution, we first need to determine the sample space Ω with a σ-algebra $\mathcal{F}$. Naturally, we choose $\Omega := \mathcal{X}$, so the sample space equals the data space. The latter must meet the following three conditions (Tsitsiklis, 2018) in order to qualify as sample space:

(1) Each outcome $\mathbf{x} \in \mathcal{X}$ is *mutually exclusive*, that is, if $\mathbf{x}$ occurs in a random trial, no other $\mathbf{x}' \in \mathcal{X}$ occurs simultaneously.

(2) The outcomes are *collectively exhaustive*, that is, the result of a random trial lies in $\mathcal{X}$.

(3) Given what we are interested in, $\mathcal{X}$ has the right *granularity*, that is, two outcomes $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ are different iff they differ in at least one relevant aspect. That is, $\mathbf{x} = \mathbf{x}'$ iff they only differ in irrelevant aspects.

Note that $\mathcal{X}$ may be an uncountable set of data points. Therefore, the power set of $\mathcal{X}$ might not be a σ-algebra. Instead, under the already met condition that $\mathcal{X}$ is a topological space, we may use the Borel σ-algebra $\mathcal{B}(\mathcal{X})$.

Definition 1.2.1 (True distribution)**.** Fix some DGP $\mathscr{D}$ with data space $\mathcal{X}$. If $(\mathcal{X}, \mathcal{B}(\mathcal{X}), P_{\mathscr{D}})$ is the probability space with probability distribution $P_{\mathscr{D}}$ over discrete data space $\mathcal{X}$ where, for all $\mathbf{x} \in \mathcal{X}$, $P_{\mathscr{D}}(\mathbf{x})$ is the probability that $\mathbf{x}$ is the result of a random generation of the DGP, then we call $P_{\mathscr{D}}$ the *true (data) distribution* of the DGP.

Taking pictures of landscapes is only one simple DGP example. Another instance is the random choice of any protein that exists on earth, or the random choice of any (theoretical) vessel that can swim on water. $P_{\mathscr{D}}$ is uniquely determined by $\mathscr{D}$ and is unknown in practice. In this work, we assume that the DGP's true distribution $P_{\mathscr{D}}$ is absolutely continuous w.r.t. the Lebesgue measure. This allows us to define a probability density function (PDF) $p_{\mathscr{D}}$ for continuous $\mathcal{X}$.

Typically, not every data point $\mathbf{x} \in \mathcal{X}$ is a possible result of the DGP.

Definition 1.2.2 (Validity)**.** Let $P_{\mathscr{D}}$ be the true distribution of a DGP $\mathscr{D}$ over some data space $\mathcal{X}$. If $P_{\mathscr{D}}(\mathbf{x}) > 0$, then $\mathbf{x}$ is called *valid*, otherwise *invalid*.

Reconsider the landscape photographing DGP: An image that depicts a cat or a plain gray void is invalid. In fact, most data points in $\mathcal{X}$ are invalid in this example.

1.3 Problem Setting: Discrete, High-dimensional, Expensive-to-evaluate, and Black-box

Continuing the landscape photographing example, assume that we want to find winter landscapes in $\mathcal{X}$ that have a lot of snow and ice. The objective function f measures the amount of snow and ice in that image.

More generally, given a DGP $\mathscr{D}$ over data space $\mathcal{X}$ and a corresponding objective function f, the overall goal is to determine a valid data point that maximizes f. That is, we have to solve the optimization problem

$$\begin{aligned} \max_{\mathbf{x}\in\mathcal{X}} \quad & f(\mathbf{x}). \\ \text{w.r.t.} \quad & \mathbf{x} \in \operatorname{supp}(P_{\mathscr{D}}). \end{aligned} \tag{OPT}$$

In this context, the elements of $\mathcal{X}$ are also called *solutions* and, thus, $\mathcal{X}$ the *solution space*. We consider the especially difficult case where

(1) $\mathcal{X}$ is discrete,

(2) $\mathcal{X}$ is very large, i.e., $\mathcal{X}$ has many (maybe infinite) dimensions (if interpretable as a vector space),

(3) f is expensive-to-evaluate, and

(4) f is black-box, i.e., there is no known analytical form of f.

These properties make optimization hard for the following reasons. First, discreteness of $\mathcal{X}$ precludes the use of gradient-based optimization methods like Stochastic Gradient Descent (SGD) as well as numeric methods such as finite differences or subgradients. Second, a high dimensionality of $\mathcal{X}$ correlates with a large number of local optima in the objective function f, causing problems for local search methods including taboo search, simulated annealing, etc. since these methods get stuck in local optima very easily. Moreover, feature-based optimization methods, e.g., greedy search and other decision strategies, become computationally infeasible due to the data space's high dimension. Third, the unavailability of the analytical form of f forbids exploiting any possibly useful structure of f such as convexity, linearity, or composition to compute a global optimum directly. Fourth, high evaluation costs of f force us to keep the number of evaluations very low, dispelling numerical optimization methods, population-based methods such as evolution strategies or genetic optimization, and local search methods.

Our landscape photographing example is not a worst-case scenario as it isn't discrete. Nevertheless, there is a number of motivating real-world examples matching even all of the above four properties, including some of the following:

Example 1.3.1 (Chemical design (Gómez-Bombarelli et al., 2016; Jin et al., 2018; Griffiths and Hernández-Lobato, 2020))**.** The data space $\mathcal{X}$ is the set of chemical molecules (like Ibuprofen, Acetone, etc.) where a molecule $\mathbf{x} \in \mathcal{X}$ is defined as a 3D structure comprising of (1) a selection of atoms of specific chemical elements and (2) covalent bonds between these atoms.[1] The value $f(\mathbf{x})$ assigned to each molecule $\mathbf{x} \in \mathcal{X}$ relates to some desirable chemical property such as stability, flammability, toxicity, heat of combustion, etc. The goal of chemical design is to find a molecule that maximizes the value of that property.

Example 1.3.2 (Protein design (Brookes et al., 2019; Kumar and Levine, 2020; Castro et al., 2022))**.** Consider the (finite) set A of all known amino acids as an alphabet of symbols. If $+$ denotes the *Kleene plus*, then $\mathcal{X} := A^+$ is the set of all possible, finite amino acid sequences with length of at least 1. Each such sequence $\mathbf{x} \in \mathcal{X}$ corresponds to a (hypothetical) protein that would result through folding[2]. The objective function f evaluates some property of that protein, e.g., fluorescence or—if the protein is an enzyme—catalytic activity. The target is to find an amino acid sequence, the corresponding protein of which maximizes the evaluated property.

Example 1.3.3 (Neural Architecture Search (NAS) (Zoph and Le, 2016; Luo et al., 2018))**.** Consider a set of images D, represented as arrays of pixels, and a set of labels (or classes) L. The task is to determine and train an end-to-end deep Neural Network (NN) $g : D \to L$ that assigns the correct label $l \in L$ to each image $\mathbf{d} \in D$. Here, $\mathcal{X}$ can be modeled as the set of all NN models $g : D \to L$. More precisely, an NN $\mathbf{x} \in \mathcal{X}$ is a structure consisting of, inter alia, neurons, weighted connections, and activation functions. The objective function f measures the performance of $\mathbf{x}$ on the labeling task (after training the NN using a pre-defined training procedure).

Example 1.3.4 (Perfume design)**.** A perfume is a liquid mixture comprising a selection of essential oils and fragrances of specific amounts. Define $\mathcal{X}$ as the set of perfumes. A jury (one single or a group of test persons) is tasked with subjectively rating the perfume's scent by assigning a score. Here, the objective function f assigns to each perfume $\mathbf{x} \in \mathcal{X}$

[1]As a machine-readable representation, one could use identifier schemes such as SMILES or InChI.

[2]For simplicity, we assume that folding is deterministic which, in general, is not the case. There exist popular models such as AlphaFold (Jumper et al., 2021) that can be used to predict the folded structure.

the overall average score that the perfume would get from the jury. The goal is to design a perfume that maximizes the jury's score.

Some more interesting design problems with existing works are material design (Mansouri Tehrani et al., 2018), movement trajectory optimization in robotics (Antonova et al., 2020), and aircraft design (Hoburg and Abbeel, 2014).

Except for the perfume design problem which can be modeled continuously, all the aforementioned examples possess the four worst case properties. In particular, $\mathcal{X}$ is very large and, hence, almost all solutions of $\mathcal{X}$ are not known a priori, i.e., they do not exist at hand. The target is to find the optimal solution $x^* \in \mathcal{X}$ which maximizes f. Like the vast majority of $\mathcal{X}$, the optimum x^* likely isn't known beforehand—it needs to be found or *generated*. This is opposed to the problems of, e.g., video recommendation or online ad placement because the set of solutions optimized over, although being high-dimensional, is finite and ”known”–no new solution is generated but rather an existing one gets picked.

Note that, indeed, the evaluation of f is expensive in the above examples. First, the solution needs to be created in reality through experiments (chemical synthesis, NN training, fragrance acquisition and composition), then their performance needs to be tested by another round of experiments (chemical tests, NN evaluation, jury assessment).

1.4 Assessing Conceivable Strategies: Model-based Optimization and Bayesian Optimization

The fact that the objective function f is black-box and expensive to evaluate can be circumvented using *Model-based Optimization (MBO)*. MBO methods replace the objective function f with a smooth and differentiable surrogate model $f_\theta : \mathcal{X} \to \mathbb{R}$ that approximates f. (Larson et al., 2019; Tripp et al., 2020) That is, the parameters θ of the surrogate model are learned so that $f_\theta(\mathbf{x}) \approx f(\mathbf{x}), \forall \mathbf{x} \in \mathcal{X}$. This can be done using empirical risk minimization. Naively optimizing f_θ is a widely used technique (Tripp et al., 2020) despite several decisive drawbacks: First, f_θ is inaccurate in regions of $\mathcal{X}$ where few or no training data was available. (Kumar and Levine, 2020) Second, the issues of high-dimensionality and discreteness with their implications still prevail.

Since we do not have access to the analytical form of f, we cannot compute any derivative. In such situations, derivative-free optimization methods are used. (Larson et al., 2019) The most popular derivative-free method, according to Fu and Levine (2021), is *Bayesian*

Optimization (BO) (Snoek et al., 2012; Snoek et al., 2015; Shahriari et al., 2016; Eismann et al., 2018). BO is the gold standard for query-efficient continuous optimization. (Stanton et al., 2022; Eismann et al., 2018) It is an iterative MBO approach. Starting from an initial dataset $\mathcal{D} \subseteq \mathcal{X}$, a surrogate gets trained on that dataset and the corresponding values. Then, iteratively, the surrogate is used to suggest new candidate solutions which then get evaluated and added to $\mathcal{D}$ in order to learn from. That is, the surrogate acts not only as a replacement for f but also as an acquisition function. Most commonly, *Gaussian Processes (GPs)* are used as the surrogate. (Maus et al., 2022; Grosnit et al., 2021) A GP can be seen as a probability distribution of functions. GPs effectively trade off exploration and exploitation while being able to quantify uncertainty, critical for the effectiveness of BO algorithms. (Deshwal and Doppa, 2021) However, BO doesn't scale well to high-dimensional data. (Kumar and Levine, 2020; Grosnit et al., 2021) In fact, BO is limited to optimizing 10 to 20 parameters at once. (Moriconi et al., 2019)

LSO overcomes the problem of high-dimensionality by constructing a lower-dimensional space and then doing BO in that space. Details of that procedure will be presented in Chapter 4.

Besides, there are other derivative-free approaches that could possibly be used to solve OPT, including reinforcement learning (RL) (Williams, 1992), the cross-entropy method (Rubinstein, 1999), and latent variable models (Garnelo et al., 2018; Kim et al., 2019). We won't cover any of these methods as they are beyond the scope of this work.

2 Generative Adversarial Networks (GANs)

A Deep Generative Model (DGM) is required to do Latent Space Optimization (LSO). Loosely speaking, a DGM is a specific type of ML model that aims to replicate a Data Generation Process (DGP). This chapter introduces DGMs briefly, covering the state of the art before dealing with Generative Adversarial Networks (GANs) that form a popular sub-class of DGMs. GANs have a vivid intuition and provide a suitable context for the reader to learn key LSO components like the latent space and the generator.

2.1 Deep Generative Models (DGMs)

2.1.1 Foundations

There exist countless DGPs in the real world, most of which have an unknown true distribution. However, not only for LSO, there is reason to model DGPs and their true distribution. Fix a DGP $\mathscr{D}$ (over data space $\mathcal{X}$) that we are interested in to model. In the ML domain, a *generative model* is considered to be a statistical model that (explicitly or implicitly) encodes a probability distribution over $\mathcal{X}$. Typically, the aim of a generative model is to resemble the true distribution $P_{\mathscr{D}}$. In this context, $\mathscr{D}$ is called *target DGP*. A generative model incorporates a generator, defined as follows.

Definition 2.1.1 (Generator)**.** Let $\mathcal{Z} := \mathbb{R}^m$ for some $m \in \mathbb{N}$. A function of the form

$$\begin{aligned} G : \mathcal{Z} &\to \mathcal{X} \\ \mathbf{z} &\mapsto \mathbf{x} \end{aligned}$$

that, for each real-valued vector $\mathbf{z} \in \mathcal{Z}$, assigns a data point $\mathbf{x}$ from a data space $\mathcal{X}$ is called *generator*.

That is, a generator is merely a function that maps real vectors into a (possibly discrete) data space $\mathscr{X}$. $\mathscr{Z}$ is called latent space and will be covered in more detail in Chapter 3. The specific integration of the generator into the generative model highly depends on the latter and will be covered in more detail only for GANs.

