Stability and Expressiveness of Deep Generative Models

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von

Lars Morten Mescheder

aus Bielefeld

Tübingen
2020

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation:	06.05.2020
Dekan:	Prof. Dr. Wolfgang Rosenstiel
1. Berichterstatter:	Prof. Dr. Andreas Geiger
2. Berichterstatter:	Prof. Dr. Philipp Hennig
3. Berichterstatter:	Dr. Ferenc Huszár

"Whenever we proceed from the known into the unknown we may hope to understand, but we may have to learn at the same time a new meaning of the word *understanding*."

Werner Karl Heisenberg

Physics and Philosophy: The Revolution in Modern Science

Abstract

In recent years, deep learning has revolutionized both machine learning and computer vision. Many classical computer vision tasks (e.g. object detection and semantic segmentation), which traditionally were very challenging, can now be solved using supervised deep learning techniques. While supervised learning is a powerful tool when labeled data is available and the task under consideration has a well-defined output, these conditions are not always satisfied. One promising approach in this case is given by generative modeling. In contrast to purely discriminative models, generative models can deal with uncertainty and learn powerful models even when labeled training data is not available. However, while current approaches to generative modeling achieve promising results, they suffer from two aspects that limit their expressiveness: (i) some of the most successful approaches to modeling image data are no longer trained using optimization algorithms, but instead employ algorithms whose dynamics are not well understood and (ii) generative models are often limited by the memory requirements of the output representation. We address both problems in this thesis: in the first part we introduce a theory which enables us to better understand the training dynamics of Generative Adversarial Networks (GANs), one of the most promising approaches to generative modeling. We tackle this problem by introducing minimal example problems of GAN training which can be understood analytically. Subsequently, we gradually increase the complexity of these examples. By doing so, we gain new insights into the training dynamics of GANs and derive new regularizers that also work well for general GANs. Our new regularizers enable us - for the first time - to train a GAN at one megapixel resolution without having to gradually increase the resolution of the training distribution. In the second part of this thesis we consider output representations in 3D for generative models and 3D reconstruction techniques. By introducing implicit representations to deep learning, we are able to extend many techniques that work in 2D to the 3D domain without sacrificing their expressiveness.

Kurzfassung

In den letzten Jahren hat Deep Learning sowohl das maschinelle Lernen als auch die maschinelle Bildverarbeitung revolutioniert. Viele klassische Computer Vision-Aufgaben, wie z.B. die Objekterkennung und semantische Segmentierung, die traditionell sehr anspruchsvoll waren, können nun mit Hilfe von überwachten Deep Learning-Techniken gelöst werden. Überwachtes Lernen ist ein mächtiges Werkzeug, wenn annotierte Daten verfügbar sind und die betrachtete Aufgabe eine eindeutige Lösung hat. Diese Bedingungen sind allerdings nicht immer erfüllt. Ein vielversprechender Ansatz ist in diesem Fall die generative Modellierung. Im Gegensatz zu rein diskriminativen Modellen können generative Modelle mit Unsicherheiten umgehen und leistungsfähige Modelle lernen, auch wenn keine annotierten Trainingsdaten verfügbar sind. Obwohl aktuelle Ansätze zur generativen Modellierung vielversprechende Ergebnisse erzielen, beeinträchtigen zwei Aspekte ihre Expressivität: (i) Einige der erfolgreichsten Ansätze zur Modellierung von Bilddaten werden nicht mehr mit Hilfe von Optimierungsalgorithmen trainiert, sondern mit Algorithmen, deren Dynamik bisher nicht gut verstanden wurde. (ii) Generative Modelle sind oft durch den Speicherbedarf der Ausgaberepräsentation begrenzt. In dieser Arbeit gehen wir auf beide Probleme ein: Im ersten Teil der Arbeit stellen wir eine Theorie vor, die es erlaubt, die Trainingsdynamik von Generative Adversarial Networks (GANs), einem der vielversprechendsten Ansätze zur generativen Modellierung, besser zu verstehen. Wir nähern uns dieser Problemstellung, indem wir minimale Beispielprobleme des GAN-Trainings vorstellen, die analytisch verstanden werden können. Anschließend erhöhen wir schrittweise die Komplexität dieser Beispiele. Dadurch gewinnen wir neue Einblicke in die Trainingsdynamik von GANs und leiten neue Regularisierer her, die auch für allgemeine GANs sehr gut funktionieren. Insbesondere ermöglichen unsere neuen Regularisierer erstmals, ein GAN mit einer Auflösung von einem Megapixel zu trainieren, ohne dass wir die Auflösung der Trainingsverteilung schrittweise erhöhen müssen. Im zweiten Teil dieser Arbeit betrachten wir Ausgaberepräsentationen für generative Modelle in 3D und für 3D-Rekonstruktionstechniken. Durch die Einführung von impliziten Repräsentationen sind wir in der Lage, viele Techniken, die in 2D funktionieren, auf den 3D-Bereich auszudehnen ohne ihre Expressivität einzuschränken.

Acknowledgements

There are many people who have supported me and without whom this work would not have been possible. First of all, I would like to thank Andreas Geiger and Sebastian Nowozin for supervising me during my studies. Supervising a student is not only about giving the right input, but also about helping the student grow and develop on a personal level. I am grateful to have found supervisors who are capable of both. I am also thankful to Philipp Hennig and Ferenc Huszár for reading and evaluating my thesis.

Moreover, I would like to thank my colleagues for creating an inspirational and fun working environment. Science can be rewarding, but what makes you get up in the morning are the people around you. In particular, I would like to thank Michael Niemeyer and Michael Oechsle, with whom I collaborated in the second half of my PhD. The time we worked together was the most rewarding for me: not only did we develop interesting scientific insights together, but also inspired each other and had fun. I would also like to thank Kevin Roth and Vaishnavh Nagarajan for their valuable input on my work presented in the first part of this thesis. My thanks also go to Hassan Abu Alhaija, Siva Karthik Mustikovela and Carsten Rother, with whom I collaborated during my PhD.

Most importantly, however, I would like to thank my wife Lena. Having a person by your side that supports you regardless of the external circumstances is the most valuable thing in the world and enables you to achieve things that you would not be able to without this support. I would also like to thank my parents, who always have an open ear when I need it. Doing a PhD can be very time-consuming and it limits the time you have for the people around you. I am extremely grateful to have people in my life who are willing to be part of this journey.

Last but not least, I would like to thank Microsoft Research and the Intel Network on Intelligent Systems for supporting me financially during my PhD. I am also thankful to the Max Planck society for creating such an amazing working environment at the Max Planck Institute for Intelligent Systems in Tübingen. I am grateful that I had the opportunity to learn and grow at this place.

Contents

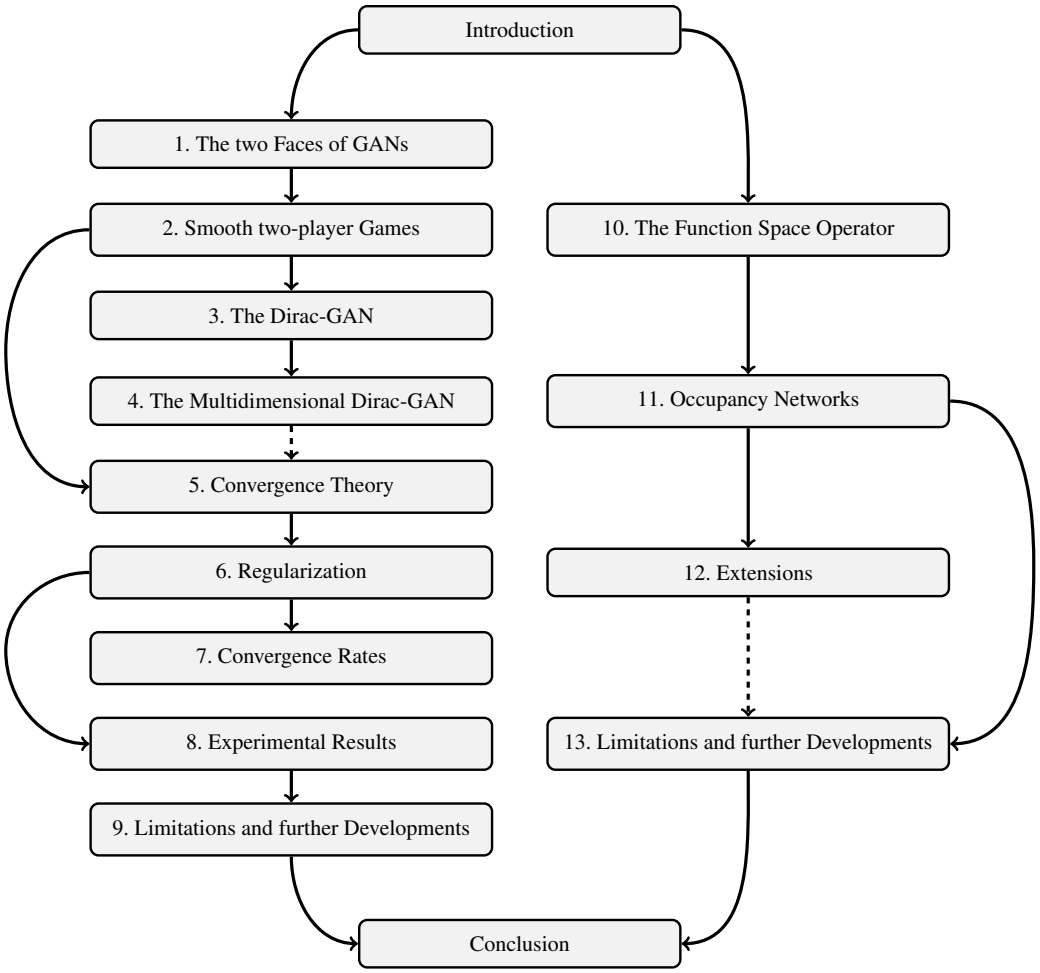


Figure 1: Dependency Graph. *Bold arrows (⟶) indicate that chapters directly build on each other. Dashed arrows (- -›) indicate that reading the first chapter is recommended, but not required, to understand the second chapter.*

List of Figures

List of Tables

Notation

Spaces

$\mathcal{X}$	Output space, e.g. images or 3D meshes. Output to generative or discriminative model.
$\mathcal{Z}$	Latent space, input to generative model.
$\mathcal{Y}$	Additional conditional information for generative model or input to discriminative model, e.g. label, single images of 3D objects, etc.
Ω_G, Ω_D	Space of trainable parameters for generator and discriminator of GAN.
$\mathcal{S}$, $\mathcal{V}$	Domain and range of function space $\mathcal{X} = \mathcal{V}^{\mathcal{S}}$ considered in Part II.
$\mathbb{N}$, $\mathbb{Z}$, $\mathbb{R}$, $\mathbb{R}_+$, $\mathbb{C}$	Natural numbers, integers, real numbers, positive real numbers and complex numbers.
$[a,b]$	Closed interval, defined by $t \in [a,b] \Leftrightarrow a \leq t \leq b$.
(a,b)	Open interval, defined by $t \in (a,b) \Leftrightarrow a < t < b$.
$[a,b)$, $(a,b]$	Half-open intervals, defined by $t \in [a,b) \Leftrightarrow a \leq t < b$ and $t \in (a,b] \Leftrightarrow a < t \leq b$
$\mathbb{R}^{n_1 \times n_2}$, $\mathbb{C}^{n_1 \times n_2}$	Space of real and complex matrices of size $n_1 \times n_2$.

Symbols

x	Element of output space $\mathcal{X}$.
z	Element of latent space $\mathcal{Z}$, often sampled from latent distribution p_0.
y	Additional conditional information in $\mathcal{Y}$ for generative model, or input to discriminative model, e.g. label, single image of 3D object, etc.
θ, ψ	Parameters of neural networks.
$G_\theta(z)$	Generator of GAN. Neural network from latent space $\mathcal{Z}$ to $\mathcal{X}$.
$D_\psi(x)$	Discriminator of GAN. Neural network from $\mathcal{X}$ to $\mathbb{R}$.
p_0	Latent distribution of GAN, probability distribution on $\mathcal{Z}$.
p_θ	Generator distribution of GAN, probability distribution on $\mathcal{X}$. Can be written as the pushforward $p_\theta = (G_\theta)_\# p_0$ of the latent distribution p_0.
φ_1, φ_2	Final activation functions defining the loss function of GAN.
f_θ	Occupancy Network. Neural network from $\mathcal{S} \times \mathcal{Z} \times \mathcal{Y}$ to $[0,1]$ with $\mathcal{S} = [0,1]^3$, that can generate and reconstruct 3D meshes.

Linear Algebra

A^T	Transposed of matrix $A \in \mathbb{C}^{n_1 \times n_2}$, defined by $(A^\mathsf{T})_{i,j} = A_{j,i}$.
$\overline{A}$	Complex conjugate of matrix $A \in \mathbb{C}^{n_1 \times n_2}$. Also defined for complex numbers (i.e. $n_1 = n_2 = 1$).
A^H	Conjugate transposed of matrix $A \in \mathbb{C}^{n_1 \times n_2}$, defined by $A^H = \overline{A}^\mathsf{T}$.
$A \cdot B$	Matrix product of $A \in \mathbb{C}^{n_1 \times n_2}$ and $B \in \mathbb{C}^{n_2 \times n_3}$. If not ambiguous, we also write AB.
A^{-1}	Matrix inverse of $A \in \mathbb{C}^{n_1 \times n_2}$.
$\operatorname{tr} A$	Trace of matrix $A \in \mathbb{C}^{n_1 \times n_2}$.
$\det A$	Determinant of matrix $A \in \mathbb{C}^{n_1 \times n_2}$.
$V^\perp$	Orthogonal complement of linear subspace $V \subseteq \mathbb{R}^n$, defined by $w \in V^\perp \Leftrightarrow \tilde{w}^\mathsf{T} \cdot w = 0$ for all $\tilde{w} \in V$.

Analysis

$\frac{\mathrm{d}f}{\mathrm{d}t}, f'(t)$	One-dimensional derivative of real valued function $f(t)$. We use $\frac{\mathrm{d}f}{\mathrm{d}t}$ instead of $f'(t)$ when we want to highlight the dependence of f on t.
$F'(\omega)$	Jacobian of function $F : \mathbb{R}^n \to \mathbb{R}^m$, which is a function $\mathbb{R}^n \to \mathbb{R}^{m \times n}$. Reduces to one-dimensional derivative if $n = m = 1$.
$\frac{\partial F}{\partial \omega_i}(\omega_1, \omega_2)$	Partial Jacobian of function $F : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \to \mathbb{R}^m$, which is a function $\mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \to \mathbb{R}^{m \times n_i}$. Reduces to one-dimensional partial derivative if $n_i = 1$.
$\nabla F(\omega)$	Gradient of function $F : \mathbb{R}^n \to \mathbb{R}$, which is a function $\mathbb{R}^n \to \mathbb{R}^n$. Equal to $\nabla F(\omega) = (F'(\omega))^\mathsf{T}$.
$\nabla_{\omega_i} F(\omega_1, \omega_2)$	Partial gradient of function $F : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \to \mathbb{R}$, which is a function $\mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \to \mathbb{R}^{n_i}$. Equal to $\nabla_{\omega_i} F(\omega_1, \omega_2) = (\frac{\partial F}{\partial \omega_i}(\omega_1, \omega_2))^\mathsf{T}$.
$\nabla^2 F(\omega)$	Hessian of function $F : \mathbb{R}^n \to \mathbb{R}$, which is a function $\mathbb{R}^n \to \mathbb{R}^{n \times n}$. Equal to $\nabla^2 F(\omega) = (\nabla F)'(\omega)$
$\nabla^2_{\omega_i, \omega_j} F(\omega_1, \omega_2)$	Partial Hessian of function $F : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \to \mathbb{R}$, which is a function $\mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \to \mathbb{R}^{n_i \times n_j}$. Equal to $\nabla^2_{\omega_i, \omega_j} F(\omega_1, \omega_2) = \frac{\partial}{\partial \omega_j} \nabla_{\omega_i} F(\omega_1, \omega_2)$.
$\partial_w F(\omega)$	Directional derivative of function $F : \mathbb{R}^n \to \mathbb{R}^m$. Defined as derivative of $t \to F(\omega + tw) - F(\omega)$ at $t = 0$. Equal to $F'(\omega)w$.
$\partial^2_w F(\omega)$	Second directional derivative of function $F : \mathbb{R}^n \to \mathbb{R}$. Defined as second derivative of $t \to F(\omega + tw) - F(\omega)$ at $t = 0$. Equal to $w^\mathsf{T} \nabla^2 F(\omega) w$.
$f^*(\xi)$	Convex conjugate of convex function $f : \mathbb{R}^n \to (-\infty, \infty]$, defined by $f^*(\xi) = \sup_{\omega \in \mathbb{R}^n} \xi^\mathsf{T} \omega - f(\omega)$.
$\operatorname{dom}(f)$	Domain of convex function, i.e. the set of all $\omega \in \mathbb{R}^n$ where $f(\omega) < \infty$.
$\mathrm{T}_{\omega^*} \mathcal{M}$	Tangent space at ω^* of $\mathcal{C}^1$-manifold $\mathcal{M} \subseteq \mathbb{R}^n$. Defined as the set of all $\gamma'(0) \in \mathbb{R}^n$ for $\mathcal{C}^1$-curves $\gamma : (-\varepsilon, \varepsilon) \to \mathcal{M}$ with $\gamma(0) = \omega^*$.

Probability Theory

$\mathcal{P}(\mathcal{X})$	Space of probability distributions on measurable space $\mathcal{X}$.
$\mathrm{E}_{x\sim p}[f(x)]$	Expectation of $f(x)$ with respect to probability distribution p.
supp p	Support of probability distribution.
$\mathcal{U}[a,b]$	Uniform distribution on closed interval $[a,b] \subseteq \mathbb{R}$.
$\phi_{\#}p$	Pushforward of probability distribution p for a mapping $\phi : \Omega_1 \to \Omega_2$. Defined by $\phi_{\#}p(A) = p(\phi^{-1}(A))$ for $A \subseteq \Omega_2$.
$\mathbb{D}(p_1 \,\|\, p_2)$	Probabilistic divergence between distributions p_1 and p_2.
$\mathrm{KL}\,(p_1 \,\|\, p_2)$	Kullback-Leibler divergence between probability distributions p_1 and p_2.

Miscellaneous

$F_2 \circ F_1$	Concatenation of mapping $F_1 : \Omega_1 \to \Omega_2$ and $F_2 : \Omega_2 \to \Omega_3$, defined by $(F_2 \circ F_1)(\omega) = F_2(F_1(\omega))$.
χ_A	Indicator function for set $A \subseteq \Omega$ defined by $\chi_A(\omega) = 1$ for $\omega \in A$ and $\chi_A(\omega) = 0$ else.
χ_0	Function that maps every ω to 0, i.e. $\chi_0(\omega) = 0$ for all ω. Equal to $\chi_\emptyset$ where $\emptyset$ denotes the empty set.
$\mathcal{B}_\varepsilon(\omega)$	Open ε-ball around $\omega \in \mathbb{R}^n$, i.e. set of points $\tilde{\omega}$ with $\|\tilde{\omega} - \omega\| < \varepsilon$.
$\lfloor r \rfloor$	Biggest integer that is less or equal to $r \in \mathbb{R}$.
$\lvert\Omega\rvert$	Cardinality of finite set Ω or n-dimensional volume of Ω if $\Omega \subseteq \mathbb{R}^n$.
$\Re(\lambda), \Im(\lambda)$	Real part and imaginary part of complex number $\lambda \in \mathbb{C}$.

Introduction

Motivation

What does it mean for a machine to understand a dataset? From the perspective of supervised learning, this question is answered by the requirement that an algorithm should generalize to unseen test cases. However, while this level of understanding is sufficient for most practical use cases of machine learning, it is unsatisfactory both from a scientific and philosophical point of view: being able to solve a repetitive task does not necessarily require any deep understanding about the data and the algorithm can find any shortcut which makes the task easier to solve. For example, a machine learning algorithm that can distinguish cats and dogs might just ignore the shape of the animal and instead solely focus on low-level details, such as the texture of fur [54].

A promising alternative approach is to require that a machine learning model should be able to generate new data. In contrast to supervised learning, this task does not allow the model to take shortcuts, since it has to model every relationship present in the data distribution. Hence, a *generative model* has to develop a much richer understanding about the data distribution. Moreover, unlike for supervised learning, (unconditional) generative models are trained on unlabeled data and are hence unsupervised. While supervised learning has been very successful for some limited tasks, unsupervised learning is often considered the holy grail of machine learning, as it is much closer to how humans learn [105].

Besides the philosophical perspective, generative models also have a number of practical applications.

First of all, many interesting real-world tasks in machine learning are ill-posed. A task is called ill-posed if it does not have a unique solution or the solution is not a continuous function of the input. This is for example the case for tasks in low-level vision like image inpainting, image denoising or image deblurring, but also for more high-level tasks like 3D reconstruction from sparse observations. While a model which was trained using supervised learning will generally predict some mean output when the problem is ill-posed, conditional generative models incorporate uncertainty and can learn to output a distribution of plausible solutions. Indeed, in recent years this approach has been successfully applied to a number of tasks such as image super-resolution [31, 106, 177], image inpainting [154, 203, 205] and image-to-image translation [27, 76, 211].

A second application of generative models is to build flexible models of a dataset that can be used for different downstream tasks. For example, it is often handy to have a low-dimensional description of 3D data [116, 130, 150, 162, 179], that can later be used to fit observations [13, 150, 179] or as an input to other tasks [44].

A third potential application of generative models is intelligent data augmentation for supervised learning [3, 4] as well as domain adaptation [50, 73, 107]. It is well-known that

supervised learning often requires millions of labeled training examples. Unfortunately, however, labeled training data is often costly or even impossible to obtain. One possible way out might be to use some fixed labeled training distribution which is augmented using generative models trained on unlabeled training data [73].

While generative modeling is a promising concept, it is a much harder problem than supervised learning. Optimization-based approaches to generative modeling are possible [37, 39, 95, 148, 164]. However, they are often limited in the complexity of the data they can generate [95, 164], are very slow [148] or put additional constraints on the neural network architecture [37, 39]. Moreover, almost all optimization-based approaches to generative modeling are based on optimizing the log-likelihood of a model which can be problematic in higher dimensions [186], as it emphasizes recall over precision. Indeed, it can be shown that replacing 99% of the generated data with random noise has only a negligible effect on the normalized log-likelihood in high dimensions [186].

Generative Adversarial Networks (GANs) [62] are a recent approach to generative modeling that rephrase the learning problem as a smooth two-player game. In contrast to most optimization-based approaches, GANs do not optimize the log-likelihood but instead focus on generating a distribution which is indistinguishable from the true data distribution. While GANs are widely viewed as one of the most promising approaches to generative modeling, they introduce a completely new style of optimization, where two neural networks are trained in an adversarial fashion. Compared to classical optimization, this new style of optimization raises many research questions, both from a practical and theoretical point of view. From a practical perspective, it is well-known that GANs are extremely challenging to train and naive optimization often does not lead to convergence. From a theoretical point of view, it would be desirable to derive a convergence theory that can answer questions about the existence and uniqueness of equilibrium points, about local and global stability of the system as well as provide convergence rates depending on the learning rates of the two networks.

In the first part of this work, we propose a theoretical approach for understanding local convergence of GANs. Starting from a very simple example of GAN training which can be understood analytically, we derive convergence criteria for GAN training by analyzing the Jacobian of a certain vector field. Our analysis also results in new regularizers for GAN training. While our focus lies on the theoretical side of GAN training, we find that our new regularizers are very effective and enable us - for the first time - to train a GAN with output resolution 1024×1024 without multiresolution training [88]. Indeed, our regularizers are now commonly used for training GANs [28, 43, 71, 81, 89, 90, 103, 111, 118, 145, 155, 185, 210], including StyleGAN [89, 90], which is widely recognized as the state-of-the-art in generative modeling.

Another important problem of generative models, but also of many discriminative models, is the dimensionality of the output. This becomes especially apparent when we try to apply generative models to the 3D domain. While generative models have recently achieved remarkable successes in generating realistic high resolution images, this success has not yet been replicated in the 3D domain. One of the main reasons is that existing representations for 3D geometry such as voxels, point clouds and meshes are either memory inefficient, suffer from discretization artifacts or cannot be efficiently inferred from data. In the second

part of this work, we describe a novel approach to 3D deep learning, which is based on directly predicting the continuous 3D Occupancy Function of an object. This approach drastically reduces the memory footprint during training, does not require any discretization and leads to representations that can be efficiently inferred from data. At inference time, we can efficiently evaluate our learned representations using fast specialized algorithms. We apply our new approach both to 3D reconstruction and generative tasks in the spatial and spatio-temporal domains, finding that it often leads to more compelling results than current state-of-the-art algorithms.

Outline

This thesis is divided into two parts, corresponding to the two topics discussed: the stability and expressiveness of deep generative models.

In Part I, we present our work on the training dynamics of GANs. In Chapter 1, we first give an introduction to GANs, both from a divergence minimization and game theoretic point of view. Both views are equally important: while the divergence minimization point of view is helpful for developing intuition and deriving new objectives, the game theoretic point of view is helpful for designing training algorithms for GANs and for developing a convergence theory. In this work we focus on the game theoretic point of view. In Chapter 2, we describe how local convergence of smooth two-player games can be understood using the Jacobian of the so-called gradient vector field and propose a simple alternative training scheme that in many cases ensures local convergence [128]. In this chapter we also lay the theoretical foundations for the convergence theory of GANs, which we develop in later chapters. In Chapter 3, we describe a simple example of GAN training, which shows that naive gradient-based optimization is not necessarily stable. This simple example is interesting, as it serves as a counterexample showing that GAN training is not always stable in the general case (unlike in the absolutely continuous case which was shown to be locally stable [136]). Moreover, our example provides some intuition of what might go wrong for more complex examples and it serves as a simple testbed for developing new training algorithms. In Chapter 4, we generalize this simple example to higher dimensions, which allows us to develop a better intuition about GAN training for high-dimensional data distributions. After having explored simple examples of GAN training, we are ready to develop a theory for general GANs. To this end, we first describe the convergence theory developed by Nagarajan and Kolter [136] for a restricted family of GANs under slightly more general conditions in Chapter 5. Unfortunately, however, one of the assumptions by Nagarajan and Kolter [136] is that the generator distribution and the true data distribution locally have the same support. This assumption is not true for our example problem from Chapter 3 and 4 and probably also not true for most real-world use cases of GANs. In Chapter 6, we therefore propose regularizers that make the training dynamics locally stable under less restrictive conditions. In Chapter 7, we present a novel analysis of the convergence behavior of GANs for finite learning rates which so far has not yet been published. Our key insight in this chapter is a new analysis of the spectrum of the Jacobian of the gradient vector field at an equilibrium point which enables us to prove eigenvalue bounds. This way,

we can prove novel convergence rates for (deterministic) GANs depending on the learning rates of the generator and discriminator. We believe that these results are interesting, as they give new insights into the conditioning of GAN systems and the role of learning rates. In Chapter 8, we apply our theory to real-world GANs. In this chapter we show that the simple regularizers that were developed in Chapter 6 stabilize training of high-resolution GANs. Indeed, to the best of our knowledge, we were the first who were able to train a GAN with an output resolution as high as 1024×1024 without resorting to additional tricks such as multiresolution training [88]. Finally, in Chapter 9, we discuss limitations of our analysis and possible future research directions.

While the stability of training algorithms is an important problem in generative modeling, generative models also suffer from another difficulty: when we want to build generative models of the real world, we usually require extremely high-dimensional outputs. For instance, when we want to build a generative model of 3D data, the dimensionality of a simple voxel representation grows cubically with the resolution of the discretization of 3D space. In Part II, we therefore focus on output representations of deep generative and discriminative models. To this end, we reinterpret the output space of a deep learning model as a function space in Chapter 10. By introducing the concept of Function Space Operator, we show how we can avoid the costly discretization of a high-dimensional space and instead learn a function with low-dimensional output. In Chapter 11, we apply this concept to generative models in the 3D domain as well as learning-based 3D reconstruction techniques. This results in a type of model we call Occupancy Network [130]. In this chapter we also discuss both training and inference for this representation. Moreover, we give further implementation details that are largely orthogonal to the method itself. Afterwards, we evaluate our new representation experimentally and compare Occupancy Networks to other 3D representations. In Chapter 12, we briefly discuss several extensions that we developed in follow-up projects [141, 145], but which are not a main part of this thesis. Finally, in Chapter 13, we discuss limitations of our representation and future research directions.

Most of the results in this work were previously published on machine learning or computer vision conferences [128, 129, 130, 141, 145]. The published papers were the result of collaborative projects. Please see Appendix G for credits. However, some chapters also include new results. This is particularly true for Chapter 4, where we extend our results for the Dirac-GAN [129] to higher dimensions, and Chapter 7, where we present a novel analysis of convergence rates for deterministic GANs in the realizable case.

Some other contributions [3, 4, 109, 126, 127, 142] of the author are not part of this thesis. Please see Appendix H for a complete list.

Part I

The Training Dynamics of GANs

“Simple experiments, simple theorems are the building blocks that help us understand more complicated systems.”

Ali Rahimi

NeurIPS 2017 Test of Time Award talk

1 The two Faces of GANs

Generative models in machine learning are models that can be trained on an unlabeled dataset and are capable of generating new data points after training is completed. As generating new content requires a good understanding of the training data at hand, such models are often regarded as a key ingredient to unsupervised learning.

In recent years, generative models have become more and more powerful. While many model classes such as PixelRNNs [148], PixelCNNs [147], real NVP [39] and Plug & Play generative networks [138] have been introduced and studied, the two most prominent ones are Variational Autoencoders (VAEs) [95, 164] and Generative Adversarial Networks (GANs) [62]. Both VAEs and GANs come with their own advantages and disadvantages: while GANs generally yield visually sharper results when applied to natural images, VAEs are attractive because they naturally yield both a generative model and an inference model.

In the first part of this thesis, we focus on GANs, as they raise many important research questions regarding their training dynamics. As a first step, we introduce GANs from a divergence minimization point of view. Afterwards, we describe a second perspective, the game theoretic point of view, which is helpful for deriving and analyzing training algorithms for GANs.

1.1 The Divergence View

1.1.1 Learning Generative Models

When learning a generative model, we would like to train a neural network $G_\theta : \mathcal{Z} \to \mathcal{X}$ with parameter vector θ that takes a latent vector $z \in \mathcal{Z}$ as input and outputs an element in a high-dimensional space $\mathcal{X}$ (e.g. an image). Given a generative model $G_\theta(\cdot)$, we can sample $z \in \mathcal{Z}$ from some prior distribution p_0 (e.g. a Gaussian distribution) to obtain a sample $G_\theta(z)$. This results in a probability distribution p_θ on $\mathcal{X}$ which we call the *generator distribution*. Assume that we are able to obtain independent samples from some *data distribution* $p_\mathcal{D}$ (e.g. an image distribution). Our goal is to adapt the parameter vector θ of $G_\theta(\cdot)$ so that $p_\theta \approx p_\mathcal{D}$.