The counterpart of generative models are discriminative models. The latter learn decision boundaries to tell data points apart. Therefore, discriminative models are typically used for labeling/categorization tasks. More precisely, let X denote the random variable that represents the data point resulting from a DGP generation trial, let $\mathcal{Y}$ denote the set of labels and Y the random variable that corresponds to a data point's label. Discriminative models aim to embody the conditional probability $P_{\mathscr{D}}(Y \mid X = \mathbf{x})$, utilized to predict the label Y for a given data point $\mathbf{x}$. In contrast to that, generative models resemble $P\mathscr{D}(X)$. Generative models that contain a deep Neural Network (NN) are called *Deep Generative Model (DGM)*.

2.1.2 State of the Art

Prominent DGMs (and their inventors) include

- Generative Adversarial Networks (GANs) (Goodfellow et al., 2014),
- Variational Autoencoders (VAEs) (Kingma and Welling, 2013),
- Normalizing Flows (Rezende and Mohamed, 2015),
- Transformers (Vaswani et al., 2017), and
- Diffusion Models (Sohl-Dickstein et al., 2015).

Other examples of (non-deep) generative models are hidden Markov models, Bayesian networks, Boltzmann machines, and Gaussian mixture models. The following popular DGM implementations vividly show that DGMs reached a well-advanced state of applied research.

StyleGAN3 (with predecessors StyleGAN and StyleGAN2) (Karras et al., 2019; Karras et al., 2020; Karras et al., 2021) is a GAN-based DGM, capable of generating high-quality images of human faces and objects. A random selection of human face images generated by StyleGAN3 is depicted in Figure 2.1. A live demonstration of StyleGAN2 can be found here[1].

[1]`https://thispersondoesnotexist.com/`

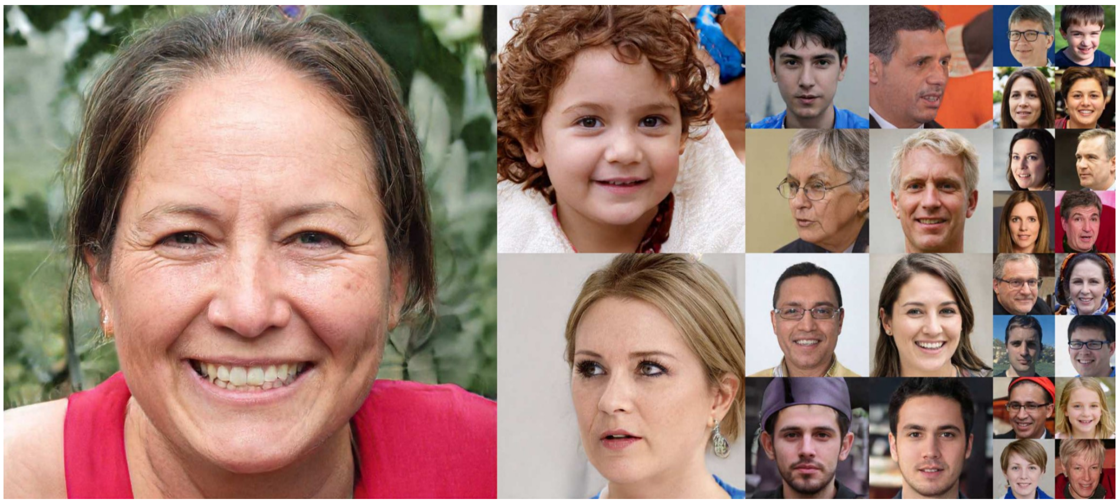


Figure 2.1: Uncurated set of images produces by StyleGAN3, taken from Karras et al. (2021).

GPT-3 (with predecessors GPT and GPT-2) (Radford et al., 2018; Radford et al., 2019; Brown et al., 2020), acronym for ”Generative Pre-trained Transformer 3”, is a powerful Transformer-based language model by OpenAI. For example, it can create text such as news articles (cf. Figure 2.2), summarize documents or translate between languages. Recently, Meta AI released a model similar to GPT-3 with comparable capabilities. (Zhang et al., 2022)

Finally, **DALL-E 2** (with predecessor DALL-E) (Ramesh et al., 2021; Ramesh et al., 2022) is a text-to-image model, taking a textual description and converting it into an image using a Diffusion Model, see also Figure 2.3.

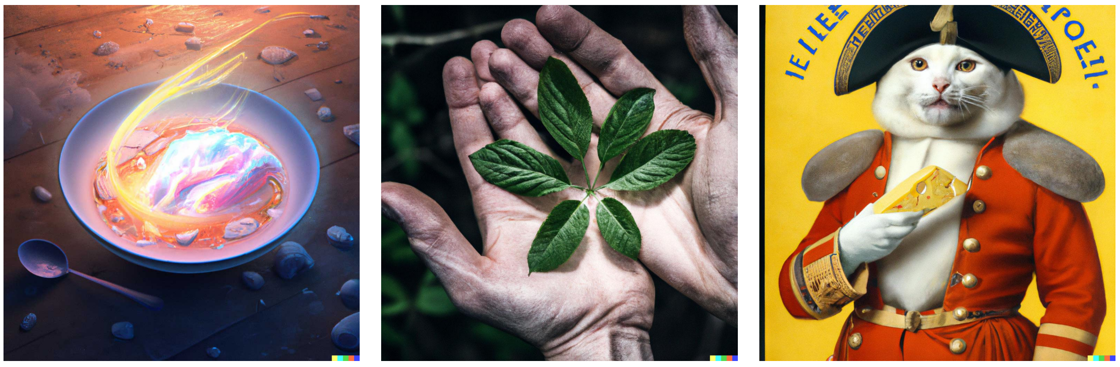
2.2 GANs

Next, we’re going through the core intuition and the mathematical details of GANs. This section mainly rests on the work of Goodfellow et al. (2014) who originally introduced GANs.

Note that the concept of LSO is DGM agnostic, that is, LSO conceptually doesn’t depend on the choice of the DGM. In practice, one should choose a DGM that is best suitable for the

Title: United Methodists Agree to Historic Split
Subtitle: Those who oppose gay marriage will form their own denomination
Article: **After two days of intense debate, the United Methodist Church has agreed to a historic split - one that is expected to end in the creation of a new denomination, one that will be "theologically and socially conservative," according to The Washington Post. The majority of delegates attending the church's annual General Conference in May voted to strengthen a ban on the ordination of LGBTQ clergy and to write new rules that will "discipline" clergy who officiate at same-sex weddings. But those who opposed these measures have a new plan: They say they will form a separate denomination by 2020, calling their church the Christian Methodist denomination.**

Figure 2.2: Artificially created news article generated by GPT-3. Generator output marked in boldface. Excerpt taken from Brown et al. (2020).



"A bowl of soup that is a portal to another dimension as digital art"

"A close up of a handpalm with leaves growing from it"

"A propaganda poster depicting a cat dressed as French emperor Napoleon holding a piece of cheese"

Figure 2.3: Exemplary output images produced by DALL-E 2 for given image captions. Examples taken from Ramesh et al. (2022) and OpenAI (2022).

given use case since each DGM type has its own strengths and weaknesses. Here, the GAN was chosen. The reasons for the choice are mainly that (1) the core idea of GANs is vivid and, hence, beneficial for the reader's intuition, (2) GANs are very popular (Pan et al., 2019), and (3) the application of GANs was observed to be quite successful (Harshvardhan et al., 2020). However, in their native form, GANs cannot model discrete data (Goodfellow et al., 2014) and, as we will see later in this section, GAN theory only works under very strict conditions. For LSO, this is not an issue due to its DGM agnosticism.

2.2.1 Architecture and Idea

Definition 2.2.1 (GAN)**.** Let $\mathcal{X} := \mathbb{R}^n, n \in \mathbb{N}$ be some continuous data space and let $\mathcal{Z} := \mathbb{R}^m, m \in \mathbb{N}$ with $m \leq n$. A *Generative Adversarial Network (GAN)* is a triple $(P_{\mathcal{Z}}, G, D)$ consisting of

- an (w.r.t. Lebesgue measure) absolutely continuous probability distribution $P_{\mathcal{Z}}$ over $\mathcal{Z}$ called *prior (noise) distribution*,
- a differentiable generator $G : \mathcal{Z} \to \mathcal{X}$, and
- a function $D : \mathcal{X} \to [0, 1]$ called *discriminator*.

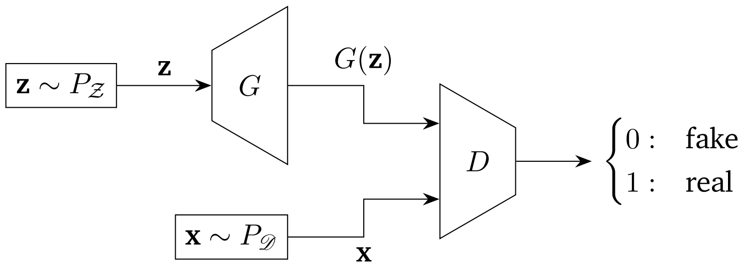

$\mathbf{z} \sim P_{\mathcal{Z}}$
$\mathbf{z}$
G
$G(\mathbf{z})$
D
$\mathbf{x} \sim P_{\mathcal{D}}$
$\mathbf{x}$
$\begin{cases} 0: & \text{fake} \\ 1: & \text{real} \end{cases}$

Figure 2.4: Structure of a GAN. G is the generator, D the discriminator, $P_{\mathcal{Z}}$ the prior distribution, and $P_{\mathcal{D}}$ is the true distribution of the target DGP.

Here, we denote with $\mathbf{x} \sim P$ that $\mathbf{x}$ is a random vector (RV) that follows the probability distribution P. Figure 2.4 shows the conceptual structure of a GAN. The generator receives a RV $\mathbf{z} \sim P_{\mathcal{Z}}$ sampled from the prior distribution and maps that vector to a data point $G(\mathbf{z})$. Since $G(\mathbf{z})$ is "produced" by the generator, we call the output *fake* data point. The discriminator D then, by random choice, receives either the fake data point or some $\mathbf{x} \sim P_{\mathcal{D}}$, a *real* data point as input. The task of D is to determine whether its input is fake or real by returning 0 for ”fake” or 1 for ”real” or any number in between, corresponding to the discriminator’s confidence. Besides, the set $\mathcal{Z}$ is called latent space and will be covered in more detail in Chapter 3.

Note that G itself is not probabilistic. Nevertheless, the combination of the prior $P_{\mathcal{Z}}$ and G can be considered as a DGP, the *DGP induced by G and $P_{\mathcal{Z}}$*. This DGP is simply the process of sampling a vector $\mathbf{z} \sim P_{\mathcal{Z}}$ from the prior and mapping it with G to some data point $\mathbf{x}$. Moreover, since G is differentiable, it is continuous and thus measurable, implying that G is a random variable mapping from the probability space $(\mathcal{Z}, \mathcal{B}(\mathcal{Z}), P_{\mathcal{Z}})$ into the measurable space $(\mathcal{X}, \mathcal{B}(\mathcal{X}))$, calling for a new distribution:

Definition 2.2.2 (Generator distribution)**.** Let $(P_{\mathcal{Z}}, G, D)$ be some GAN. The probability distribution P_G induced by the random variable G is called the *generator distribution* (w.r.t. generator G and prior $P_{\mathcal{Z}}$).

That is, P_G is defined over the measurable space $(\mathcal{X}, \mathcal{B}(\mathcal{X}))$ via $P_G(X) := P_{\mathcal{Z}}(G^{-1}(X))$ for all $X \in \mathcal{B}(\mathcal{X})$ (here, $G^{-1}(X) := \{\mathbf{z} \in \mathcal{Z} \mid G(\mathbf{z}) \in X\}$). Accordingly, p_G denotes the probability density function of P_G which, by the way, exists since $P_{\mathcal{Z}}$ is absolutely continuous. Obviously, P_G is exactly the true distribution of the DGP induced by G and $P_{\mathcal{Z}}$.

On a side note, the prior $P_{\mathcal{Z}}$ is commonly defined as a multivariate standard normal distribution, i.e., $P_{\mathcal{Z}} := \mathcal{N}(\mathbf{0}, \mathbf{I})$ (with zero-vector $\mathbf{0}$ and identity matrix $\mathbf{I}$).

Core intuition behind GANs The goal of G is to produce fake instances that look as real as possible, i.e., instances that could have come from the true distribution $P_{\mathscr{D}}$ with high probability. Intuitively, G tries to ”fool” D. Throughout the training, the latter needs to learn how to distinguish the increasingly realistic-looking fake instances from real ones. For example, consider the police-counterfeiter example provided by Goodfellow et al. (2014): A team of criminals, the counterfeiters, try to produce fake currency, whereas the police attempt to identify the fake currency. The counterfeiters and the police represent the generator G and the discriminator D, respectively. Besides, the central bank’s money printing machine producing the real, legitimate currency can be interpreted as the target DGP $\mathscr{D}$ here. Over time, both actors learn to improve: The counterfeiters produce more realistic-looking currency by optimizing their strategy, i.e., by converging the generator distribution P_G towards the true distribution $P_{\mathscr{D}}$.

As in the police-counterfeiter example above, G and D are *adversaries* in the game of producing and identifying fake data points, hence the DGM’s name.

2.2.2 Mathematical Theory Behind GANs

The ultimate goal of a GAN is to learn the target DGP. That is, we need to find a generator G so that $P_G = P_{\mathscr{D}}$. To this end, we state

Definition 2.2.3 (GAN value function)**.** The *value function* of a GAN $(P_{\mathcal{Z}}, G, D)$ is defined as

$$V(G, D) := \mathbb{E}_{\mathbf{x}\sim \mathrm{P}_{\mathscr{D}}}\left[\log\left(D(\mathbf{x})\right)\right] + \mathbb{E}_{\mathbf{z}\sim \mathrm{P}_{\mathcal{Z}}}\left[\log\left(1 - D(G(\mathbf{z}))\right)\right].$$

We model the opposition of G and D with the following minimax game:

$$\min_G \max_D V(G, D). \tag{OptGAN}$$

The objective of G is to minimize the value V while D has the opposite target. Since log is monotonously increasing, we note the following key intuition:

- D can increase V through increasing $D(\mathbf{x})$ and decreasing $D(G(\mathbf{z}))$. Doing so corresponds to D being better able to discriminate between real and fake data.
- G can decrease V only via the second expectation term. In detail, G needs to drive $D(G(\mathbf{z}))$ towards 1 as close as possible, corresponding to G ”fooling” D to ”believe” that the fake sample $G(\mathbf{z})$ is real.

The value function and the minimax game were proposed by Goodfellow et al. (2014). Their hypothesis is that $P_G = P_{\mathscr{D}}$ in the (unique) Nash equilibrium[2] of OptGAN. Their proof, however, is incomplete. This section shows their hypothesis for a special case, namely where G is a diffeomorphism. Henceforth, let $(P_{\mathcal{Z}}, G, D)$ be a fixed GAN.

Proposition 2.2.4. *If the generator G is a diffeomorphism whose Jacobian $JG(\mathbf{z})$ is non-zero for all $\mathbf{z} \in \mathcal{Z}$ and both the PDF $p_{\mathcal{Z}}$ and the discriminator D are continuous, then*

$$\mathbb{E}_{\mathbf{z}\sim \mathrm{P}_{\mathcal{Z}}}\left[\log(1 - D(G(\mathbf{z})))\right] = \mathbb{E}_{\mathbf{x}\sim \mathrm{P}_{\mathrm{G}}}\left[\log(1 - D(\mathbf{x}))\right].$$

[2] A Nash equilibrium is a pair (G, D) where unilateral improvement of OptGAN by only G or D is prohibited. (Mescheder et al., 2018)

Proof. First, by the multivariate PDF transform rule (Ostwald, 2021), the PDF p_G of the distribution P_G is given by

$$p_G(\mathbf{x}) = p_{\mathcal{Z}}(G^{-1}(\mathbf{x})) \cdot \left|\det(JG^{-1}(\mathbf{x}))\right| , \tag{2.1}$$

where $\det(JG^{-1}(\mathbf{x}))$ denotes the Jacobian determinant of G^{-1} evaluated at $\mathbf{x}$ and G^{-1} the inverse of G (which exists since G is a diffeomorphism). According to the properties of the expectation value,

$$\mathbb{E}_{\mathbf{z}\sim P_{\mathcal{Z}}}\left[\log(1 - D(G(\mathbf{z})))\right] = \int_{\mathcal{Z}} p_{\mathcal{Z}}(\mathbf{z}) \cdot \log(1 - D(G(\mathbf{z})))\,\mathrm{d}\mathbf{z}\,.$$

The function $f(\mathbf{z}) := p_{\mathcal{Z}}(\mathbf{z}) \cdot \log(1 - D(G(\mathbf{z})))$ is a concatenation of continuous functions and, thus, continuous. Therefore, and since G is continuously differentiable, we can apply the substitution rule for integrals. The above term is equivalent to

$$\begin{aligned}\int_{G^{-1}(\mathcal{X})} f(\mathbf{z})\,\mathrm{d}\mathbf{z} &\overset{\text{Subst.}}{=} \int_{\mathcal{X}} f(G^{-1}(\mathbf{x})) \cdot \left|\det(JG^{-1}(\mathbf{x}))\right| \mathrm{d}\mathbf{x} \\ &= \int_{\mathcal{X}} p_{\mathcal{Z}}(G^{-1}(\mathbf{x})) \cdot \log(1 - D(\mathbf{x})) \cdot \left|\det(JG^{-1}(\mathbf{x}))\right| \mathrm{d}\mathbf{x}\,.\end{aligned}$$

Using the identity of the PDF p_G (Eqn. 2.1), the integral becomes

$$\int_{\mathcal{X}} p_G(\mathbf{x}) \cdot \log(1 - D(\mathbf{x}))\,\mathrm{d}\mathbf{x}$$

which is the same as

$$\mathbb{E}_{\mathbf{x}\sim P_G}\left[\log(1 - D(\mathbf{x}))\right].$$

□

Goodfellow et al. (2014) used the non-trivial identity of Proposition 2.2.4 without any proof or comment. In particular, the authors didn't state the requirements that are needed for this identity to hold. Here, inter alia, we presuppose G to be a diffeomorphism, which is a very strong requirement. The implications of this are discussed in Chapter 5.

From now on, we assume that the requirements of Proposition 2.2.4 are met. The rest of this subsection shows that $P_G = P_{\mathscr{D}}$ in a Nash equilibrium of the minimax game OptGAN (if it exists).

Consider the sub-problem receiving when dropping the $\min_G$ part from OptGAN, i.e.,

$$\max_D V(G, D) \tag{OptD}$$

for some fixed generator G.

Lemma 2.2.5. *The optimal discriminator D^*_G for OptD, if it exists, is the discriminator with*

$$D^*_G(\mathbf{x}) = \frac{p_{\mathscr{D}}(\mathbf{x})}{p_{\mathscr{D}}(\mathbf{x}) + p_G(\mathbf{x})}, \qquad \forall \mathbf{x} \in \mathcal{X}.$$

Proof. By Proposition 2.2.4, by the properties of the expectation value, and by the linearity of the integral,

$$\begin{aligned} V(G, D) &= \mathbb{E}_{\mathbf{x} \sim \mathrm{P}_{\mathscr{D}}} \left[\log\left(D(\mathbf{x})\right) \right] + \mathbb{E}_{\mathbf{x} \sim \mathrm{P_G}} \left[\log(1 - D(\mathbf{x})) \right] \\ &= \int_{\mathcal{X}} p_{\mathscr{D}}(\mathbf{x}) \cdot \log(D(\mathbf{x}))\, \mathrm{d}\mathbf{x} + \int_{\mathcal{X}} p_G(\mathbf{x}) \cdot \log(1 - D(\mathbf{x}))\, \mathrm{d}\mathbf{x} \\ &= \int_{\mathcal{X}} \left(p_{\mathscr{D}}(\mathbf{x}) \cdot \log(D(\mathbf{x})) + p_G(\mathbf{x}) \cdot \log(1 - D(\mathbf{x})) \right) \mathrm{d}\mathbf{x}\,. \end{aligned} \tag{2.2}$$

D is optimal if it maximizes the integrand of 2.2. This is relevant only for all $\mathbf{x} \in \operatorname{supp}(P_{\mathscr{D}}) \cup \operatorname{supp}(P_G)$ as, for all other $\mathbf{x}$, the integrand is zero and, hence, the discriminator has no influence on the objective. Fix $\mathbf{x} \in \operatorname{supp}(P_{\mathscr{D}}) \cup \operatorname{supp}(P_G)$ and let

$$\begin{aligned} v : [0, 1] &\to \mathbb{R} \\ x &\mapsto a \log(x) + b \log(1 - x) \end{aligned}$$

be a simplified representation of the integrand where $a := p_{\mathscr{D}}(\mathbf{x})$ and $b := p_G(\mathbf{x})$. The maximizer of v also maximizes the integrand. That is, the output of an optimal discriminator must be that maximizer. The function v is concave since it is the sum of concave functions (because log is concave). Therefore, and because v is defined over a compact interval, v has a unique maximum. It can be found at the zero of the function's derivative. It holds $v'(x) = \frac{a}{x} - \frac{b}{1-x}$ which is zero at $x = \frac{a}{a+b}$. This value is indeed in $[0, 1]$ and inserting the PDFs for a and b concludes the proof. ☐

Keep in mind that Lemma 2.2.5 only holds if the optimal discriminator—which needs to be differentiable—exists. This may not be the case for non-continuous $p_{\mathscr{D}}$.

Definition 2.2.6. The optimal value of OptD, i.e.,

$$C(G) := \max_D V(G, D)$$

is called the *virtual training criterion* of the generator G.

Theorem 2.2.7. *The generator G^* is the global minimizer for*

$$\min_G C(G)$$

if and only if $P_{G^*} = P_{\mathscr{D}}$. In that case, $C(G^*) = -\log 4$.

Proof. By the definition of the Kullback-Leibler (KL) divergence, by Lemma 2.2.5, and by the properties of the expectation value, observe that

$$\begin{aligned}
\mathrm{KL}\left(P_{\mathscr{D}} \middle\| \frac{P_{\mathscr{D}} + P_G}{2}\right) &= \int_{\mathcal{X}} p_{\mathscr{D}}(\mathbf{x}) \log\left(2 \frac{p_{\mathscr{D}}(\mathbf{x})}{p_{\mathscr{D}}(\mathbf{x}) + p_G(\mathbf{x})}\right) \mathrm{d}\mathbf{x} \\
&= \int_{\mathcal{X}} p_{\mathscr{D}}(\mathbf{x})(\log 2 + \log D_G^*(\mathbf{x}))\, \mathrm{d}\mathbf{x} \\
&= \log 2 + \mathbb{E}_{\mathbf{x}\sim \mathrm{P}_{\mathscr{D}}}[\log D_G^*(\mathbf{x})].
\end{aligned}$$

Analogously,

$$\mathrm{KL}\left(P_G \middle\| \frac{P_{\mathscr{D}} + P_G}{2}\right) = \log 2 + \mathbb{E}_{\mathbf{x}\sim \mathrm{P}_G}[\log D_G^*(\mathbf{x})].$$

Therefore, inserting into the definition of $C(G)$, we get

$$\begin{aligned}
C(G) &= \mathbb{E}_{\mathbf{x}\sim \mathrm{P}_{\mathscr{D}}}\left[\log\left(D_G^*(\mathbf{x})\right)\right] + \mathbb{E}_{\mathbf{x}\sim \mathrm{P}_G}\left[\log(1 - D_G^*(\mathbf{x}))\right] \\
&= \mathrm{KL}\left(P_{\mathscr{D}} \middle\| \frac{P_{\mathscr{D}} + P_G}{2}\right) + \mathrm{KL}\left(P_G \middle\| \frac{P_{\mathscr{D}} + P_G}{2}\right) - \log 4 \\
&= 2\,\mathrm{JSD}(P_{\mathscr{D}} \| P_G) - \log 4,
\end{aligned}$$

where JSD is the Jensen-Shannon divergence. The claim follows by the fact that the JSD term is non-negative and that it is zero iff $P_{\mathscr{D}} = P_G$. □

Theorem 2.2.7 shows $P_{\mathscr{D}} = P_G$ only under very specific conditions. Moreover, note that it is not immediately clear if there is no other Nash equilibrium. We leave this question open as it would exceed this Thesis's scope.

2.2.3 Training Algorithm

In order to apply ML to a GAN $(P_\mathcal{Z}, G, D)$, we need to parameterize G and D. However, parameterization strongly restricts the space of functions taken by both G and D. Therefore, the theoretical results above do not apply. Let G_θ and D_φ denote the parameterized versions of G and D with parameters θ and φ, respectively. Typically, deep convolutional NNs are used as the parameterization of G_θ and D_φ. (Goodfellow, 2016) NNs have proven to have strong performance on many different tasks. In practice, the expressiveness of an NN corresponds to its size. (*AI Index Report* 2022) Empirical results show that for many tasks a manageable size is sufficient. (Pan et al., 2019; Karras et al., 2021)

In order to train a GAN, we require the generator G_θ and the discriminator D_φ to be differentiable w.r.t. their parameters θ and φ, respectively. Otherwise, it would be impossible to compute the gradients of G_θ and D_φ required for training their parameters.

Recall the Nash equilibrium of the minimax game OptGAN. The assertion is that we reached the goal $P_{G_\theta} = P_\mathcal{D}$ if we found a Nash equilibrium (if there is exactly one). That is, we need to find a parameter assignment (θ^*, φ^*) that prohibits unilateral improvement by only G_θ or D_φ, respectively.

Definition 2.2.8 (GAN loss)**.** The *loss* of a GAN $(P_\mathcal{Z}, G_\theta, D_\varphi)$ with parameters θ and φ for a real-valued vector $\mathbf{z} \in \mathcal{Z}$ and a data point $\mathbf{x} \in \mathcal{X}$ is

$$l(\theta, \varphi, \mathbf{z}, \mathbf{x}) := \log(D_\varphi(\mathbf{x})) + \log(1 - D_\varphi(G_\theta(\mathbf{z}))).$$

The procedure for training a parameterized GAN is displayed in Algorithm 1. It uses Stochastic Gradient Descent (SGD) with small batches (minibatches). The gradients listed in the algorithm are the respective derivatives of the GAN loss.

Conjecture 2.2.9 (Convergence of Algorithm 1)**.** *Given that G_θ and D_φ have sufficient capacity so that*

(1) at each (outer) step of Algorithm 1, D_θ is allowed to reach its optimum and

(2) P_{G_θ} is updated improving (reducing) the criterion

$$\mathbb{E}_{\mathbf{x}\sim \mathrm{P}_\mathcal{D}}\left[\log\left(D^*_{G_\theta}(\mathbf{x})\right)\right] + \mathbb{E}_{\mathbf{x}\sim P_{G_\theta}}\left[\log(1 - D^*_{G_\theta}(\mathbf{x}))\right],$$

then P_{G_θ} converges to $P_\mathcal{D}$ (for possibly infinitely many training steps).