In supervised learning, we usually train our models by minimizing a loss function between the predictions of our model and the supervision signal, e.g. a ground truth label. Ideally, we would like to do a similar thing for training generative models. To do so, we need a distance measure (loss function) between probability distributions. We call such a distance measure a *divergence* between probability distributions. Let $\mathcal{P}(\mathcal{X})$ denote the space of probability measures on some measurable space $\mathcal{X}$. Formally, we define

Divergence	$\varphi_1(t)$	$\varphi_2(t)$	$\mathcal{F}$
Indicator-divergence [128]	t	t	$D : \mathcal{X} \to \mathbb{R}$
f-divergence [143]	$f'(\exp(t))$	$f'_{rev}(\exp(t))$	$D : \mathcal{X} \to \mathbb{R}$
symmetric f-divergence	$f'(\exp(t))$	$f'(\exp(t))$	$D : \mathcal{X} \to \mathbb{R}$
Kullback-Leibler [143]	$1+t$	$\exp(t)$	$D : \mathcal{X} \to \mathbb{R}$
reverse Kullback-Leibler [143]	$\exp(t)$	$1+t$	$D : \mathcal{X} \to \mathbb{R}$
modified Jensen-Shannon [62]	$-\log(1+\exp(-t))$	$-\log(1+\exp(-t))$	$D : \mathcal{X} \to \mathbb{R}$
squared Hellinger [143]	$1-\frac{1}{2}\exp(-t/2)$	$1-\frac{1}{2}\exp(-t/2)$	$D : \mathcal{X} \to \mathbb{R}$
Wasserstein divergence [5]	t	t	$D : \mathcal{X} \to \mathbb{R}, \forall x : \|\nabla D(x)\| \leq 1$
Neural Network divergence [7]	$\varphi_1(t)$	$\varphi_2(t)$	Neural network $D_\psi : \mathcal{X} \to \mathbb{R}$

Table 1.1: Probabilistic Divergences. *Overview of different divergences between probability distributions. Here, the modified Jensen-Shannon divergence is defined as* $2\,\mathrm{JS}(p_\mathcal{D}, q_\theta) - \log(4)$.

Definition 1.1. *We call a function* $\mathbb{D} : \mathcal{P}(\mathcal{X}) \times \mathcal{P}(\mathcal{X}) \to [0,\infty]$*, such that* $\mathbb{D}(p_1 \,\|\, p_2) \geq 0$ *and* $\mathbb{D}(p_1 \,\|\, p_1) = 0$ *for all* $p_1, p_2 \in \mathcal{P}(\mathcal{X})$ *a* pseudo-divergence *on* $\mathcal{X}$*. If additionally* $\mathbb{D}(p_1 \,\|\, p_2) = 0$ *implies* $p_1 = p_2$*, we call* $\mathbb{D}$ *a* divergence*.*

Examples of divergences on $\mathcal{X} = \mathbb{R}^n$ include the Kullback-Leibler divergence and the Wasserstein divergence.

Assume that we are given some target distribution $p_\mathcal{D}$ from which we can draw i.i.d. samples and a parametric family of distributions p_θ that also allows us to draw i.i.d. samples. Our goal is to find θ^* that minimizes the divergence $\mathbb{D}(p_\theta \,\|\, p_\mathcal{D})$, i.e. we want to solve the optimization problem

$$\min_\theta \mathbb{D}(p_\theta \,\|\, p_\mathcal{D}). \tag{1.1}$$

Without further knowledge about $\mathbb{D}$, this optimization problem is not tractable. Fortunately, however, many divergences that are used in practice can be represented in the following form [5, 62, 143]:

$$\mathbb{D}(p_1 \,\|\, p_2) = \max_{D \in \mathcal{F}} \mathrm{E}_{x\sim p_1}[\varphi_1(D(x))] + \mathrm{E}_{x\sim p_2}[\varphi_2(-D(x))] \tag{1.2}$$

for some function class $\mathcal{F} \subseteq \mathcal{X} \to \mathbb{R}$ and functions $\varphi_1, \varphi_2 : \mathbb{R} \to \mathbb{R}$. Here, the function $D \in \mathcal{F}$ discriminates between samples from p_1 and those from p_2 and is therefore called *discriminator*. Together with (1.1), this leads to min-max problems of the form

$$\min_\theta \max_{D \in \mathcal{F}} \mathrm{E}_{x\sim p_\theta}[\varphi_1(D(x))] + \mathrm{E}_{x\sim p_\mathcal{D}}[\varphi_2(-D(x))] \tag{1.3}$$

In the following sections, we discuss several of these divergences. First, we take a look at the *Indicator divergence*, which is 0 if $p_1 = p_2$ and ∞ otherwise [128]. Afterwards we discuss f-divergences [143], which include the Jensen-Shannon divergence [62]. Finally, we briefly describe the Wasserstein divergence [5] and Neural Network divergences [7]. An overview of the different divergences is given in Table 1.1.

1.1.2 Indicator Divergence

The simplest probabilistic divergence is arguably given by the Indicator divergence. The Indicator divergence simply assigns 0 to a pair (p_1, p_2) of probability measures when $p_1 = p_2$ and ∞ otherwise. Formally, we have

Definition 1.2. *The Indicator divergence $\mathbb{D}_\infty$ is defined as*

$$\mathbb{D}_\infty(p_1 \,\|\, p_2) = \begin{cases} 0 & \text{if } p_1 = p_2 \\ \infty & \text{else} \end{cases} \tag{1.4}$$

Indeed, it is easy to find a dual representation as in (1.2) for the Indicator divergence:

Lemma 1.3. *The Indicator divergence can be written as*

$$\mathbb{D}_\infty(p_1 \,\|\, p_2) = \sup_{D:\mathcal{X}\to\mathbb{R}} \mathrm{E}_{x\sim p_1}[D(x)] + \mathrm{E}_{x\sim p_2}[-D(x)] \tag{1.5}$$

Here, the supremum is taken over all measurable functions $D : \mathcal{X} \to \mathbb{R}$.

Proof. If $p_1 = p_2$, then the right hand side of (1.5) is obviously 0. For $p_1 \neq p_2$, there exists a measurable set A such that $p_1(A) \neq p_2(A)$. Setting $D = \alpha\chi_A$ where χ_A is the indicator function for A, i.e. $\chi_A(x) = 1$ for $x \in A$ and $\chi_A(x) = 0$ else, and $\alpha \in \mathbb{R}$ we see that the right hand side is bigger than $\alpha(p_1(A) - p_2(A))$ for any $\alpha \in \mathbb{R}$. For $|\alpha| \to \infty$ this yields the conclusion. □

Lemma 1.3 shows that the Indicator divergence directly leads to a min-max problem as in (1.3). When we impose additional constraints on the probability distributions p_1 and p_2, Lemma 1.3 also holds when we impose restrictions on the space of representable functions $\mathcal{F}$: for example, when we assume that p_1 and p_2 are Radon-measures on a locally compact Hausdorff space $\mathcal{X}$, Lemma 1.3 still holds when we restrict $\mathcal{F}$ to the class of continuous functions $\mathcal{C}(\mathcal{X})$.[1]

1.1.3 f-Divergences

A very common family of divergences is given by f-divergences. For a convex lower semicontinuous function $f : \mathbb{R}_+ \to \mathbb{R}$ that satisfies $f(1) = 0$, we can define

$$\mathbb{D}_f(p_1 \,\|\, p_2) := \mathrm{E}_{x\sim p_2}\left[f\left(\frac{p_1(x)}{p_2(x)} \right) \right] \tag{1.6}$$

for probability densities[2] p_1, p_2. Using Jensen's inequality [78], it is easy to see that $\mathbb{D}_f$ indeed defines as pseudo-divergence. Moreover, if f is strictly convex, $\mathbb{D}_f$ defines a

[1] This is a direct consequence of the Riesz–Markov–Kakutani representation theorem [48, 82, 122] which states that the dual space the space of continuous functions with compact support $\mathcal{C}_c(\mathcal{X})$ is isomorphic to the space of Radon-measures on $\mathcal{X}$, see e.g. Rudin [170] for details.

[2] For simplicity, we restrict our discussion here to the case where p_1 and p_2 have densities and we identify the measures with the corresponding densities. For the general case, where p_1 and p_2 are arbitrary probability measures we can define $\mathbb{D}_f(p_1 \,\|\, p_2)$ using (1.11).

Divergence	$f(t)$	$f_{rev}(t)$	$f'(t)$	$f'_{rev}(t)$
Kullback-Leibler	$t\log(t)$	$-\log(t)$	$\log(t)+1$	$-\frac{1}{t}$
Pearson χ^2	$(t-1)^2$	$t\left(1-\frac{1}{t}\right)^2$	$2(t-1)$	$1-\frac{2}{t^2}$
Jensen-Shannon	$-\frac{1}{2}(1+t)\log\left(\frac{1+t}{2}\right)+\frac{1}{2}t\log(t)$	same	$\frac{1}{2}\log\left(\frac{2t}{1+t}\right)$	same
Squared Hellinger	$(\sqrt{t}-1)^2$	same	$1-\frac{1}{2\sqrt{t}}$	same

Table 1.2: f-Divergences *Examples of f-divergences. While the Jensen-Shannon and Squared Hellinger divergences are symmetric, the Kullback-Leibler and the Pearson χ^2 divergences are not.*

divergence on $\mathcal{X}$.

For every convex lower semicontinuous function f, we can define the convex conjugate (also called Fenchel-conjugate) [15]

$$f^*(\tau) = \sup_{t\in\mathrm{dom}(f)} \tau t - f(t) \tag{1.7}$$

It can be shown [15] that the convex conjugate of f is again convex lower semicontinuous and satisfies $f^{**}(t) = f(t)$ for all $t \in \mathrm{dom}(f)$.

The f-divergence is symmetric, i.e. $\mathbb{D}_f(p_1 \,\|\, p_2) = \mathbb{D}_f(p_2 \,\|\, p_1)$, if $f(t) = t\,f(\frac{1}{t})$. For the following discussion, it is helpful to define $f_{rev}(t) = t\,f(\frac{1}{t})$. We have

Lemma 1.4. *For a convex lower semicontinuous function $f : \mathbb{R}_+ \to \mathbb{R}$ with $f(1) = 0$ let $f_{rev} : \mathbb{R}_+ \to \mathbb{R}$ be defined as $f_{rev}(t) := t\,f(\frac{1}{t})$ for $t \neq 0$ and $f_{rev}(0) = \lim_{t\to 0} t\,f(\frac{1}{t})$. Then f_{rev} also defines a lower semicontinuous function with $f_{rev}(1) = 0$. Moreover, if f is continuously differentiable, we have $f'_{rev}(t) = -f^*(f'(\frac{1}{t}))$. The f-divergence corresponding to f_{rev} is given by $\mathbb{D}_{f_{rev}}(p_1 \,\|\, p_2) = \mathbb{D}_f(p_2 \,\|\, p_1)$.*

Proof. It can be shown that for f convex lower semicontinuous, the *perspective* $\tilde{f}(t,s) := sf\left(\frac{t}{s}\right)$ is also convex lower semicontinuous [15].[3] This directly implies that f_{rev} is convex lower semicontinuous.[4]

If f is differentiable, we have

$$f'_{rev}(t) = f\left(\frac{1}{t}\right) - \frac{1}{t}f'\left(\frac{1}{t}\right) \tag{1.8}$$

Using the fact [15] that $f^*(f'(t)) = t\,f'(t) - f(t)$ we hence see that

$$f'_{rev}(t) = -f^*\left(f'\left(\frac{1}{t}\right)\right) \tag{1.9}$$

[3] This statement can be proved by using the fact that a function f is convex lower semicontinuous if and only if the epigraph $\mathrm{epi}(f) := \{(t,s) \mid s \geq f(t)\}$ is closed and convex. See Boyd and Vandenberghe [15] for a formal proof.

[4] For the special case where f is twice continuously differentiable, this can also be seen by calculating $f''_{rev}(t) = \frac{1}{t^3}f''(t) \geq 0$ for $t > 0$.

Moreover, we have

$$\mathbb{D}_{f_{rev}}(p_1 \,\|\, p_2) = \mathrm{E}_{x\sim p_2}\left[\frac{p_1(x)}{p_2(x)} f\left(\frac{p_2(x)}{p_1(x)}\right)\right] = \mathrm{E}_{x\sim p_1}\left[f\left(\frac{p_2(x)}{p_1(x)}\right)\right] = \mathbb{D}_f(p_2 \,\|\, p_1) \quad (1.10)$$

□

We can use the convex conjugate to obtain a representation of an f-divergence as in (1.2). The basic idea was described by Nguyen, Wainwright, and Jordan [137] and later adapted to the GAN setting by Nowozin, Cseke, and Tomioka [143]. Here, we present a slightly modified version of this result using f_{rev}. For simplicity, we now assume that f - and therefore also f_{rev} - are continuously differentiable.

Lemma 1.5. *An f-divergence with $f : \mathbb{R}_+ \to \mathbb{R}$ continuously differentiable can be written as*

$$\mathbb{D}_f(p_1 \,\|\, p_2) = \sup_{D:\mathcal{X}\to\mathbb{R}} \mathrm{E}_{x\sim p_1}\left[f'\left(\mathrm{e}^{D(x)}\right)\right] + \mathrm{E}_{x\sim p_2}\left[f'_{rev}\left(\mathrm{e}^{-D(x)}\right)\right] \quad (1.11)$$

Proof. Following Nguyen, Wainwright, and Jordan [137] and Nowozin, Cseke, and Tomioka [143], we have

$$\begin{aligned}
\mathbb{D}_f(p_1 \,\|\, p_2) &= \mathrm{E}_{x\sim p_2}\left[f\left(\frac{p_1(x)}{p_2(x)}\right)\right] \\
&= \mathrm{E}_{x\sim p_2}\left[\sup_{\tau\in\mathrm{dom}(f^*)} \frac{p_1(x)}{p_2(x)}\tau - f^*(\tau)\right] \\
&= \sup_{T:\mathcal{X}\to\mathrm{dom}(f^*)} \mathrm{E}_{x\sim p_2}\left[\frac{p_1(x)}{p_2(x)}T(x) - f^*(T(x))\right] \\
&= \sup_{T:\mathcal{X}\to\mathrm{dom}(f^*)} \mathrm{E}_{x\sim p_1}\left[T(x)\right] + \mathrm{E}_{x\sim p_2}\left[-f^*(T(x))\right]
\end{aligned} \quad (1.12)$$

To simplify this expression we now use that f is differentiable. By exploiting that $\mathrm{dom}(f) = \mathbb{R}_+$ and the range of f' is dense in $\mathrm{dom}(f^*)$ [20], we can rewrite T as $T(x) = f'\left(\mathrm{e}^{D(x)}\right)$ with $D : \mathcal{X} \to \mathbb{R}$. Using Lemma 1.4 we obtain

$$\mathbb{D}_f(p_1 \,\|\, p_2) = \sup_{D:\mathcal{X}\to\mathbb{R}} \mathrm{E}_{x\sim p_1}\left[f'\left(\mathrm{e}^{D(x)}\right)\right] + \mathrm{E}_{x\sim p_2}\left[f'_{rev}\left(\mathrm{e}^{-D(x)}\right)\right] \quad (1.13)$$

□

For symmetric f, we directly obtain

Corollary 1.6. *For symmetric f, i.e. when $f_{rev} = f$, we have*

$$\mathbb{D}_f(p_1 \,\|\, p_2) = \sup_{D:\mathcal{X}\to\mathbb{R}} \mathrm{E}_{x\sim p_1}\left[f'\left(\mathrm{e}^{D(x)}\right)\right] + \mathrm{E}_{x\sim p_2}\left[f'\left(\mathrm{e}^{-D(x)}\right)\right] \quad (1.14)$$

Proof. This is a direct consequence of Lemma 1.5, since in this case we have $f'_{rev} = f'$. □

Examples of f-divergences are given in Table 1.2. For instance, when we set $f(t) = t\log(t)$ we obtain the Kullback-Leibler divergence and Lemma 1.5 yields

$$\mathrm{KL}\left(p_1 \,\|\, p_2\right) = \sup_{D:\mathcal{X}\to\mathbb{R}} \mathrm{E}_{x\sim p_1}\left[1 + D(x)\right] + \mathrm{E}_{x\sim p_2}\left[-\mathrm{e}^{D(x)}\right] \tag{1.15}$$

Similarly, we obtain the following expression for the Jensen-Shannon divergence that was also considered in the original GAN paper [62]: for $\varphi(t) = -\log(1+\exp(-t))$, we have

$$2\,\mathrm{JS}(p_1 \,\|\, p_2) - \log(4) = \sup_{D:\mathcal{X}\to\mathbb{R}} \mathrm{E}_{x\sim p_1}\left[\varphi(D(x))\right] + \mathrm{E}_{x\sim p_2}\left[\varphi(-D(x))\right] \tag{1.16}$$

1.1.4 Wasserstein Divergence

While f-divergences describe a rich family of probabilistic divergences, they do not take the metric of the underlying space $\mathcal{X}$ into account. However, if the underlying space $\mathcal{X}$ has a metric structure, it can be desirable to use divergences that exploit the metric of $\mathcal{X}$.

We can define a divergence [59, 85, 86, 190], called *Wasserstein divergence*[5], that takes the metric of the underlying space into account using the theory of optimal transport. Intuitively, we want to define $\mathbb{D}_W(p_1 \,\|\, p_2)$ as the minimum cost of moving the mass of p_1 to p_2. To this end, we introduce a *transport plan* $\pi \in \mathcal{P}(\mathcal{X}\times\mathcal{X})$, i.e. a probability distribution on the Cartesian product $\mathcal{X}\times\mathcal{X}$. This transport plan describes what percentage of mass is moved from one point to another. The requirement that π moves p_1 to p_2 then means that π is "mapped" to p_1 and p_2 respectively when we apply the projection mappings $\mathrm{proj}_1(x_1,x_2) = x_1$ and $\mathrm{proj}_2(x_1,x_2) = x_2$ to π. To make this formal, we introduce the pushforward $\phi_\# p$ of a probability distribution for a mapping $\phi : \Omega_1 \to \Omega_2$: for $A \subseteq \Omega_2$ measurable we define

$$(\phi_\# p)(A) := p\left(\{\omega \in \Omega_1 \mid \phi(\omega) \in A\}\right) \tag{1.17}$$

We are now ready to define the Wasserstein divergence.

Definition 1.7. *The Wasserstein divergence is defined as the minimal cost of transporting mass between p_1 and p_2, i.e.*

$$\mathbb{D}_W(p_1 \,\|\, p_2) := \inf_{\substack{\pi\in\mathcal{P}(\mathcal{X}\times\mathcal{X}) \\ (\mathrm{proj}_1)_\#\pi = p_1 \\ (\mathrm{proj}_2)_\#\pi = p_2}} \mathrm{E}_{x_1,x_2\sim\pi}\left[\|x_1 - x_2\|\right] \tag{1.18}$$

One important feature of the Wasserstein divergence is that it defines a metric for weak-convergence [191]. As a consequence, the mapping $\theta \to \mathbb{D}_W(p_\theta \,\|\, p_\mathcal{D})$ is continuous when p_θ is defined by a neural network, i.e. $p_\theta = (G_\theta)_\# p_0$ with $G_\theta : \mathcal{Z} \to \mathcal{X}$ a generative neural network and p_0 the prior distribution (e.g. a Gaussian distribution). It has been argued that this is an important property to achieve stable training of GANs [5, 6].

Again, we can find a dual representation of $\mathbb{D}_W$:

[5] The Wasserstein divergence is also called Wasserstein-distance of order one [191].

Lemma 1.8. *We have*

$$\mathbb{D}_W(p_1 \,\|\, p_2) = \sup_{\substack{D \in \mathcal{C}_1(\mathcal{X}) \\ \forall x \in \mathcal{X}: \|\nabla D(x)\| \leq 1}} \mathrm{E}_{x \sim p_1}[D(x)] + \mathrm{E}_{x \sim p_2}[-D(x)] \tag{1.19}$$

Proof. See for example Villani [191], Theorem 5.9 and Remark 6.5. □

While the Wasserstein divergence leads to a meaningful training objective for the generator when the discriminator is held exactly optimal [6], this can generally not be achieved in practice. We discuss this limitation and the arising problems in Chapter 3. Moreover, the constraint $\|\nabla D(x)\| \leq 1$ can be difficult to enforce in the context of GANs. Practitioners therefore usually resort to techniques that make this constraint approximately true [64, 134].

1.1.5 Neural Network Divergence

So far, we have considered probabilistic divergences where the function class $\mathcal{F}$ in (1.3) is extremely large, e.g. all measurable or all Lipschitz continuous functions. However, such expressive $\mathcal{F}$ cannot directly be represented using a parametric model. In practice, $\mathcal{F}$ is therefore approximated with a parametric family of functions, e.g. parameterized by a neural network $D_\psi(x)$. Similarly, we represent p_θ with a generator $G_\theta : \mathcal{Z} \to \mathcal{X}$ that maps some noise vector $z \in \mathcal{Z}$ that is sampled from some latent distribution p_0 to $G_\theta(z) \in \mathcal{X}$. Formally, p_θ is the pushforward of the measure p_0: $p_\theta = (G_\theta)_{\#} p_0$. The combination of the two neural networks $D_\psi(\cdot)$ and $G_\theta(\cdot)$ is called a Generative Adversarial Network (GAN) [62].

Of course, when minimizing the divergence with respect to this approximated family, we no longer minimize the correct divergence. However, it can be verified that taking any class of functions in (1.3) leads to a (pseudo-)divergence for appropriate choices of φ_1 and φ_2. Therefore, some authors call these divergences *Neural Network divergences* [7].

To make this concept formal, we first introduce the notion of a *valid* pair of functions:

Definition 1.9. *We call a pair (φ_1, φ_2) of functions $\varphi_1, \varphi_2 : \mathbb{R} \to \mathbb{R}$* valid*, if*

i) $\varphi_1(t) + \varphi_2(-t) \leq 0$ *for all* $t \in \mathbb{R}$ *and* $\varphi_1(0) + \varphi_2(0) = 0$

ii) $\varphi_1'(0)$ *and* $\varphi_2'(0)$ *both exist and* $\varphi_1'(0), \varphi_2'(0) \neq 0$

We call (φ_1, φ_2) strictly valid *if* (φ_1, φ_2) *is valid and* $\varphi_1(t) + \varphi_2(-t) = 0$ *implies* $t = 0$.

As it turns out, f-divergences lead to valid activation functions:

Lemma 1.10. *Let $f : \mathbb{R}_+ \to \mathbb{R}$ be convex with $f(1) = 0$ and $f_{rev}(t) = t\, f\left(\frac{1}{t}\right)$. Moreover, assume that f is twice differentiable at 1 and that $f''(1) \neq 0$. Then (φ_1, φ_2) with*

$$\varphi_1(t) := f'(\exp(t)) \quad \text{and} \quad \varphi_2(t) := f'_{rev}(\exp(t)) \tag{1.20}$$

is valid. If f is strictly convex, then (φ_1, φ_2) is strictly valid.

Proof. Let $s = \exp(t)$. Then, using (1.8),

$$\begin{aligned} \varphi_1(t) + \varphi_2(-t) &= f'(s) + f'_{rev}\left(\frac{1}{s}\right) \\ &= f'(s) + f(s) - s\,f'(s) \\ &= -(f(1) - f(s) - f'(s)(1-s)) \end{aligned} \tag{1.21}$$

The negative of the last expression is known as the *Bregman divergence* [16] of f from s to 1 and is always greater or equal to zero. Moreover, if f is strictly convex, the Bregman-divergence is zero if and only if $s = 1$, i.e. $t = 0$. □

Our next example shows that we can also define valid activations using concave functions:

Lemma 1.11. *Let $\varphi_1 = \varphi_2 = \varphi$ for a concave function $\varphi : \mathbb{R} \to \mathbb{R}$. Assume that $\varphi(0) = 0$ and that $\varphi'(0)$ exists and is non-zero. Then (φ_1, φ_2) is valid. If additionally φ is strictly concave, then (φ_1, φ_2) is strictly valid.*

Proof. For $\varphi_1 = \varphi_2 = \varphi$ with φ concave we have

$$\varphi_1(t) + \varphi_2(-t) = 2\frac{\varphi(t) + \varphi(-t)}{2} \leq 2\,\varphi\left(\frac{t + (-t)}{2}\right) = 2\,\varphi(0) = 0 \tag{1.22}$$

Moreover, the inequality is strict for $t \neq 0$ if φ is strictly concave. □

Let us take a look at two examples: $\varphi(t) = -\log(1 + \exp(-t)) + \log(2)$ satisfies the assumptions of Lemma 1.11. Indeed, this is the activation function considered in the original GAN-paper [62] and corresponds (up to a factor) to the Jensen-Shannon divergence (Table 1.2). Similarly, $\varphi(t) = -(1+t)^2$ also satisfies the assumptions of Lemma 1.11. This activation function was considered in Least-Squares-GAN [121]. In both cases, (φ, φ) therefore defines a valid pair of activation functions. Indeed, it is easy to check that (φ, φ) is even strictly valid in both cases.

We can now prove some elementary properties of valid activation functions:

Lemma 1.12. *Assume that (φ_1, φ_2) is valid. Then $\varphi_1'(0) = \varphi_2'(0)$ and, if φ_1, φ_2 are twice differentiable at zero, $\varphi_1''(0) + \varphi_2''(0) \leq 0$.*

Proof. Consider the function $\xi(t) := \varphi_1(t) + \varphi_2(-t)$. If (φ_1, φ_2) is valid, then ξ has a local maximum at $t = 0$. Using elementary analysis, this implies $\xi'(0) = 0$ and $\xi''(0) \leq 0$, hence the assertion. □

Under what condition do activation functions (φ_1, φ_2) lead to probabilistic divergences? The next lemma shows that a pair of valid activation functions always leads to a pseudo-divergence:

Lemma 1.13. *Assume that (φ_1, φ_2) is valid and let $\mathcal{F}$ denote any class of functions from $\mathcal{X}$ to $\mathbb{R}$. Assume that the indicator function χ_0 that maps every $x \in \mathcal{X}$ to 0 is in $\mathcal{F}$. Then*

$$\mathbb{D}_{\mathcal{F}}(p_1 \,\|\, p_2) := \max_{D \in \mathcal{F}} \mathrm{E}_{x \sim p_1}[\varphi_1(D(x))] + \mathrm{E}_{x \sim p_2}[\varphi_2(-D(x))] \tag{1.23}$$

defines a pseudo-divergence on $\mathcal{X}$. If additionally for every $p_1 \neq p_2$ there exists $D \in \mathcal{F}$ such that

$$\mathrm{E}_{x \sim p_1}[\varphi_1(D(x))] + \mathrm{E}_{x \sim p_2}[\varphi_2(-D(x))] > 0 \tag{1.24}$$

then $\mathbb{D}_{\mathcal{F}}$ is a divergence on $\mathcal{X}$.

Proof. Let

$$\eta(p_1, p_2, D) := \mathrm{E}_{x \sim p_1}[\varphi_1(D(x))] + \mathrm{E}_{x \sim p_2}[\varphi_2(-D(x))] \tag{1.25}$$

so that $\mathbb{D}_{\mathcal{F}}(p_1 \,\|\, p_2) = \sup_{D \in \mathcal{F}} \eta(p_1, p_2, D)$. First note that $\eta(p_1, p_2, \chi_0) = 0$ and hence $\mathbb{D}_{\mathcal{F}}(p_1 \,\|\, p_2) \geq 0$. On the other hand $\eta(p_1, p_1, D) \leq 0$ for all $D \in \mathcal{F}$ and hence $\mathbb{D}_{\mathcal{F}}(p_1 \,\|\, p_1) \leq 0$. Together, this implies $\mathbb{D}_{\mathcal{F}}(p_1 \,\|\, p_1) = 0$, showing that $\mathbb{D}_{\mathcal{F}}$ is a pseudo-divergence.

Now assume that for every $p_1 \neq p_2$ there exists $D \in \mathcal{F}$ such that

$$\mathrm{E}_{x \sim p_1}[\varphi_1(D(x))] + \mathrm{E}_{x \sim p_2}[\varphi_2(-D(x))] > 0 \tag{1.26}$$

This means that for $p_1 \neq p_2$ there is D with $\eta(p_1, p_2, D) > 0$ and hence $\mathbb{D}_{\mathcal{F}}(p_1 \,\|\, p_2) > 0$. This shows that in this case $\mathbb{D}_{\mathcal{F}}$ is a divergence. □

It should be noted that the requirement that there is D with

$$\mathrm{E}_{x \sim p_1}[\varphi_1(D(x))] + \mathrm{E}_{x \sim p_2}[\varphi_2(-D(x))] > 0 \tag{1.27}$$

for all probability distributions p_1, p_2 with $p_1 \neq p_2$ is very strong and implies that $\mathcal{F}$ is extremely expressive (cf. Arora et al. [7]). However, in practice we are usually only interested in a restricted class of probability distributions ($p_{\mathcal{D}}$ and p_θ) and it therefore suffices if this condition is true for distributions from this class, i.e. if there is D with $\mathrm{E}_{x \sim p_\theta}[\varphi_1(D(x))] + \mathrm{E}_{x \sim p_{\mathcal{D}}}[\varphi_2(-D(x))] > 0$ for $p_\theta \neq p_{\mathcal{D}}$.

1.2 The Game Theoretic View

1.2.1 Training Algorithms of GANs

While elegant, the derivation of GANs from a divergence minimization point of view in Section 1.1 has several problems. The first issue is that by approximating the discriminator with a neural network, we only optimize a lower bound to the true divergence. While Lemma 1.13 shows that under some circumstances this lower bound still defines a valid (but different) divergence, it can have very different properties from the divergence we want to optimize. Indeed, while neural networks are usually good at approximating the true divergence in lower dimensions, it becomes more and more difficult in higher dimensions [33]. The second, more severe issue is that the optimization problem in (1.3) requires us to

find the optimal discriminator for every update of the generator. Since the discriminator is itself a neural network, this is clearly intractable.

In practice, GANs are therefore usually trained using Simultaneous Gradient Descent (SimGD) or Alternating Gradient Descent (AltGD). In SimGD we perform simultaneous gradient updates on the parameters of the generator and discriminator, respectively. Similarly, in AltGD we perform the gradient updates in an alternating fashion. Both algorithms can hence be understood as training the two networks, $G_\theta(\cdot)$ and $D_\psi(\cdot)$, at the same time.

Unfortunately, however, by doing so we no longer solve an optimization problem and the resulting algorithms can have very different properties from gradient-based algorithms in optimization. Indeed, by applying SimGD and AltGD we are implicitly trying to find a Nash equilibrium of the two-player zero-sum game where the first player (the generator $G_\theta(\cdot)$) tries to minimize

$$\mathcal{L}(\theta,\psi) := \mathrm{E}_{x\sim p_\theta}\left[\varphi_1(D_\psi(x))\right] + \mathrm{E}_{x\sim p_\mathcal{D}}\left[\varphi_2(-D_\psi(x))\right] \tag{1.28}$$

while the second player (the discriminator $D_\psi(\cdot)$) tries to maximize it. This directly raises multiple question: (i) Does a GAN always have a (pure) Nash equilibrium? (ii) Does this equilibrium solve the original problem (1.3) and, finally, (iii) do SimGD and/or AltGD converge to this equilibrium point?

In the next section we will see that the answer to the first two questions (under some assumptions) is affirmative. However, the third question is much more difficult to answer and we will dedicate most of Part I to an answer.

1.2.2 Equilibria

Let us consider the two-player zero-sum game where the two players try to minimize and maximize

$$\mathcal{L}(\theta,\psi) := \mathrm{E}_{x\sim p_\theta}\left[\varphi_1(D_\psi(x))\right] + \mathrm{E}_{x\sim p_\mathcal{D}}\left[\varphi_2(-D_\psi(x))\right] \tag{1.29}$$

A (pure) *Nash equilibrium* is a pair (θ^*,ψ^*) that satisfies

$$\inf_\theta \mathcal{L}(\theta,\psi^*) = \mathcal{L}(\theta^*,\psi^*) = \sup_\psi \mathcal{L}(\theta^*,\psi) \tag{1.30}$$

For general two-player games, there can be zero, one or multiple pure Nash-equilibria. In this section we consider the special case of the game defined by the objective in (1.29). Let us assume that there is θ^* such that $p_{\theta^*} = p_\mathcal{D}$ and ψ^* such that $D_{\psi^*}(x) = 0$ for all $x \in \mathcal{X}$. Under what conditions does (θ^*,ψ^*) define a (pure) Nash equilibrium? If (θ^*,ψ^*) defines a Nash equilibrium, under what conditions is it the unique Nash equilibrium?

The following Lemma provides an answer to first question:

Lemma 1.14. *Assume that (φ_1,φ_2) is valid. Moreover, assume that there is (θ^*,ψ^*) such that $p_{\theta^*} = p_\mathcal{D}$ and $D_{\psi^*} = \chi_0$. Then (θ^*,ψ^*) is a (pure) Nash equilibrium for the zero-sum game where the ψ-player tries to maximize $\mathcal{L}(\theta,\psi)$ while the θ-player tries to minimize it.*

Proof. To show that (θ^*, ψ^*) is a pure Nash equilibrium, we have to show that

$$\sup_{\psi} \mathcal{L}(\theta^*, \psi) = \mathcal{L}(\theta^*, \psi^*) = \inf_{\theta} \mathcal{L}(\theta, \psi^*) \tag{1.31}$$

However, we have for every θ

$$\mathcal{L}(\theta, \psi^*) = \mathrm{E}_{x\sim p_\theta}[\varphi_1(0)] + \mathrm{E}_{x\sim p_\mathcal{D}}[\varphi_2(0)] = \varphi_1(0) + \varphi_2(0) = 0 \tag{1.32}$$

an hence,

$$\mathcal{L}(\theta^*, \psi^*) = 0 = \inf_{\theta} \mathcal{L}(\theta, \psi^*) \tag{1.33}$$

Similarly, by Definition 1.9,

$$\begin{aligned} \mathcal{L}(\theta^*, \psi) &= \mathrm{E}_{x\sim p_\mathcal{D}}\left[\varphi_1(D_\psi(x))\right] + \mathrm{E}_{x\sim p_\mathcal{D}}\left[\varphi_2(-D_\psi(x))\right] \\ &= \mathrm{E}_{x\sim p_\mathcal{D}}\left[\varphi_1(D_\psi(x)) + \varphi_2(-D_\psi(x))\right] \leq 0 \end{aligned} \tag{1.34}$$

and hence, together with (1.32),

$$\mathcal{L}(\theta^*, \psi^*) = 0 = \sup_{\psi} \mathcal{L}(\theta^*, \psi) \tag{1.35}$$

This shows that (θ^*, ψ^*) is a pure Nash equilibrium. □

So far, we have shown that (θ^*, ψ^*) with $p_{\theta^*} = p_\mathcal{D}$ and $D_{\psi^*} = \chi_0$ defines a Nash equilibrium. However, the game might also have other pure Nash-equilibria. The following lemma provides insight to what extent these equilibria can occur.

Lemma 1.15. *Assume that (φ_1, φ_2) is valid. Moreover, assume that there is (θ^*, ψ^*) with $p_{\theta^*} = p_\mathcal{D}$, $D_{\psi^*}(x) = 0$ for $x \in \operatorname{supp} p_\mathcal{D}$ and*

$$\mathrm{E}_{x\sim p_\mathcal{D}}\left[\nabla_\psi D_\psi(x)\big|_{\psi=\psi^*}\right] \neq \mathrm{E}_{x\sim p_\theta}\left[\nabla_\psi D_\psi(x)\big|_{\psi=\psi^*}\right] \tag{1.36}$$

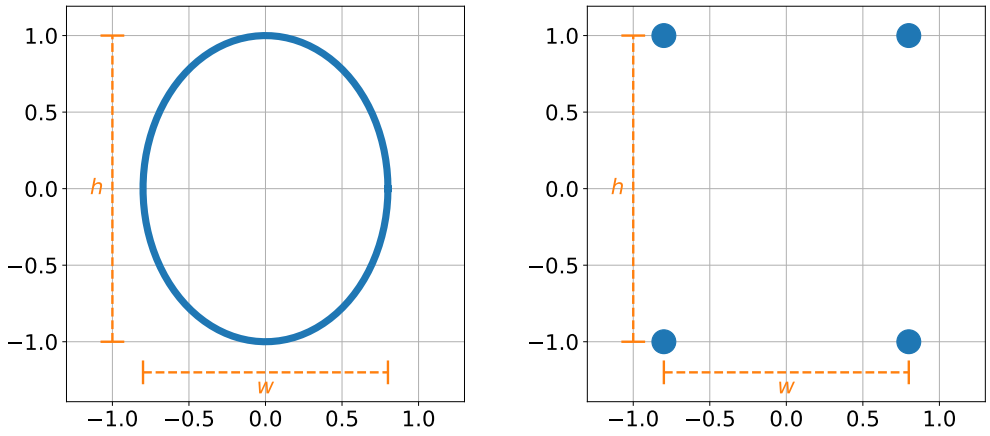
for every θ with $p_\theta \neq p_\mathcal{D}$. Then every pure Nash equilibrium (θ_{NE}, ψ_{NE}) satisfies $p_{\theta_{NE}} = p_\mathcal{D}$. If we further assume that (φ_1, φ_2) is strictly valid, we have $D_{\psi_{NE}}(x) = 0$ for almost all $x \in \operatorname{supp} p_\mathcal{D}$.

Proof. Assume that $p_{\theta_{NE}} \neq p_\mathcal{D}$. As before, let $\mathcal{L}(\theta, \psi)$ be defined as in (1.29). Using (1.32), we see that $\mathcal{L}(\theta_{NE}, \psi^*) = 0$. Moreover, since $\varphi_1'(0) = \varphi_2'(0) \neq 0$ by Definition 1.9 and Lemma 1.12, $\nabla_\psi \mathcal{L}(\theta_{NE}, \psi^*)$ is

$$\mathrm{E}_{x\sim p_{\theta_{NE}}}\left[\varphi_1'(0)\,\nabla_\psi D_\psi(x)\big|_{\psi=\psi^*}\right] - \mathrm{E}_{x\sim p_\mathcal{D}}\left[\varphi_2'(0)\,\nabla_\psi D_\psi(x)\big|_{\psi=\psi^*}\right] \neq 0 \tag{1.37}$$

This implies that there is $\tilde{\psi}$ in a neighborhood of ψ^* with $\mathcal{L}(\theta_{NE}, \tilde{\psi}) > 0$. Since, (θ_{NE}, ψ_{NE}) is a Nash equilibrium this shows that

$$\mathcal{L}(\theta_{NE}, \psi_{NE}) = \sup_{\psi} \mathcal{L}(\theta_{NE}, \psi) \geq \mathcal{L}(\theta_{NE}, \tilde{\psi}) > 0 \tag{1.38}$$



(a) Ellipse *(b) Four Points*

Figure 1.1: Example Distributions. *This figure visualizes the two example data distributions $p_{\mathcal{D}}$ that we consider in this chapter. In both cases, the discriminator $D_\psi(x)$ is given by a quadratic function.*

On the other hand, by (1.34) we have $\mathcal{L}(\theta^*, \psi) \leq 0$ for all ψ and hence

$$\mathcal{L}(\theta^*, \psi_{NE}) \leq 0 < \mathcal{L}(\theta_{NE}, \psi_{NE}) = \inf_\theta \mathcal{L}(\theta, \psi_{NE}) \tag{1.39}$$

which is a contradiction. This shows that $p_{\theta_{NE}} = p_{\mathcal{D}}$.

Now assume that (φ_1, φ_2) is strictly valid, i.e. $\varphi_1(t) + \varphi_2(-t) < 0$ for $t \neq 0$. We have already seen that $p_{\theta_{NE}} = p_{\mathcal{D}}$. Therefore,

$$\mathcal{L}(\theta_{NE}, \psi_{NE}) := \mathrm{E}_{x \sim p_{\mathcal{D}}} \left[\varphi_1(D_{\psi_{NE}}(x)) + \varphi_2(-D_{\psi_{NE}}(x)) \right] \leq 0 \tag{1.40}$$

We also see that $\mathcal{L}(\theta_{NE}, \psi_{NE}) < 0$ if not $D_{\psi_{NE}}(x) = 0$ for almost all $x \in \operatorname{supp} p_{\mathcal{D}}$. In this case, using (1.32),

$$\mathcal{L}(\theta_{NE}, \psi^*) = 0 > \mathcal{L}(\theta_{NE}, \psi_{NE}) = \sup_\psi \mathcal{L}(\theta_{NE}, \psi) \tag{1.41}$$

which is a contradiction. We hence have $D_{\psi_{NE}}(x) = 0$ for almost all $x \in \operatorname{supp} p_{\mathcal{D}}$, which shows the assertion. □

It should be noted that Lemma 1.15 only asserts that $D_{\psi_{NE}}(x) = 0$ on $\operatorname{supp} p_{\mathcal{D}}$. However, $D_\psi(x) = 0$ for $x \in \operatorname{supp} p_{\mathcal{D}}$ and $p_\theta = p_{\mathcal{D}}$ does not necessarily imply that (θ, ψ) is a Nash equilibrium if $\operatorname{supp} p_{\mathcal{D}} \neq \mathcal{X}$, since $D_\psi(\cdot)$ might have non-zero gradients orthogonal to $\operatorname{supp} p_{\mathcal{D}}$. We will have more to say on this case in Chapter 3.

Let us consider two examples, visualized in Figure 1.1:

Example (Ellipse): Assume that p_0 is given by a uniform distribution on $[0, 2\pi]$, i.e. $p_0 = \mathcal{U}[0, 2\pi]$, and $p_{\mathcal{D}}$ is a uniform distribution on an ellipse with width w and height h in the two-dimensional plane (Figure 1.1a). Moreover, let

$$G_\theta(z) = \begin{pmatrix} \theta_1 \cos(z) \\ \theta_2 \sin(z) \end{pmatrix} \quad \text{and} \quad D_\psi(x) = \psi_1 x_1^2 + \psi_2 x_2^2 \tag{1.42}$$

We see that for $\theta_1^* = \frac{w}{2}$ and $\theta_2^* = \frac{h}{2}$, we have $p_{\theta^*} = p_{\mathcal{D}}$ and for $\psi^* = 0$ we have $D_{\psi^*} = \chi_0$. For a valid pair of activation functions (φ_1, φ_2), the assumptions of Lemma 1.14 are hence satisfied. This shows that (θ^*, ψ^*) defines a Nash equilibrium.

Is (θ^*, ψ^*) the unique Nash equilibrium? We have

$$\nabla_\psi D_\psi(x) = \begin{pmatrix} x_1^2 \\ x_2^2 \end{pmatrix} \tag{1.43}$$

Therefore,

$$\mathrm{E}_{x \sim p_\theta} \left[\nabla_\psi D_\psi(x) \right] = \mathrm{E}_{z \sim \mathcal{U}[0,2\pi]} \left[\begin{pmatrix} \theta_1^2 \cos(z)^2 \\ \theta_2^2 \sin(z)^2 \end{pmatrix} \right] = \frac{1}{2} \begin{pmatrix} \theta_1^2 \\ \theta_2^2 \end{pmatrix} \tag{1.44}$$

and similarly,

$$\mathrm{E}_{x \sim p_{\mathcal{D}}} \left[\nabla_\psi D_\psi(x) \right] = \frac{1}{2} \begin{pmatrix} \theta_1^{*2} \\ \theta_2^{*2} \end{pmatrix} \tag{1.45}$$

Hence, for $p_\theta \neq p_{\mathcal{D}}$

$$\mathrm{E}_{x \sim p_{\mathcal{D}}} \left[\nabla_\psi D_\psi(x) \big|_{\psi = \psi^*} \right] \neq \mathrm{E}_{x \sim p_\theta} \left[\nabla_\psi D_\psi(x) \big|_{\psi = \psi^*} \right] \tag{1.46}$$

Lemma 1.15 is hence applicable, showing that every Nash equilibrium (θ_{NE}, ψ_{NE}) satisfies $p_{\theta_{NE}} = p_{\mathcal{D}}$ and, if (φ_1, φ_2) is strictly valid, $D_{\psi_{NE}}(x) = 0$ for $x \in \operatorname{supp} p_{\mathcal{D}}$, which implies $D_{\psi_{NE}} = \chi_0$.

Example (Four Points): We again consider the discriminator

$$D_\psi(x) = \psi_1 x_1^2 + \psi_2 x_2^2 \tag{1.47}$$

but define the generator by the discrete distribution

$$p_\theta = \frac{1}{4} (\delta_{(-\theta_1, -\theta_2)} + \delta_{(-\theta_1, \theta_2)} + \delta_{(\theta_1, -\theta_2)} + \delta_{(\theta_1, \theta_2)}) \tag{1.48}$$

Moreover, let $p_{\mathcal{D}} = p_{\theta^*}$ for some $\theta^* = (\frac{w}{2}, \frac{h}{2})^\mathsf{T} \in \mathbb{R}^2$ (Figure 1.1b). As before, for $\psi^* = 0$ we have $D_{\psi^*} = \chi_0$. For a valid pair of activation functions (φ_1, φ_2), the assumptions of Lemma 1.14 are hence again satisfied, showing that (θ^*, ψ^*) defines a Nash equilibrium. Moreover,

$$\mathrm{E}_{x \sim p_\theta} \left[\nabla_\psi D_\psi(x) \right] = \begin{pmatrix} \theta_1^2 \\ \theta_2^2 \end{pmatrix} \tag{1.49}$$

Hence, for $p_\theta \neq p_\mathcal{D}$

$$\mathrm{E}_{x\sim p_\mathcal{D}}\left[\nabla_\psi D_\psi(x)\big|_{\psi=\psi^*}\right] \neq \mathrm{E}_{x\sim p_\theta}\left[\nabla_\psi D_\psi(x)\big|_{\psi=\psi^*}\right] \tag{1.50}$$

Lemma 1.15 is hence again applicable, showing that every Nash equilibrium (θ_{NE}, ψ_{NE}) satisfies $p_{\theta_{NE}} = p_\mathcal{D}$. Moreover, for strictly valid (φ_1, φ_2), $D_{\psi_{NE}}(x) = 0$ for $x \in \operatorname{supp} p_\mathcal{D}$, which implies $D_{\psi_{NE}} = \chi_0$.

The remaining question is if SimGD or AltGD converge to the Nash equilibrium in these two examples. Surprisingly, while the two examples are formally very similar, they behave differently in terms of convergence. We will revisit these two examples in Chapter 5 where we derive a convergence theory for GAN training.

1.3 Conclusion

We have seen that GANs can be analyzed from two different points of view: the divergence and the game theoretic perspective. Interestingly, both points of view are complementary and lead to different theoretic insights: while the divergence point of view is useful for deriving novel training objectives, the game theoretic point of view allows to analyze the training dynamics in a rigorous way. Adopting the later point of view, we have derived theoretic criteria for the existence and uniqueness of Nash equilibria for GANs, some of which were already stated informally by Goodfellow et al. [62]. While these results are insightful, they are not sufficient to ensure stability of our training algorithms. In the next chapters, we therefore analyze the training dynamics further to obtain a holistic picture of the convergence of GAN training.

2 Smooth two-player Games

In Chapter 1 we have seen how Generative Adversarial Networks (GANs) can be understood both from a divergence minimization and game theoretical point of view. While the first perspective is helpful for deriving novel training objectives, the second perspective is useful for developing a convergence theory of GAN training. Since we are interested in gaining a better understanding of the training dynamics of GANs, we focus on the game theoretical perspective.

In this chapter we first analyze local convergence in general smooth two-player games. By restricting our attention to *local* instead of global convergence, we can linearize the non-linear system and therefore use tools from linear algebra to derive convergence criteria and convergence rates. Our theory can be regarded as an extension of the convergence theory for Simultaneous Gradient Descent (SimGD) in continuous games by Ratliff, Burden, and Sastry [163] and Nowozin, Cseke, and Tomioka [143]. In particular, we extend the theory by Ratliff, Burden, and Sastry [163] to Alternating Gradient Descent (AltGD) and analyze the role of eigenvalues with non-zero imaginary part. This also leads to interesting results regarding the convergence rate of SimGD. Finally, we propose Consensus Optimization [128], an alternative optimization algorithm that has better convergence properties than SimGD and AltGD and apply it to simple GAN problems.

In this thesis we focus on the *deterministic* training dynamics of GANs. While in practice GANs are usually trained with stochastic algorithms, we believe that a thorough understanding of the deterministic case is an important first step towards a complete theory. To extend our results to the stochastic case, we would have to use tools from stochastic approximation theory [167, 178] which requires more technical assumptions. In contrast, by focusing on the deterministic case, we can derive a clean theory which emphasizes structural insights over the technical details.[1]

2.1 Definitions

A differentiable two-player game is defined by two cost functions $\mathcal{L}_1(\theta, \psi)$ and $\mathcal{L}_2(\theta, \psi)$ defined over a common space $(\theta, \psi) \in \Omega_1 \times \Omega_2$. Here, $\Omega_1 \subseteq \mathbb{R}^n$ corresponds to the possible actions of the first player and $\Omega_2 \subseteq \mathbb{R}^m$ corresponds to the possible actions of the second player. The goal of the first player is to minimize $\mathcal{L}_1$, while the second player tries to minimize $\mathcal{L}_2$. In the context of GANs, Ω_1 is the set of possible parameter values for the

[1] It is important to note that many convergence theorems from stochastic approximation theory are based on linear convergence of the corresponding deterministic algorithm [14]. Many of our results can hence be generalized to the stochastic case when we use some additional technical assumptions (e.g. uniformly bounded noise). For completeness, we provide a brief discussion of some of these results in Appendix B.2.

generator and Ω_2 is the set of possible parameter values for the discriminator. In this case, we denote Ω_1 and Ω_2 by Ω_G and Ω_D, respectively. We call a game a zero-sum game if $\mathcal{L}_2 = -\mathcal{L}_1$. In this case, we define $\mathcal{L}(\theta, \psi) := \mathcal{L}_1(\theta, \psi) = -\mathcal{L}_2(\theta, \psi)$. Note that the derivation of the GAN-game in Chapter 1 leads to a zero-sum game. However, in practice people often employ a non-saturating objective for the generator [62], which leads to a game that is not zero-sum. While we focus on the zero-sum case here for a simpler exposition, our convergence theory can also be extended to a non-saturating objective, see Section 6.6.1 for details.

Our goal is to find a Nash equilibrium of the game, i.e. a point (θ^*, ψ^*) given by the two conditions

$$\theta^* \in \operatorname*{argmin}_{\theta} \mathcal{L}_1(\theta, \psi^*) \quad \text{and} \quad \psi^* \in \operatorname*{argmin}_{\psi} \mathcal{L}_2(\theta^*, \psi) \tag{2.1}$$

We call a point (θ^*, ψ^*) a local Nash equilibrium, if (2.1) holds in a local neighborhood of (θ^*, ψ^*).

Every differentiable two-player game defines a vector field

$$v(\theta, \psi) = \begin{pmatrix} -\nabla_\theta \mathcal{L}_1(\theta, \psi) \\ -\nabla_\psi \mathcal{L}_2(\theta, \psi) \end{pmatrix} \tag{2.2}$$

We call $v(\cdot)$ the *associated gradient vector field*.

Our goal is to find a Nash equilibrium of the two-player game. To this end, we first derive a simple characterization of local Nash-equilibria, which was also considered by Ratliff, Burden, and Sastry [163]. Recall that a stationary point of a vector field is a point where the vector field vanishes, i.e. a point (θ^*, ψ^*) where $v(\theta^*, \psi^*) = 0$.

Lemma 2.1. *If (θ^*, ψ^*) is a local Nash equilibrium, then (θ^*, ψ^*) is a stationary point of $v(\cdot)$, i.e. $v(\theta^*, \psi^*) = 0$, and $\nabla^2_\theta \mathcal{L}_1(\theta^*, \psi^*)$, $\nabla^2_\psi \mathcal{L}_2(\theta^*, \psi^*)$ are both positive semi-definite. Conversely, if (θ^*, ψ^*) is a stationary point of $v(\cdot)$ and $\nabla^2_\theta \mathcal{L}_1(\theta^*, \psi^*)$, $\nabla^2_\psi \mathcal{L}_2(\theta^*, \psi^*)$ are both positive definite, then (θ^*, ψ^*) is a local Nash equilibrium.*

Proof. By definition, (θ^*, ψ^*) is a local Nash equilibrium if and only if (2.1) holds in some neighborhood of (θ^*, ψ^*). Using basic results from analysis, any such point (θ^*, ψ^*) satisfies $\nabla_\theta \mathcal{L}_1(\theta^*, \psi^*) = 0$, $\nabla_\psi \mathcal{L}_2(\theta^*, \psi^*) = 0$ with $\nabla^2_\theta \mathcal{L}_1(\theta^*, \psi^*)$ and $\nabla^2_\psi \mathcal{L}_2(\theta^*, \psi^*)$ positive semi-definite.

Conversely, if (θ^*, ψ^*) is a stationary point of $v(\cdot)$ with $\nabla^2_\theta \mathcal{L}_1(\theta^*, \psi^*)$ and $\nabla^2_\psi \mathcal{L}_2(\theta^*, \psi^*)$ positive definite, then (θ^*, ψ^*) satisfies (2.1) and is therefore a local Nash equilibrium. □

Note that Lemma 2.1 does not show that (θ^*, ψ^*) is a Nash equilibrium if one of $\nabla^2_\theta \mathcal{L}_1(\theta^*, \psi^*)$ or $\nabla^2_\psi \mathcal{L}_2(\theta^*, \psi^*)$ is only positive semi-definite instead of positive definite. Indeed, we will see in the next chapters that this is a typical situation for GAN training.

For the special case of zero-sum two-player games, we have $\mathcal{L}(\theta, \psi) = \mathcal{L}_1(\theta, \psi) = -\mathcal{L}_2(\theta, \psi)$ and thus

$$v'(\theta, \psi) = \begin{pmatrix} -\nabla^2_\theta \mathcal{L}(\theta, \psi) & -\nabla^2_{\psi,\theta} \mathcal{L}(\theta, \psi)^\mathsf{T} \\ \nabla^2_{\psi,\theta} \mathcal{L}(\theta, \psi) & \nabla^2_\psi \mathcal{L}(\theta, \psi) \end{pmatrix} \tag{2.3}$$

Algorithm 1 Simultaneous Gradient Descent (SimGD)

while not converged **do**
  $\delta_\theta \leftarrow -\nabla_\theta \mathcal{L}_1(\theta, \psi)$
  $\delta_\psi \leftarrow -\nabla_\psi \mathcal{L}_2(\theta, \psi)$
  $\theta \leftarrow \theta + h_1 \delta_\theta$
  $\psi \leftarrow \psi + h_2 \delta_\psi$
end while

Algorithm 2 Alternating Gradient Descent (AltGD)

while not converged **do**
  $\theta \leftarrow \theta - h_1 \nabla_\theta \mathcal{L}_1(\theta, \psi)$
  $\psi \leftarrow \psi - h_2 \nabla_\psi \mathcal{L}_2(\theta, \psi)$
end while

Hence,

Corollary 2.2. *If (θ^*, ψ^*) is a local Nash equilibrium of a zero-sum game, then (θ^*, ψ^*) is a stationary point of $v(\cdot)$. Moreover, $\nabla^2_\theta \mathcal{L}(\theta^*, \psi^*)$ is positive semi-definite and $\nabla^2_\psi \mathcal{L}(\theta^*, \psi^*)$ is negative semi-definite. Conversely, if (θ^*, ψ^*) is a stationary point of $v(\cdot)$, $\nabla^2_\theta \mathcal{L}(\theta^*, \psi^*)$ is positive definite and $\nabla^2_\psi \mathcal{L}_2(\theta^*, \psi^*)$ is negative definite, then (θ^*, ψ^*) is a local Nash equilibrium.*

Proof. This follows directly from Lemma 2.1, using $\mathcal{L} = \mathcal{L}_1 = -\mathcal{L}_2$. □

Example (Bilinear game) Consider the two-player zero-sum game defined by $\mathcal{L}(\theta, \psi) = \theta \cdot \psi$ with $\theta, \psi \in \mathbb{R}$. In this case, $v(\theta, \psi) = (-\psi, \theta)^\mathsf{T}$. It is easy to see that the unique Nash equilibrium of this game is at $(\theta^*, \psi^*) = (0, 0)$. The Jacobian $v'(\theta^*, \psi^*)$ is given by

$$v'(\theta^*, \psi^*) = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \tag{2.4}$$

Hence, $\nabla^2_\theta \mathcal{L}(\theta^*, \psi^*) = 0$ is positive semi-definite, but not positive definite. Similarly, $\nabla^2_\psi \mathcal{L}(\theta^*, \psi^*) = 0$ is negative semi-definite, but not negative definite.

This example was also described by Salimans et al. [173] as an example of a two-player game where SimGD does not converge. Interestingly, we will see in Chapter 3 that a variant of this game also occurs for GAN training.

2.2 Algorithms

A natural algorithm for finding Nash-equilibria of general smooth two-player games is Simultaneous Gradient Descent (SimGD), which we briefly discussed in the context of zero-sum games in Chapter 1. SimGD for games was described in several works, for example by Ratliff, Burden, and Sastry [163] and, more recently also in the context of GANs,

by Nowozin, Cseke, and Tomioka [143]. The idea is simple and is illustrated in Algorithm 1. We iteratively update the parameters of the two players by simultaneously applying gradient descent to the cost functions of the two players. This can also be understood as applying the Euler-method to the ordinary differential equation

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix}\theta(t)\\ \psi(t)\end{pmatrix} = v(\theta(t),\psi(t)) \tag{2.5}$$

where $v(\cdot)$ is the associated gradient vector field of the two-player game.

An alternative is given by Alternating Gradient Descent (AltGD), described in Algorithm 2. AltGD was first mentioned in the context of GANs by Goodfellow et al. [62]. The idea is similar to SimGD, but instead of performing simultaneous gradient updates on the parameters of the two players, we update the parameters in an alternating fashion.

Both algorithms can be described as a fixed point algorithm that applies an *update operator* $F(\theta,\psi)$ to the parameter values (θ,ψ) of the two players [129]. For example, SimGD corresponds to the update operator

$$F(\theta,\psi) = \begin{pmatrix}\theta - h_1\nabla_\theta\mathcal{L}(\theta,\psi)\\ \psi - h_2\nabla_\psi\mathcal{L}(\theta,\psi)\end{pmatrix} \tag{2.6}$$

Similarly, AltGD can be described by an operator $F = F_2 \circ F_1$ where F_1 and F_2 perform an update for the first and second player, respectively:

$$F_1(\theta,\psi) = \begin{pmatrix}\theta - h_1\nabla_\theta\mathcal{L}_1(\theta,\psi)\\ \psi\end{pmatrix} \qquad F_2(\theta,\psi) = \begin{pmatrix}\theta\\ \psi - h_2\nabla_\psi\mathcal{L}_2(\theta,\psi)\end{pmatrix} \tag{2.7}$$

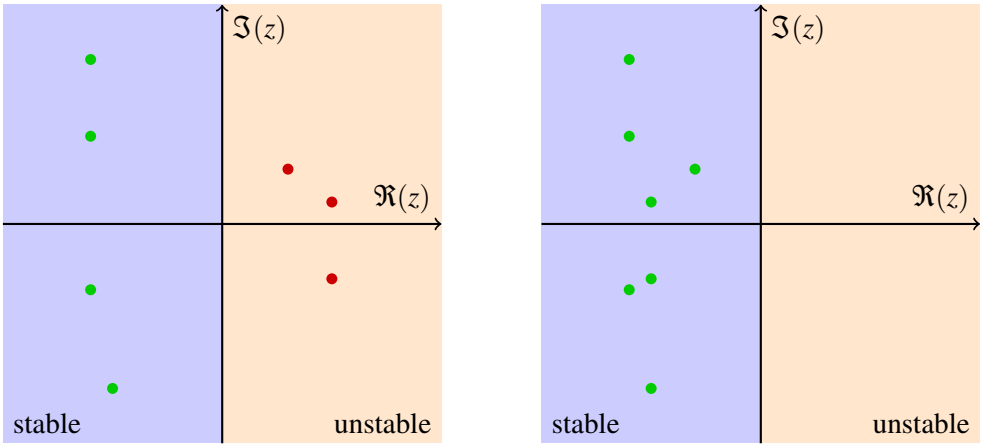
To understand convergence of these algorithms, we hence have to understand the dynamics of sequences of the form

$$(\theta,\psi),\ F(\theta,\psi),\ F(F(\theta,\psi)),\ F(F(F(\theta,\psi))),\ \ldots \tag{2.8}$$

We say that update operator $F : \Omega_1 \times \Omega_2 \to \Omega_1 \times \Omega_2$ defines a *discrete dynamical system*. In the next section we analyze local convergence of such systems near equilibrium points when F is a $\mathcal{C}^1$-mapping.

2.3 Convergence

In this section we apply results from the theory of discrete dynamical systems (Appendix B) to understand the convergence properties of SimGD and AltGD. As we state formally in Theorem B.2, linear convergence of F near a fixed point (θ^*,ψ^*) is entirely determined by the eigenvalues of the Jacobian $F'(\theta^*,\psi^*)$ at that equilibrium point: if all eigenvalues have absolute value smaller than one, then the fixed point iteration (2.8) converges with linear rate to (θ^*,ψ^*) in a neighborhood of (θ^*,ψ^*). In this case, we say that the discrete dynamical system defined by F is *exponentially stable* at (θ^*,ψ^*). Conversely, if $F'(\theta^*,\psi^*)$ has one eigenvalue with absolute value greater or equal to one, the fixed point iteration (2.8)



(a) *Not stable* ***(b)*** *Stable*

Figure 2.1: Convergence (Continuous). *The continuous dynamical system* (2.5) *is exponentially stable at a stationary point if and only if all eigenvalues lie in the left half-plane. We see that the left system is not stable, since there are eigenvalues in the right half-plane. In contrast, the system on the right is stable.*

is generally not linearly convergent to (θ^*, ψ^*), see Lemma B.3.

These results are analogues to the stability theory (Figure 2.1) for differential equations [94] like the one in (2.5) near a stationary point (θ^*, ψ^*): the stationary point is exponentially stable if and only if all eigenvalues of $v'(\theta^*, \psi^*)$ have negative real-part. The continuous system can be interpreted as optimizing (θ, ψ) with infinitesimal learning rates and was considered by Nagarajan and Kolter [136] in concurrent work.

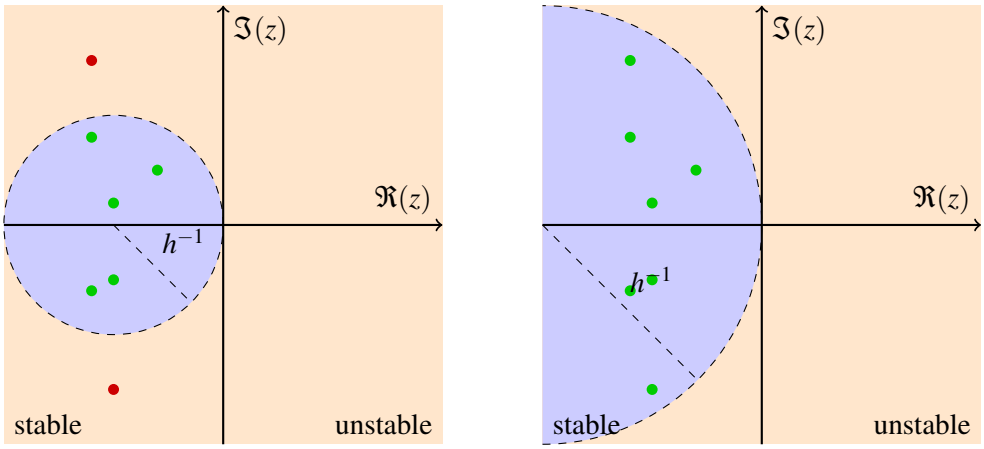
To understand convergence of SimGD and AltGD, we have to understand when the Jacobian of the corresponding update operator F has only eigenvalues with absolute value smaller than one. For simplicity, we consider equal learning rates for both players in this section, i.e. $h_1 = h_2 = h$. An analysis of the influence of different learning rates on convergence of GANs is provided in Chapter 7.