Algorithm 1: Minibatch SGD for GANs

```
Input :GAN $(P_{\mathcal{Z}}, G_\theta, D_\varphi)$ with parameters $\theta$ and $\varphi$, sample size $p$
for # training steps do
    for # discriminator updates do
        sample $\{\mathbf{z}^{(1)}, ..., \mathbf{z}^{(p)}\}$ from $P_{\mathcal{Z}}$
        sample $\{\mathbf{x}^{(1)}, ..., \mathbf{x}^{(p)}\}$ from $P_{\mathcal{D}}$
        update $D_\varphi$ by ascending its stochastic gradient
            $\nabla_\varphi \frac{1}{p} \sum_{i=1}^{p} \left[\log D_\varphi(\mathbf{x}^{(i)}) + \log(1 - D_\varphi(G_\theta(\mathbf{z}^{(i)})))\right]$
    sample $\{\mathbf{z}^{(1)}, ..., \mathbf{z}^{(p)}\}$ from $P_{\mathcal{Z}}$
    update $G_\theta$ by descending its stochastic gradient
        $\nabla_\theta \frac{1}{p} \sum_{i=1}^{p} \left[\log(1 - D_\varphi(G_\theta(\mathbf{z}^{(i)})))\right]$
```

A proof attempt of Conjecture 2.2.9 can be found in Goodfellow et al. (2014) and is omitted here. The capacity constraint of that conjecture requires G_θ and D_φ to be flexible enough so that optimization is possible as specified in the conjecture's conditions (1) and (2). The capacity constraint is very hard to satisfy in practice since model parameterization strongly limits the model's capacity. Therefore, this conjecture is only of theoretical interest. As of 2018, ”there were no convergence proofs for GAN models, even in very simple settings.” (Li et al., 2018) In a nutshell, GAN theory is hardly transferable into practice, too.

2.2.4 Issues with GANs: Mode Collapse, Vanishing Gradients & High Sample Complexity

The bad applicability of GAN theory to practice is not the only downside of GANs. Training GANs turns out to be unstable. The most common issue encountered in application is non-convergence due to *mode collapse* and *vanishing gradients*. (Goodfellow, 2016; Wiatrak et al., 2020) Moreover, GANs require a lot of training data, i.e., sample complexity is high.

Mode collapse is the situation in which the generator is highly non-injective, i.e., many different latent vectors $\mathbf{z} \in \mathcal{Z}$ are mapped (close) to a single data point $\mathbf{x} \in \mathcal{X}$. (Goodfellow, 2016) Mode collapse corresponds to *overfitting*, a common problem encountered in ML. Overfitting is an effect observed during training where a generative model's distribution strongly deviates from the true distribution while only supporting data points from the training dataset. Here, this means that $\mathbf{x} \in \operatorname{supp}(P_G)$ for all $\mathbf{x}$ in the training dataset, but $\mathbf{x} \notin \operatorname{supp}(P_G)$ for most $\mathbf{x} \in \operatorname{supp}(P_{\mathscr{D}})$.

Intuitively, in the case of mode collapse, the generator G overfits so that it tends to produce only one but (for the discriminator) most plausible data point. During the learning process, the discriminator learns to reject that, and only that, data point, making it easy for the generator to switch to a different plausible output in data space. This process repeats endlessly, with generator and discriminator wandering around in data space.[3]

Vanishing gradients is a common problem in ML where the gradient passed through the NN architecture becomes too small and, thus, training is slow or stops. The generator's gradient may vanish when the discriminator becomes too accurate. (Goodfellow, 2016)

Numerous stabilization and balancing methods exist in order to tackle mode collapse and vanishing gradients. (Wiatrak et al., 2020; Mescheder et al., 2018; Kodali et al., 2017; Goodfellow, 2016)

[3]Mode collapse may be caused due to an unwanted, implicit swap of $\min_G \max_D$ in the objective function of OptGAN during training. (Goodfellow, 2016) The minimax and maximin solutions are different. In fact, the maximin version encourages the generator to focus on a single point $\mathbf{x} \in \mathcal{X}$ most plausible to the discriminator.

3 Latent Space

One approach to deal with the discreteness of our original optimization problem OPT is to transform the problem's discrete solution space $\mathcal{X}$ into a continuous representation space. We receive such a continuous representation space simply by training a DGM like a GAN or a VAE on $\mathcal{X}$. That space is called *latent space* and will be presented in this chapter. More precisely, after a brief definition of latent space, its most important properties will be highlighted, though only on a qualitative and empirical level. The lack of rigor in this chapter is mainly due to the relatively under-explored theory of the latent space in current literature.

3.1 Definition

Definition 3.1.1 (Latent space)**.** Let $G : \mathcal{Z} \to \mathcal{X}$ be a generator and let $P_{\mathcal{Z}}$ be a prior distribution. The generator's continuous domain $\mathcal{Z} = \mathbb{R}^m, m \in \mathbb{N}$ is called *latent space*. Its elements are called *latent vectors* or *noise vectors*. Moreover, for $\mathbf{z} \in \mathcal{Z}$, we refer to the data point $\mathbf{x} := G(\mathbf{z})$ as the *phenotype* of $\mathbf{z}$ (w.r.t. generator G).

The term "phenotype"—inspired by genes and their effect on the appearance of a living being—is novel in this context and was introduced for the sake of readability. In essence, $\mathcal{Z}$ is simply $\mathbb{R}^m$ for some $m \in \mathbb{N}$, but its elements all become a meaning when $\mathcal{Z}$ is considered as the domain of a generator G, because G assigns a data point to each latent vector. This relation is the basis of latent space theory and implies useful topological properties. Besides, the dimension m of $\mathcal{Z}$ can be chosen arbitrarily. Typically, and as an assumption for the rest of this work, it is most useful to choose m much smaller than the dimension of $\mathcal{X}$. However, there are some limitations to consider that are further discussed in Chapter 5.

Since the latent vector $\mathbf{z}$ has fewer entries than its phenotype $\mathbf{x} := G(\mathbf{z})$ it can be seen as a compact and virtual high-level representation of $\mathbf{x}$, hence the name. The following example, taken from Karras et al. (2020) who published StyleGAN2, will be revisited throughout this chapter.

Example 3.1.2 (Facial images). Consider the DGP $\mathscr{D}$ of taking photos from human faces where $\mathcal{X} := [0, 255]^{1024\times1024\times3}$ is the data space of RGB images with pixel resolution 1024^2 and pixel values ranging continuously from 0 to 255. Then, $\mathcal{X}$ has a dimension of more than 3 million. A good choice of the latent space for a DGM to train on is $\mathcal{Z} := \mathbb{R}^{512}$. In fact, StyleGAN2 (Karras et al., 2020) which was designed to learn $\mathscr{D}$ uses this dimensionality choice and is one of the state-of-the-art face generation models.

3.2 Qualitative Properties

Let $\mathbf{z}_1, \mathbf{z}_2 \in \mathcal{Z}$ be two latent vectors with phenotypes $\mathbf{x}_1, \mathbf{x}_2$, respectively. Empirical studies have shown (Bojanowski et al., 2017; N. Chen et al., 2018; Shen et al., 2020; Voynov and Babenko, 2020) that the latent space exhibits at least three interesting properties that often—but not always—hold.

Locality If $\mathbf{z}_1$ and $\mathbf{z}_2$ are close (regarding, e.g., the Euclidean metric), then their phenotypes are close as well regarding some data-specific distance measure. In the case of the facial image example, the LPIPS distance can be used.

Valid interpolation The interpolated latent vector

$$\mathbf{z} := \lambda\mathbf{z}_1 + (1-\lambda)\mathbf{z}_2, \qquad \lambda \in [0, 1]$$

has a valid phenotype $\mathbf{x}$. It is crucial to note that this property is very different from the case where the interpolated phenotypes collapse to some invalid ”average” data point.

Varying λ creates an interpolation path in $\mathcal{Z}$. Moving $\mathbf{z}$ along that path generates a nonlinear interpolation path in $\mathcal{X}$ with meaningful phenotypes and smooth transitions between them. In the case of facial images, one face becomes morphed into another face as visualized in Figure 3.1. Besides, not all (high-level) properties of the interpolated results need to lie in their respective property interval that is spanned by $\mathbf{x}_1$ and $\mathbf{x}_2$ as can be seen in the aforementioned figure. More specifically, on one hand, face perspective and gender appear to lie in between while, on the other hand, age, mouth shape, and background color do not.

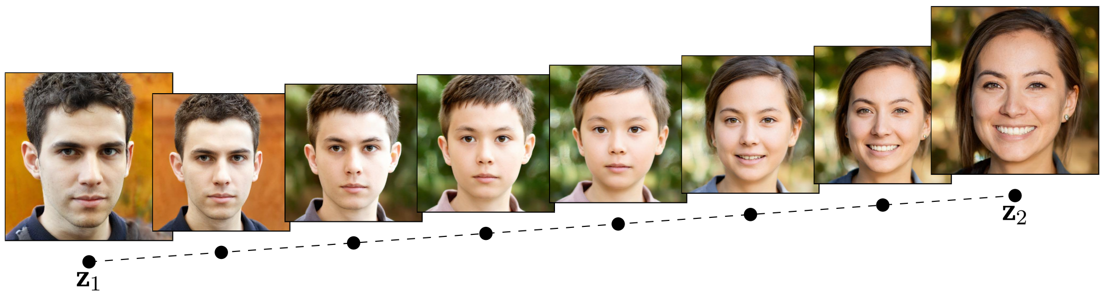


Figure 3.1: Image morphing that results from interpolation in latent space $\mathcal{Z}$. The latent vectors $\mathbf{z}_1$ and $\mathbf{z}_2$ are the endpoints of the interpolation line, the face images are the corresponding phenotypes. Images were generated using `https://facemorph.me`.

Vector arithmetic Let $\mathbf{x}_1$ and $\mathbf{x}_2$ be same except for a single binary property which is exhibited by $\mathbf{x}_2$ but not by $\mathbf{x}_1$. Let $\mathbf{z} \in \mathcal{Z}$ be some latent vector the phenotype $\mathbf{x}$ of which doesn't have said property. Then,

$$\mathbf{x}' := G(\mathbf{z} + (\mathbf{z}_2 - \mathbf{z}_1))$$

is similar to $\mathbf{x}$ except it has said property.

Intuitively, adding the vector $\mathbf{z}_2 - \mathbf{z}_1$ to $\mathbf{z}$ adds the missing property to $\mathbf{x}$. For example, in the case of human faces, this can be the property of having eyeglasses. That is, if the faces $\mathbf{x}$ and $\mathbf{z}_1$ don't wear eyeglasses but $\mathbf{z}_2$ does, then the face $\mathbf{x}'$ is a modification of face $\mathbf{x}$ wearing eyeglasses. This example is visualized in Figure 3.2. In fact, there is evidence that, "for any binary semantic, there exists a hyperplane in the latent space serving as the separation boundary." (Shen et al., 2020)

Disentanglement Since the generator G may be any function, the individual entries of an input $\mathbf{z} \in \mathcal{Z}$ usually do not have any interpretable effect on the output's semantics. (X. Chen et al., 2016) In contrast to that, in a *disentangled* latent space, each dimension of $\mathbf{z}$ has its own independent semantic meaning in data space. That is, varying the latent

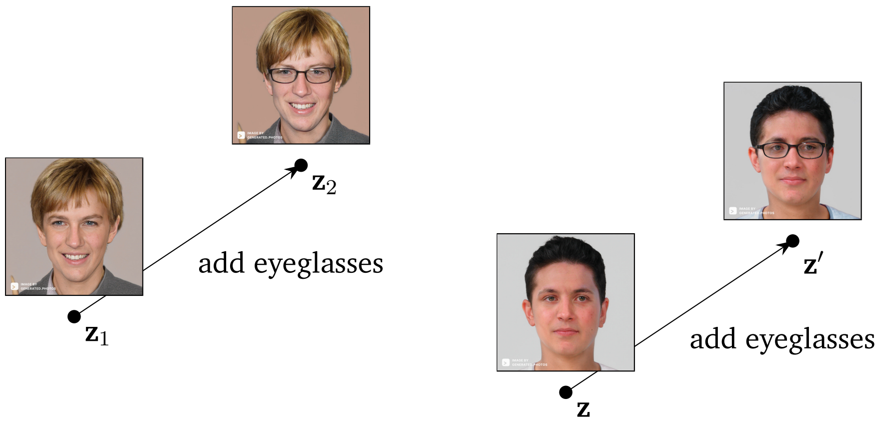


Figure 3.2: Vector arithmetic in the latent space. The vector $\mathbf{z}_2 - \mathbf{z}_1$ corresponds to "adding eyeglasses" to a given face image. This works even if $\mathbf{z}_1$ and $\mathbf{z}$ belong to completely different faces. Images generated with Generated Media (2022).

vector $\mathbf{z}$ in only a single entry changes only one interpretable aspect of the phenotype, see Figure 3.3 for a demonstration. Various disentanglement methods exist (X. Chen et al., 2016; Higgins et al., 2016; Liu et al., 2020; Lee et al., 2020; Ramesh et al., 2018; Voynov and Babenko, 2020), including learning a disentangled latent space in the first place, as done with StyleGAN2 (Karras et al., 2020), or identifying interpretable directions which can be used for a change of basis (Shen et al., 2020).

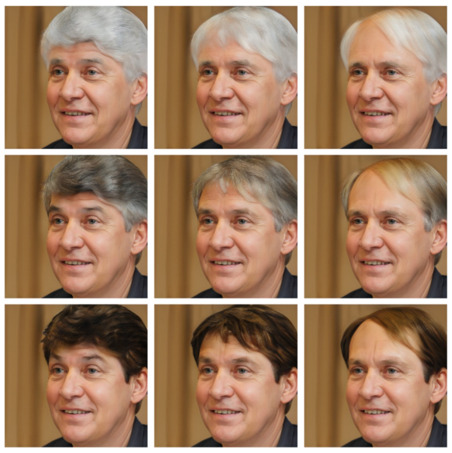


Figure 3.3: Single attribute manipulation done inside a disentangled latent space. Rows show variations in the latent dimension for "amount of hair", columns show variations of hair color. Image taken from Z. Wu et al. (2020). Note that the disentanglement here isn't perfect since, for example, changing the amount of hair also affects the position of the eyebrows.

Understanding the GANs' success to learn the target DGP $\mathscr{D}$ is still an open field of research. (Shen et al., 2020) One plausible explanation is the Manifold Hypothesis stating that (high-dimensional) real-world data lies on low-dimensional manifolds embedded within the high-dimensional space (DeepAI, 2021). That is, $\text{supp}(P_{\mathscr{D}})$ is a low-dimensional manifold in $\mathcal{X}$. This hypothesis is supported by the observation that low-dimensional choices of $\mathcal{Z}$ are still sufficient, as can be seen with StyleGAN2. In other words, the choice of the dimensionality of $\mathcal{Z}$ depends on the manifold's dimension.

3.3 Topology

Revisiting the interpolation example from Figure 3.1, the interpolation line used to generate the images is the Euclidean shortest path between $\mathbf{z}_1$ and $\mathbf{z}_2$. Nonetheless, the interpolated phenotypes certainly do not form a shortest path in data space. Otherwise, the background would not change to green and the person's age would not drop significantly on the interpolation line. This observation implies that shortest paths in $\mathcal{X}$ (w.r.t. some data-specific metric) do not necessarily correspond to shortest paths in $\mathcal{Z}$ w.r.t. the Euclidean metric. In fact, the underlying geometric structure of the latent space is in most cases not Euclidean. (Michelis and Becker, 2021) However, knowing the latent space geometry is essential for DGM evaluation. (Michelis and Becker, 2021)

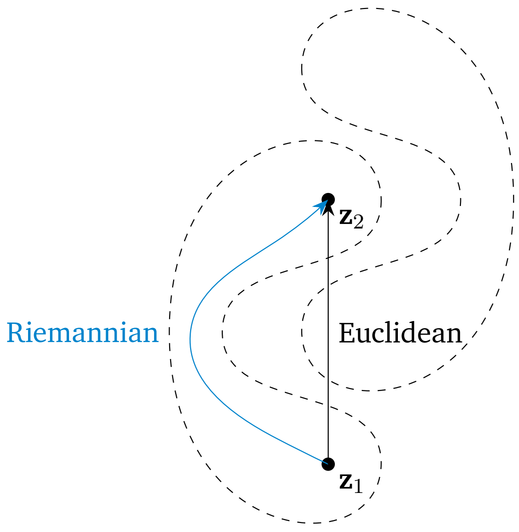


Figure 3.4: Shortest paths in latent space. The dashed lines highlight latent regions of two different classes, e.g., faces with or without glasses. Inspired by Arvanitidis et al. (2017).

Data points that (w.r.t. $P_{\mathscr{D}}$) lie in low-density regions of $\mathcal{X}$ are pushed together in $\mathcal{Z}$, making stationary distances poor proxies for similarity. (N. Chen et al., 2018) Therefore, several different works (N. Chen et al., 2018; Arvanitidis et al., 2017; Michelis and Becker, 2021; Shen et al., 2020) suggest to use a Riemannian induced metric

$M := J^T J$, where $J := \frac{\partial G}{\partial \mathbf{z}}$ is the Jacobian of the generator. It is also called the ”pull-back metric.” (Michelis and Becker, 2021) In experiments with VAEs, Arvanitidis et al. (2017) observed that interpolation walks from one $\mathbf{z}_1 \in \mathcal{Z}$ to some other $\mathbf{z}_2 \in \mathcal{Z}$ generated smoother transitions when using Riemannian shortest paths instead of Euclidean ones. Figure 3.4 showcases the situation: Generating images along the Euclidean interpolation line would result in a sequence that contains instances that deviate strongly from both endpoints. The Riemannian shortest path avoids this by staying in the same class.

4 Latent Space Optimization (LSO)

4.1 The Core Idea: Optimize via the Latent Space Using a Surrogate Function

Reconsider our original problem OPT and recall that, in the especially hard case, the data space $\mathcal{X}$ is discrete and the objective function f is non-differentiable, black-box, and expensive to evaluate.

Latent Space Optimization (LSO) is a heuristic optimization method that can be used to solve unhandy optimization problems like OPT. LSO does it by performing model-based Bayesian optimization in the latent space $\mathcal{Z}$ of a generator G. To this end, the following function acts as a surrogate for f.

Definition 4.1.1 (Latent objective function)**.** The function

$$\begin{aligned} h : \mathcal{Z} &\to \mathbb{R} \\ \mathbf{z} &\mapsto h(\mathbf{z}) := f(G(\mathbf{z})) \end{aligned}$$

is called *latent objective function*, cf. Figure 4.1.

$$\begin{tikzcd}
& \mathcal{X} \arrow[dr, "f"] & \\
\mathcal{Z} \arrow[ur, "G"] \arrow[rr, "{h, H}"'] & & \mathbb{R}
\end{tikzcd}$$

Figure 4.1: The functions and sets involved in LSO.

The core approach of LSO is to

(1) train a differentiable model H (called *latent objective model*) that approximates the (unknown) latent objective function h and

(2) optimize

$$\max_{\mathbf{z}\in\mathcal{Z}} H(\mathbf{z}) \tag{LSOPT}$$

yielding a solution $\hat{\mathbf{z}}$.

The expectation is that $h(\hat{\mathbf{z}})$ is close to the global optimum of OPT. In fact, under strong conditions, we can easily conclude the following useful relation between the global optimizers of OPT and LSOPT.

Proposition 4.1.2. *Let $\mathbf{x}^*$ be any global optimizer of OPT for which $\mathbf{x}^* \in G(\mathcal{Z})$ and assume that $H = h$. If $\hat{\mathbf{z}}$ is a global optimizer of LSOPT, then its phenotype $\hat{\mathbf{x}} := G(\hat{\mathbf{z}})$ is a global optimizer of OPT.*

Proof. Let $\hat{\mathbf{z}}$ be a global optimizer of LSOPT. Since $\mathbf{x}^*$ is an element of the image of G, there exists $\mathbf{z}^* \in \mathcal{Z}$ with $G(\mathbf{z}^*) = \mathbf{x}^*$. Because $\hat{\mathbf{z}}$ is a global optimizer, $H(\hat{\mathbf{z}}) \geq H(\mathbf{z}^*)$. Analogously, since $\mathbf{x}^*$ is optimal, $f(\mathbf{x}^*) \geq f(G(\hat{\mathbf{z}}))$. Using $H = h$ and $h = f \circ G$, we get

$$H(\hat{\mathbf{z}}) \geq H(\mathbf{z}^*) = f(G(\mathbf{z}^*)) = f(\mathbf{x}^*) \geq f(G(\hat{\mathbf{z}})) = H(\hat{\mathbf{z}}),$$

so

$$f(\mathbf{x}^*) = H(\hat{\mathbf{z}}) = f(G(\hat{\mathbf{z}})),$$

concluding the proof. □

The requirement that the generator's image $G(\mathcal{Z})$ contains an optimal solution is met with a high chance if G is well-trained, i.e., if $P_G \approx P_{\mathcal{D}}$. This is due to the fact that $\mathrm{supp}(P_G)$ is then roughly congruent to $\mathrm{supp}(P_{\mathcal{D}})$ which contains $\mathbf{x}^*$. Unfortunately, with today's methods, we only can approximate G since we rely on parameterization. The same is true for H, so the condition $H = h$ is hardly achievable. The LSO algorithm, as we will see in the next section, contains an approach to lay the foundations so that Proposition 4.1.2 can be (heuristically) applied. That is, the algorithm tries to learn proper G and H, maximizing the chances to derive a near-optimal solution.

4.2 The Native LSO Algorithm

Definition 4.2.1. Let $G : \mathcal{Z} \to \mathcal{X}$ be some generator. A function

$$G^{-1} : \mathcal{X} \to \mathcal{Z}$$

with $G(G^{-1}(\mathbf{x})) = \mathbf{x}$ for all $\mathbf{x} \in G(\mathcal{Z})$ is called *inverse model* of G.

In practice, the inverse model is not necessarily known, depending on the used DGM. For example, G^{-1} exists readily in the case of VAEs, but in the case of native GANs, G^{-1} needs to be constructed/trained first. The problem of finding the inverse model is called *source attribution*. Note that only data points in the image $G(\mathcal{Z})$ of G have a latent vector since G is not surjective on $\mathcal{X}$ in general. In fact, it can't if $\mathcal{Z}$ has a lower dimension than $\mathcal{X}$.

In the ML context, we need to "train" the generator G first, using a training dataset $\mathcal{D} := \{(\mathbf{x}_i, f(\mathbf{x}_i)) \mid \mathbf{x}_i \in \mathcal{X}, 1 \leq i \leq N, N \in \mathbb{N}\}$. (For notation simplicity, we only denote the first element of the pair if we refer to specific elements of $\mathcal{D}$.) Training means, an ML algorithm is applied to construct a G that is able to reconstruct all $\mathbf{x}_i$ of $\mathcal{D}$ and whose generator distribution resembles the true distribution (by the ML generalization principle). The internals of the ML algorithm are DGM-dependent and skipped here. We suppose that the ML training algorithm produces a desired G.

For LSO to work, we need to assume that each data point $\mathbf{x}_i$ of $\mathcal{D}$ lies in $\text{supp}(P_G)$ (which is typically the case after training). Obviously, $\text{supp}(P_G) \subseteq G(\mathcal{Z})$, therefore our assumption implies $\mathbf{x}_i \in G(\mathcal{Z})$. On a sidenote, the relation $\text{supp}(P_G) \subseteq G(\mathcal{Z})$ might be strict, because $\text{supp}(P_\mathcal{Z}) \subseteq \mathcal{Z}$ can be strict as well. Finally, we need to assume that we can compute the inverse model G^{-1}. Given these requirements, repeating steps 1 and 2 described above for a fixed number of iterations M then gives us the LSO algorithm in its native form, displayed in Algorithm 2, cf. Siivola et al. (2021) and Tripp et al. (2020).

The conjecture here is that with each iteration, H converges to h and, thus, the chances increase that the output of Algorithm 2 is (close to) the optimal solution, motivated by Proposition 4.1.2. Briefly, LSO effectively tackles the difficulties of OPT as follows:

- LSO optimizes on the continuous space $\mathcal{Z}$, avoiding the discreteness of $\mathcal{X}$.
- H is known, differentiable, and cheap to evaluate in contrast to f.

However, there is a particular situation in which LSO (in its native form) fails.

Algorithm 2: Latent Space Optimization

```
Input  : Dataset $\mathcal{D} = \{(\mathbf{x}_i, f(\mathbf{x}_i))\}_{i=1}^N$, query budget $M$, objective function $f$,
         generator $G$
Output : solution $\mathbf{x} \in \mathcal{X}$ that is expected to have a value close to the optimum of OPT
Train generator $G$ on $\mathcal{D}$
Compute latent vectors to obtain dataset
 $\mathcal{D}_\mathcal{Z} = \{(\mathbf{z}, f(\mathbf{x})) \mid \mathbf{z} = G^{-1}(\mathbf{x}), (\mathbf{x}, f(\mathbf{x})) \in \mathcal{D}\}$
for $1, \dots, M$ do
    Train a latent objective model $H$ on $\mathcal{D}_\mathcal{Z}$
    Obtain solution $\hat{\mathbf{z}} \in \mathcal{Z}$ by optimizing $H$ over $\mathcal{Z}$ (LSOPT)
    $\hat{\mathbf{x}} \leftarrow G(\hat{\mathbf{z}})$
    Add $(\hat{\mathbf{x}}, f(\hat{\mathbf{x}}))$ to $\mathcal{D}$ and add $(\hat{\mathbf{z}}, f(\hat{\mathbf{x}}))$ to $\mathcal{D}_\mathcal{Z}$
return $\arg\max_{(\mathbf{x},\cdot)\in\mathcal{D}} f(\mathbf{x})$
```

4.3 An Important Failure Mode of Native LSO

In the following, let $\mathbf{x}^*$ be a global optimizer of OPT.

Proposition 4.3.1. *If $\mathbf{x}^* \notin G(\mathcal{Z})$ then $\mathbf{x}^*$ cannot be found using Algorithm 2.*

Proof. If, for the sake of contradiction, $\mathbf{x}^*$ were the output of Algorithm 2, then $\mathbf{x}^*$ must have been element of $\mathcal{D}$ right before termination. Hence, $\mathbf{x}^*$ was either included in the initial $\mathcal{D}$ or it was added to it during the loop. In the first case, $\mathbf{x}^* \in G(\mathcal{Z})$ because we assumed that all data points of $\mathcal{D}$ are in $\operatorname{supp}(P_G)$ after training. In the second case, $\mathbf{x}^*$ must have been the result of the computation of $G(\hat{\mathbf{z}})$ where $\hat{\mathbf{z}}$ was the optimal solution of LSOPT. In any of both cases, $\mathbf{x}^* \in G(\mathcal{Z})$, opposing the proposition's assumption. □

However, *even if any* optimal $\mathbf{x}^*$ is element of $G(\mathcal{Z})$, it can be hard to find a solution as good as $\mathbf{x}^*$. To understand the reason, we need

Definition 4.3.2. Let $\varepsilon > 0$ be small. A bounded subset $\mathcal{Z}' \subsetneq \mathcal{Z}$ with

$$\int_{\mathcal{Z}'} p_\mathcal{Z}(\mathbf{z})\, \mathrm{d}\mathbf{z} > 1 - \varepsilon$$

is called *feasible (latent) region*.

In words, a feasible latent region $\mathcal{Z}'$ is a bounded subset of the latent space $\mathcal{Z}$ that entails practically all the probability mass of the prior $P_{\mathcal{Z}}$. Even if $\mathcal{Z}$ and $\text{supp}(P_{\mathcal{Z}})$ are unbounded, $\mathcal{Z}'$ exists since $P_{\mathcal{Z}}$ has finite volume.