The next lemma describes the eigenvalues of $F'(\theta^*, \psi^*)$ for SimGD:

Lemma 2.3. *The eigenvalues of the Jacobian of the update operator for SimGD with equal learning rates $h_1 = h_2 = h$ are given by $1 + h\lambda$ with λ the eigenvalues of $v'(\theta^*, \psi^*)$. Assume that $v'(\theta^*, \psi^*)$ has only eigenvalues with negative real part. The eigenvalues of the Jacobian of the update operator F for SimGD are then all in the unit circle if and only if*

$$h < \frac{1}{|\Re(\lambda)|} \frac{2}{1 + \left(\frac{\Im(\lambda)}{\Re(\lambda)}\right)^2} \tag{2.9}$$

for all eigenvalues λ of $v'(\theta^*, \psi^*)$.



(a) *Not Stable ($h = 1.0$)* ***(b)*** *Stable ($h = 0.5$)*

Figure 2.2: Convergence (Discretized). *The discretized dynamical system (i.e. SimGD) is exponentially stable at a stationary point if and only if all eigenvalues lie in the unit circle of radius $1/h$ which is tangent to the imaginary axis where h denotes the learning rate. We see that in this example, the discretized dynamical system is not stable for $h = 1.0$, but is stable for $h = 0.5$. In general, stability can be achieved for some $h > 0$ as long as all eigenvalues lie strictly in the left half-plane.*

Proof. For SimGD we have

$$F(\theta, \psi) = (\theta, \psi) + h\, v(\theta, \psi) \tag{2.10}$$

and hence $F'(\theta^*, \psi^*) = I + h\, v'(\theta^*, \psi^*)$. The eigenvalues are therefore given by $\mu = 1 + h\lambda$ with λ the eigenvalues of $v'(\theta^*, \psi^*)$.

To see when $|\mu| < 1$, we write $\lambda = -a + ib$ with $a, b \in \mathbb{R}$ and $a > 0$. Then

$$|\mu|^2 = (1 - ha)^2 + h^2 b^2 \tag{2.11}$$

which is smaller than one if and only if

$$h < \frac{2a}{a^2 + b^2} \tag{2.12}$$

Dividing both the numerator and denominator by a^2 shows the assertion. □

The results from Lemma 2.3 are visualized in Figure 2.2: to achieve convergence for SimGD we have to choose h so that all eigenvalues of the Jacobian $v'(\theta^*, \psi^*)$ lie in a circle with radius h^{-1} that is tangent to the y-axis.

Equation 2.9 shows that there are two major factors that determine the maximum possible learning rate h: (i) the maximum value of $\Re(\lambda)$ and (ii) the maximum value q of $|\Im(\lambda)/\Re(\lambda)|$. Note that as q goes to infinity, we have to choose h according to $\mathcal{O}(q^{-2})$ which can quickly become extremely small. This can also be seen in Figure 2.2: if $F'(\theta^*, \psi^*)$

has an eigenvalue with small absolute real part but big imaginary part, h needs to be chosen extremely small to still achieve convergence. Moreover, even if we make h small enough, most eigenvalues of $F'(\theta^*, \psi^*)$ will be very close to 1, which leads to very slow convergence of the algorithm. Note that this problem of SimGD does not arise for standard gradient descent in optimization, where the Jacobian $F'(\theta^*, \psi^*)$ is symmetric and therefore has only real eigenvalues.

Let us now consider the case of AltGD:

Lemma 2.4. *Assume that $v'(\theta^*, \psi^*)$ has only eigenvalues with negative real part. The eigenvalues of the Jacobian of the update operator for AltGD with equal learning rates $h_1 = h_2 = h$ are all in the unit circle for $h > 0$ small enough.*

Proof. The Jacobian of the update operator $F = F_2 \circ F_1$ at an equilibrium point (θ^*, ψ^*) is

$$F'(\theta^*, \psi^*) = F_2'(\theta^*, \psi^*) \cdot F_1'(\theta^*, \psi^*) \tag{2.13}$$

However, we have

$$F_i'(\theta^*, \psi^*) = I + h\, v_i'(\theta^*, \psi^*) \tag{2.14}$$

for $i \in \{1, 2\}$ where

$$v_1(\theta, \psi) = \begin{pmatrix} -\nabla_\theta \mathcal{L}_1(\theta, \psi) \\ 0 \end{pmatrix} \quad \text{and} \quad v_2(\theta, \psi) = \begin{pmatrix} 0 \\ -\nabla_\psi \mathcal{L}_2(\theta, \psi) \end{pmatrix} \tag{2.15}$$

denote the components of the gradient vector field. Hence

$$\begin{aligned} F'(\theta^*, \psi^*) &= I + h\,(v_1'(\theta^*, \psi^*) + v_2'(\theta^*, \psi^*)) + h^2\, v_2'(\theta^*, \psi^*) v_1'(\theta^*, \psi^*) \\ &= I + h\,(v'(\theta^*, \psi^*) + h\, v_2'(\theta^*, \psi^*) v_1'(\theta^*, \psi^*)) \end{aligned} \tag{2.16}$$

For $h > 0$ small enough, all eigenvalues of

$$v'(\theta^*, \psi^*) + h\, v_2'(\theta^*, \psi^*) v_1'(\theta^*, \psi^*) \tag{2.17}$$

will be arbitrarily close to the eigenvalues of $v'(\theta^*, \psi^*)$.

Because all eigenvalues of $v'(\theta^*, \psi^*)$ have negative real-part, all eigenvalues of $F'(\theta^*, \psi^*)$ will hence lie inside the unit circle for $h > 0$ small enough. □

Lemma 2.3 and Lemma 2.4 give a simple convergence criterion for SimGD and AltGD for isolated stationary points (θ^*, ψ^*) of the gradient vector field. However, for neural networks there is usually a manifold of reparameterizations [38, 136] of the network that lead to equivalent equilibrium points. In the following, let $\mathcal{M} \subseteq \Omega_1 \times \Omega_2$ denote a $\mathcal{C}^1$-manifold such that $\mathcal{M}$ consists only of stationary points of $v(\cdot)$, i.e. $v(\theta, \psi) = 0$ for all $(\theta, \psi) \in \mathcal{M}$. To also handle these equilibria, we prove Theorem 2.5. The proof is based on Theorem B.4 which states that it is sufficient to analyze the eigenvalues of $B^\mathsf{T} F'(\theta^*, \psi^*) B$ where

$$B = (b_1, \ldots, b_l) \in \mathbb{R}^{(n+m)\times l} \tag{2.18}$$

denotes an orthogonal basis of $(\mathrm{T}_{(\theta^*,\psi^*)}\mathcal{M})^\perp$. Here, $(\mathrm{T}_{(\theta^*,\psi^*)}\mathcal{M})^\perp$ is the orthogonal complement of the tangent space of $\mathcal{M}$ at $(\theta^*.\psi^*)$.

Theorem 2.5. *Assume that $\mathcal{M} \subseteq \Omega_1 \times \Omega_2$ is a $\mathcal{C}^1$-manifold and that $v(\theta,\psi) = 0$ for $(\theta,\psi) \in \mathcal{M}$. Let $(\theta^*,\psi^*) \in \mathcal{M}$ and let $B \in \mathbb{R}^{(n+m)\times l}$ denote an orthogonal basis of $(\mathrm{T}_{(\theta^*,\psi^*)}\mathcal{M})^\perp$. If all eigenvalues of $B^\mathsf{T} v'(\theta^*,\psi^*)B$ have negative real-part, then both SimGD and AltGD are linearly convergent to $\mathcal{M}$ in a neighborhood of (θ^*,ψ^*) for small enough learning rates $h > 0$.*

Proof. To prove convergence, we want to apply Theorem B.4. To this end, we only have to show that all eigenvalues of $B^\mathsf{T} F'(\theta^*,\psi^*)B$ have absolute value smaller than one.

However, if F denotes the updater operator of SimGD, we have

$$B^\mathsf{T} F'(\theta^*,\psi^*)B = B^\mathsf{T}(I + h\,v'(\theta^*,\psi^*))B = I + h\,B^\mathsf{T} v'(\theta^*,\psi^*)B \tag{2.19}$$

Here, we used that $B^\mathsf{T} B = I$, which holds because the columns of B are orthonormal. As in the proof of Lemma 2.3, we therefore see that all eigenvalues of $B^\mathsf{T} F'(\theta^*,\psi^*)B$ have absolute value smaller than one for h small enough if all eigenvalues of $B^\mathsf{T} v'(\theta^*,\psi^*)B$ have negative real-part.

Similarly, if F denotes the update operator of AltGD, we have as in the proof of Lemma 2.4

$$\begin{aligned} B^\mathsf{T} F'(\theta^*,\psi^*)B &= B^\mathsf{T}\left(I + h\,v'(\theta^*,\psi^*) + h^2\,v_2'(\theta^*,\psi^*)v_1'(\theta^*,\psi^*)\right)B \\ &= I + h\left(B^\mathsf{T} v'(\theta^*,\psi^*)B + h B^\mathsf{T} v_2'(\theta^*,\psi^*)v_1'(\theta^*,\psi^*)B\right) \end{aligned} \tag{2.20}$$

with $v_1(\cdot)$ and $v_2(\cdot)$ as in the proof of Lemma 2.4. However, as in the proof of Lemma 2.4, we see that for small $h > 0$ the eigenvalues of

$$B^\mathsf{T} v'(\theta^*,\psi^*)B + h B^\mathsf{T} v_2'(\theta^*,\psi^*)v_1'(\theta^*,\psi^*)B \tag{2.21}$$

will be arbitrarily close to the eigenvalues of $B^\mathsf{T} v'(\theta^*,\psi^*)B$. For $h > 0$ small enough all eigenvalues of $B^\mathsf{T} F'(\theta^*,\psi^*)B$ will therefore have absolute value smaller than one.

All in all, Theorem B.4 now implies that both SimGD and AltGD are convergent to $\mathcal{M}$ with linear rate in a neighborhood of (θ^*,ψ^*). □

2.4 Consensus Optimization

In Section 2.3 we have seen that both SimGD and AltGD are locally convergent to a local Nash equilibrium if $v'(\theta^*,\psi^*)$ has only eigenvalues with negative real part. However, it is easy to see that the Jacobian $v'(\theta^*,\psi^*)$ for the bilinear game from Section 2.1 has the eigenvalues $\pm i$. Theorem 5.8 is hence not applicable. Indeed, it can be shown that neither SimGD nor AltGD converge for this simple example.

In this section we describe Consensus Optimization (ConOpt) [128], a general algorithm that can be applied to general smooth two-player zero-sum games to make the training dynamics exponentially stable. While Consensus Optimization can introduce spurious points

of attraction to the training dynamics, we find that it can be successfully applied to GANs, enabling us to train architectures that do not converge for vanilla SimGD or AltGD.

Related approaches to Consensus Optimization include the extragradient method [58, 101], optimistic mirror descent [34, 125, 161] and the prediction method [202] that lead to local convergence in smooth two-player games. A similar regularization term as in Consensus Optimization was also independently proposed by Nagarajan and Kolter [136]. However, Nagarajan and Kolter [136] proposed to only regularize the component $\nabla_\psi \mathcal{L}(\theta, \psi)$ of the gradient vector field corresponding to the discriminator parameters. Moreover, the regularization term is only added to the generator objective to give the generator more foresight.

All these related approaches and Consensus Optimization have in common that they focus on arbitrary smooth two-player games, but do not exploit the special structure of GANs. As a result, they may be ill-behaved far away from the equilibrium point and can also introduce spurious points of attraction to the GAN training dynamics. In Chapter 3 we derive a regularizer for the special case of GANs which does not suffer from these issues, but still makes the equilibrium for GANs exponentially stable.

2.4.1 Derivation

Finding stationary points of the vector field $v(\cdot)$ is equivalent to solving the equation $v(\theta, \psi) = 0$. In the context of two-player games this means that we have to solve

$$\nabla_\theta \mathcal{L}_1(\theta, \psi) = 0 \quad \text{and} \quad \nabla_\psi \mathcal{L}_2(\theta, \psi) = 0. \tag{2.22}$$

A simple strategy for finding such stationary points is to minimize $R(\theta, \psi) = \frac{1}{2}\|v(\theta, \psi)\|^2$ for (θ, ψ). Unfortunately, this can result in unstable stationary points of $v(\cdot)$ or other local minima of $\frac{1}{2}\|v(\theta, \psi)\|^2$ and in practice it does not work well.

We therefore consider a modified vector field $\tilde{v}(\cdot)$ that is as close as possible to the original vector field $v(\cdot)$, but at the same time still minimizes $R(\theta, \psi)$ (at least locally). A sensible candidate for such a vector field is

$$\tilde{v}(\theta, \psi) = v(\theta, \psi) - \gamma \nabla R(\theta, \psi) \tag{2.23}$$

for some $\gamma > 0$. A simple calculation shows that the gradient $\nabla R(\theta, \psi)$ is given by

$$\nabla R(\theta, \psi) = v'(\theta, \psi)^\mathsf{T} v(\theta, \psi) \tag{2.24}$$

This vector field is the gradient vector field associated to the modified two-player game given by the two modified cost functions

$$\tilde{\mathcal{L}}_1(\theta, \psi) = \mathcal{L}_1(\theta, \psi) + \gamma R(\theta, \psi) \quad \text{and} \quad \tilde{\mathcal{L}}_2(\theta, \psi) = \mathcal{L}_2(\theta, \psi) + \gamma R(\theta, \psi) \tag{2.25}$$

The regularizer $R(\theta, \psi)$ encourages agreement between the two players. We therefore call the resulting algorithm *Consensus Optimization* (Algorithm 3).[2] Note that in a stochastic

[2]This algorithm requires backpropagation through the squared norm of the gradient with respect to the weights

Algorithm 3 Consensus Optimization

while not converged **do**
  $\delta_\theta \leftarrow -\nabla_\theta(\mathcal{L}_1(\theta,\psi) + \gamma R(\theta,\psi))$
  $\delta_\psi \leftarrow -\nabla_\psi(\mathcal{L}_2(\theta,\psi) + \gamma R(\theta,\psi))$
  $\theta \leftarrow \theta + h\,\delta_\theta$
  $\psi \leftarrow \psi + h\,\delta_\psi$
end while

setting, we can obtain unbiased gradients of $R(\theta,\psi)$ by evaluating $v(\cdot)$ on two different batches and taking the inner product of the results.[3]

2.4.2 Convergence

For analyzing convergence, we consider a more general algorithm than in Section 2.4.1 which is given by iteratively applying a function $F : \Omega_1 \times \Omega_2 \to \Omega_1 \times \Omega_2$ of the form

$$F(\theta,\psi) = (\theta,\psi) + hW(\theta,\psi)\,v(\theta,\psi) \tag{2.26}$$

for some learning rate $h > 0$ and an invertible matrix $W(\theta,\psi)$. Consensus Optimization is a special case of this algorithm for $W(\theta,\psi) = I - \gamma v'(\theta,\psi)^\mathsf{T}$. We assume that γ^{-1} is not an eigenvalue of $v'(\theta,\psi)^\mathsf{T}$ for any (θ,ψ), so that $W(\theta,\psi)$ is indeed invertible.

Lemma 2.6. *Assume $h > 0$ and $W(\theta,\psi)$ invertible for all $(\theta,\psi) \in \Omega_1 \times \Omega_2$. Then (θ^*,ψ^*) is a fixed point of* (2.26) *if and only if it is a stationary point of $v(\cdot)$. Moreover, if (θ^*,ψ^*) is a stationary point of $v(\cdot)$, we have*

$$F'(\theta^*,\psi^*) = I + hW(\theta^*,\psi^*)v'(\theta^*,\psi^*) \tag{2.27}$$

Proof. If $v(\theta^*,\psi^*) = 0$, then $F(\theta^*,\psi^*) = (\theta^*,\psi^*)$, so (θ^*,ψ^*) is a fixed point of F. Conversely, if (θ^*,ψ^*) satisfies $F(\theta^*,\psi^*) = (\theta^*,\psi^*)$, we have $W(\theta^*,\psi^*)\,v(\theta^*,\psi^*) = 0$. Because we assume $W(\theta^*,\psi^*)$ to be invertible, this shows $v(\theta^*,\psi^*) = 0$.

Recall that $\Omega_1 \subseteq \mathbb{R}^n$ and $\Omega_2 \subseteq \mathbb{R}^m$ and consider $w \in \mathbb{R}^{n+m}$. The directional derivative of $F(\theta,\psi)$ in the direction of w is given by

$$\partial_w F(\theta,\psi) = w + h\,\partial_w W(\theta,\psi)\,v(\theta,\psi) + hW(\theta,\psi)\partial_w v(\theta,\psi) \tag{2.28}$$

For a fixed point (θ^*,ψ^*) we have by the first part of the proof $v(\theta^*,\psi^*) = 0$ and therefore

$$\begin{aligned} F'(\theta^*,\psi^*)w &= \partial_w F(\theta^*,\psi^*) \\ &= w + hW(\theta^*,\psi^*)\partial_w v(\theta^*,\psi^*) \\ &= (I + hW(\theta^*,\psi^*)v'(\theta^*,\psi^*))w \end{aligned} \tag{2.29}$$

of the network. This is sometimes called *double backpropagation* [41] and is supported by modern deep learning frameworks such as Tensorflow [1] and PyTorch [153].

[3] In practice, we find that Consensus Optimization also works when we evaluate $R(\theta,\psi)$ only on one batch and we use this simplified version in our experiments.

Since $w \in \mathbb{R}^{n+m}$ was arbitrary, this shows

$$F'(\theta^*, \psi^*) = I + hW(\theta^*, \psi^*)\, v'(\theta^*, \psi^*) \tag{2.30}$$

□

We are now ready to state our convergence theorem. To this end, we want to apply Theorem 2.5. We therefore again consider a manifold $\mathcal{M}$ so that for all $(\theta, \psi) \in \mathcal{M}$ we have $v(\theta, \psi) = 0$. We further assume that for any $w \in \mathbb{R}^{n+m}$ which is not in $\mathrm{T}_{(\theta^*, \psi^*)}\mathcal{M}$ the directional derivative $\partial_w v(\theta^*, \psi^*)$ is non-zero. The meaning of this assumption is that every time we leave the manifold of equilibrium points $\mathcal{M}$, the vector field becomes non-zero.

Theorem 2.7. *Assume that $\mathcal{M} \subseteq \Omega_1 \times \Omega_2$ is a $\mathcal{C}^1$-manifold and that $v(\theta, \psi) = 0$ for all $(\theta, \psi) \in \mathcal{M}$. Let $(\theta^*, \psi^*) \in \mathcal{M}$ be a local Nash equilibrium of the zero-sum game defined by $\mathcal{L}(\theta, \psi)$. Assume that for any $w \in \mathbb{R}^{n+m}$ with $w \notin \mathrm{T}_{(\theta^*, \psi^*)}\mathcal{M}$ we have $\partial_w v(\theta^*, \psi^*) \neq 0$. Then Consensus Optimization is linearly convergent towards $\mathcal{M}$ in a neighborhood of (θ^*, ψ^*) for small enough learning rates $h > 0$.*

Proof. For simplicity, we first define $J := v'(\theta^*, \psi^*)$ and $\tilde{J} := \tilde{v}'(\theta^*, \psi^*)$. In order to apply Theorem 2.5, we have to show that $B^\mathsf{T}\tilde{J}B$ has only eigenvalues with negative real-part where $B \in \mathbb{R}^{(n+m)\times l}$ defines an orthogonal basis of $(\mathrm{T}_{(\theta^*, \psi^*)}\mathcal{M})^\perp$.

Let $\lambda \in \mathbb{C}$ denote an eigenvalue of $B^\mathsf{T}\tilde{J}B$ with corresponding eigenvector $w \in \mathbb{C}^{n+m}$, $\|w\| = 1$, and let $\tilde{w} = Bw$. Since $\|w\| = 1$, we also have $\|\tilde{w}\| = 1$. By Lemma 2.6 and (2.24) we have $\tilde{J} = J - \gamma J^\mathsf{T} J$ and therefore

$$\lambda = w^H \left(B^\mathsf{T}\tilde{J}B\right) w = \tilde{w}^H \tilde{J}\tilde{w} = \tilde{w}^H J\tilde{w} - \gamma \|J\tilde{w}\|^2 \tag{2.31}$$

Using Corollary 2.2 and (2.3), we see that the real part of $\tilde{w}^H J\tilde{w}$ for $\tilde{w} = (\tilde{w}_1^\mathsf{T}, \tilde{w}_2^\mathsf{T})^\mathsf{T}$ with $\tilde{w}_1 \in \mathbb{C}^n$, $\tilde{w}_2 \in \mathbb{C}^m$ is

$$\begin{aligned} \Re(\tilde{w}^H J\tilde{w}) &= \frac{1}{2}\tilde{w}^H \left(J + J^\mathsf{T}\right) \tilde{w} \\ &= -\frac{1}{2}\tilde{w}_1^H \nabla_\theta^2 \mathcal{L}(\theta^*, \psi^*)\, \tilde{w}_2 + \frac{1}{2}\tilde{w}_2^H \nabla_\psi^2 \mathcal{L}(\theta^*, \psi^*)\, \tilde{w}_2 \\ &\leq 0 \end{aligned} \tag{2.32}$$

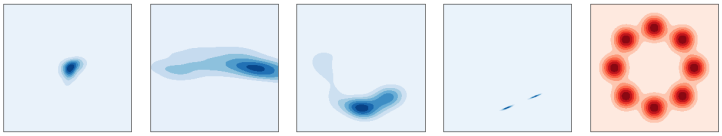
because we assumed (θ^*, ψ^*) to be a local Nash equilibrium. Therefore,

$$\Re(\lambda) = \Re\left(\tilde{w}^H J\tilde{w}\right) - \gamma\|J\tilde{w}\|^2 \leq -\gamma\|J\tilde{w}\|^2 \tag{2.33}$$

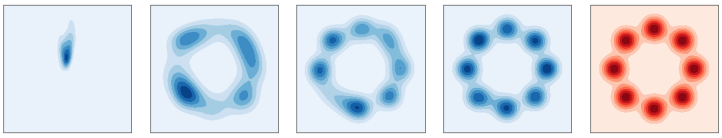
Let $\tilde{w}_r$ and $\tilde{w}_i$ denote the real and imaginary part of $\tilde{w} = Bw$, respectively. Since $\tilde{w}_i, \tilde{w}_r \in (\mathrm{T}_{(\theta^*, \psi^*)}\mathcal{M})^\perp$ and $\tilde{w} \neq 0$, either $\tilde{w}_r$ or $\tilde{w}_i$ is not in $\mathrm{T}_{(\theta^*, \psi^*)}\mathcal{M}$. As a result, by assumption,

$$J\tilde{w} = J\tilde{w}_r + iJ\tilde{w}_i = \partial_{\tilde{w}_r} v'(\theta^*, \psi^*) + i\,\partial_{\tilde{w}_i} v'(\theta^*, \psi^*) \neq 0 \tag{2.34}$$

Together with (2.33) this yields $\Re(\lambda) < 0$. Theorem 2.5 now shows the assertion. □



(a) *Simultaneous Gradient Descent*



(b) *Consensus optimization*

Figure 2.3: Mixture of Gaussians. *Comparison of SimGD and Consensus Optimization on a circular mixture of Gaussians. The images depict from left to right the resulting densities of the algorithm after* 0*,* 5000*,* 10000 *and* 20000 *iterations as well as the target density (in red).*

2.4.3 Experiments

Mixture of Gaussians In our first experiment we evaluate Consensus Optimization on a simple $2D$-example where our goal is to learn a mixture of 8 Gaussian distributions with standard deviations equal to 10^{-2} and modes uniformly distributed around the unit circle. While simplistic, training algorithms for GANs often fail to converge even on such simple examples without extensive fine-tuning of the architecture and hyperparameters [131].

For both the generator and discriminator we use fully-connected neural networks with 4 hidden layers and 16 hidden units in each layer. For all layers, we use ReLU-nonlinearities. We use a 16-dimensional Gaussian prior for the latent code z and set up the game between the generator and discriminator using the cost functions from [62]. To test Consensus Optimization, we run both SimGD and Consensus Optimization with RMSProp[4] and a learning rate of 10^{-4} for 20000 steps. For Consensus Optimization, we use a regularization parameter of $\gamma = 10$.

The results produced by SimGD and Consensus Optimization for 0, 5000, 10000 and 20000 iterations are depicted in Figure 2.3. We see that while SimGD jumps around the modes of the distribution and fails to converge, Consensus Optimization converges smoothly to the target distribution (shown in red). Figure 2.4 shows the empirical distribution of the eigenvalues of the Jacobian of $v(\cdot)$ and the regularized vector field $\tilde{v}(\cdot)$. It can be seen that near the Nash equilibrium most eigenvalues are indeed very close to the imaginary axis and that the proposed modification of the vector field used in Consensus Optimization moves the eigenvalues to the left.

[4]While RMSProp is slightly more complex than gradient descent, it can be interpreted as gradient descent with a mechanism of automatically rescaling the weights. If we assume this rescaling to be (approximately) constant near an equilibrium point, our theory still applies.

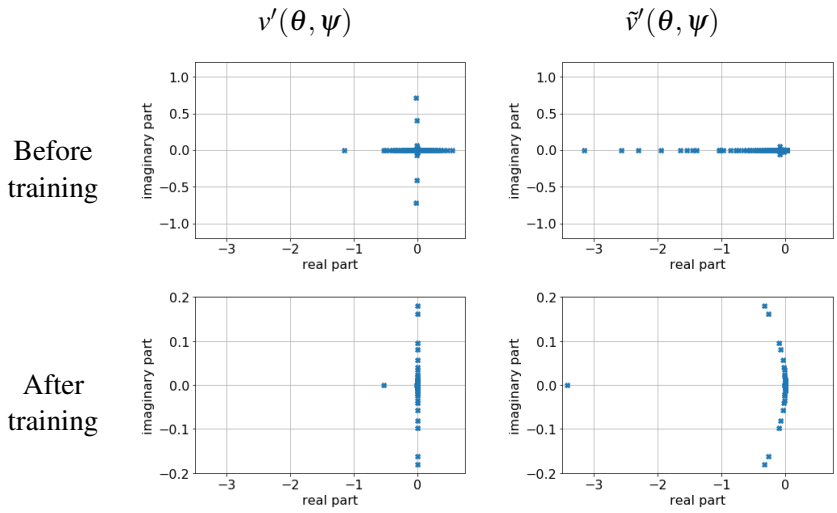
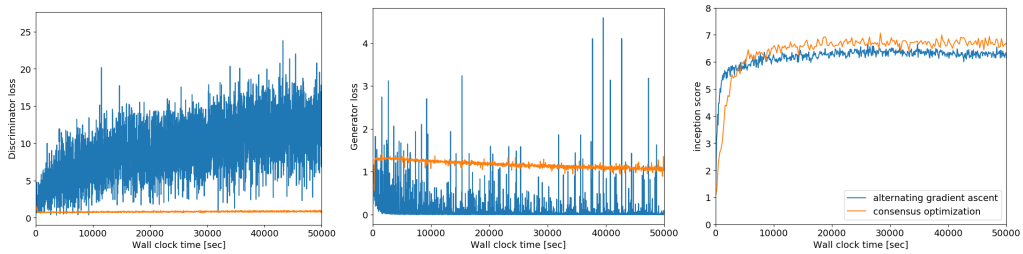


Figure 2.4: Eigenvalues. *Empirical distribution of eigenvalues before and after training using Consensus Optimization. The first column shows the distribution of the eigenvalues of the Jacobian $v'(\theta, \psi)$ of the unmodified vector field $v(\theta, \psi)$. The second column shows the eigenvalues of the Jacobian $\tilde{v}'(\theta, \psi)$ of the regularized vector field $\tilde{v}(\theta, \psi) = v(\theta, \psi) - \gamma \nabla R(\theta, \psi)$ used in Consensus Optimization. We see that $v'(\theta, \psi)$ has eigenvalues close to the imaginary axis near the Nash equilibrium. As predicted theoretically, this is not the case for the regularized vector field $\tilde{v}(\theta, \psi)$. For visualization purposes, the real part of the spectrum of $\tilde{v}'(\theta, \psi)$ before training was clipped.*

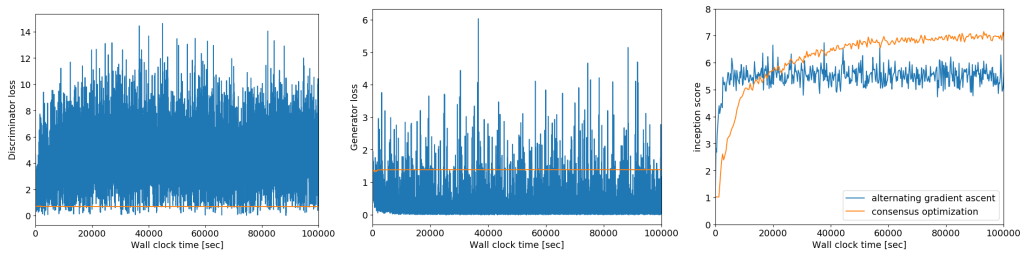
CIFAR-10 In our second experiment, we apply Consensus Optimization to the CIFAR-10 dataset [102], using a DC-GAN-like architecture [160] without Batch Normalization [75] in the generator or discriminator. These architectures are known to be hard to optimize using SimGD or AltGD [5, 160].

We see that Consensus Optimization successfully trains the models and we also observe that unlike when using vanilla AltGD, the generator and discriminator losses remain almost constant during training. This is illustrated in Figure 2.5. For a quantitative evaluation, we also measure the Inception score [173] over time (Figure 2.5c), showing that Consensus Optimization compares favorably to a DC-GAN trained with AltGD. The improvement of Consensus Optimization over AltGD is even more significant when we use four instead of three convolutional layers (Figure 2.6). A qualitative comparison is shown in Figure 2.7. We see that Consensus Optimization successfully stabilizes GAN training in cases where SimGD and AltGD fail. Interestingly, however, it fails for the standard setting of a DC-GAN where we use Batch Normalization. We believe that this due the presence of spurious attractors in the GAN training dynamics when we use Consensus Optimization: while Consensus Optimization stabilizes the training dynamics, it can make formerly instable stationary points of the gradient vector field stable if the regularization parameter is chosen to be high. This may lead to poor solutions.



(a) *Discriminator loss* ***(b)*** *Generator loss* ***(c)*** *Inception score*

Figure 2.5: Training Dynamics. *(a) and (b): Comparison of the generator and discriminator losses on a DC-GAN architecture with three convolutional layers trained on CIFAR-10 for Consensus Optimization (without Batch Normalization) and AltGD (with Batch Normalization). We observe that while AltGD leads to highly fluctuating losses, Consensus Optimization successfully stabilizes the training and makes the losses almost constant during training. (c): Comparison of the Inception score over time which was computed using* 6400 *samples. We see that on this architecture both methods have comparable rates of convergence and Consensus Optimization achieves slightly better end results.*



(a) *Discriminator loss* ***(b)*** *Generator loss* ***(c)*** *Inception score*

Figure 2.6: Training Dynamics. *(a) and (b): Comparison of the generator and discriminator losses on a DC-GAN architecture with four convolutional layers trained on CIFAR-10 for Consensus Optimization (without Batch Normalization) and AltGD (with Batch Normalization). (c): Comparison of the Inception score over time which was computed using* 6400 *samples. We see that on this architecture Consensus Optimization achieves much better end results.*

Finally, we note that Consensus Optimization becomes less stable for deeper architectures. We attribute this result both to the presence of spurious attractors in the regularized gradient vector field and that our regularizer can lead to eigenvalues with big absolute value, which makes the Jacobian of the gradient vector field ill-conditioned. Indeed, recent works [9, 55, 140] suggest that considerably better results can be obtained by adapting the matrix $W(\theta, \psi)$ in (2.26) so that the diagonal blocks are given by identity matrices. In Chapter 6 we introduce a different regularizer for GAN training that is better adapted to GANs and which therefore does not have the same problems, but still makes the training dynamics exponentially stable.

2.5 Conclusion

In this chapter we have taken a step back and looked at local convergence for general smooth two-player games. By extending the theory of Ratliff, Burden, and Sastry [163] we have seen that local convergence for both Simultaneous Gradient Descent and Alternating Gradient Descent is determined by the properties of the eigenvalues of the gradient vector field. We have also derived a simple algorithm that ensures local convergence of general smooth two-player games, albeit at the price of potentially introducing spurious attractors to the game dynamics. While these results are directly applicable to GANs, we will see in the next chapters that for the special case of GANs we can get additional theoretical insights. In particular, we construct simple examples of GANs that do not converge and derive novel regularizers that ensure local convergence of general GANs under mild technical assumptions.

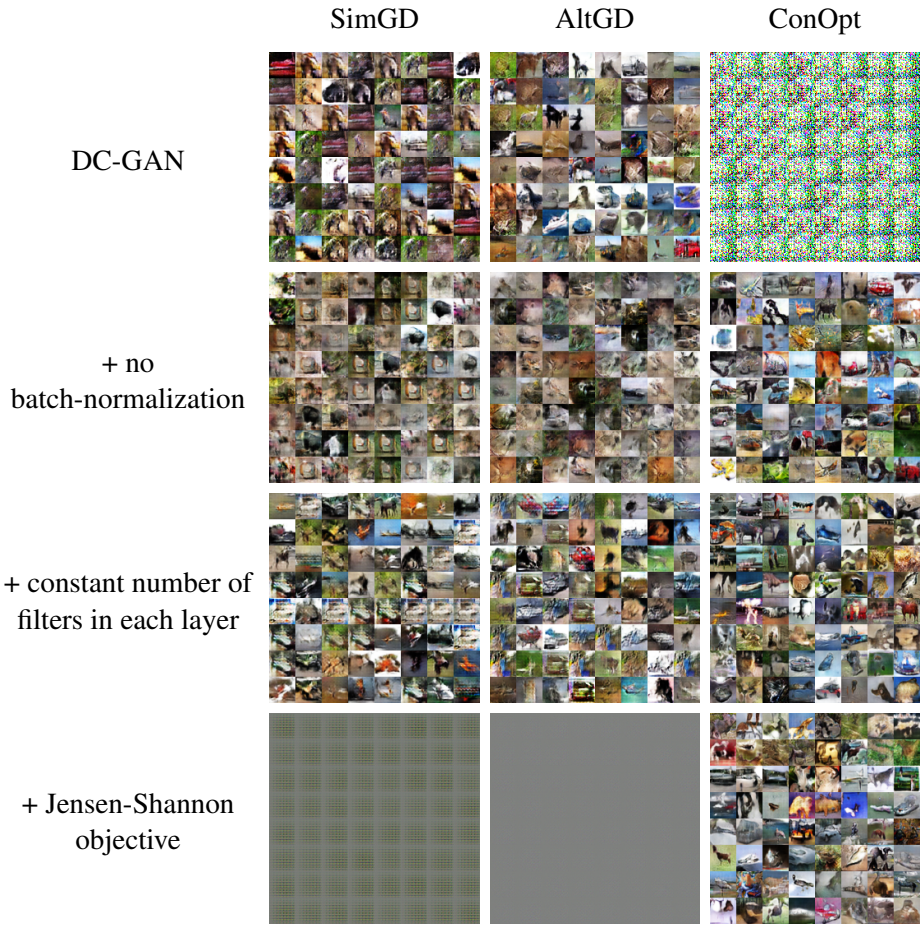


Figure 2.7: Qualitative Comparison. *Comparison of the results obtained by training different architectures with SimGD, AltGD and Consensus Optimization: we train a standard DC-GAN architecture (with* 4 *convolutional layers), a DC-GAN architecture without Batch Normalization in neither the discriminator nor the generator, a DC-GAN architecture that additionally has a constant number of filters in each layer and a DC-GAN architecture that additionally uses the Jensen-Shannon objective for the generator. This objective was used for the derivations in [62], but not used in practice as it was found to hamper convergence. While Consensus Optimization struggles with the original DC-GAN architecture (with Batch Normalization), it successfully trains all other architectures.*

3 The Dirac-GAN

In Chapter 1 we have seen how Generative Adversarial Networks (GANs) can be understood both from a divergence and a game-theoretic point of view. While the former view yields an interpretation of *what* we optimize, the second one leads to concrete algorithms such as Simultaneous Gradient Descent (SimGD) and Alternating Gradient Descent (AltGD). However, in Chapter 2 we have seen that SimGD and AltGD need not converge to a Nash equilibrium for general smooth two-player games, even locally. Importantly, standard convergence results for gradient descent in optimization do not directly transfer to smooth two-player games.

In this chapter we start our discussion of GAN convergence with a simple, yet highly informative example of GAN training: we design a minimal example of GAN training that can be understood analytically. This example, which we call the *Dirac-GAN*, serves various purposes: first, from a theoretic point of view, it serves as a counterexample which shows that naive gradient descend-based GAN training does not always converge. Second, it gives us a framework to formally analyze various algorithms and regularization strategies for GAN training and to understand what behavior we can expect in more complex situations. Finally, it serves as a testbed to develop new techniques to stabilize GAN training which we can generalize to more complex examples. Indeed, we will see in Chapter 6 and 8 that regularization techniques which we develop for the Dirac-GAN generalize remarkably well to more complex GAN architectures.

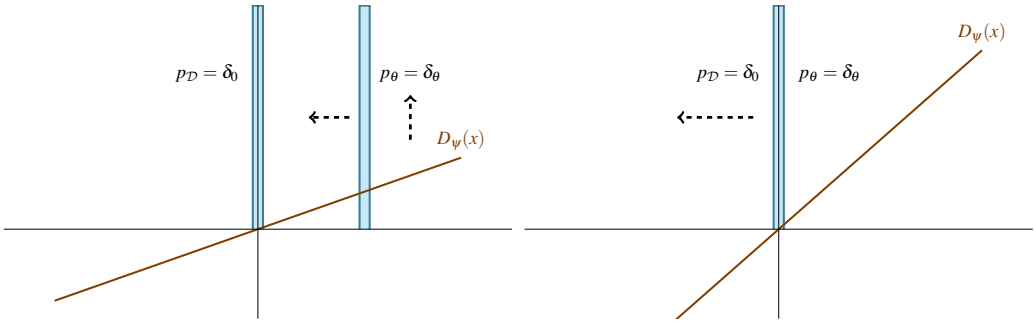
3.1 The one-dimensional Dirac-GAN

In this section we introduce a minimal one-dimensional example of GAN training, which we call the *Dirac-GAN*. In Chapter 4, we generalize this example to higher dimensions which will lead to additional insights.

How could a minimal example of GAN training look like? To fully specify a GAN, we have to specify the data distribution $p_{\mathcal{D}}$, the distribution produced by the generator p_θ and the discriminator D_ψ. The simplest possible data distribution $p_{\mathcal{D}}$ is arguably given by a single number in $\mathbb{R}$. To even further simplify the discussion[1], we set this number to zero: $p_{\mathcal{D}} = \delta_0$. The generator also produces just one number, which we denote by $\theta \in \mathbb{R}$. To discriminate between two numbers, a linear discriminator is sufficient: $D_\psi(x) = \psi \cdot x$ with $\psi \in \mathbb{R}$. In summary, we define the *Dirac-GAN* as follows:

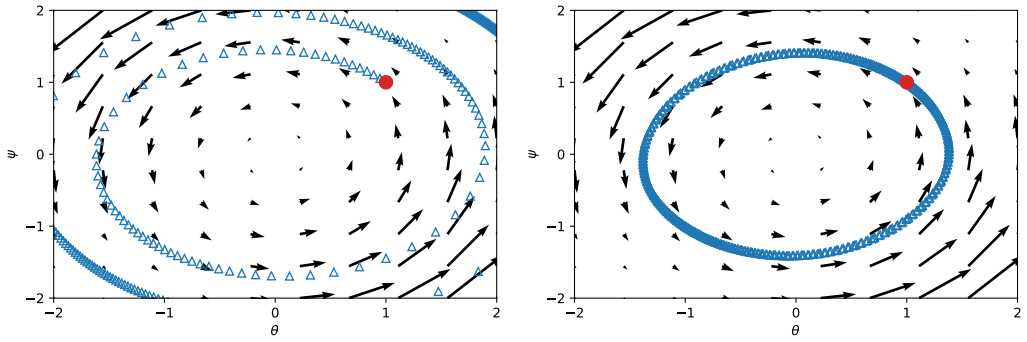
Definition 3.1. *The one-dimensional* Dirac-GAN *consists of a (univariate) generator distribution $p_\theta = \delta_\theta$ and a linear discriminator $D_\psi(x) = \psi \cdot x$. The true data distribution $p_{\mathcal{D}}$ is given by a Dirac-distribution concentrated at zero.*

[1] We remove this assumption in Chapter 4.



(a) $t = t_0$ **(b)** $t = t_1$

Figure 3.1: Dirac-GAN. *Visualization of the Dirac-GAN which shows that gradient descent-based GAN optimization is not always convergent: (a) In the beginning, the discriminator pushes the generator towards the true data distribution and the discriminator's slope increases. (b) When the generator reaches the target distribution, the slope of the discriminator is largest, pushing the generator away from the target distribution. This results in oscillatory training dynamics that never converge.*



(a) *SimGD* **(b)** *AltGD*

Figure 3.2: Training Dynamics. *Training behavior of the Dirac-GAN. The starting iterate is marked in red.*

Note that for the Dirac-GAN, both the generator and the discriminator have exactly one parameter. This situation is visualized in Figure 3.1. In this setup, the GAN training objective (1.28) is given by

$$\mathcal{L}(\theta, \psi) = \varphi_1(\psi\,\theta) + \varphi_2(0) \tag{3.1}$$

The usage of a linear discriminator in Definition 3.1 might appear restrictive. However, from a divergence point of view (Section 1.1) the class of linear discriminators is in fact as powerful as the class of all real-valued functions for this example: when we use $\varphi_1(t) = -\log(1+\exp(-t))$ and we take the supremum over ψ in (3.1), we obtain (up to scalar and additive constants) the Jensen-Shannon divergence between p_θ and $p_\mathcal{D}$. The same holds true for all other divergences considered in Section 1.1.

Interestingly, GAN training does not converge in this simple setup.

Lemma 3.2. *Assume that (φ_1, φ_2) is valid (Definition 1.9). The unique Nash equilibrium of the game defined by* (3.1) *is given by $(\theta^*, \psi^*) = (0,0)$. Moreover, the Jacobian of the gradient vector field at the equilibrium point has the two eigenvalues $\pm\varphi_1'(0)\,i$ which are both on the imaginary axis.*

Proof. First note that $p_{\theta^*} = p_\mathcal{D}$ and $p_{\psi^*} = \chi_0$. By Lemma 1.14, (θ^*, ψ^*) hence defines a (pure) Nash equilibrium. Moreover, for $\theta \neq \theta^* = 0$,

$$\mathrm{E}_{x\sim p_\mathcal{D}}\left[\frac{\partial}{\partial\psi} D_\psi(x)\Big|_{\psi=\psi^*}\right] = 0 \neq \theta = \mathrm{E}_{x\sim p_\theta}\left[\frac{\partial}{\partial\psi} D_\psi(x)\Big|_{\psi=\psi^*}\right] \tag{3.2}$$

By Lemma 1.15 every Nash equilibrium (θ_{NE}, ψ_{NE}) hence satisfies $p_{\theta_{NE}} = p_\mathcal{D}$, i.e. $\theta_{NE} = 0$. We obtain the gradient vector field $v(\theta, \psi)$ by differentiating the loss (3.1) with respect to θ and ψ:

$$v(\theta, \psi) = \begin{pmatrix} -\varphi_1'(\theta\,\psi)\,\psi \\ \varphi_1'(\theta\,\psi)\,\theta \end{pmatrix} \tag{3.3}$$

In particular, since $\theta_{NE} = 0$,

$$v(\theta_{NE}, \psi_{NE}) = \begin{pmatrix} -\varphi_1'(0)\,\psi_{NE} \\ 0 \end{pmatrix} \tag{3.4}$$

Because (θ_{NE}, ψ_{NE}) is a pure Nash equilibrium, we have $v(\theta_{NE}, \psi_{NE}) = 0$ (Lemma 2.1). Because (φ_1, φ_2) is valid, we have $\varphi_1'(0) \neq 0$ and thus $\psi_{NE} = 0$. This shows that $(\theta^*, \psi^*) = (0,0)$ is indeed the unique Nash equilibrium.

Moreover, the Jacobian $v'(\theta, \psi)$ of $v(\cdot)$ is given by

$$\begin{pmatrix} -\varphi_1''(\theta\,\psi)\,\psi^2 & -\varphi_1'(\theta\,\psi) - \varphi_1''(\theta\,\psi)\,\theta\,\psi \\ \varphi_1'(\theta\,\psi) + \varphi_1''(\theta\,\psi)\,\theta\,\psi & \varphi_1''(\theta\,\psi)\,\theta^2 \end{pmatrix} \tag{3.5}$$

Evaluating (3.5) at the Nash equilibrium $\theta^* = \psi^* = 0$, we obtain

$$v'(0,0) = \begin{pmatrix} 0 & -\varphi_1'(0) \\ \varphi_1'(0) & 0 \end{pmatrix} \tag{3.6}$$

which has the eigenvalues $\pm\varphi_1'(0)\,i$. □

We now take a closer look at the training dynamics produced by various algorithms for training the Dirac-GAN. First, we consider the (idealized) continuous system in (2.5): while Lemma 3.2 shows that the continuous system is generally not linearly convergent to the equilibrium point, it could in principle converge with a sublinear convergence rate. However, this is not the case as the next lemma shows:

Lemma 3.3. *The integral curves of the gradient vector field $v(\theta,\psi)$ do not converge to the Nash equilibrium. More specifically, every integral curve $(\theta(t),\psi(t))$ of the gradient vector field $v(\theta,\psi)$ satisfies $\theta(t)^2 + \psi(t)^2 = const$ for all $t \in [0,\infty)$.*

Proof. Let $R(\theta,\psi) := \frac{1}{2}(\theta^2+\psi^2)$. Then

$$\frac{\mathrm{d}}{\mathrm{d}t} R(\theta(t),\psi(t)) = \theta(t)v_1(\theta(t),\psi(t)) + \psi(t)v_2(\theta(t),\psi(t)) = 0 \tag{3.7}$$

showing that $R(\theta,\psi)$ is indeed constant for all $t \in [0,\infty)$. □

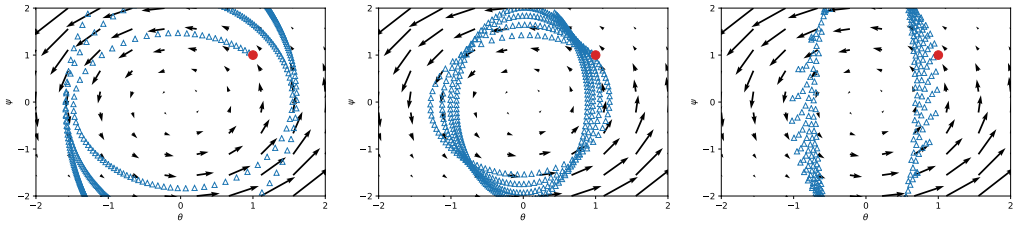
Note that our results do not contradict the results of Nagarajan and Kolter [136] and Heusel et al. [68]: our example violates Assumption IV by Nagarajan and Kolter [136] that the support of the generator distribution is equal to the support of the true data distribution near the equilibrium (see Chapter 5). It also violates the assumption[2] by Heusel et al. [68] that the optimal discriminator parameter vector is a continuous function of the current generator parameters. In fact, unless $\theta = 0$, there is not even an optimal discriminator parameter for the Dirac-GAN. Indeed, we find that two time-scale updates as suggested by Heusel et al. [68] do not lead to convergence for the Dirac-GAN (see Figure 3.3). However, our example seems to be a prototypical situation for (unregularized) GAN training which usually deals with distributions that are concentrated on lower dimensional manifolds [6].

We now take a closer look at the *discretized system*.

Lemma 3.4. *For SimGD with learning rates h_g and h_d for the generator and discriminator, respectively, the Jacobian of the update operator $F(\theta,\psi)$ has eigenvalues $\lambda_{1/2} = 1 \pm \sqrt{h_g h_d}\varphi_1'(0)i$ with absolute values $\sqrt{1+h_g h_d \varphi_1'(0)^2}$ at the Nash equilibrium. Independently of the learning rate, SimGD is therefore not stable near the equilibrium. Even stronger, for every initial condition and learning rates $h_g, h_d > 0$, the weighted squared norm*

$$\frac{\theta_k^2}{h_g} + \frac{\psi_k^2}{h_d} \tag{3.8}$$

[2] This assumption is usually even violated by Wasserstein GANs (WGANs), as the optimal discriminator parameter vector as a function of the current generator parameters can have discontinuities near the Nash equilibrium. See Section 3.3.2 for details.



(a) SimGD *(b) AltGD ($n_d = 1$)* *(c) AltGD ($n_d = 5$)*

Figure 3.3: Two time-scale Training. *Convergence properties of our GAN using two time-scale training as proposed by Heusel et al. [68]. For the Dirac-GAN we do not see any sign of convergence when training with two time-scales. The starting iterate is marked in red.*

of the iterates (θ_k, ψ_k) obtained by SimGD is monotonically increasing.

Proof. The update operator for SimGD is given by

$$F(\theta, \psi) = \begin{pmatrix} \theta - h_g \varphi_1'(\theta\psi)\psi \\ \psi + h_d \varphi_1'(\theta\psi)\theta \end{pmatrix} \tag{3.9}$$

The Jacobian at the equilibrium point $(\theta^*, \psi^*) = (0,0)$ is therefore

$$F'(0,0) = \begin{pmatrix} 1 & -h_g \varphi_1'(0) \\ h_d \varphi_1'(0) & 1 \end{pmatrix} \tag{3.10}$$

An easy calculation shows that the eigenvalues are given by

$$\lambda_{1/2} = 1 + \sqrt{h_g h_d}\, \varphi_1'(0)\, i \tag{3.11}$$

To see the weighted squared norms of the iterates (θ_k, ψ_k) are monotonically increasing, we calculate

$$\begin{aligned} \frac{\theta_{k+1}^2}{h_g} + \frac{\psi_{k+1}^2}{h_d} &= \frac{1}{h_g}(\theta_k - h_g \varphi_1'(\theta_k\psi_k)\psi_k)^2 + \frac{1}{h_d}(\psi_k + h_d \varphi_1'(\theta_k\psi_k)\theta_k)^2 \\ &= \frac{\theta_k^2}{h_g} + \frac{\psi_k^2}{h_d} + \varphi_1'(\theta_k\psi_k)^2(h_g\psi_k^2 + h_d\theta_k^2) \\ &> \frac{\theta_k^2}{h_g} + \frac{\psi_k^2}{h_d} \end{aligned} \tag{3.12}$$

□

The behavior of SimGD for the Dirac-GAN is visualized in Figure 3.2a. Similarly, for AltGD we have

Lemma 3.5. *For AltGD with n_g generator and n_d discriminator updates and learning rates*

h_g and h_d, the Jacobian of the update operator $F(\theta, \psi)$ has eigenvalues

$$\lambda_{1/2} = 1 - \frac{\alpha^2}{2} \pm \sqrt{\left(1 - \frac{\alpha^2}{2}\right)^2 - 1} \tag{3.13}$$

with $\alpha := \sqrt{n_g n_d h_g h_d}\, \varphi_1'(0)$ at the equilibrium. For $\alpha \leq 2$, all eigenvalues are hence on the unit circle. Moreover for $\alpha > 2$, there are eigenvalues outside the unit circle.

Proof. The update operators for AltGD are given by

$$F_G(\theta, \psi) = \begin{pmatrix} \theta - h_g \varphi_1'(\theta\psi)\psi \\ \psi \end{pmatrix} \tag{3.14}$$

$$F_D(\theta, \psi) = \begin{pmatrix} \theta \\ \psi + h_d \varphi_1'(\theta\psi)\theta \end{pmatrix} \tag{3.15}$$

The Jacobians of these operators at the equilibrium are given by

$$F_G'(0,0) = \begin{pmatrix} 1 & -h_g \varphi_1'(0) \\ 0 & 1 \end{pmatrix} \tag{3.16}$$

$$F_D'(0,0) = \begin{pmatrix} 1 & 0 \\ h_d \varphi_1'(0) & 1 \end{pmatrix} \tag{3.17}$$

As a result, the Jacobian of the combined update operator is

$$\begin{aligned} (F_D^{n_d} \circ F_G^{n_g})'(0,0) &= F_D'(0,0)^{n_d} \cdot F_G'(0,0)^{n_g} \\ &= \begin{pmatrix} 1 & -n_g h_g \varphi_1'(0) \\ n_d h_d \varphi_1'(0) & -n_g n_d h_g h_d \varphi_1'(0)^2 + 1 \end{pmatrix} \end{aligned} \tag{3.18}$$

An easy calculation shows that the eigenvalues of this matrix are

$$\lambda_{1/2} = 1 - \frac{\alpha^2}{2} \pm \sqrt{\left(1 - \frac{\alpha^2}{2}\right)^2 - 1} \tag{3.19}$$

with $\alpha = \sqrt{n_g n_d h_g h_d}\, |\varphi_1'(0)|$ which are on the unit circle if and only if $\alpha \leq 2$. Moreover for $\alpha > 2$ all eigenvalues are on the real axis and we have

$$1 - \frac{\alpha^2}{2} - \sqrt{\left(1 - \frac{\alpha^2}{2}\right)^2 - 1} \leq 1 - \frac{\alpha^2}{2} < -1 \tag{3.20}$$

which is outside the unit circle. □

Even though Lemma 3.5 shows that AltGD does not converge with linear rate to the Nash equilibrium, it could in principle converge with a sublinear convergence rate. However, this is very unlikely because – as Lemma 3.3 shows – even the continuous system does not converge. Indeed, we empirically find that AltGD oscillates in stable cycles around the

equilibrium and shows no sign of convergence (Figure 3.2b).

3.2 Where do instabilities come from?

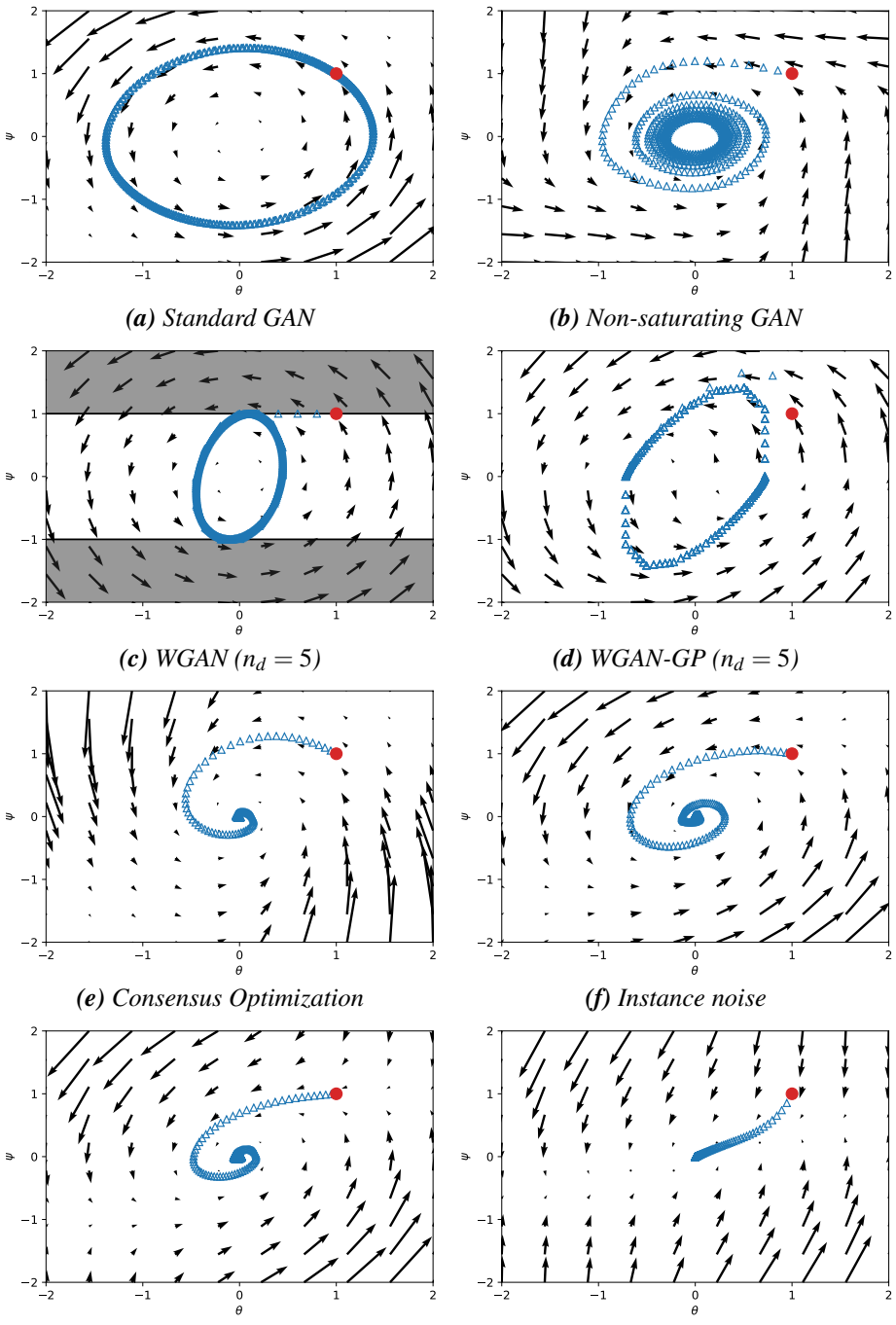
Our simple example shows that naive gradient-based GAN optimization does not always converge to the equilibrium point. To get a better understanding of what can go wrong for more complicated GANs, it is instructive to analyze these instabilities in depth for this simple example problem.

To understand the instabilities, we have to take a closer look at the oscillatory behavior that GANs exhibit both for the Dirac-GAN and for more complex systems. An intuitive explanation for the oscillations is given in Figure 3.1: when the generator is far from the true data distribution, the discriminator pushes the generator towards the true data distribution. At the same time, the discriminator becomes more certain, which increases the discriminator's slope (Figure 3.1a). Now, when the generator reaches the target distribution (Figure 3.1b), the slope of the discriminator is largest, pushing the generator away from the target distribution. As a result, the generator moves away again from the true data distribution and the discriminator has to change its slope from positive to negative. After a while, we end up with a similar situation as in the beginning of training, only on the other side of the true data distribution. This process repeats indefinitely and does not converge.

Another way to look at this is to consider the local behavior of the training algorithm near the Nash equilibrium. Indeed, near the Nash equilibrium, there is nothing that pushes the discriminator towards having zero slope on the true data distribution. Even if the generator is initialized *exactly* on the target distribution, there is no incentive for the discriminator to move to the equilibrium discriminator. As a result, training is unstable near the equilibrium.

This phenomenon of discriminator gradients orthogonal to the data distribution can also arise for more complex examples: as long as the data distribution is concentrated on a low dimensional manifold and the class of discriminators is big enough, there is no incentive for the discriminator to produce zero gradients orthogonal to the tangent space of the data manifold and hence converge to the equilibrium discriminator. Even if the generator produces *exactly* the true data distribution, there is no incentive for the discriminator to produce zero gradients orthogonal to the tangent space. When this happens, the discriminator does not provide useful gradients for the generator orthogonal to the data distribution and the generator does not converge.

Note that these instabilities can only arise if the true data distribution is concentrated on a lower dimensional manifold. Indeed, Nagarajan and Kolter [136] showed that - under some suitable assumptions - gradient descent-based GAN optimization is locally convergent for absolutely continuous distributions. Unfortunately, this assumption may not be satisfied for data distributions like natural images to which GANs are commonly applied [6]. Moreover, even if the data distribution is absolutely continuous but concentrated along some lower dimensional manifold, the eigenvalues of the Jacobian of the gradient vector field will be very close to the imaginary axis, resulting in a highly ill-conditioned problem. We observed this effect in Section 2.4.3 where we examined the spectrum of the Jacobian for a data distribution given by a circular mixture of Gaussian distributions with small variance.



(a) Standard GAN *(b) Non-saturating GAN*

(c) WGAN ($n_d = 5$) *(d) WGAN-GP ($n_d = 5$)*

(e) Consensus Optimization *(f) Instance noise*

(g) Gradient penalty *(h) Gradient penalty (CR)*

Figure 3.4: Regularization. *Convergence properties of different GAN training algorithms using AltGD with recommended number of discriminator updates per generator update ($n_d = 1$ if not noted otherwise). The shaded area in Figure 3.4c visualizes the set of forbidden values for the discriminator parameter ψ. The starting iterate is marked in red.*

3.3 Regularization

As we have seen in Section 3.1, unregularized GAN training does not always converge to the Nash equilibrium. In this section we discuss how several regularization techniques that have recently been proposed influence convergence of the Dirac-GAN.

3.3.1 Nonsaturating GAN

Especially in the beginning of training, the discriminator can reject samples produced by the generator with high confidence [62]. When this happens, the loss for the generator may saturate so that the generator receives almost no gradient information anymore.

To circumvent this problem Goodfellow et al. [62] introduced a nonsaturating objective for the generator. In nonsaturating GANs, the generator objective is replaced with[3]

$$\max_{\theta} \mathrm{E}_{x\sim p_\theta}\left[\varphi_1(-D_\psi(x))\right] \tag{3.21}$$

In our example, this is $\max_\theta \varphi_1(-\psi\theta)$.

While the nonsaturating generator objective was originally motivated by global stability considerations, we investigate its effect on local convergence. A linear analysis similar to normal GANs yields

Lemma 3.6. *The unique Nash equilibrium for the nonsaturating Dirac-GAN is given by $\theta^* = \psi^* = 0$. The eigenvalues of the Jacobian of the gradient vector field at the equilibrium are $\pm\varphi_1'(0)i$ which are both on the imaginary axis.*

Proof. The gradient vector field for the nonsaturating GAN is given by

$$v(\theta,\psi) = \begin{pmatrix} -\varphi_1'(-\theta\psi)\psi \\ \varphi_1'(\theta\psi)\theta \end{pmatrix} \tag{3.22}$$

As in the proof of Lemma 3.2, we see that $(\theta^*,\psi^*) = (0,0)$ defines the unique Nash equilibrium for the nonsaturating GAN.