As identified by Tripp et al. (2020), often much of the training data is low-scoring, i.e., has low objective values. Consequently, most of the feasible latent region's phenotypes are low-scoring as well. Hence, the feasible region devotes little or no space to high-scoring solutions, causing many such solutions to lie outside the generator distribution's support $\text{supp}(P_G)$. Hence, even if an optimal solution $\mathbf{x}^*$ is in the image of G, it likely lies outside the region $G(\mathcal{Z}')$. This is problematic for the reason that G usually generates low-quality (and, thus, low-scoring) results for latent vectors of $\mathcal{Z} \setminus \mathcal{Z}'$ which is due to the lack of training inside that region. For the same reason, the latent objective model H performs less accurately outside $\mathcal{Z}'$. See Figure 4.2 for a visualization of this failure mode.

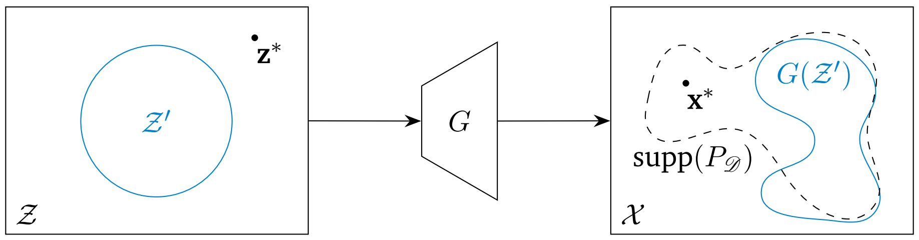


Figure 4.2: Failure mode of LSO where the optimal solution $\mathbf{x}^*$ lies in the image of G but outside the data region $G(\mathcal{Z}')$. $\mathbf{z}^*$ is the latent vector of $\mathbf{x}^*$. Note that, in this example, $\text{supp}(P_{\mathscr{D}})$ is bounded, but it may be unbounded in general.

4.4 Addressing the Failure Mode with Weighted Retraining

In order to settle the previously described failure mode, Tripp et al. (2020) propose to modify the native LSO algorithm. The modification introduces a recurring update of G to shift its distribution to put more probability mass on high-scoring data. That is, we dismiss the idea that G matches its distribution the true distribution $P_{\mathscr{D}}$ and, instead, increase the chance that G covers an optimal solution $\mathbf{x}^*$ of OPT with high probability.

Note that in the native LSO algorithm, Algorithm 2, each iteration of the for loop yields new information in the form of a new latent vector $\hat{\mathbf{z}}$, its phenotype $G(\hat{\mathbf{z}})$, and the identified true value $f(G(\hat{\mathbf{z}}))$. This information is used only by the objective model H but ignored by the generator G. That is, G remains unchanged during the whole LSO process. One idea is to use the information to *retrain* G so that it can improve modeling high-scoring data. Retraining can be done via SGD every few iterations of LSO using the newly acquired information. Moreover, in the standard case, SGD samples batches of training data from $\mathcal{D}$ using a uniform probability distribution. However, we can modify that distribution to foster high-scoring solutions in $\mathcal{D}$. That is, we can increase the SGD sampling probability for high-scoring solutions and decrease it for low-scoring ones. Tripp et al. (2020) call this approach *weighting* and propose one possible new sampling distribution which we omit here. In general, the authors define a *weighting function* as $w : \mathcal{D} \to \mathbb{R}$, for which

$$w(d) > 0, \forall d \in \mathcal{D} \qquad \text{and} \qquad \sum_{d \in \mathcal{D}} w(d) = 1.$$

Algorithm 3 shows the modified LSO algorithm as proposed by Tripp et al. (2020), where modifications compared to Algorithm 2 are highlighted in blue. The periodic retraining of G is expected to reshape $G(\mathcal{Z}')$ as sketched in Figure 4.3. In words, the assertion is that weighted retraining ensures $\mathbf{x}^* \in G(\mathcal{Z}')$, ultimately solving the failure mode.

Algorithm 3: Latent Space Optimization with Weighted Retraining

```
Input  : Dataset $\mathcal{D} = \{(\mathbf{x}_i, f(\mathbf{x}_i))\}_{i=1}^N$, query budget $M$, objective fn. $f$, generator $G$,
         retrain frequency $r$, weighting fn. $w$
Output : solution $\mathbf{x} \in \mathcal{X}$ that is expected to have a value close to the optimum of OPT
for $1, ..., \frac{M}{r}$ do
    Train generator $G$ on $\mathcal{D}$ weighted by $w$
    Compute latent vectors to obtain dataset
      $\mathcal{D}_\mathcal{Z} = \{(\mathbf{z}, f(\mathbf{x})) \mid \mathbf{z} = G^{-1}(\mathbf{x}), (\mathbf{x}, f(\mathbf{x})) \in \mathcal{D}\}$
    for $1, ..., r$ do
        Train a latent objective model $H$ on $\mathcal{D}_\mathcal{Z}$
        Obtain solution $\hat{\mathbf{z}} \in \mathcal{Z}$ by optimizing $H$ over $\mathcal{Z}$ (LSOPT)
        $\hat{\mathbf{x}} \leftarrow G(\hat{\mathbf{z}})$
        Add $(\hat{\mathbf{x}}, f(\hat{\mathbf{x}}))$ to $\mathcal{D}$ and add $(\hat{\mathbf{z}}, f(\hat{\mathbf{x}}))$ to $\mathcal{D}_\mathcal{Z}$
return $\arg\max_{(\mathbf{x},\cdot) \in \mathcal{D}} f(\mathbf{x})$
```

On a sidenote, retraining the generator G changes the assignment of data points to the latent vectors. Therefore, after each modification of G, the dataset $\mathcal{D}_\mathcal{Z}$ needs to be updated entirely before being reused by the latent objective model H for training. This is opposed to the native LSO algorithm where the latent vector of each data point in $\mathcal{D}$ is calculated only once.

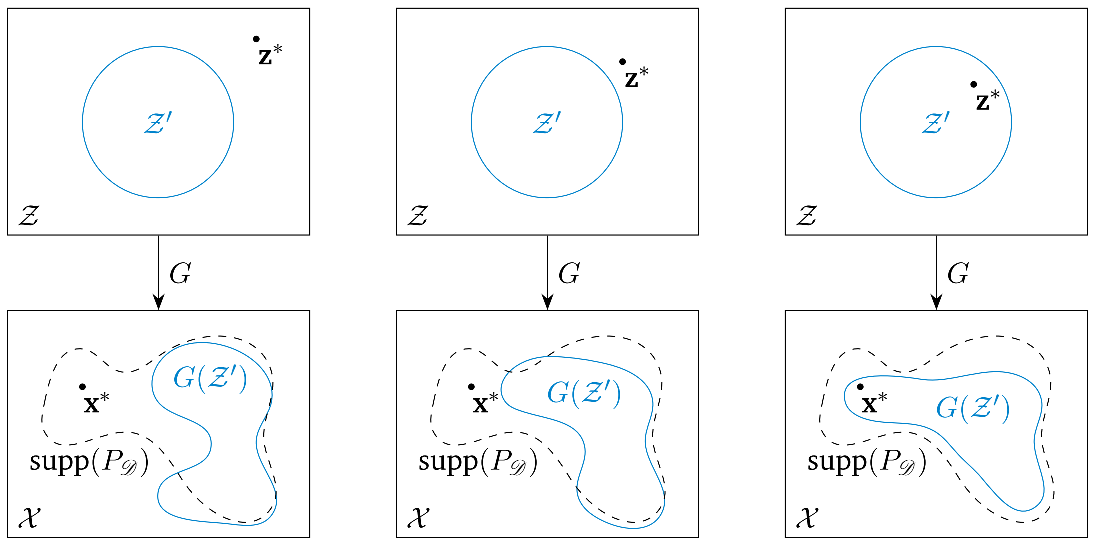


Figure 4.3: The expected change of $G(\mathcal{Z}')$ and $\mathbf{z}^*$ during LSO with weighted retraining, from the beginning (left) where the optimal solution $\mathbf{x}^*$ is not in $G(\mathcal{Z}')$ to the end (right) where finally $\mathbf{x}^* \in G(\mathcal{Z}')$. Note that, due to the changes of G, the latent vector $\mathbf{z}^*$ of the optimal solution changes as well.

4.5 The Latent Objective Model H

The LSOPT solving step of the LSO algorithm is typically done in the setting of Bayesian Optimization (BO). A popular choice for H is a Gaussian Process (GP) (Rasmussen, 2003). (Tripp et al., 2020) Essentially, based on the information contained in $\mathcal{D}_{\mathcal{Z}}$, H in some sense "proposes" a latent vector to consider next. Therefore, H is also called *acquisition function*. (Siivola et al., 2021)

The selection strategy of the acquisition function can be arbitrary. Note that, instead of always selecting a sample with the highest expected value, it may be more useful to choose a sample with a high potential for knowledge gain. That is, exploring yet unevaluated regions of the latent space $\mathcal{Z}$ can reveal new areas of high value. This *exploration-exploitation trade-off* is a common dilemma found in the BO literature. (González et al., 2015; Moriconi et al., 2019; Shahriari et al., 2016) GPs are a suitable tool for this situation as GPs not only predict values for latent vectors but also provide a value of uncertainty. In fact, recent GP-based BO surpasses non-GP BO methods. (Ahn et al., 2022)

4.6 Related Work

The earliest work on LSO was published by González et al. (2015) who apply BO using GP to a reduced, continuous representation of high-dimensional and discrete data to solve the problem of protein design. The first ones to perform LSO with DGMs were Gómez-Bombarelli et al. (2016). (Deshwal and Doppa, 2021) Inspired by these pioneers, an extensive number of methodological LSO works followed. (X. Lu et al., 2018; Eismann et al., 2018; Kajino, 2019; Moriconi et al., 2019; Tripp et al., 2020; Bartunov et al., 2020; Grosnit et al., 2021; Notin et al., 2021; M. Lu et al., 2021; Siivola et al., 2021; Maus et al., 2022; Castro et al., 2022)

Despite being still an emerging technique, LSO is applied to a quickly growing range of tasks and domains. This includes chemical design (Gómez-Bombarelli et al., 2016; Jin et al., 2018; Griffiths and Hernández-Lobato, 2020), protein/gene design (Castro et al., 2022; González et al., 2015), physics (Park et al., 2022; Tucci et al., 2021), neural architecture search (Luo et al., 2018), and robotics movement trajectories (Antonova et al., 2020). Explicit recommendations for LSO practitioners were collected in the work of Siivola et al. (2021).

In the literature, LSO is sometimes referred to as ”latent space BO” (Maus et al., 2022; Stanton et al., 2022), ”continuous latent search” (Bartunov et al., 2020), and, in the case of VAEs, ”VAE-BO” (Grosnit et al., 2021).

Several issues of LSO were identified and addressed. For example, as mentioned above, Tripp et al. (2020) introduced weighted retraining to emphasize high-scoring latent regions, ultimately increasing the chance of finding an optimal solution. Similarly, X. Lu et al. (2018) propose Structure Generating VAEs (SG-VAEs) that can be used to actively guide the search towards certain latent regions.

Generators do not guarantee to produce valid outputs, especially when fed with a latent vector far outside of the feasible region $\mathcal{Z}'$. To avoid invalid outputs, Grammar Variational Autoencoders (GVAE) were introduced by Kusner et al. (2017). GVAEs enforce the generation of only valid examples through the use of context-free grammar. In their case, $\mathcal{X}$ is a set of production rule sequences. Each such production rule corresponds to a valid solution. Nevertheless, creating a context-free grammar is not always practically possible (think of $\mathcal{X}$ being the set of images depicting human faces) and even if it is possible, it requires the definition of a set of appropriate production rules. Another model achieving 100 % validity was proposed by Kajino (2019). Other and more general LSO advancements were provided by Maus et al. (2022), Grosnit et al. (2021), and Deshwal and Doppa (2021).

4.6.1 Optimization Methods Related to LSO

Kumar and Levine (2020) use an approach very similar to LSO. They propose Model Inversion Networks (MINs) as an MBO method. The core idea is to train a model that, inversely to the objective function f, maps from value space $\mathcal{Y}$ (often $\mathbb{R}$) into data space $\mathcal{X}$, allowing for optimization over the low-dimensional value space $\mathcal{Y}$. In order to foster generation diversity, a latent space $\mathcal{Z}$ is added to the domain of the MIN. More precisely, a MIN f_θ^{-1} is defined as the map

$$f_\theta^{-1} : \mathcal{Y} \times \mathcal{Z} \to \mathcal{X},$$

mapping a conditional variable or label $\mathbf{y} \in \mathcal{Y}$ together with a random vector $\mathbf{z} \in \mathcal{Z}$ from some prior distribution to a data point $\mathbf{x} \in \mathcal{X}$. Using a GAN as a MIN instance, the setting is similar to that of a C-GAN (Mirza and Osindero, 2014). Compared to the method proposed by Kumar and Levine (2020), LSO doesn't incorporate the value space $\mathcal{Y}$ as a part of the generator's domain. Thus, the setting of LSO is simpler, avoiding optimization in $\mathcal{Y}$ which, besides, may be discrete. In contrast to LSO, where a generator can be trained on *unlabeled* data, a MIN requires the entire training set to be labeled, i.e., evaluated. This is infeasible for large amounts of data in the case where the objective function is expensive to evaluate. If $(\mathbf{y}, \mathbf{z}) \in \mathcal{Y} \times \mathcal{Z}$ is an input for a MIN, $\mathbf{y}$ is, in the LSO context, simply another entry in the latent vector with the only difference that it is interpretable by humans. If $\mathbf{y}$ represents the expected value of the corresponding phenotype, it could be used to guide LSO into regions of $\mathcal{Z}$ with high $\mathbf{y}$.

Fu and Levine (2021) introduce an optimization method for high-dimensional, expensive-to-evaluate problems by utilizing the normalized maximum likelihood (NML) estimator. Their method, which they call Normalized Maximum Likelihood Estimation for MBO (NEMO), can be applied to high-dimensional design problems in chemistry, biology, and materials engineering. The strongest difference to LSO lies in the fact that NEMO is done in data space $\mathcal{X}$ whereas LSO is done in latent space $\mathcal{Z}$, avoiding discreteness and high dimensionality.