Moreover, the Jacobian $v'(\theta,\psi)$ is

$$\begin{pmatrix} \varphi_1''(-\theta\psi)\psi^2 & -\varphi_1'(-\theta\psi) + \varphi_1''(-\theta\psi)\theta\psi \\ \varphi_1'(\theta\psi) + \varphi_1''(\theta\psi)\theta\psi & \varphi_1''(\theta\psi)\theta^2 \end{pmatrix} \tag{3.23}$$

At $\theta^* = \psi^* = 0$ we therefore have

$$v'(0,0) = \begin{pmatrix} 0 & -\varphi_1'(0) \\ \varphi_1'(0) & 0 \end{pmatrix} \tag{3.24}$$

with eigenvalues $\lambda_{1/2} = \pm\varphi_1'(0)i$. □

Lemma 3.6 implies that SimGD is not locally convergent for a nonsaturating GAN and any learning rate $h > 0$, because the eigenvalues of the Jacobian of the corresponding update

[3] Goodfellow et al. [62] used $\varphi_1(t) = \varphi_2(t) = -\log(1+\exp(-t))$.

operator F all have absolute value larger than one (Lemma 2.3). While Lemma 3.6 also rules out linear convergence towards the Nash equilibrium in the continuous case (i.e. for $h \to 0$), the continuous training dynamics could in principle still converge with a sublinear convergence rate. Indeed, we find this to be the case for the Dirac-GAN. We have

Lemma 3.7. *For every integral curve of the gradient vector field of the nonsaturating Dirac-GAN we have*

$$\frac{\mathrm{d}}{\mathrm{d}t}(\theta(t)^2 + \psi(t)^2) = 2\theta\psi\left[\varphi_1'(\theta\psi) - \varphi_1'(-\theta\psi)\right] \tag{3.25}$$

For concave φ_1, this is non-positive. Moreover, for $\varphi_1''(0) < 0$, the continuous training dynamics of the nonsaturating Dirac-GAN converge with logarithmic convergence rate.

Proof. The gradient vector field for the nonsaturating Dirac-GAN is given by

$$v(\theta, \psi) = \begin{pmatrix} -\varphi_1'(-\theta\psi)\psi \\ \varphi_1'(\theta\psi)\theta \end{pmatrix} \tag{3.26}$$

Hence, we have

$$\frac{\mathrm{d}}{\mathrm{d}t}(\theta(t)^2 + \psi(t)^2) = v_1(\theta,\psi)\theta + v_2(\theta,\psi)\psi = 2\theta\psi\left[\varphi_1'(\theta\psi) - \varphi_1'(-\theta\psi)\right] \tag{3.27}$$

For concave φ_1, we have

$$\frac{\varphi_1'(\theta\psi) - \varphi_1'(-\theta\psi)}{2\theta\psi} \leq 0 \tag{3.28}$$

and hence

$$\frac{\mathrm{d}}{\mathrm{d}t}(\theta(t)^2 + \psi(t)^2) = 4\theta^2\psi^2\,\frac{\varphi_1'(\theta\psi) - \varphi_1'(-\theta\psi)}{2\theta\psi} \leq 0 \tag{3.29}$$

Now assume that $\varphi_1''(0) < 0$. To intuitively understand why the continuous system converges with logarithmic convergence rate, we express the training dynamics in polar coordinates $(\theta, \psi) = (\sqrt{w}\cos(\phi), \sqrt{w}\sin(\phi))$. We have, by (3.27),

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}w &= 2\theta\psi\left[\varphi_1'(\theta\psi) - \varphi_1'(-\theta\psi)\right] \\ &= 4\varphi_1''(0)\theta^2\psi^2 + \mathcal{O}(|\theta\psi|^4) \\ &= \varphi_1''(0)w^2\sin^2(2\phi) + \mathcal{O}(w^4) \end{aligned} \tag{3.30}$$

Similarly, the temporal derivative of the angle ϕ is

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\phi &= \frac{1}{w}\begin{pmatrix} -\psi \\ \theta \end{pmatrix}^{\mathsf{T}}\begin{pmatrix} -\varphi_1'(-\theta\psi)\psi \\ \varphi_1'(\theta\psi)\theta \end{pmatrix} \\ &= \frac{\varphi_1'(\theta\psi)\theta^2 + \varphi_1'(-\theta\psi)\psi^2}{w} \\ &= \varphi_1'(0) + \mathcal{O}(w) \end{aligned} \tag{3.31}$$

When we ignore higher order terms, we can solve this system analytically[4] for ϕ and w:

$$\phi(t) = \varphi_1'(0)(t - t_0) \tag{3.33}$$

$$w(t) = \frac{2}{-\varphi_1''(0)t + \frac{\varphi_1''(0)}{4\varphi_1'(0)}\sin(4\varphi_1'(0)(t-t_0)) + c} \tag{3.34}$$

The training dynamics are hence convergent with logarithmic convergence rate $\mathcal{O}\left(\frac{1}{\sqrt{t}}\right)$.

For a more formal proof, first note that w is non-increasing by the first part of the proof. Moreover, for every $\varepsilon > 0$ there is $\delta > 0$ such that for $w < \delta$:

$$\varphi_1'(0) - \varepsilon \leq \frac{\mathrm{d}}{\mathrm{d}t}\phi \leq \varphi_1'(0) + \varepsilon \tag{3.35}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}w \leq (\varphi_1''(0)\sin^2(2\phi) + \varepsilon)w^2. \tag{3.36}$$

This implies that for every time interval $[0,T]$, $\phi(t)$ is in

$$\bigcup_{k\in\mathbb{Z}} \left[\frac{\pi}{8} + k\frac{\pi}{2}, \frac{3\pi}{8} + k\frac{\pi}{2}\right] \tag{3.37}$$

for t in a union of intervals $Q_T \subseteq [0,T]$ with total length at least $\beta\lfloor \alpha T \rfloor$ where

$$\alpha = \tfrac{2}{\pi}(\varphi_1'(0) - \varepsilon) \qquad \text{and} \qquad \beta = \tfrac{\pi}{4}\left(\varphi_1'(0) + \varepsilon\right)^{-1} \tag{3.38}$$

Importantly, for ε small enough, α, β are positive constants that are independent of T. For these $t \in Q_T$ we have $\sin^2(2\phi(t)) \geq \frac{1}{2}$. Because $\varphi_1''(0) < 0$, this shows

$$\frac{\mathrm{d}}{\mathrm{d}t}w(t) \leq \left(\frac{1}{2}\varphi_1''(0) + \varepsilon\right) w(t)^2 \tag{3.39}$$

for $t \in Q_T$ and ε small enough. Solving the right hand formally yields for some $c > 0$

$$w(t) \leq \frac{1}{-(\frac{1}{2}\varphi_1''(0) + \varepsilon)t + c} \tag{3.40}$$

As $w(t)$ is non-increasing for $t \notin Q_T$ and the total length of Q_T is at least $\beta\lfloor \alpha T \rfloor$ this shows that

$$w(T) \leq \frac{1}{-(\frac{1}{2}\varphi_1''(0) + \varepsilon)\beta\lfloor \alpha T \rfloor + c} \tag{3.41}$$

The training dynamics hence converge with logarithmic convergence rate $\mathcal{O}\left(\frac{1}{\sqrt{t}}\right)$. □

[4] For solving the ODE we use the *separation of variables*-technique and the identity

$$\int 2\sin^2(ax)\mathrm{d}x = x - \frac{\sin(2ax)}{2a}. \tag{3.32}$$

Note that the standard choice $\varphi_1(t) = -\log(1+\exp(-t))$ is concave and satisfies $\varphi_1''(0) = -\frac{1}{4} < 0$. Lemma 3.6 is hence applicable and shows that the continuous training dynamics of the nonsaturating Dirac-GAN converge for the standard choice of φ_1, albeit with an extremely slow convergence rate. However, this analysis relies strongly on the fact that we assumed that the data distribution is concentrated at zero and the proof of Lemma 3.7 does not transfer to the general case (i.e. $p_{\mathcal{D}} = p_{x_0}$) that we examine in Chapter 4. Nonetheless, Lemma 3.7 highlights the benefits of the non-saturating objective proposed by Goodfellow et al. [62] for the stability of GAN training. The training behavior of the nonsaturating GAN on our example problem is visualized in Figure 3.4b.

3.3.2 Wasserstein GAN

As discussed in Section 1.1, the two-player GAN game can be interpreted as minimizing a probabilistic divergence between the true data distribution and the distribution produced by the generator [62, 143]. This divergence is obtained by considering the best-response strategy for the discriminator, resulting in an objective function that only contains the generator parameters. Many recent regularization techniques for GANs are based on the observation [6] that this divergence may be discontinuous with respect to the parameters of the generator or may even take on infinite values if the support of the data distribution and the generator distribution do not match.

To make the divergence continuous with respect to the parameters of the generator, Arjovsky, Chintala, and Bottou [5] replace the Jensen-Shannon divergence used in the original derivation of GANs [62] with the Wasserstein divergence (Section 1.1.4), resulting in a model called Wasserstein GAN (WGAN). In practice, Arjovsky, Chintala, and Bottou [5] propose to use $\varphi_1(t) = \varphi_2(t) = t$ and restrict the class of discriminators to Lipschitz continuous functions with Lipschitz constant equal to 1. While a WGAN converges if the discriminator is always trained until convergence, in practice WGANs are usually trained by running only a fixed finite number of discriminator updates per generator update. However, near the Nash equilibrium the optimal discriminator parameters can have a discontinuity as a function of the current generator parameters: for the Dirac-GAN, the optimal discriminator has to move from $\psi = -1$ to $\psi = 1$ when θ changes signs. As the gradients get smaller near the equilibrium point, the gradient updates do not lead to convergence for the discriminator. Overall, the training dynamics are again determined by the Jacobian of the gradient vector field near the Nash equilibrium.

Lemma 3.8. *A WGAN trained with SimGD or AltGD with a fixed number of discriminator updates per generator update and learning rates $h_g, h_d > 0$ does generally not converge to the Nash equilibrium for the Dirac-GAN.*

Proof. Let us first consider SimGD. Assume that the iterates (θ_k, ψ_k) converge to the equilibrium point $(0,0)$. Note that $(\theta_{k+1}, \psi_{k+1}) \neq 0$ if $(\theta_k, \psi_k) \neq 0$. We can therefore assume without loss of generality that $(\theta_k, \psi_k) \neq 0$ for all $k \in \mathbb{N}$.

Because $\lim_{k\to\infty} \psi_k = 0$, there exists k_0 such that for all $k \geq k_0$ we have $|\psi_k| < 1$. For

$k \geq k_0$ we therefore have

$$\begin{pmatrix} \theta_{k+1} \\ \psi_{k+1} \end{pmatrix} = \begin{pmatrix} 1 & -h_g \\ h_d & 1 \end{pmatrix} \begin{pmatrix} \theta_k \\ \psi_k \end{pmatrix} \tag{3.42}$$

For $k \geq k_0$, the iterates are therefore given by

$$\begin{pmatrix} \theta_k \\ \psi_k \end{pmatrix} = A^{k-k_0} \begin{pmatrix} \theta_{k_0} \\ \psi_{k_0} \end{pmatrix} \quad \text{with} \quad A = \begin{pmatrix} 1 & -h_g \\ h_d & 1 \end{pmatrix} \tag{3.43}$$

However, the eigenvalues of A are given by $\lambda_{1/2} = 1 \pm \sqrt{h_g h_d} i$ which both have absolute value $\sqrt{1 + h_g h_d} > 1$. This contradicts the assumption that (θ_k, ψ_k) converges to $(0,0)$.

A similar argument also hold for AltGD. In this case, A is given by

$$\begin{pmatrix} 1 & 0 \\ h_d & 1 \end{pmatrix}^{n_d} \begin{pmatrix} 1 & -h_g \\ 0 & 1 \end{pmatrix}^{n_g} = \begin{pmatrix} 1 & -n_g h_g \\ n_d h_d & 1 - n_g n_d h_g h_d \end{pmatrix} \tag{3.44}$$

The eigenvalues of A as in (3.44) are given by

$$1 - \frac{n_g n_h h_g h_d}{2} \pm \sqrt{\left(1 - \frac{n_g n_d h_g h_d}{2}\right)^2 - 1} \tag{3.45}$$

At least one of these eigenvalues has absolute value greater or equal to one. Note that for almost all initial conditions (θ_0, ψ_0), the inner product between the eigenvector corresponding to the eigenvalue with absolute value bigger than one will be nonzero for all $k \in \mathbb{N}$. Since the recursion in (3.42) is linear, this contradicts the fact that $(\theta_k, \psi_k) \to (0,0)$, showing that AltGD generally does not converge to the Nash equilibrium either. □

The training behavior of the WGAN is visualized in Figure 3.4c. The convergence properties of WGANs were also considered by Nagarajan and Kolter [136] who showed that even for absolutely continuous densities and infinitesimal learning rates, WGANs are not always convergent.

3.3.3 Wasserstein GAN-GP

In practice, it can be hard to enforce the Lipschitz-constraint for WGANs. A practical solution to this problem was proposed by Gulrajani et al. [64], who derived a simple gradient penalty with a similar effect as the Lipschitz-constraint. The resulting training objective is commonly referred to as Wasserstein GAN with Gradient Penalties (WGAN-GP).

Similarly to WGANs, we find that WGAN-GP does not converge for the Dirac-GAN. A similar analysis also applies to the DRAGAN-regularizer proposed by Kodali et al. [98].

The regularizer proposed by Gulrajani et al. [64] is given by

$$R(\psi) = \frac{\gamma}{2} \mathrm{E}_{\hat{x}} \left(\|\nabla_x D_\psi(\hat{x})\| - 1 \right)^2 \tag{3.46}$$

where $\hat{x}$ is sampled uniformly on the line segment between two random points $x_1 \sim p_\theta$, $x_2 \sim p_\mathcal{D}$.

For the Dirac-GAN, this regularizer simplifies to

$$R(\psi) = \frac{\gamma}{2}(|\psi| - 1)^2 \tag{3.47}$$

The corresponding gradient vector field is given by

$$\tilde{v}(\theta, \psi) = \begin{pmatrix} -\psi \\ \theta - \text{sign}(\psi)\gamma(|\psi| - 1) \end{pmatrix} \tag{3.48}$$

Note that the gradient vector field has a discontinuity at the equilibrium point, as the gradient vector field takes on values with norm bigger than some fixed constant in every neighborhood of the equilibrium point. As a result, we have

Lemma 3.9. *WGAN-GP trained with SimGD or AltGD with a fixed number of generator and discriminator updates and learning rates $h_g, h_d > 0$, does generally not converge to the Nash equilibrium for the Dirac-GAN.*

Proof. First, consider SimGD. Assume that the iterates (θ_k, ψ_k) converge to the equilibrium point $(0,0)$. For almost all initial conditions[5] we have $(\theta_k, \psi_k) \neq (0,0)$ for all $k \in \mathbb{N}$. This implies

$$|\psi_{k+1} - \psi_k| = h_d|\theta_k - \gamma\psi_k - \text{sign}(\psi_k)| \tag{3.49}$$

and hence $\lim_{k\to\infty} |\psi_{k+1} - \psi_k| = h_d \neq 0$, showing that (θ_k, ψ_k) is not a Cauchy sequence. This contradicts the assumption that (θ_k, ψ_k) converges to the equilibrium point $(0,0)$.

A similar argument also holds for AltGD. □

The training behavior of WGAN-GP on our example problem is visualized in Figure 3.4d. Despite these negative results, WGAN-GP has been successfully applied in practice [64, 88] and we leave a theoretical analysis of these empirical results to future research.

3.3.4 Consensus optimization

In Section 2.4 we have derived Consensus Optimization [128]. Consensus Optimization is an algorithm that attempts to solve the problem of eigenvalues with zero real-part by introducing a regularization term that explicitly moves the eigenvalues to the left. The regularization term in Consensus Optimization is given by

$$R(\theta, \psi) = \frac{\gamma}{2}\|v(\theta, \psi)\|^2 = \frac{\gamma}{2}(\|\nabla_\theta \mathcal{L}(\theta, \psi)\|^2 + \|\nabla_\psi \mathcal{L}(\theta, \psi)\|^2) \tag{3.50}$$

As was shown in Section 2.4, Consensus Optimization converges locally for small learning rates $h > 0$. Indeed, for the Dirac-GAN we have

[5] Depending on γ and h modulo a set of measure 0.

Lemma 3.10. *For the Dirac-GAN, the eigenvalues of the Jacobian of the gradient vector field for Consensus Optimization at the equilibrium point are given by*

$$\lambda_{1/2} = -\gamma\varphi_1'(0)^2 \pm i\varphi_1'(0) \tag{3.51}$$

In particular, all eigenvalues have negative real part $-\gamma\varphi_1'(0)^2$. Hence, Consensus Optimization is locally convergent with linear convergence rate for small enough learning rates.

Proof. As shown in Section 2.4, the Jacobian of the modified vector field $\tilde{v}$ at the equilibrium point is

$$\tilde{v}'(0,0) = v'(0,0) - \gamma v'(0,0)^\intercal v'(0,0) \tag{3.52}$$

In our case, this is

$$\begin{pmatrix} -\gamma\varphi_1'(0)^2 & -\varphi_1'(0) \\ \varphi_1'(0) & -\gamma\varphi_1'(0)^2. \end{pmatrix} \tag{3.53}$$

A simple calculation shows that the eigenvalues of $\tilde{v}'(0,0)$ are given by

$$\lambda_{1/2} = -\gamma\varphi_1'(0)^2 \pm i\varphi_1'(0) \tag{3.54}$$

□

A visualization of Consensus Optimization for the Dirac-GAN is given in Figure 3.4e.

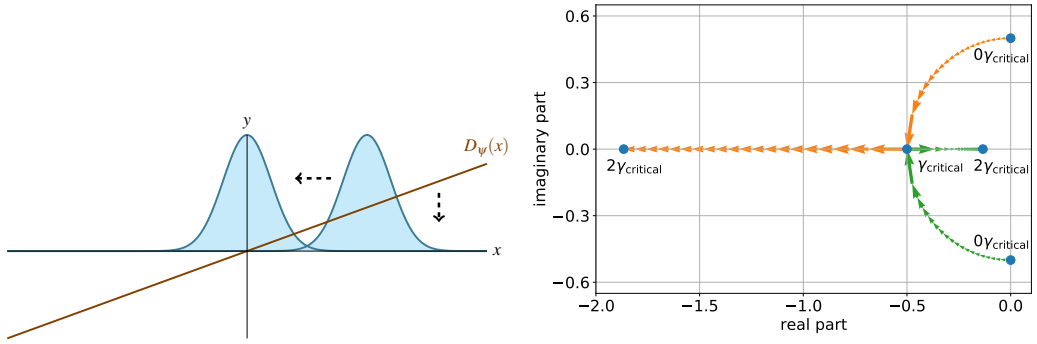
Unfortunately, Consensus Optimization has the drawback that it can introduce new spurious points of attraction to the GAN training dynamics. While this is usually not a problem for simple examples, it can be a problem for more complex ones like deep neural networks. In the next sections we therefore consider two alternative regularizers that make the dynamics exponentially stable but are better behaved far away from the equilibrium point.

3.3.5 Instance noise

As we have seen in the previous sections, unregularized GAN training is not always convergent. Moreover, we have also seen that WGAN [5] and WGAN-GP [64] do not always lead to convergence either. While Consensus Optimization makes the training dynamics locally convergent, it can also introduce new spurious points of attraction to the GAN training dynamics. These results raise the question if we can find regularizers that are better adapted to GAN training, but still lead to local convergence.

A common technique to stabilize GANs is to add *instance noise* [6, 177], i.e. independent Gaussian noise, to the data points. While the original motivation was to make the probabilistic divergence between data and generator distribution well-defined for distributions that do not have common support, this does not clarify the effects of instance noise on the *training algorithm* itself and its ability to find a Nash equilibrium. Interestingly, however, it was recently shown [136] that in the case of absolutely continuous distributions, gradient descent-based GAN optimization is - under suitable assumptions - locally convergent.

Indeed, for the Dirac-GAN we have:



(a) *Example with instance noise* ***(b)*** *Eigenvalues*

Figure 3.5: Instance Noise. *While unregularized GAN training is inherently unstable, instance noise can stabilize it. (a) Near the Nash equilibrium, the discriminator of the Dirac-GAN is pushed towards the zero discriminator. (b) As we increase the noise level σ from 0 to σ_{critical}, the real part of the eigenvalues at the equilibrium point becomes negative and the absolute value of the imaginary part becomes smaller. For noise levels bigger than σ_{critical} all eigenvalues are real-valued and GAN training hence behaves like a normal optimization problem.*

Lemma 3.11. *When using Gaussian instance noise with standard deviation σ, the eigenvalues of the Jacobian of the gradient vector field are given by*

$$\lambda_{1/2} = -\frac{\rho_2\sigma^2}{2} \pm \sqrt{\frac{\rho_2^2\sigma^4}{4} - \rho_1^2} \tag{3.55}$$

with $\rho_1 = \varphi_1'(0)$ and $\rho_2 = -\varphi_1''(0) - \varphi_2''(0)$. In particular, all eigenvalues of the Jacobian have negative real-part at the Nash equilibrium if (φ_1, φ_2) is strictly valid and $\sigma > 0$. Hence, SimGD and AltGD are both locally convergent for small enough learning rates.

Proof. When using instance noise, the GAN training objective (1.28) is given by

$$\mathrm{E}_{\tilde{\theta}\sim\mathcal{N}(\theta,\sigma^2)}\left[\varphi_1(\tilde{\theta}\psi)\right] + \mathrm{E}_{x\sim\mathcal{N}(0,\sigma^2)}\left[\varphi_2(-x\psi)\right] \tag{3.56}$$

The corresponding gradient vector field is hence given by

$$\tilde{v}(\theta,\psi) = \mathrm{E}_{\tilde{\theta},x}\left[\begin{pmatrix} -\psi\varphi_1'(\tilde{\theta}\psi) \\ \tilde{\theta}\varphi_1'(\tilde{\theta}\psi) - x\varphi_2'(-x\psi) \end{pmatrix}\right] \tag{3.57}$$

The Jacobian $\tilde{v}'(\theta,\psi)$ is therefore

$$\mathrm{E}_{\tilde{\theta},x}\left[\begin{pmatrix} -\varphi_1''(\tilde{\theta}\psi)\psi^2 & -\varphi_1'(\tilde{\theta}\psi) - \varphi_1''(\tilde{\theta}\psi)\tilde{\theta}\psi \\ \varphi_1'(\tilde{\theta}\psi) + \varphi_1''(\tilde{\theta}\psi)\tilde{\theta}\psi & \varphi_1''(\tilde{\theta}\psi)\tilde{\theta}^2 + x^2\varphi_2''(-x\psi) \end{pmatrix}\right] \tag{3.58}$$

Evaluating (3.58) at $(\theta^*, \psi^*) = (0,0)$ yields

$$\tilde{v}'(0,0) = \begin{pmatrix} 0 & -\rho_1 \\ \rho_1 & -\rho_2\sigma^2 \end{pmatrix} \tag{3.59}$$

with $\rho_1 = \varphi_1'(0) = \varphi_2'(0)$ and $\rho_2 = -\varphi_1''(0) - \varphi_2''(0)$. A straightforward calculation shows that the eigenvalues of $\tilde{v}'(0,0)$ are given by

$$\lambda_{1/2} = -\frac{\rho_2\sigma^2}{2} \pm \sqrt{\frac{\rho_2^2\sigma^4}{4} - \rho_1^2} \tag{3.60}$$

Note that by Definition 1.9 $\rho_1 \neq 0$ and by Lemma 1.12 $\rho_2 > 0$ if (φ_1, φ_2) is strictly valid. In this case, both λ_1 and λ_2 therefore have negative real part. □

Interestingly, Lemma 3.11 shows that there is a critical noise level given by $\sigma^2_{\text{critical}} = 2|\rho_1|/\rho_2$. If the noise level is smaller than the critical noise level, the eigenvalues of the Jacobian have non-zero imaginary part which results in a rotational component in the gradient vector field near the equilibrium point. If the noise level is larger than the critical noise level, all eigenvalues of the Jacobian become real-valued and the rotational component in the gradient vector field disappears. The optimization problem is best behaved when we select $\sigma = \sigma_{\text{critical}}$: in this case we can even achieve quadratic convergence for $h_g = h_d = |\rho_1|^{-1}$. The effect of instance noise on the eigenvalues is visualized in Figure 3.5b, which shows the traces of the two eigenvalues as we increase σ from 0 to $2\,\sigma_{\text{critical}}$.

Figure 3.4f shows the training behavior of the GAN with instance noise, showing that instance noise indeed creates a strong radial component in the gradient vector field which makes the training algorithm converge.

3.3.6 Zero-centered gradient penalties

Motivated by the success of instance noise to make the f-divergence between two distributions well-defined, Roth et al. [169] derived a local approximation to instance noise that results in a zero-centered[6] gradient penalty for the discriminator.

For the Dirac-GAN, a penalty on the squared norm of the gradients of the discriminator (no matter where) results in the regularizer

$$R(\psi) = \frac{\gamma}{2}\psi^2 \tag{3.61}$$

This regularizer does not include the weighting terms considered by Roth et al. [169]. However, the same analysis can also be applied to the regularizer with the additional weighting, yielding almost exactly the same results.[7]

[6] In contrast to the gradient regularizers used in WGAN-GP [64] and DRAGAN [98] which are not zero-centered.

[7] Please see Section 6.6.3 for an analysis.

Lemma 3.12. *When using the gradient regularizer from* (3.61) *for the discriminator, the eigenvalues of the Jacobian of the gradient vector field at the equilibrium point are given by*

$$\lambda_{1/2} = -\frac{\gamma}{2} \pm \sqrt{\frac{\gamma^2}{4} - \rho_1^2} \tag{3.62}$$

with $\rho_1 = \varphi_1'(0) = \varphi_2'(0)$. *In particular, for* $\gamma > 0$ *all eigenvalues have negative real part. Hence, SimGD and AltGD are both locally convergent for small enough learning rates.*

Proof. The regularized gradient vector field becomes

$$\tilde{v}(\theta, \psi) = \begin{pmatrix} -\varphi_1'(\theta\psi)\psi \\ \varphi_1'(\theta\psi)\theta - \gamma\psi \end{pmatrix} \tag{3.63}$$

The Jacobian $\tilde{v}'(\theta, \psi)$ is therefore given by

$$\begin{pmatrix} -\varphi_1''(\theta\psi)\psi^2 & -\varphi_1'(\theta\psi) - \varphi_1''(\theta\psi)\theta\psi \\ \varphi_1'(\theta\psi) + \varphi_1''(\theta\psi)\theta\psi & \varphi_1''(\theta\psi)\theta^2 - \gamma \end{pmatrix} \tag{3.64}$$

Evaluating (3.64) at $\theta^* = \psi^* = 0$ yields

$$\tilde{v}'(0,0) = \begin{pmatrix} 0 & -\rho_1 \\ \rho_1 & -\gamma \end{pmatrix} \tag{3.65}$$

whose eigenvalues are given by

$$\lambda_{1/2} = -\frac{\gamma}{2} \pm \sqrt{\frac{\gamma^2}{4} - \rho_1^2} \tag{3.66}$$

□

Like for instance noise, there is a critical regularization parameter $\gamma_{\text{critical}} = 2|\rho_1|$ that results in a locally rotation-free vector field. A visualization of the training behavior of the Dirac-GAN with gradient penalty is shown in Figure 3.4g. Figure 3.4h illustrates the training behavior of the GAN with gradient penalty and Critical Regularization (CR). In particular, we see that near the Nash equilibrium the vector field does not have a rotational component anymore and hence behaves like a normal optimization problem.

3.4 Conclusion

In this chapter we have introduced a minimal example of GAN training, which shows that neither Simultaneous Gradient Descent nor Alternating Gradient Descent converge for general (unregularized) GANs. However, as we have seen, for our example there is a simple solution to stabilize the training dynamics: both instance noise and zero-centered gradient penalties make the GAN training dynamics of the one-dimensional Dirac-GAN exponentially stable. Interestingly, both kinds of regularizer have the same effect on the

eigenvalues of the Jacobian of the gradient vector field. This raises the question if these regularizers can also stabilize GAN training for more complex examples. To answer this question we first generalize our results to higher dimensions in the next chapter. In Chapter 5 and 6, we apply our results to arbitrary GANs, finding that many properties of our simple example carry over to the general case.

4 The Multidimensional Dirac-GAN

In Chapter 3, we have introduced a simple one-dimensional example of GAN training, which we call the one-dimensional Dirac-GAN. In this chapter we generalize our results from Chapter 3 to higher dimensions. While analyzing the one-dimensional Dirac-GAN is instructive to better understand the training dynamics of GANs, even more insights can be gained by considering the multidimensional generalization. Indeed, when the data distribution for the Dirac-GAN is not concentrated at zero, the convergence behavior of the Dirac-GAN can only be fully understood in higher dimensions.[1]

Recall that the one-dimensional Dirac-GAN is defined by a data distribution $p_{\mathcal{D}} = \delta_0$, a generator $p_\theta = \delta_\theta$ and a linear discriminator $D_\psi(x) = \psi \cdot x$ where $\theta, \psi \in \mathbb{R}$. As we have seen in Chapter 3, we can analytically solve the resulting GAN training dynamics. This leads to important insights into the convergence behavior of GAN training. However, the one-dimensional Dirac-GAN is a very simple model and so far it is unclear if we can find a similar model for high-dimensional distributions (e.g. image distributions).

It turns out that this is possible. The one-dimensional Dirac-GAN from Chapter 3 can be easily generalized to multiple dimensions: in this case, the generator produces a Dirac distribution concentrated at a point $\theta \in \mathbb{R}^n$ and the discriminator is given by a linear function $D_\psi(x) = \psi^\mathsf{T} x$ with $\psi \in \mathbb{R}^n$. In addition, we would like to remove the requirement that the true data distribution is concentrated at 0. We therefore assume that the true data distribution is given by $p_{\mathcal{D}} = \delta_{x_0}$ for some $x_0 \in \mathbb{R}^n$.

In summary, we define the multidimensional Dirac-GAN as follows:

Definition 4.1. *The multidimensional* Dirac-GAN *consists of a generator distribution $p_\theta = \delta_\theta$ with $\theta \in \mathbb{R}^n$ and a linear discriminator $D_\psi(x) = \psi^\mathsf{T} x$ with $\psi \in \mathbb{R}^n$. The true data distribution $p_{\mathcal{D}}$ is given by a Dirac-distribution concentrated at $x_0 \in \mathbb{R}^n$.*

4.1 Unregularized Training

We first consider the case where the multidimensional Dirac-GAN is trained without additional regularization, as proposed in the original GAN paper [62]. In this case, the GAN training objective (1.28) becomes

$$\mathcal{L}(\theta, \psi) = \varphi_1(\psi^\mathsf{T}\theta) + \varphi_2(-\psi^\mathsf{T} x_0) \tag{4.1}$$

[1] Also see Nie and Patel [140] who used a multidimensional version of our example [129] as a starting point for their analysis of GAN training.

By computing the gradients with respect to θ and ψ, we obtain the gradient vector field

$$v(\theta,\psi) = \begin{pmatrix} -\varphi_1'(\psi^\mathsf{T}\theta)\,\psi \\ \varphi_1'(\psi^\mathsf{T}\theta)\,\theta - \varphi_2'(-\psi^\mathsf{T}x_0)\,x_0 \end{pmatrix} \tag{4.2}$$

We first consider existence and uniqueness of Nash-equilibria for the multidimensional Dirac-GAN.

Lemma 4.2. *Assume that (φ_1,φ_2) is strictly valid (Definition 1.9) and consider $(\theta^*,\psi^*) = (x_0,0)$. Then (θ^*,ψ^*) is the unique (pure) Nash equilibrium of the GAN-game where the generator and discriminator try to minimize and maximize* (4.1)*, respectively*

Proof. Obviously, for $(\theta^*,\psi^*) = (x_0,0)$ we have $p_{\theta^*} = p_{\mathcal{D}}$ and $D_{\psi^*} = \chi_0$. By Lemma 1.14, (θ^*,ψ^*) hence defines a Nash equilibrium.

To see that the equilibrium is unique, we apply Lemma 1.15. To this end, we use that $\nabla_\psi D_\psi(x) = x$. Hence, for $\theta \neq x_0$:

$$\mathrm{E}_{\,x\sim p_{\mathcal{D}}}\left[\nabla_\psi D_\psi(x)\big|_{\psi=\psi^*}\right] = x_0 \neq \theta = \mathrm{E}_{\,x\sim p_\theta}\left[\nabla_\psi D_\psi(x)\big|_{\psi=\psi^*}\right] \tag{4.3}$$

Lemma 1.15 is hence applicable, showing that every Nash equilibrium (θ_{NE},ψ_{NE}) satisfies $p_{\theta_{NE}} = p_{\mathcal{D}}$ and $D_{\psi_{NE}}(x) = 0$ for $x \in \operatorname{supp}\ p_{\mathcal{D}}$, i.e. $\theta_{NE} = x_0$ and $\psi_{NE}^\mathsf{T} x_0 = 0$. As a result, the gradient vector field (4.2) at (θ_{NE},ψ_{NE}) is

$$v(\theta_{NE},\psi_{NE}) = \begin{pmatrix} -\varphi_1'(0)\,\psi_{NE} \\ 0 \end{pmatrix} \tag{4.4}$$

Since (θ_{NE},ψ_{NE}) is a Nash equilibrium, we have $v(\theta_{NE},\psi_{NE}) = 0$ by Lemma 2.1 and therefore, using $\varphi_1'(0) \neq 0$, $\psi_{NE} = 0$. This shows that $(\theta^*,\psi^*) = (x_0,0)$ is the unique Nash equilibrium. □

As before, we can understand the local convergence behavior of Simultaneous Gradient Descent (SimGD) and Alternating Gradient Descent (AltGD) by looking at the Jacobian of the gradient vector field (4.2) at the equilibrium point. To this end, we again define $\rho_1 = \varphi_1'(0) = \varphi_2'(0)$ and $\rho_2 = -\varphi_1''(0) - \varphi_2''(0)$.

Lemma 4.3. *The Jacobian at the equilibrium point $(\theta^*,\psi^*) = (x_0,0)$ for the multidimensional Dirac-GAN is given by:*

$$v'(\theta^*,\psi^*) = \begin{pmatrix} 0 & -\rho_1 I \\ \rho_1 I & -\rho_2\, x_0 x_0^\mathsf{T} \end{pmatrix} \tag{4.5}$$

with $\rho_1 = \varphi_1'(0) = \varphi_2'(0)$ and $\rho_2 = -\varphi_1''(0) - \varphi_2''(0)$. The Jacobian has $n-1$ eigenvalues at $\rho_1\, i$ and $-\rho_1\, i$ respectively. The other two eigenvalues are

$$-\frac{\rho_2\|x_0\|^2}{2} \pm \sqrt{\frac{\rho_2^2\|x_0\|^4}{4} - \rho_1^2} \tag{4.6}$$

In particular, the system is exponentially stable if and only if $n = 1$, $\rho_2 > 0$ *and* $x_0 \neq 0$.

Proof. First, by taking the derivative of (4.2), we obtain the following expression for the Jacobian $v'(\theta, \psi)$:

$$\begin{pmatrix} -\varphi_1''(\psi^\mathsf{T}\theta)\,\psi\psi^\mathsf{T} & -\varphi_1'(\psi^\mathsf{T}\theta)\,I - \varphi_1''(\psi^\mathsf{T}\theta)\,\psi\theta^\mathsf{T} \\ \varphi_1'(\psi^\mathsf{T}\theta)\,I + \varphi_1''(\psi^\mathsf{T}\theta)\,\theta\psi^\mathsf{T} & \varphi_1''(\psi^\mathsf{T}\theta)\,\theta\theta^\mathsf{T} + \varphi_2''(-\psi^\mathsf{T}x_0)\,x_0x_0^\mathsf{T} \end{pmatrix} \tag{4.7}$$

Evaluating this expression at $(\theta^*, \psi^*) = (x_0, 0)$ yields

$$v'(\theta^*, \psi^*) = \begin{pmatrix} 0 & -\rho_1 I \\ \rho_1 I & -\rho_2 x_0 x_0^\mathsf{T} \end{pmatrix} \tag{4.8}$$

with $\rho_1 = \varphi_1'(0) = \varphi_2'(0)$ and $\rho_2 = -\varphi_1''(0) - \varphi_2''(0)$.

Let $J := v'(\theta^*, \psi^*)$. To compute the eigenvalues of J, we compute the characteristic polynomial $\chi(\lambda)$:

$$\chi(\lambda) = \det(\lambda I - J) = \det \begin{pmatrix} \lambda I & \rho_1 I \\ -\rho_1 I & \lambda I + \rho_2 x_0 x_0^\mathsf{T} \end{pmatrix} \tag{4.9}$$

Assume for now that $\lambda \neq 0$. Using Lemma A.1 about the determinant of block matrices, we obtain

$$\begin{aligned} \chi(\lambda) &= \det(\lambda I) \det\left(\left(\lambda + \frac{\rho_1^2}{\lambda}\right) I + \rho_2 x_0 x_0^\mathsf{T}\right) \\ &= \det\left((\lambda^2 + \rho_1^2) I + \rho_2 \lambda\, x_0 x_0^\mathsf{T}\right) \end{aligned} \tag{4.10}$$

Let $R := (\lambda^2 + \rho_1^2) I$. Then

$$\det(R) = (\lambda^2 + \rho_1^2)^n \qquad \text{and} \qquad x_0^T R^{-1} x_0 = \frac{\|x_0\|^2}{\lambda^2 + \rho_1^2} \tag{4.11}$$

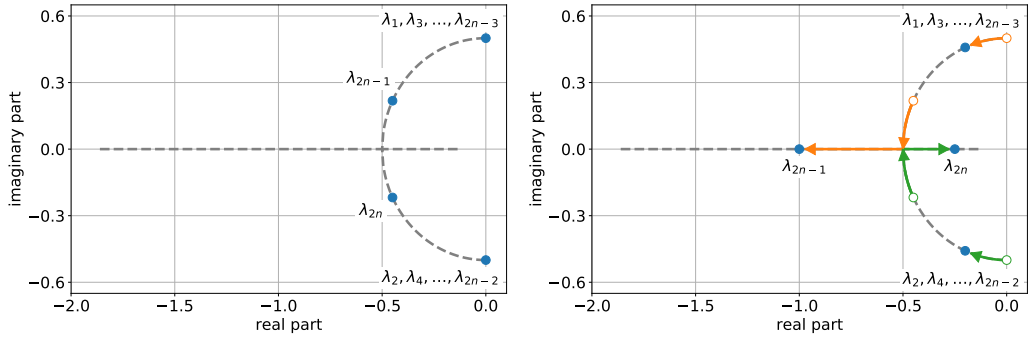
By the matrix determinant lemma (Lemma A.2), we therefore obtain

$$\begin{aligned} \chi(\lambda) &= (1 + \rho_2 \lambda\, x_0^\mathsf{T} R^{-1} x_0) \det(R) \\ &= \left(1 + \frac{\rho_2 \|x_0\|^2}{\lambda^2 + \rho_1^2}\,\lambda\right) (\lambda^2 + \rho_1^2)^n \\ &= \xi(\lambda)(\lambda^2 + \rho_1^2)^{n-1} \end{aligned} \tag{4.12}$$

where

$$\xi(\lambda) = \lambda^2 + \rho_2 \|x_0\|^2 \lambda + \rho_1^2 \tag{4.13}$$

By continuity, this result also holds for $\lambda = 0$. The Jacobian $v'(\theta^*, \psi^*)$ hence has $n-1$ eigenvalues at $-\rho_1 i$ and $+\rho_1 i$, respectively. The other two eigenvalues are given by the



(a) *Unregularized* ***(b)*** *Regularized*

Figure 4.1: Eigenvalues. *(a) The Jacobian of the multidimensional Dirac-GAN has $2n-2$ eigenvalues on the imaginary axis and two eigenvalues in the left half plane. (b) When we use instance noise or zero-centered gradient penalties, all eigenvalues are in the left half plane. As we increase the noise level / regularization parameter, the eigenvalues move along a circle until they hit the real axis. Afterwards, the eigenvalues move in opposite directions along the real axis.*

zeros of $\xi(\lambda)$, i.e.

$$-\frac{\rho_2\|x_0\|^2}{2} \pm \sqrt{\frac{\rho_2^2\|x_0\|^4}{4} - \rho_1^2} \tag{4.14}$$

□

The eigenvalues of the Jacobian $v'(\theta^*, \psi^*)$ for the multidimensional Dirac-GAN are visualized in Figure 4.1a.

As discussed in Section 3.3.1, in practice GANs are often trained with a nonsaturating loss for the generator [62]. In this case, the gradient vector field (4.2) becomes

$$v(\theta, \psi) = \begin{pmatrix} -\varphi_1'(-\psi^\mathsf{T}\theta)\,\psi \\ \varphi_1'(\psi^\mathsf{T}\theta)\,\theta - \varphi_2'(-\psi^\mathsf{T}x_0)\,x_0 \end{pmatrix} \tag{4.15}$$

As it turns out, near the equilibrium point the training dynamics of the nonsaturating GAN behaves similarly to the unmodified case:

Remark 4.4. *The results from Lemma 4.3 also hold for the non-saturating case* (4.15).

Proof. The Jacobian $v'(\theta, \psi)$ of the modified gradient vector field in (4.15) is

$$\begin{pmatrix} \varphi_1''(-\psi^\mathsf{T}\theta)\,\psi\psi^\mathsf{T} & -\varphi_1'(-\psi^\mathsf{T}\theta)\,I + \varphi_1''(-\psi^\mathsf{T}\theta)\,\psi\theta^\mathsf{T} \\ \varphi_1'(\psi^\mathsf{T}\theta)\,I + \varphi_1''(\psi^\mathsf{T}\theta)\,\theta\psi^\mathsf{T} & \varphi_1''(\psi^\mathsf{T}\theta)\,\theta\theta^\mathsf{T} + \varphi_2''(-\psi^\mathsf{T}x_0)\,x_0x_0^\mathsf{T} \end{pmatrix} \tag{4.16}$$

Evaluating at $(\theta^*, \psi^*) = (x_0, 0)$ yields

$$v'(\theta^*, \psi^*) = \begin{pmatrix} 0 & -\rho_1 I \\ \rho_1 I & -\rho_2 x_0 x_0^\mathsf{T} \end{pmatrix} \tag{4.17}$$

with $\rho_1 = \varphi_1'(0) = \varphi_2'(0)$ and $\rho_2 = -\varphi_1''(0) - \varphi_2''(0)$. In particular, the Jacobian in (4.17) is equal to the Jacobian (4.5) and hence also has the same eigenvalues. □

Interestingly, Lemma 4.3 shows that the one-dimensional Dirac-GAN is exponentially stable for $x_0 \neq 0$. However, for $n \geq 2$, this is no longer the case. We hence see that the case $x_0 \neq 0$ can only be understood when we generalize the one-dimensional Dirac-GAN to higher dimensions.

4.2 Regularized Training

In the last section we have seen that the gradient vector field of the unregularized multidimensional Dirac-GAN has eigenvalues on the imaginary axis when $n \geq 2$. Unregularized training of the multidimensional Dirac-GAN is therefore generally not stable. We have also seen that multidimensional case provides new insights regarding the case where the true data distribution is not concentrated at zero. We now consider multidimensional generalizations of the regularizers that have stabilized the one-dimensional Dirac-GAN: instance noise and zero-centered gradient penalties.

4.2.1 Instance Noise

Let us first consider the case where we add isotropic Gaussian noise to both the generator and the true data distribution. As in Section 3, we see that the regularized gradient vector field is given by

$$\tilde{v}(\theta, \psi) = \mathrm{E}_{\tilde{\theta} \sim \mathcal{N}(\theta, \sigma^2 I), x \sim \mathcal{N}(x_0, \sigma^2 I)} \begin{pmatrix} -\varphi_1'(\psi^\mathsf{T} \tilde{\theta})\, \psi \\ \varphi_1'(\psi^\mathsf{T} \tilde{\theta})\, \tilde{\theta} - \varphi_2'(-\psi^\mathsf{T} x)\, x \end{pmatrix} \tag{4.18}$$

Lemma 4.5. *When using isotropic instance noise with standard deviation σ, the Jacobian of the regularized gradient vector field* (4.18) *at the equilibrium $(\theta^*, \psi^*) = (x_0, 0)$ is*

$$\tilde{v}'(\theta^*, \psi^*) = \begin{pmatrix} 0 & -\rho_1 I \\ \rho_1 I & -\rho_2 \left(\sigma^2 I + x_0 x_0^\mathsf{T}\right) \end{pmatrix} \tag{4.19}$$

The Jacobian has $2n-2$ eigenvalues at

$$\frac{-\rho_2 \sigma^2}{2} \pm \sqrt{\frac{-\rho_2^2 \sigma^4}{2} - \rho_1^2} \tag{4.20}$$

Moreover, the other two eigenvalues are

$$-\frac{\rho_2(\sigma^2 + \|x_0\|^2)}{2} \pm \sqrt{\frac{\rho_2^2(\sigma^2 + \|x_0\|^2)^2}{4} - \rho_1^2} \tag{4.21}$$

Proof. The Jacobian of $\tilde{v}'(\theta,\psi)$ is

$$\mathrm{E}_{\tilde{\theta},x}\begin{pmatrix} -\varphi_1''(\psi^\mathsf{T}\tilde{\theta})\,\psi\psi^\mathsf{T} & -\varphi_1'(\psi^\mathsf{T}\tilde{\theta})\,I-\varphi_1''(\psi^\mathsf{T}\tilde{\theta})\,\psi\tilde{\theta}^\mathsf{T} \\ \varphi_1'(\psi^\mathsf{T}\tilde{\theta})\,I+\varphi_1''(\psi^\mathsf{T}\tilde{\theta})\,\tilde{\theta}\,\psi^\mathsf{T} & \varphi_2''(\psi^\mathsf{T}\tilde{\theta})\,\tilde{\theta}\tilde{\theta}^\mathsf{T}+\varphi_2''(-\psi^\mathsf{T}x)\,xx^\mathsf{T} \end{pmatrix} \tag{4.22}$$

where $\tilde{\theta}\sim\mathcal{N}(\theta,\sigma^2 I)$ and $x\sim\mathcal{N}(x_0,\sigma^2 I)$. Evaluating at $(\theta^*,\psi^*)=(x_0,0)$ yields

$$\tilde{v}'(\theta^*,\psi^*)=\begin{pmatrix} 0 & -\rho_1 I \\ \rho_1 I & -\rho_2\left(\sigma^2 I+x_0x_0^\mathsf{T}\right) \end{pmatrix} \tag{4.23}$$

As in the proof of Lemma 4.3, we apply Lemma A.1 to obtain the characteristic polynomial:

$$\begin{aligned}\chi(\lambda)&=\det\begin{pmatrix} \lambda I & \rho_1 I \\ -\rho_1 I & \lambda I+\rho_2\left(\sigma^2 I+x_0x_0^\mathsf{T}\right)\end{pmatrix}\\ &=\det(\lambda I)\det\left(\left(\lambda+\frac{\rho_1^2}{\lambda}\right)I+\rho_2\left(\sigma^2 I+x_0x_0^\mathsf{T}\right)\right)\\ &=\det\left((\lambda^2+\rho_2\sigma^2\lambda+\rho_1^2)I+\rho_2\lambda\,x_0x_0^\mathsf{T}\right)\end{aligned} \tag{4.24}$$

Now, let

$$R:=(\lambda^2+\rho_2\sigma^2\lambda+\rho_1^2)I \tag{4.25}$$

A straightforward calculation shows that

$$\det(R)=(\lambda^2+\rho_2\sigma^2\lambda+\rho_1^2)^n \tag{4.26}$$

$$x_0^T R^{-1}x_0=\frac{\|x_0\|^2}{\lambda^2+\rho_2\sigma^2\lambda+\rho_1^2} \tag{4.27}$$

By the matrix determinant lemma (Lemma A.2), we obtain

$$\begin{aligned}\chi(\lambda)&=(1+\rho_2\lambda\,x_0^\mathsf{T}R^{-1}x_0)\det(R)\\ &=\left(1+\frac{\rho_2\|x_0\|^2\lambda}{\lambda^2+\rho_2\sigma^2\lambda+\rho_1^2}\right)(\lambda^2+\rho_2\sigma^2\lambda+\rho_1^2)^n\\ &=\xi(\lambda)(\lambda^2+\rho_2\sigma^2\lambda+\rho_1^2)^{n-1}\end{aligned} \tag{4.28}$$

with

$$\xi(\lambda)=\lambda^2+\rho_2(\sigma^2+\|x_0\|^2)\lambda+\rho_1^2 \tag{4.29}$$

The Jacobian $v'(\theta^*,\psi^*)$ hence has $2n-2$ eigenvalues at

$$\frac{-\rho_2\sigma^2}{2}\pm\sqrt{\frac{-\rho_2^2\sigma^4}{2}-\rho_1^2} \tag{4.30}$$

The other two eigenvalues are given by the zeros of $\xi(\lambda)$, i.e.

$$-\frac{\rho_2(\sigma^2+\|x_0\|^2)}{2} \pm \sqrt{\frac{\rho_2^2(\sigma^2+\|x_0\|^2)^2}{4} - \rho_1^2} \tag{4.31}$$

□

The eigenvalues of the Jacobian $v'(\theta^*, \psi^*)$ for the multidimensional Dirac-GAN with instance noise are visualized in Figure 4.1b.

So far, we have only considered the case where we add isotropic white Gaussian instance noise with standard deviation σ to both the true data distribution and the generator distribution. However, it is also interesting to consider the case where we add anisotropic noise to the distributions. Adapting the proof of Lemma 4.5, we obtain

Lemma 4.6. *When we add anisotropic Gaussian instance noise with covariance matrix Σ both to the generator and data distribution, i.e. $\tilde{\theta} \sim \mathcal{N}(\theta, \Sigma)$ and $x \sim \mathcal{N}(x_0, \Sigma)$, the Jacobian of gradient vector field at $(\theta^*, \psi^*) = (x_0, 0)$ is given by*

$$\tilde{v}'(\theta^*, \psi^*) = \begin{pmatrix} 0 & -\rho_1 I \\ \rho_1 I & -\rho_2 \left(\Sigma + x_0 x_0^\mathsf{T}\right) \end{pmatrix} \tag{4.32}$$

The eigenvalues of $\tilde{v}'(\theta^*, \psi^*)$ are

$$-\frac{\rho_2 \mu_k}{2} \pm \sqrt{\frac{\rho_2^2 \mu_k^2}{4} - \rho_1^2} \tag{4.33}$$

where μ_k $(k = 1, \ldots, n)$ denote the eigenvalues of $\Sigma + x_0 x_0^\mathsf{T}$.

Proof. As in the proof of Lemma 4.5, we see that the Jacobian at the equilibrium point is

$$\tilde{v}'(\theta^*, \psi^*) = \begin{pmatrix} 0 & -\rho_1 I \\ \rho_1 I & -\rho_2 \left(\Sigma + x_0 x_0^\mathsf{T}\right) \end{pmatrix} \tag{4.34}$$

Moreover, using Lemma A.1 about the determinant of block matrices, we see

$$\begin{aligned} \chi(\lambda) &= \det(\lambda I) \det\left(\left(\lambda + \frac{\rho_1^2}{\lambda}\right) I + \rho_2 \left(\Sigma + x_0 x_0^\mathsf{T}\right)\right) \\ &= \det\left((\lambda^2 + \rho_1^2) I + \rho_2 \lambda \left(\Sigma + x_0 x_0^\mathsf{T}\right)\right) \end{aligned} \tag{4.35}$$

Let $\chi_{-\rho_2\lambda\left(\Sigma + x_0 x_0^\mathsf{T}\right)}$ denote the characteristic polynomial of $-\rho_2\lambda\left(\Sigma + x_0 x_0^\mathsf{T}\right)$. We can then rewrite (4.35) as

$$\chi(\lambda) = \chi_{-\rho_2\lambda\left(\Sigma + x_0 x_0^\mathsf{T}\right)}\left(\lambda^2 + \rho_1^2\right) \tag{4.36}$$

Hence, the eigenvalues of $v'(\theta^*, \psi^*)$ can be calculated by solving

$$\lambda^2 + \rho_2 \mu \lambda + \rho_1^2 = 0 \tag{4.37}$$

for all eigenvalues μ of $\Sigma + x_0 x_0^\mathsf{T}$. This shows that the eigenvalues λ are given by

$$-\frac{\rho_2 \mu_k}{2} \pm \sqrt{\frac{\rho_2^2 \mu_k^2}{4} - \rho_1^2} \tag{4.38}$$

with μ_k ($k = 1, \dots, n$) the eigenvalues of $\Sigma + x_0 x_0^\mathsf{T}$. □

When we use anisotropic instance noise, the eigenvalues hence behave similarly to the isotropic case, visualized in Figure 4.1b. However, in this case the eigenvalues lie at different locations on the dashed gray line.

4.2.2 Zero-centered gradient penalties

So far, we have considered instance noise as a regularizer for the multidimensional Dirac-GAN. However, introducing noise into GAN training can make training less stable and also leads to slower convergence. In particular, Lemma 4.5 shows that $\nabla^2_\psi \mathcal{L}(\theta^*, \psi^*)$ is ill-conditioned[2] if σ^2 is much smaller than $\|x_0\|^2$. That means that in practice we have to choose $\sigma \approx \|x_0\|$ to obtain a convergent system. However, in high-dimensional spaces this becomes quickly intractable, as σ^2 is the noise level added to each coordinate whereas $\|x_0\|^2$ is the sum of the squared coordinates and therefore scales linearly with the dimensionality of x_0. While instance noise hence leads to a stable system in the deterministic case, the amount of instance noise required to stabilize the system can quickly become intractable in high-dimensional spaces.

Similarly as we did for the one-dimensional Dirac-GAN in Section 3.3.6, we therefore consider zero-centered gradient penalties, a deterministic regularizer, as an alternative. Because zero-centered gradient penalties are deterministic, they do not introduce additional noise into the training.

As we have seen in Section 3.3.6, zero-centered gradient penalties have the same effect on the local convergence behavior of the one-dimensional Dirac-GAN as instance noise. As it turns out, this statement generalizes to higher dimensions. When we use zero-centered gradient penalties, the gradient vector field (2.2) is given by

$$\tilde{v}(\theta, \psi) = \begin{pmatrix} -\varphi_1'(\psi^\mathsf{T}\theta)\psi \\ \varphi_1'(\psi^\mathsf{T}\theta)\theta - \varphi_2'(-\psi^\mathsf{T} x_0)x_0 - \gamma\psi \end{pmatrix} \tag{4.39}$$

Calculating the Jacobian and its eigenvalues at the equilibrium point is now straightforward:

Lemma 4.7. *When using zero-centered gradient penalties with regularization parameter γ, the Jacobian of the gradient vector field* (4.39) *at* $(\theta^*, \psi^*) = (x_0, 0)$ *is*

$$\tilde{v}'(\theta^*, \psi^*) = \begin{pmatrix} 0 & -\rho_1 I \\ \rho_1 I & -\gamma I - \rho_2 x_0 x_0^\mathsf{T} \end{pmatrix} \tag{4.40}$$

[2] We analyze the conditioning of GANs in depth in Chapter 7.

The Jacobian has $2n-2$ eigenvalues at

$$-\frac{\gamma}{2} \pm \sqrt{\frac{\gamma^2}{4} - \rho_1^2} \tag{4.41}$$

Moreover, the other two eigenvalues are

$$-\frac{\gamma + \rho_2 \|x_0\|^2}{2} \pm \sqrt{-\frac{(\gamma + \rho_2 \|x_0\|^2)^2}{4} - \rho_1^2} \tag{4.42}$$

Proof. The proof is similar to the proof of Lemma 4.5. In this case,

$$R := (\lambda^2 + \gamma\lambda + \rho_1^2) I \tag{4.43}$$

and therefore

$$\det(R) = (\lambda^2 + \gamma\lambda + \rho_1)^n \tag{4.44}$$

$$x_0^T R^{-1} x_0 = \frac{\|x_0\|^2}{\lambda^2 + \gamma\lambda + \rho_1^2} \tag{4.45}$$

Hence,

$$\begin{aligned} \chi(\lambda) &= (1 + \rho_2 \lambda\, x_0^\mathsf{T} R^{-1} x_0) \det(R) \\ &= \left(1 + \frac{\rho_2 \|x_0\|^2}{\lambda^2 + \gamma\lambda + \rho_1^2}\, \lambda\right) (\lambda^2 + \gamma\lambda + \rho_1^2)^n \\ &= \xi(\lambda)(\lambda^2 + \gamma\lambda + \rho_1^2)^{n-1} \end{aligned} \tag{4.46}$$

with

$$\xi(\lambda) = \lambda^2 + (\gamma + \rho_2 \|x_0\|^2)\lambda + \rho_1^2 \tag{4.47}$$

The Jacobian $v'(\theta^*, \psi^*)$ therefore has $2n-2$ eigenvalues at

$$-\frac{\gamma}{2} \pm \sqrt{\frac{\gamma^2}{4} - \rho_1^2} \tag{4.48}$$

The other two eigenvalues are again given by the zeros of $\xi(\lambda)$, i.e.

$$-\frac{\gamma + \rho_2 \|x_0\|^2}{2} \pm \sqrt{\frac{(\gamma + \rho_2 \|x_0\|^2)^2}{4} - \rho_1^2} \tag{4.49}$$

□

Lemma 4.7 shows that zero-centered gradient penalties have the same effect on local stability of the multidimensional Dirac-GAN as instance noise (Figure 4.1b). However, in contrast to instance noise, zero-centered gradient penalties are a deterministic regularizer and therefore do not have the same drawbacks as instance noise which introduces additional noise into the training.

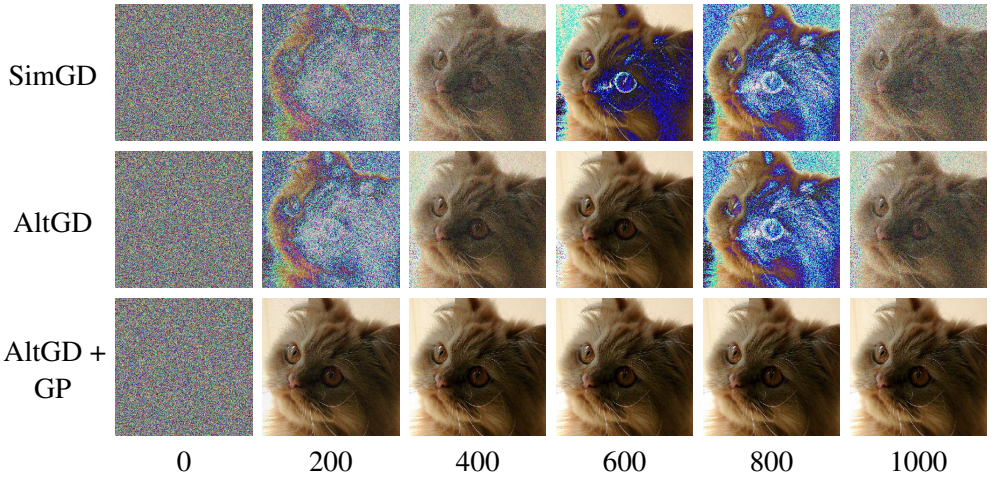


Figure 4.2: Multidimensional Dirac-GAN: *Qualitative results for the multidimensional Dirac-GAN over the number of iterations. While unregularized GAN training does not converge, training with zero-centered gradient penalties leads to convergence.*

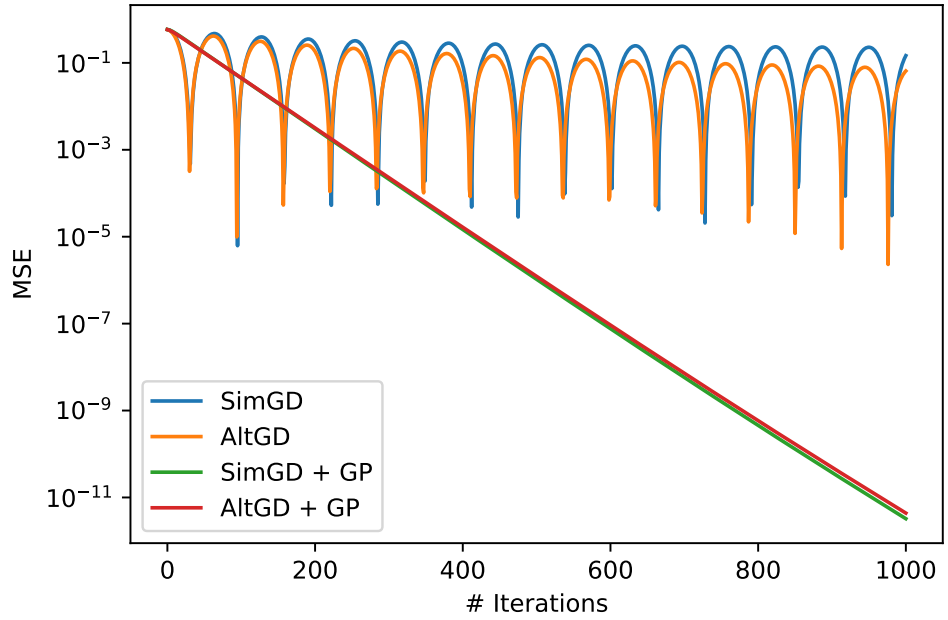


Figure 4.3: Multidimensional Dirac-GAN: *Quantitative results for the multidimensional Dirac-GAN. The graph shows the Mean Squared Error (MSE) between the target image x_0 and the generated sample θ over the number of iterations. While unregularized GAN training does not converge, training with zero-centered gradient penalties leads to convergence.*

4.3 Simulation

In this section we experimentally test our results from the previous sections. To this end, we consider a multidimensional Dirac-GAN where x_0 is given by an RGB image of resolution 256×256. The dimensionality of the problem is hence $n = 256 \cdot 256 \cdot 3 \approx 197k$. In Section 3.3.6 we have seen that $\|x_0\|^2$ has a strong effect on the conditioning of the system. We therefore rescale the pixel values[3] of x_0 to $[0, 1/256]$, so that $\|x_0\|^2 \in [0,1]$. For all experiments, we use learning rates of $h_g = h_d = 0.1$ for the generator and discriminator, respectively. For regularized training, we use zero-centered gradient penalties with $\gamma = 1$. We initialize the discriminator weights ψ with zeros and the generator weights θ with random numbers in $[0, 1/256]$.

Figure 4.2 shows a qualitative comparison of unregularized and regularized training of the multidimensional Dirac-GAN. We clearly see that neither SimGD nor AltGD converge to the true data distribution. In contrast, SimGD and AltGD with zero-centered gradient penalties exhibit fast convergence to the Nash equilibrium: even after 200 iterations, the resulting image is visually not distinguishable from the target image. Figure 4.3 shows the Mean Squared Error (MSE) between the target image x_0 and the image θ produced by the generator. We clearly see that unregularized training does not converge. In contrast, training with zero-centered gradient penalties leads to linear convergence towards the equilibrium point, both for SimGD and AltGD.

4.4 Conclusion

In this chapter we have seen that our results from Chapter 3 generalize to higher-dimensional spaces. More specifically, we have introduced and analyzed a multidimensional generalization of the one-dimensional Dirac-GAN. Interestingly, while qualitatively the multidimensional Dirac-GAN behaves similar to the one-dimensional case, it also yields additional insights. Indeed, the multidimensional case exhibits two kinds of eigenvalues with different influences on the GAN training dynamics. In the next chapter we extend our results even further and analyze the stability of general (unregularized) GANs.