LSO requires a DGM that, additionally, needs to be trained in advance. Moriconi et al. (2019) instead propose to learn a response surface to perform LSO on, avoiding the need for a DGM. Besides, the direct application of BO to high-dimensional data without any lower-dimensional latent space or DGM was conducted in Wang et al. (2016).

5 Discussion

5.1 Limitations of LSO

First and foremost, LSO as an ML-based optimization method is purely heuristic and, thus, delivers no quality guarantees for the end result. Although the theoretic motivation is sound, the conditions are practically not satisfiable or verifiable. As a consequence, current LSO theory mostly comprises conjectures and assertions supported by empirical observations. In order to derive quantifiable guarantees, the theory needs to be further investigated on a principled, formal level, e.g., by restricting the DGM or the problem space, left for future work.

Moreover, the success of LSO depends heavily on the quality of the DGM. More specifically, the DGM's generator is required (1) to produce only a small share of invalid solutions (because each evaluation of an invalid solution poses a waste of expensive evaluation resources) and (2) to cover as much of the solution space as possible. The latter is related to the intensely studied challenge of getting a high output diversity. Output diversity depends a lot on the training dataset which, besides, needs to be large as well—a typical issue for ML methods.

It is hard to choose the right dimensionality of the latent space $\mathcal{Z} := \mathbb{R}^m, m \in \mathbb{N}$ because the right size is task-dependent and the exact consequences of varying the dimension are not well explored. (Siivola et al., 2021) In any case, if m is chosen too small, the generator performs badly and shows low diversity while a too large m promotes overfitting and hinders generalization. (Siivola et al., 2021) Despite everything, m is still large in practice (Maus et al., 2022; Deshwal and Doppa, 2021), weakening the advantage of optimizing in $\mathcal{Z}$ as BO is practically limited to optimizing 10–20 parameters. (Moriconi et al., 2019) Therefore, follow-up work proposed LOL-BO (Maus et al., 2022), which is an optimization strategy that uses the notion of trust regions, a concept applied in BO to high-dimensional problems.

As already mentioned, sampling outside the feasible region $\mathcal{Z}'$ leads to low-quality outputs with high probability. Therefore, it is desirable to stay inside $\mathcal{Z}'$—which is simple in the case of GANs by just obeying $P_{\mathcal{Z}}$. In contrast to that, this is rather difficult for VAEs. Thus, Notin et al. (2021) propose an importance sampling-based estimator that approximates the epistemic uncertainty of the generator in order to determine if a given latent vector is in $\mathcal{Z}'$.

Recall LSO with weighted retraining (Algorithm 3). That algorithm extends high-value regions in latent space. However, training the latent objective model H inside the modified latent space appears challenging in practice. (Tripp et al., 2020) This is because, in general, high-value latent vectors might be split up into numerous disconnected regions across the latent space. Indeed, that low connectivity hinders GPs (or BO in general) to fit to the latent objective. (Grosnit et al., 2021) Additionally, it may happen that the generator didn't learn to generate the information necessary to evaluate data instances, making it difficult for H to learn the objective function. (Deshwal and Doppa, 2021)

Moreover, Algorithm 3 requires frequent retraining of the generator. Retraining may be expensive in specific applications. Nevertheless, evaluations of the objective function may be magnitudes more expensive (as is the case of drug synthesis), relativizing model retraining costs. (Tripp et al., 2020)

5.2 Limitations of the Showed GAN Theory

In contrast to Goodfellow et al. (2014), who provided an incomplete proof for the optimal discriminator (Lemma 2.2.5), we showed it for the very special case when G is a diffeomorphism, both D and p_G are continuous, and the optimal discriminator D_G^* exists. These requirements, are radical, especially the diffeomorphism condition. It implies that $\mathcal{Z}$ and $\mathcal{X}$ need to have the same dimension and that G is invertible. First, in contrast to, e.g., VAEs and NFs, GANs have no inverse generator by nature. Second, $\text{dim}(\mathcal{Z}) = \text{dim}(\mathcal{X})$ prohibits us from constructing a latent space that has a lower dimension than $\mathcal{X}$. For the very desirable case $\text{dim}(\mathcal{Z}) < \text{dim}(\mathcal{X})$, the optimal discriminator property of Lemma 2.2.5 is *not* valid (Pulford and Kondrashov, 2021). Even if Theorem 2.2.7 can be shown for this case, the theory is not applicable in practice due to parameterization. In a nutshell, although having a simple key intuition, GANs have an immature theory and, thus, might not be the right choice for LSO.

5.3 Broader Impact

On the one hand, LSO is a possible candidate for the design of novel solutions to many real-world problems. For example, it could be used to create new drugs and vaccines to cure intractable illnesses like cancer or HIV, or to stop the ongoing spread of COVID, flu etc. LSO could also accelerate the progress on the genetic enhancement of different life forms, including humans.

On the other hand, as with most ML approaches, researchers and applied scientists should be aware of the capabilities that arise with these techniques, good and bad. Dual use examples exist already. As an instance, the latent space of DGMs can be straightforwardly used to manipulate the semantics of photos or videos. Results of such manipulations occurred publicly on the web as so-called *deepfakes* and are considered to be an increasing threat[1]. (*AI Index Report* 2022) Luckily, multiple methods to detect deepfakes exist (*AI Index Report* 2022; Nguyen et al., 2019), but their capabilities are limited at some point. (Karras et al., 2019) LSO, in particular, could be abused to circumvent such detectors and to produce higher fidelity output. Furthermore, adversaries could exploit LSO for the design of destructive weapons—physical, chemical as well as biological. That danger is particularly high during conflicts between modern civilizations like the current Russo-Ukrainian war.

[1]The German Federal Office for Information Security extensively informs about the danger of deepfakes here: `https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Informationen-und-Empfehlungen/Kuenstliche-Intelligenz/Deepfakes/deepfakes_node.html`

6 Conclusion

This work presented Latent Space Optimization (LSO), a heuristic optimization method for problems with discrete and high-dimensional solution space and with an expensive-to-evaluate black-box objective. LSO relies on the low-dimensional, continuous latent space of a Deep Generative Model (DGM) trained on the solution space. An easy-to-evaluate surrogate function that maps from the latent space into solution space replaces the unhandy objective function. Several methodological adjustments, like weighted retraining, are expected to increase LSO's chances to return high-valued solutions.

A rigorous introduction to the theory of Generative Adversarial Networks (GANs) was given, a popular DGM representative. Also, the state of the art of DGMs was reviewed briefly. Then, this work presented useful properties of the latent space on a qualitative level—a more formal theory is yet to be established. Finally, the LSO method was introduced, and an important failure mode was examined and addressed. Despite being only about six years old, the collected literature shows that LSO gained remarkable attention, not only in theory but also in practice. Eventually, this work discussed LSO limitations and pointed out potential cases of misuse.

In addition, a proof for the GAN's optimal discriminator was provided for a special case. The original GAN paper (Goodfellow et al., 2014) skipped that proof, including critical requirements that limit the applicability of current GAN theory dramatically. However, this restriction is not crucial for LSO since LSO is model-agnostic. A more principled mathematical investigation of LSO, especially in combination with a DGM that has a well-established theory—regarding both the model and the latent space—could be an interesting future direction.

Acronyms

AI	Artificial Intelligence.
BO	Bayesian Optimization.
DGM	Deep Generative Model.
DGP	Data Generation Process.
GAN	Generative Adversarial Network.
GP	Gaussian Process.
LSO	Latent Space Optimization.
MBO	Model-based Optimization.
ML	Machine Learning.
NF	Normalizing Flow.
NN	Neural Network.
PDF	Probability Density Function.
SGD	Stochastic Gradient Descent.
VAE	Variational Autoencoder.

Bibliography

Ahn, Jaeyeon, Taehyeon Kim, and Seyoung Yun (2022). “Mold into a Graph: Efficient Bayesian Optimization over Mixed-Spaces”. In: *ArXiv* abs/2202.00893.

AI Index Report (2022). Tech. rep. Cordura Hall 201 Panama Street Stanford University Stanford, CA 94305: Stanford Institute for Human-Centered Artificial Intelligence. URL: `https://aiindex.stanford.edu/wp-content/uploads/2022/03/2022-AI-Index-Report_Master.pdf`.

Antonova, Rika, Akshara Rai, Tianyu Li, and Danica Kragic (2020). “Bayesian Optimization in Variational Latent Spaces with Dynamic Compression”. In: *Proceedings of the Conference on Robot Learning*. Ed. by Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura. Vol. 100. Proceedings of Machine Learning Research. PMLR, pp. 456–465. URL: `https://proceedings.mlr.press/v100/antonova20a.html`.

Arvanitidis, Georgios, Lars Kai Hansen, and Søren Hauberg (2017). “Latent Space Oddity: On the Curvature of Deep Generative Models”. In: *arXiv preprint arXiv:1710.11379*.

Bartunov, Sergey, Vinod Nair, Peter Battaglia, and Timothy P. Lillicrap (2020). “Continuous Latent Search for Combinatorial Optimization”. In: *Learning Meets Combinatorial Algorithms at NeurIPS2020*. URL: `https://openreview.net/forum?id=P3FX9pUev-`.

Bojanowski, Piotr, Armand Joulin, David Lopez-Paz, and Arthur Szlam (2017). “Optimizing the Latent Space of Generative Networks”. In: *arXiv preprint arXiv:1707.05776*.

Brookes, David, Hahnbeom Park, and Jennifer Listgarten (Sept. 2019). “Conditioning by Adaptive Sampling for Robust Design”. In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, pp. 773–782. URL: `https://proceedings.mlr.press/v97/brookes19a.html`.

Brown, Tom, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. (2020). “Language Models are Few-shot Learners”. In: *Advances in Neural Information Processing Systems* 33, pp. 1877–1901.

Castro, Egbert, Abhinav Godavarthi, Julian Rubinfien, Kevin B. Givechian, Dhananjay Bhaskar, and Smita Krishnaswamy (2022). *Guided Generative Protein Design using Regularized Transformers*. DOI: 10.48550/ARXIV.2201.09948. URL: https://arxiv.org/abs/2201.09948.

Chen, Nutan, Alexej Klushyn, Richard Kurle, Xueyan Jiang, Justin Bayer, and Patrick Smagt (2018). "Metrics for Deep Generative Models". In: *International Conference on Artificial Intelligence and Statistics*. PMLR, pp. 1540–1550.

Chen, Xi, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel (2016). "InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems* 29.

DeepAI (2021). *What is the Manifold Hypothesis?* Accessed September 17, 2021. URL: https://deepai.org/machine-learning-glossary-and-terms/manifold-hypothesis.

Deshwal, Aryan and Jana Doppa (2021). "Combining Latent Space and Structured Kernels for Bayesian Optimization over Combinatorial Spaces". In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan. Vol. 34. Curran Associates, Inc., pp. 8185–8200. URL: https://proceedings.neurips.cc/paper/2021/file/44e76e99b5e194377e955b13fb12f630-Paper.pdf.

Eismann, Stephan, Daniel Levy, Rui Shu, Stefan Bartzsch, and Stefano Ermon (2018). "Bayesian Optimization and Attribute Adjustment". In: *Proc. 34th Conference on Uncertainty in Artificial Intelligence*.

Fu, Justin and Sergey Levine (2021). "Offline Model-Based Optimization via Normalized Maximum Likelihood Estimation". In: DOI: 10.48550/ARXIV.2102.07970. URL: https://arxiv.org/abs/2102.07970.

Garnelo, Marta, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and S. M. Ali Eslami (Oct. 2018). "Conditional Neural Processes". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 1704–1713. URL: https://proceedings.mlr.press/v80/garnelo18a.html.

Generated Media, Inc. (2022). *Generated Photos*. Accessed Aug 14, 2022. URL: https://generated.photos/.

Gómez-Bombarelli, Rafael, David Duvenaud, José Miguel Hernández-Lobato, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alán Aspuru-Guzik (2016). "Automatic Chemical Design Using a Data-driven Continuous Representation of Molecules".

In: *CoRR* abs/1610.02415. arXiv: 1610.02415. URL: http://arxiv.org/abs/1610.02415.

González, Javier, Joseph Longworth, David C. James, and Neil D. Lawrence (2015). *Bayesian Optimization for Synthetic Gene Design*. DOI: 10.48550/ARXIV.1505.01627. URL: https://arxiv.org/abs/1505.01627.

Goodfellow, Ian (2016). "NIPS 2016 Tutorial: Generative Adversarial Networks". In: DOI: 10.48550/ARXIV.1701.00160.

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio (2014). "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems* 27.

Griffiths, Ryan-Rhys and José Miguel Hernández-Lobato (Jan. 2020). "Constrained Bayesian optimization for automatic chemical design using variational autoencoders". In: *Chemical science* 11.2, pp. 577–586. ISSN: 2041-6520. DOI: 10.1039/c9sc04026a.

Grosnit, Antoine et al. (2021). *High-Dimensional Bayesian Optimisation with Variational Autoencoders and Deep Metric Learning*. DOI: 10.48550/ARXIV.2106.03609. URL: https://arxiv.org/abs/2106.03609.

Harshvardhan, G. M., Mahendra Kumar Gourisaria, Manjusha Pandey, and Siddharth Swarup Rautaray (2020). "A Comprehensive Survey and Analysis of Generative Models in Machine Learning". In: *Computer Science Review* 38. ISSN: 1574-0137. DOI: https://doi.org/10.1016/j.cosrev.2020.100285.

Higgins, Irina, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner (2016). "Beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework". In: URL: https://openreview.net/forum?id=Sy2fzU9gl.

Hoburg, Warren and Pieter Abbeel (2014). "Geometric Programming for Aircraft Design Optimization". In: *AIAA Journal* 52.11, pp. 2414–2426. DOI: 10.2514/1.J052732. eprint: https://doi.org/10.2514/1.J052732. URL: https://doi.org/10.2514/1.J052732.

Jin, Wengong, Regina Barzilay, and Tommi Jaakkola (Oct. 2018). "Junction Tree Variational Autoencoder for Molecular Graph Generation". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 2323–2332.

Jumper, John, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. (2021). "Highly Accurate Protein Structure Prediction with AlphaFold". In: *Nature* 596.7873, pp. 583–589.

Kajino, Hiroshi (Sept. 2019). "Molecular Hypergraph Grammar with its Application to Molecular Optimization". In: *Proceedings of the 36th International Conference on*

Machine Learning. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, pp. 3183–3191. URL: `https://proceedings.mlr.press/v97/kajino19a.html`.

Karras, Tero, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila (2021). "Alias-free Generative Adversarial Networks". In: *Advances in Neural Information Processing Systems* 34.

Karras, Tero, Samuli Laine, and Timo Aila (2019). "A Style-based Generator Architecture for Generative Adversarial Networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4401–4410.

Karras, Tero, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila (2020). "Analyzing and Improving the Image Quality of StyleGAN". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8110–8119.

Kim, Hyunjik, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh (2019). *Attentive Neural Processes*. DOI: `10.48550/ARXIV.1901.05761`. URL: `https://arxiv.org/abs/1901.05761`.

King, Gary (2020). *3. Data Generation Processes*. Accessed May 7, 2022. URL: `https://www.youtube.com/watch?v=kaL1KzDTotc`.

Kingma, Diederik P. and Max Welling (2013). "Auto-Encoding Variational Bayes". In: DOI: `10.48550/ARXIV.1312.6114`. URL: `https://arxiv.org/abs/1312.6114`.

Kodali, Naveen, Jacob Abernethy, James Hays, and Zsolt Kira (2017). "On Convergence and Stability of GANs". In: *arXiv preprint arXiv:1705.07215*. DOI: `10.48550/ARXIV.1705.07215`. URL: `https://arxiv.org/abs/1705.07215`.

Kumar, Aviral and Sergey Levine (2020). "Model Inversion Networks for Model-based Optimization". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., pp. 5126–5137. URL: `https://proceedings.neurips.cc/paper/2020/file/373e4c5d8edfa8b74fd4b6791d0cf6dc-Paper.pdf`.

Kusner, Matt J., Brooks Paige, and José Miguel Hernández-Lobato (June 2017). "Grammar Variational Autoencoder". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, pp. 1945–1954. URL: `https://proceedings.mlr.press/v70/kusner17a.html`.

Larson, Jeffrey, Matt Menickelly, and Stefan M. Wild (2019). "Derivative-free Optimization Methods". In: *Acta Numerica* 28, pp. 287–404. DOI: `10.1017/S0962492919000060`.

Lee, Wonkwang, Donggyun Kim, Seunghoon Hong, and Honglak Lee (2020). "High-fidelity Synthesis with Disentangled Representation". In: *European Conference on Computer Vision*. Springer, pp. 157–174.

Li, Jerry, Aleksander Madry, John Peebles, and Ludwig Schmidt (Oct. 2018). "On the Limitations of First-Order Approximation in GAN Dynamics". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 3005–3013. URL: `https://proceedings.mlr.press/v80/li18d.html`.

Liu, Bingchen, Yizhe Zhu, Zuohui Fu, Gerard De Melo, and Ahmed Elgammal (2020). "OOGAN: Disentangling GAN with One-hot Sampling and Orthogonal Regularization". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 04, pp. 4836–4843.

Lu, Minfang, Shuai Ning, Shuangrong Liu, Fengyang Sun, Bo Yang, Bo Zhang, Junteng Zheng, and Lin Wang (2021). *OPT-GAN: Black-Box Global Optimization via Generative Adversarial Nets*. DOI: `10.48550/ARXIV.2102.03888`. URL: `https://arxiv.org/abs/2102.03888`.

Lu, Xiaoyu, Javier Gonzalez, Zhenwen Dai, and Neil Lawrence (Oct. 2018). "Structured Variationally Auto-encoded Optimization". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 3267–3275. URL: `https://proceedings.mlr.press/v80/lu18c.html`.

Luo, Renqian, Fei Tian, Tao Qin, Enhong Chen, and Tie-Yan Liu (2018). "Neural Architecture Optimization". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc. URL: `https://proceedings.neurips.cc/paper/2018/file/933670f1ac8ba969f32989c312faba75-Paper.pdf`.

Mansouri Tehrani, Aria, Anton O. Oliynyk, Marcus Parry, Zeshan Rizvi, Samantha Couper, Feng Lin, Lowell Miyagi, Taylor D. Sparks, and Jakoah Brgoch (Aug. 2018). "Machine Learning Directed Search for Ultraincompressible, Superhard Materials". In: *Journal of the American Chemical Society* 140.31, pp. 9844–9853. ISSN: 0002-7863. DOI: `10.1021/jacs.8b02717`.

Maus, Natalie, Haydn T. Jones, Juston S. Moore, Matt J. Kusner, John Bradshaw, and Jacob R. Gardner (2022). *Local Latent Space Bayesian Optimization over Structured Inputs*. DOI: `10.48550/ARXIV.2201.11872`. URL: `https://arxiv.org/abs/2201.11872`.

Mescheder, Lars, Andreas Geiger, and Sebastian Nowozin (2018). "Which Training Methods for GANs do Actually Converge?" In: *International Conference on Machine Learning*. PMLR, pp. 3481–3490.

Michelis, Mike Yan and Quentin Becker (2021). "On Linear Interpolation in the Latent Space of Deep Generative Models". In: *arXiv preprint arXiv:2105.03663*.

Mirza, Mehdi and Simon Osindero (2014). “Conditional Generative Adversarial Nets”. In: *arXiv preprint arXiv:1411.1784*.

Moriconi, Riccardo, Marc P. Deisenroth, and K. S. Sesh Kumar (2019). “High-dimensional Bayesian Optimization Using Low-dimensional Feature Spaces”. In: DOI: `10.48550/ARXIV.1902.10675`. URL: `https://arxiv.org/abs/1902.10675`.

Nguyen, Thanh Thi, Quoc Viet Hung Nguyen, Cuong M. Nguyen, Dung Nguyen, Duc Thanh Nguyen, and Saeid Nahavandi (2019). “Deep Learning for Deepfakes Creation and Detection: A Survey”. In: *arXiv preprint arXiv:1909.11573*.

Notin, Pascal, José Miguel Hernández-Lobato, and Yarin Gal (2021). “Improving Black-box Optimization in VAE Latent Space Using Decoder Uncertainty”. In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan. Vol. 34. Curran Associates, Inc., pp. 802–814. URL: `https://proceedings.neurips.cc/paper/2021/file/06fe1c234519f6812fc4c1baae25d6af-Paper.pdf`.

OpenAI (2022). *DALL-E 2*. Accessed May 8, 2022. URL: `https://openai.com/dall-e-2/`.

Ostwald, Dirk (2021). *Lecture "Statistics for Data Science"*. Accessed Oct 3, 2022. URL: `https://www.ewi-psy.fu-berlin.de/einrichtungen/arbeitsbereiche/computational_cogni_neurosc/teaching/Statistics_for_Data_Science_20_211/5_Transformations.pdf`.

Pan, Zhaoqing, Weijie Yu, Xiaokai Yi, Asifullah Khan, Feng Yuan, and Yuhui Zheng (2019). “Recent Progress on Generative Adversarial Networks (GANs): A Survey”. In: *IEEE Access* 7, pp. 36322–36333. DOI: `10.1109/ACCESS.2019.2905015`.

Park, S.M., H.G. Yoon, D.B. Lee, J.W. Choi, H.Y. Kwon, and C. Won (2022). “Optimization of Physical Quantities in the Autoencoder Latent Space”. In: *Scientific Reports* 12.1, pp. 1–9.

Pulford, Graham W. and Kirill Kondrashov (2021). “Convergence and Optimality Analysis of Low-Dimensional Generative Adversarial Networks Using Error Function Integrals”. In: *IEEE Access* 9, pp. 165366–165384. DOI: `10.1109/ACCESS.2021.3133762`.

Radford, Alec, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever (2018). “Improving Language Understanding by Generative Pre-training”. In: URL: `https://www.cs.ubc.ca/~amuham01/LING530/papers/radford2018improving.pdf`.

Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. (2019). “Language Models are Unsupervised Multitask Learners”. In: *OpenAI Blog* 1.8, p. 9.

Ramesh, Aditya, Youngduck Choi, and Yann LeCun (2018). “A Spectral Regularizer for Unsupervised Disentanglement”. In: *arXiv preprint arXiv:1812.01161*.

Ramesh, Aditya, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen (2022). "Hierarchical Text-conditional Image Generation with Clip Latents". In: *arXiv preprint arXiv:2204.06125*.

Ramesh, Aditya, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever (2021). "Zero-shot Text-to-image Generation". In: *International Conference on Machine Learning*. PMLR, pp. 8821–8831.

Rasmussen, Carl Edward (2003). "Gaussian Processes in Machine Learning". In: *Summer school on machine learning*. Springer, pp. 63–71.

Rezende, Danilo and Shakir Mohamed (2015). "Variational Inference with Normalizing Flows". In: *International conference on machine learning*. PMLR, pp. 1530–1538.

Rubinstein, Reuven (1999). *The Cross-Entropy Method for Combinatorial and Continuous Optimization*. English. DOI: 10.1023/A:1010091220143.

Shahriari, Bobak, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas (2016). "Taking the Human Out of the Loop: A Review of Bayesian Optimization". In: *Proceedings of the IEEE* 104.1, pp. 148–175. DOI: 10.1109/JPROC.2015.2494218.

Shen, Yujun, Jinjin Gu, Xiaoou Tang, and Bolei Zhou (2020). "Interpreting the Latent Space of GANs for Semantic Face Editing". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9243–9252.

Siivola, Eero, Andrei Paleyes, Javier González, and Aki Vehtari (2021). "Good Practices for Bayesian Optimization of High Dimensional Structured Spaces". In: *Applied AI Letters* 2.2, e24. DOI: https://doi.org/10.1002/ail2.24. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/ail2.24. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/ail2.24.

Snoek, Jasper, Hugo Larochelle, and Ryan P Adams (2012). "Practical Bayesian Optimization of Machine Learning Algorithms". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger. Vol. 25. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2012/file/05311655a15b75fab86956663e1819cd-Paper.pdf.

Snoek, Jasper, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams (July 2015). "Scalable Bayesian Optimization Using Deep Neural Networks". In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, pp. 2171–2180. URL: https://proceedings.mlr.press/v37/snoek15.html.

Sohl-Dickstein, Jascha, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli (July 2015). "Deep Unsupervised Learning using Nonequilibrium Thermodynamics". In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France:

PMLR, pp. 2256–2265. URL: https://proceedings.mlr.press/v37/sohl-dickstein15.html.

Stanton, Samuel, Wesley Maddox, Nate Gruver, Phillip Maffettone, Emily Delaney, Peyton Greenside, and Andrew Gordon Wilson (2022). "Accelerating Bayesian Optimization for Biological Sequence Design with Denoising Autoencoders". In: DOI: 10.48550/ARXIV.2203.12742. URL: https://arxiv.org/abs/2203.12742.

Tripp, Austin, Erik Daxberger, and José Miguel Hernández-Lobato (2020). "Sample-efficient Optimization in the Latent Space of Deep Generative Models via Weighted Retraining". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., pp. 11259–11272. URL: https://proceedings.neurips.cc/paper/2020/file/81e3225c6ad49623167a4309eb4b2e75-Paper.pdf.

Tsitsiklis, John (2018). *Lecture "Introduction to Probability"*. Accessed May 7, 2022. URL: https://ocw.mit.edu/courses/res-6-012-introduction-to-probability-spring-2018/resources/sample-space/.

Tucci, Mauro, Sami Barmada, Alessandro Formisano, and Dimitri Thomopulos (2021). "A Regularized Procedure to Generate a Deep Learning Model for Topology Optimization of Electromagnetic Devices". In: *Electronics* 10.18. ISSN: 2079-9292. DOI: 10.3390/electronics10182185. URL: https://www.mdpi.com/2079-9292/10/18/2185.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin (2017). "Attention Is All You Need". In: *Advances in Neural Information Processing Systems* 30.

Voynov, Andrey and Artem Babenko (2020). "Unsupervised Discovery of Interpretable Directions in the GAN Latent Space". In: *International Conference on Machine Learning*. PMLR, pp. 9786–9796.

Wang, Ziyu, Frank Hutter, Masrour Zoghi, David Matheson, and Nando de Feitas (2016). "Bayesian Optimization in a Billion Dimensions via Random Embeddings". In: *Journal of Artificial Intelligence Research* 55, pp. 361–387.

Wiatrak, Maciej, Stefano V. Albrecht, and Andrew Nystrom (2020). *Stabilizing GANs: A Survey Stabilizing Generative Adversarial Networks*. Tech. rep. tech. rep.

Williams, Ronald J. (May 1992). "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning". In: *Mach. Learn.* 8.3–4, pp. 229–256. ISSN: 0885-6125. DOI: 10.1007/BF00992696. URL: https://doi.org/10.1007/BF00992696.

Wu, Zongze, Dani Lischinski, and Eli Shechtman (2020). *StyleSpace Analysis: Disentangled Controls for StyleGAN Image Generation*. DOI: 10.48550/ARXIV.2011.12799. URL: https://arxiv.org/abs/2011.12799.

Zhang, Susan et al. (2022). *OPT: Open Pre-trained Transformer Language Models*. DOI: 10.48550/ARXIV.2205.01068. URL: https://arxiv.org/abs/2205.01068.

Zoph, Barret and Quoc V. Le (2016). *Neural Architecture Search with Reinforcement Learning*. DOI: 10.48550/ARXIV.1611.01578. URL: https://arxiv.org/abs/1611.01578.