[3] Using Corollary A.4, it can be shown that this is equivalent to choosing a large regularization parameter γ as well as a small learning rate h_d for the discriminator and a large learning rate h_g for the generator.

5 Convergence Theory

In the last two chapters, we have analyzed the Dirac-GAN in depth. While this simple example leads to interesting insights, we have not yet answered if and how these results transfer to more complex cases. In this chapter we develop general mathematical tools to understand GAN training in the general case.

To generalize our convergence results from the previous chapters to arbitrary GANs, we have to formulate several assumptions. To this end, we build on the four assumptions formulated by Nagarajan and Kolter [136], but generalize them. Nagarajan and Kolter [136] were able to prove local convergence of the continuous GAN dynamics under these four assumptions. In this chapter we prove a discrete version of these results using our theory from Chapter 2.

Unfortunately, while the four assumptions are usually (approximately) satisfied as long as both p_θ and $p_\mathcal{D}$ are absolutely continuous distributions, the fourth condition is not satisfied in more realistic situations. Indeed, for the one-dimensional Dirac-GAN from Chapter 3 and the multidimensional Dirac-GAN from Chapter 4, the fourth assumption is not satisfied. In Chapter 6 we therefore build on the results from this chapter and show how regularization helps overcome the limitations of the fourth assumption.

In addition to the results presented in this chapter, in Appendix D we describe another form of stable equilibrium, called Energy Solution, that may exist for unregularized GAN training. However, this kind of solution requires a more expressive discriminator and can be ill-conditioned. Please see Appendix D for a discussion.

5.1 Assumptions

In this section we formulate slightly generalized versions of the assumptions by Nagarajan and Kolter [136]. As we want to focus on theoretical insights instead of technical details, we assume that $G_\theta(z)$ and $D_\psi(x)$ are infinitely often differentiable in their parameters[1] and inputs θ, ψ, x and z. Moreover, we assume that we can always exchange differentiation and integration (i.e. expectations). This condition is satisfied, for example, when the corresponding derivatives are uniformly bounded.

Recall that we denote the parameter spaces of the generator $G_\theta(\cdot)$ and the discriminator $D_\psi(\cdot)$ by $\Omega_G \subseteq \mathbb{R}^n$ and $\Omega_D \subseteq \mathbb{R}^m$, respectively. Let $(\theta^*, \psi^*) \in \Omega_G \times \Omega_D$ denote an equilibrium point of the (unregularized) training dynamics. In our convergence analysis, we consider the realizable case, i.e. as in Section 1.2 we assume that there are generator parameters that make the generator produce the true data distribution.

[1] Note that this condition is not satisfied when we implement $G_\theta(\cdot)$ and $D_\psi(\cdot)$ with ReLU activation functions. However, we can always approximate the activation functions with infinitely differentiable functions.

Assumption I. *We have* $p_{\theta^*} = p_{\mathcal{D}}$ *and* $D_{\psi^*}(x) = 0$ *for* $x \in \operatorname{supp} p_{\mathcal{D}}$.

Similarly to Nagarajan and Kolter [136], we assume that φ_1 and φ_2 satisfy the following property:

Assumption II. φ_1 *and* φ_2 *are twice differentiable at* 0 *and* (φ_1, φ_2) *is strictly valid.*[2]

Note that by Lemma 1.12 we have $\varphi_1'(0) = \varphi_2'(0) \neq 0$ and $\varphi_1''(0) + \varphi_2''(0) < 0$. For brevity, we again define $\rho_1 := \varphi_1'(0) = \varphi_2'(0)$ and $\rho_2 := -\varphi_1''(0) - \varphi_2''(0)$.

The convergence proof is complicated by the fact that for neural networks, there generally is not a single equilibrium point (θ^*, ψ^*), but a submanifold of equivalent equilibria corresponding to different parameterizations of the same function. We therefore define the *reparameterization manifolds* $\mathcal{M}_G$ and $\mathcal{M}_D$. To this end, let

$$h(\psi) := \mathrm{E}_{x \sim p_{\mathcal{D}}} \left[|D_\psi(x)|^2 \right] \tag{5.1}$$

The *reparameterization manifolds* are then defined as

$$\mathcal{M}_G := \{\theta \mid p_\theta = p_{\mathcal{D}}\} \quad \mathcal{M}_D := \{\psi \mid h(\psi) = 0\} \tag{5.2}$$

To prove local convergence, we have to assume some regularity properties for $\mathcal{M}_G$ and $\mathcal{M}_D$ near the equilibrium point. To state these assumptions, we need

$$g(\theta) := \mathrm{E}_{x \sim p_\theta} \left[\nabla_\psi D_\psi(x)|_{\psi=\psi^*} \right] \tag{5.3}$$

Assumption III. *There are* ε*-balls* $\mathcal{B}_\varepsilon(\theta^*)$ *and* $\mathcal{B}_\varepsilon(\psi^*)$ *around* θ^* *and* ψ^* *so that* $\mathcal{M}_G \cap \mathcal{B}_\varepsilon(\theta^*)$ *and* $\mathcal{M}_D \cap \mathcal{B}_\varepsilon(\psi^*)$ *define* $\mathcal{C}^1$*- manifolds. Moreover, the following holds:*
(i) if $w \in \mathbb{R}^n$ *is not in the tangent space of* $\mathcal{M}_D$ *at* ψ^**, then* $\partial_w^2 h(\psi^*) \neq 0$.
(ii) if $w \in \mathbb{R}^m$ *is not in the tangent space of* $\mathcal{M}_G$ *at* θ^**, then* $\partial_w g(\theta^*) \neq 0$.

While formally similar, the two conditions in Assumption III have very different meanings: the first condition is a simple regularity property that means that the geometry of $\mathcal{M}_D$ can be locally described by the second derivative of h. The second condition implies that the discriminator is strong enough so that it can detect any deviation from the equilibrium generator distribution. Indeed, this is the only point where we assume that the class of representable discriminators is sufficiently expressive (and excludes, for example, the trivial case $D_\psi(x) = 0$ for all ψ and x). Indeed, the second condition can be regarded as a local version of the condition in Lemma 1.15 that we needed to ensure uniqueness of the Nash equilibrium. Note that Assumption III is similar to the third assumption from Nagarajan and Kolter [136], but more general. Indeed, Nagarajan and Kolter [136] only consider linear manifolds $\mathcal{M}_G$ and $\mathcal{M}_D$.

Finally, we state a fourth assumption, which was also originally proposed by Nagarajan and Kolter [136], on the support of the data and generator distributions.

Assumption IV. *There is an* ε*-balls* $\mathcal{B}_\varepsilon(\theta^*)$ *so that* $\operatorname{supp}(p_\theta) = \operatorname{supp}(p_{\mathcal{D}})$ *for all* $\theta \in \mathcal{B}_\varepsilon(\theta^*)$.

[2] See Definition 1.9 for the definition of "strictly valid".

Assumption IV is the most restrictive assumption and is typically only satisfied for absolutely continuous data and generator distributions (e.g. Gaussian distributions). Indeed, Assumption IV does not hold for the (unregularized) Dirac-GAN from Chapter 3 and 4. In Chapter 6 we introduce regularizers that enable us to remove this assumption.

5.2 Linearization

To prove convergence of the GAN training dynamics, we first need to understand the local structure of the gradient vector field $v(\cdot)$. Recall that the gradient vector field $v(\cdot)$ is defined as

$$v(\theta, \psi) := \begin{pmatrix} -\nabla_\theta \mathcal{L}(\theta, \psi) \\ \nabla_\psi \mathcal{L}(\theta, \psi) \end{pmatrix} \tag{5.4}$$

with

$$\mathcal{L}(\theta, \psi) = \mathrm{E}_{x \sim p_\theta} \left[\varphi_1(D_\psi(x)) \right] + \mathrm{E}_{x \sim p_\mathcal{D}} \left[\varphi_2(-D_\psi(x)) \right] \tag{5.5}$$

We first derive an explicit representation of $v(\cdot)$.

Lemma 5.1. *The gradients of $\mathcal{L}(\theta, \psi)$ with respect to θ and ψ are given by*

$$\begin{aligned} \nabla_\theta \mathcal{L}(\theta, \psi) &= \nabla_\theta \mathrm{E}_{x \sim p_\theta} \left[\varphi_1(D_\psi(x)) \right] && (5.6) \\ &= \mathrm{E}_{z \sim p_0} \left[\varphi_1'(D_\psi(G_\theta(z))) \left[\frac{\partial G_\theta}{\partial \theta}(z) \right]^\mathsf{T} \cdot \nabla_x D_\psi(G_\theta(z)) \right] && (5.7) \\ \nabla_\psi \mathcal{L}(\theta, \psi) &= \mathrm{E}_{x \sim p_\theta} \left[\varphi_1'(D_\psi(x)) \nabla_\psi D_\psi(x) \right] - \mathrm{E}_{x \sim p_\mathcal{D}} \left[\varphi_2'(-D_\psi(x)) \nabla_\psi D_\psi(x) \right] && (5.8) \end{aligned}$$

Proof. To compute $\nabla_\theta \mathcal{L}(\theta, \psi)$, we apply the multivariate chain rule to

$$\mathrm{E}_{x \sim p_\theta} \left[\varphi_1(D_\psi(x)) \right] = \mathrm{E}_{z \sim p_0} \left[\varphi_1(D_\psi(G_\theta(z))) \right] \tag{5.9}$$

To compute $\nabla_\psi \mathcal{L}(\theta, \psi)$, we apply the multivariate chain rule to (5.5). □

As discussed in Chapter 2, to understand local convergence near an equilibrium point (θ^*, ψ^*) we have to analyze the Jacobian $v'(\theta^*, \psi^*)$.

Lemma 5.2 (Nagarajan and Kolter)**.** *Assume that (θ^*, ψ^*) satisfies Assumption I and IV. The Jacobian of the gradient vector field $v(\theta, \psi)$ at (θ^*, ψ^*) is then*

$$v'(\theta^*, \psi^*) = \begin{pmatrix} 0 & -K_{DG}^\mathsf{T} \\ K_{DG} & -K_{DD} \end{pmatrix} \tag{5.10}$$

The terms K_{DD} and K_{DG} are given by

$$\begin{aligned} K_{DG} &= \rho_1 \frac{\partial}{\partial \theta} \mathrm{E}_{x \sim p_\theta} \left[\nabla_\psi D_\psi(x) \right] \Big|_{\theta = \theta^*, \psi = \psi^*} && (5.11) \\ K_{DD} &= \rho_2 \, \mathrm{E}_{x \sim p_\mathcal{D}} \left[\nabla_\psi D_{\psi^*}(x) \nabla_\psi D_{\psi^*}(x)^\mathsf{T} \right] \big|_{\psi = \psi^*} && (5.12) \end{aligned}$$

Proof. First note that by the definition of $v(\cdot)$ in (5.4), the Jacobian $v'(\theta^*, \psi^*)$ of $v(\cdot)$ is given by

$$\begin{pmatrix} -\nabla_\theta^2 \mathcal{L}(\theta^*, \psi^*) & -\nabla_{\psi,\theta}^2 \mathcal{L}(\theta^*, \psi^*)^\mathsf{T} \\ \nabla_{\psi,\theta}^2 \mathcal{L}(\theta^*, \psi^*) & \nabla_\psi^2 \mathcal{L}(\theta^*, \psi^*) \end{pmatrix} \tag{5.13}$$

Let us first consider $\nabla_\theta^2 \mathcal{L}(\theta^*, \psi^*)$. By Lemma 5.1, we have

$$\nabla_\theta \mathcal{L}(\theta, \psi) = \nabla_\theta \mathrm{E}_{x \sim p_\theta} \left[\varphi_1(D_\psi(x)) \right] \tag{5.14}$$

However, we have $D_{\psi^*}(x) = 0$ for all $x \in \text{supp } p_\mathcal{D}$ by Assumption I. Moreover, by Assumption IV there is $\varepsilon > 0$ such that for all $\theta \in \mathcal{B}_\varepsilon(\theta^*)$ we have supp $p_\theta =$ supp $p_\mathcal{D}$. This shows that for $\theta \in \mathcal{B}_\varepsilon(\theta^*)$ we have $\nabla_\theta \mathcal{L}(\theta, \psi^*) = 0$ and therefore also $\nabla_\theta^2 \mathcal{L}(\theta^*, \psi^*) = 0$.

To compute $\nabla_{\psi,\theta}^2 \mathcal{L}(\theta^*, \psi^*)$, we compute the partial Jacobian of (5.8) with respect to θ:

$$\begin{aligned} \nabla_{\psi,\theta}^2 \mathcal{L}(\theta^*, \psi^*) &= \frac{\partial}{\partial \theta} \mathrm{E}_{x \sim p_\theta} \left[\varphi_1'(D_\psi(x)) \nabla_\psi D_\psi(x) \right] \big|_{\theta=\theta^*, \psi=\psi^*} \\ &= \rho_1 \frac{\partial}{\partial \theta} \mathrm{E}_{x \sim p_\theta} \left[\nabla_\psi D_\psi(x) \right] \big|_{\theta=\theta^*, \psi=\psi^*} \end{aligned} \tag{5.15}$$

Here, we again used $D_{\psi^*}(x) = 0$ for $x \in \text{supp } p_\mathcal{D}$ (Assumption I) and supp $p_\theta =$ supp $p_\mathcal{D}$ for $\theta \in \mathcal{B}_\varepsilon(\theta^*)$ (Assumption IV).

Finally, consider $\nabla_\psi^2 \mathcal{L}(\theta^*, \psi^*)$. By taking the partial Jacobian of (5.8) with respect to ψ, we obtain

$$\begin{aligned} \nabla_\psi^2 \mathcal{L}(\theta, \psi) = &\mathrm{E}_{x \sim p_\theta} \left[\varphi_1''(D_\psi(x)) \nabla_\psi D_\psi(x) \nabla_\psi D_\psi(x)^\mathsf{T} + \varphi_2'(D_\psi(x)) \nabla_\psi^2 D_\psi(x) \right] \\ &- \mathrm{E}_{x \sim p_\mathcal{D}} \left[-\varphi_1''(-D_\psi(x)) \nabla_\psi D_\psi(x) \nabla_\psi D_\psi(x)^\mathsf{T} + \varphi_2'(-D_\psi(x)) \nabla_\psi^2 D_\psi(x) \right] \end{aligned} \tag{5.16}$$

Evaluating at (θ^*, ψ^*) hence yields, by Assumption I,

$$\nabla_\psi^2 \mathcal{L}(\theta^*, \psi^*) = -\rho_2 \, \mathrm{E}_{x \sim p_\mathcal{D}} \left[\nabla_\psi D_\psi(x) \nabla_\psi D_\psi(x)^\mathsf{T} \right] \big|_{\psi=\psi^*} \tag{5.17}$$

□

5.3 Convergence

We now prove our first convergence theorem. This theorem was first derived by Nagarajan and Kolter [136] for the continuous GAN training dynamics under slightly more restrictive assumptions. Here, we provide a proof of a similar theorem for the discretized training dynamics (i.e. SimGD or AltGD) under our more general assumptions.

To show convergence, we want to apply Theorem 2.5. To this end, we have to show that $\mathcal{M}_G \times \mathcal{M}_D$ consists only of stationary points of $v(\cdot)$ in a neighborhood of (θ^*, ψ^*) and that $v'(\theta^*, \psi^*)$ only has eigenvalues with negative real-part when restricted to

$$\left(\mathrm{T}_{(\theta^*, \psi^*)} \left(\mathcal{M}_G \times \mathcal{M}_D \right) \right)^\perp = \left(\mathrm{T}_{\theta^*} \mathcal{M}_G \right)^\perp \times \left(\mathrm{T}_{\psi^*} \mathcal{M}_D \right)^\perp \tag{5.18}$$

We first show that $\mathcal{M}_G \times \mathcal{M}_D$ consists only of stationary points of $v(\cdot)$ near (θ^*, ψ^*).

Lemma 5.3. *Assume that (θ^*, ψ^*) satisfies Assumption I and Assumption IV. Then there is $\varepsilon > 0$ such that $(\mathcal{M}_G \cap \mathcal{B}_\varepsilon(\theta^*)) \times \mathcal{M}_D$ consists only of stationary points of $v(\cdot)$, i.e. $v(\theta, \psi) = 0$ for all $(\theta, \psi) \in (\mathcal{M}_G \cap \mathcal{B}_\varepsilon(\theta^*)) \times \mathcal{M}_D$.*

Proof. We have to show that $\nabla_\theta \mathcal{L}(\theta, \psi) = 0$ and $\nabla_\psi \mathcal{L}(\theta, \psi) = 0$ for all $(\theta, \psi) \in (\mathcal{M}_G \cap \mathcal{B}_\varepsilon(\theta^*)) \times \mathcal{M}_D$ for some $\varepsilon > 0$.

We first consider $\nabla_\psi \mathcal{L}(\theta, \psi)$. By Lemma 5.1, we have

$$\nabla_\psi \mathcal{L}(\theta, \psi) = \mathrm{E}_{x \sim p_\theta}\left[\varphi_1'(D_\psi(x)) \nabla_\psi D_\psi(x)\right] - \mathrm{E}_{x \sim p_\mathcal{D}}\left[\varphi_2'(-D_\psi(x)) \nabla_\psi D_\psi(x)\right] \tag{5.19}$$

However, for $(\theta, \psi) \in \mathcal{M}_G \times \mathcal{M}_D$ we have $p_\theta = p_\mathcal{D}$ and $D_\psi(x) = 0$ for all $x \in \operatorname{supp} p_\mathcal{D}$. This shows that

$$\nabla_\psi \mathcal{L}(\theta, \psi) = \mathrm{E}_{x \sim p_\mathcal{D}}\left[\varphi_1'(0) \nabla_\psi D_\psi(x) - \varphi_2'(0) \nabla_\psi D_\psi(x)\right] = 0 \tag{5.20}$$

Now consider $\nabla_\theta \mathcal{L}(\theta, \psi)$. We use $D_\psi(x) = 0$ for $x \in \operatorname{supp} p_\mathcal{D}$ and $\psi \in \mathcal{M}_D$ as well as $\operatorname{supp} p_\theta = \operatorname{supp} p_\mathcal{D}$ for $\theta \in \mathcal{B}_\varepsilon(\theta^*)$ by Assumption IV. As in the proof of Lemma 5.2, this shows that

$$\nabla_\theta \mathcal{L}(\theta, \psi) = \nabla_\theta \mathrm{E}_{x \sim p_\theta}\left[\varphi_1(0)\right] = 0 \tag{5.21}$$

for all $(\theta, \psi) \in \mathcal{B}_\varepsilon(\theta^*) \times \mathcal{M}_D$ with some $\varepsilon > 0$. All in all, we see that $v(\theta, \psi) = 0$ for all $(\theta, \psi) \in (\mathcal{M}_G \cap \mathcal{B}_\varepsilon(\theta^*) \times \mathcal{M}_D$. □

Lemma 5.3 allows us to directly draw a conclusion about the interaction term $K_{DG} = \nabla^2_{\psi,\theta} L(\theta^*, \psi^*)$ from Lemma 5.2.

Lemma 5.4. *Let K_{DG} be defined as in Lemma 5.2. Then the following assertions hold:*

i) $K_{DG} w = 0$ *for all* $w \in \mathrm{T}_{\theta^*} \mathcal{M}_G$

ii) $K_{DG}^\mathsf{T} w = 0$ *for all* $w \in \mathrm{T}_{\psi^*} \mathcal{M}_D$

iii) $K_{DG} w \in (\mathrm{T}_{\psi^*} \mathcal{M}_D)^\perp$ *for all* $w \in \mathbb{R}^n$

iv) $K_{DG}^\mathsf{T} w \in (\mathrm{T}_{\theta^*} \mathcal{M}_G)^\perp$ *for all* $w \in \mathbb{R}^m$

Proof. Let $w \in \mathrm{T}_{\theta^*} \mathcal{M}_G$. By definition of the tangent space $\mathrm{T}_{\theta^*} \mathcal{M}_G$, there is a $\mathcal{C}^1$-curve $\theta : (-\varepsilon, \varepsilon) \to \mathcal{M}_G$ with $\theta(0) = \theta^*$ and $\theta'(0) = w$. Hence, by Lemma 5.2,

$$K_{DG} w = \frac{\mathrm{d}}{\mathrm{d}t} \nabla_\psi \mathcal{L}(\theta(t), \psi^*)\bigg|_{t=0} \tag{5.22}$$

However, by Lemma 5.3, $\nabla_\psi \mathcal{L}(\theta(t), \psi^*) = 0$ for all $t \in (-\varepsilon, \varepsilon)$ for ε small enough, since $(\theta(t), \psi^*) \in \mathcal{M}_G \times \mathcal{M}_D$. This shows $K_{DG} w = 0$ for all $w \in \mathrm{T}_{\theta^*} \mathcal{M}_G$, which is (i). We can prove (ii) in a similar way, using

$$K_{DG}^\mathsf{T} w = -\frac{\mathrm{d}}{\mathrm{d}t} \nabla_\theta \mathcal{L}(\theta^*, \psi(t))\bigg|_{t=0} = 0 \tag{5.23}$$

for a curve $\psi : (-\varepsilon, \varepsilon) \to \mathcal{M}_D$ with $\psi(0) = \psi^*$ and $\psi'(0) = w$ if $w \in \mathrm{T}_{\psi^*}\mathcal{M}_D$.

To show (iii), let $w \in \mathbb{R}^n$ and $\tilde{w} \in \mathrm{T}_{\psi^*}\mathcal{M}_D$. Using (ii) from the first part of the proof, we obtain

$$\tilde{w}^\mathsf{T}(K_{DG}w) = (K_{DG}^\mathsf{T}\tilde{w})^\mathsf{T}w = 0^\mathsf{T} \cdot w = 0 \tag{5.24}$$

Since, $\tilde{w} \in \mathrm{T}_{\psi^*}\mathcal{M}_D$ was arbitrary, this shows $K_{DG}w \in (\mathrm{T}_{\psi^*}\mathcal{M}_D)^\perp$ for all $w \in \mathbb{R}^n$, which is (iii). The last assertion (iv) follows in the same way from (i). □

Next, we examine the eigenvalues of $v'(\theta^*, \psi^*)$. We first state a simple criterion from Nagarajan and Kolter [136] under what conditions the Jacobian has only eigenvalues with negative real-part. In Chapter 7 we extend this lemma to obtain tight bounds on the real and imaginary part of the eigenvalues.

Here, we give a short proof of a lemma by Nagarajan and Kolter [136].

Lemma 5.5 (Nagarajan and Kolter)**.** *Assume $J \in \mathbb{R}^{(n+m)\times(n+m)}$ is of the following form:*

$$J = \begin{pmatrix} 0 & -P^\mathsf{T} \\ P & -Q \end{pmatrix} \tag{5.25}$$

where $Q \in \mathbb{R}^{m\times m}$ is a symmetric positive definite matrix and $P \in \mathbb{R}^{m\times n}$ has full column rank. Then all eigenvalues λ of J satisfy $\Re(\lambda) < 0$.

Proof. Let $w^\mathsf{T} = (w_1^\mathsf{T}, w_2^\mathsf{T}) \in \mathbb{C}^{n+m}$ with $w_1 \in \mathbb{C}^n$ and $w_2 \in \mathbb{C}^m$ denote some eigenvector of J with corresponding eigenvalues $\lambda \in \mathbb{C}$. Then

$$\Re(\lambda) = \frac{\lambda + \bar{\lambda}}{2} = \frac{1}{2}w^H(J + J^\mathsf{T})w = -w_2^H Q w_2. \tag{5.26}$$

Because Q is positive definite, we have $\Re(\lambda) \leq 0$. Moreover, it suffices to show that $w_2 \neq 0$ to prove $\Re(\lambda) < 0$.

Assume that $w_2 = 0$. Because w is an eigenvector of J, we have $Pw_1 - Qw_2 = \lambda w_2$ and therefore $Pw_1 = 0$. Because P has full-column rank, this shows $w_1 = 0$ and hence $w = 0$. However, this contradicts the fact that w is an eigenvector of J. All in all, this show that $w_2 \neq 0$ and thus $\Re(\lambda) \leq -w_2^H Q w_2 < 0$ as required. □

To apply Lemma 5.5, we have to show that K_{DD} is positive definite and K_{DG} has full column rank (at least orthogonal to $\mathcal{M}_D$ and $\mathcal{M}_G$). However, we have

Lemma 5.6. *Assume that Assumption I, II, III and IV hold. If $w \in \mathbb{R}^m$ is not in the tangent space of $\mathcal{M}_D$ at ψ^*, then $w^\mathsf{T}K_{DD}w > 0$.*

Proof. By Lemma 5.2, we have

$$w^\mathsf{T}K_{DD}w = \rho_2 \, \mathrm{E}_{x\sim p_\mathcal{D}}\left[\left(\nabla_\psi D_{\psi^*}(x)^\mathsf{T}w\right)^2\right] \tag{5.27}$$

By Assumption II, we have $\rho_2 > 0$. Hence, $w^\mathsf{T}K_{DD}w \geq 0$ and $w^\mathsf{T}K_{DD}w = 0$ implies

$$\nabla_\psi D_\psi(x)^\mathsf{T}w\big|_{\psi=\psi^*} = 0 \tag{5.28}$$

for all $x \in \operatorname{supp} p_{\mathcal{D}}$. Let

$$h(\psi) := \mathrm{E}_{x \sim p_{\mathcal{D}}} \left[|D_\psi(x)|^2 \right] \tag{5.29}$$

Using the fact that $D_\psi(x) = 0$ for $x \in \operatorname{supp} p_{\mathcal{D}}$, we see that the Hessian of $h(\psi)$ at ψ^* is

$$\nabla_\psi^2 h(\psi^*) = 2\,\mathrm{E}_{x \sim p_{\mathcal{D}}} \left[\nabla_\psi D_{\psi^*}(x) \nabla_\psi D_{\psi^*}(x)^\mathsf{T} \right] \tag{5.30}$$

The second directional derivative $\partial_w^2 h(\psi^*)$ is therefore

$$\partial_w^2 h(\psi^*) = 2\,\mathrm{E}_{x \sim p_{\mathcal{D}}} \left[\left(\nabla_\psi D_{\psi^*}(x)^\mathsf{T} w \right)^2 \right] = 0 \tag{5.31}$$

By Assumption III, this can only hold if w is in the tangent space of $\mathcal{M}_D$ at ψ^*. □

Lemma 5.7. *Assume that Assumption I, II, III and IV hold. If $w \in \mathbb{R}^n$ is not in the tangent space $\mathrm{T}_{\theta^*}\mathcal{M}_G$ of $\mathcal{M}_G$ at θ^*, then $K_{DG} w \notin \mathrm{T}_{\psi^*}\mathcal{M}_D$*

Proof. By Lemma 5.4 we have $K_{DG} w \in (\mathrm{T}_{\psi^*}\mathcal{M}_D)^\perp$. It therefore suffices to show $K_{DG} w \neq 0$ to prove $K_{DG} w \notin \mathrm{T}_{\psi^*}\mathcal{M}_D$. However, by Lemma 5.2, we have

$$\begin{aligned} K_{DG} w &= \rho_1 \left[\frac{\partial}{\partial \theta} \mathrm{E}_{x \sim p_\theta} \left[\nabla_\psi D_\psi(x) \right] \Big|_{\theta=\theta^*, \psi=\psi^*} \right] w \\ &= \rho_1 \partial_w g(\theta) \end{aligned} \tag{5.32}$$

for

$$g(\theta) := \mathrm{E}_{p_\theta(x)} \left[\nabla_\psi D_\psi(x) \big|_{\psi=\psi^*} \right] \tag{5.33}$$

By Assumption III and because $\rho_1 \neq 0$ (Assumption II), this implies $K_{DG} w \neq 0$ if w is not in the tangent space of $\mathcal{M}_G$ at θ^*. Together with Lemma 5.4 this yields the assertion.

□

We are now ready to state the first convergence result. Nagarajan and Kolter [136] proved a continuous version of this theorem under slightly more restrictive assumptions. Here, we prove a discrete version, using the tools that we developed in Chapter 2.

Theorem 5.8. *Assume Assumption I, II, III and IV hold for (θ^*, ψ^*). For small enough learning rates, SimGD and AltGD are both convergent to $\mathcal{M}_G \times \mathcal{M}_D$ in a neighborhood of (θ^*, ψ^*). Moreover, the rate of convergence is at least linear.*

Proof. We want to show that both SimGD and AltGD are locally convergent to $\mathcal{M}_G \times \mathcal{M}_D$ by applying Theorem 2.5. First note that by Lemma 5.3 there is $\varepsilon > 0$ such that $(\mathcal{M}_G \cap \mathcal{B}_\varepsilon(\theta^*)) \times \mathcal{M}_D$ consists only of stationary points of the gradient vector field.

Let $\mathrm{T}_{\theta^*}\mathcal{M}_G$ and $\mathrm{T}_{\psi^*}\mathcal{M}_D$ denote the tangent spaces of $\mathcal{M}_G$ and $\mathcal{M}_D$ at θ^* and ψ^* and let $B_G \in \mathbb{R}^{n \times l_1}$ and $B_D \in \mathbb{R}^{m \times l_2}$ denote orthogonal bases of $(\mathrm{T}_{\theta^*}\mathcal{M}_G)^\perp$ and $(\mathrm{T}_{\psi^*}\mathcal{M}_D)^\perp$, respectively. Moreover, let

$$B := \begin{pmatrix} B_G & 0 \\ 0 & B_D \end{pmatrix} \in \mathbb{R}^{(n+m) \times (l_1 \times l_2)} \tag{5.34}$$

First note that the columns of B define a basis of

$$\left(\mathrm{T}_{(\theta^*,\psi^*)}\left(\mathcal{M}_G \times \mathcal{M}_D\right)\right)^{\perp} = \left(\mathrm{T}_{\theta^*}\mathcal{M}_G\right)^{\perp} \times \left(\mathrm{T}_{\psi^*}\mathcal{M}_D\right)^{\perp} \tag{5.35}$$

To apply Theorem 2.5 we have to show that $B^{\mathsf{T}} v'(\theta^*, \psi^*) B$ has only eigenvalues with negative real-part.

Using Lemma 5.2, we see that $B^{\mathsf{T}} v'(\theta, \psi) B$ is

$$B^{\mathsf{T}} v'(\theta^*, \psi^*) B = \begin{pmatrix} 0 & -B_G^{\mathsf{T}} K_{DG}^{\mathsf{T}} B_D \\ B_D^{\mathsf{T}} K_{DG} B_G & -B_D^{\mathsf{T}} K_{DD} B_D \end{pmatrix} \tag{5.36}$$

We now show that $B_D^{\mathsf{T}} K_{DD} B_D$ is positive definite and $B_D^{\mathsf{T}} K_{DG} B_G$ has full column rank.

To this end, first note that for any $w \in \mathbb{R}^{l_1}$, we have $B_D w \in \left(\mathrm{T}_{\psi^*}\mathcal{M}_D\right)^{\perp}$ and hence $B_D w \notin \mathrm{T}_{\psi^*}\mathcal{M}_D$ for $w \neq 0$. Therefore, by Lemma 5.6, we have

$$w^{\mathsf{T}} \left(B_D^{\mathsf{T}} K_{DD} B_D\right) w = (B_D w)^{\mathsf{T}} K_{DD} (B_D w) > 0 \tag{5.37}$$

for $w \neq 0$. As a result, we see that $B_D^{\mathsf{T}} K_{DD} B_D$ is symmetric positive definite.

Similarly, since $B_G w \notin \mathrm{T}_{\theta^*}\mathcal{M}_G$ for $w \neq 0$, we can apply Lemma 5.7 to obtain $K_{DG} B_G w \notin \mathrm{T}_{\psi^*}\mathcal{M}_D$. However, this implies that $K_{DG} B_G w$ has a non-zero component orthogonal to $\mathrm{T}_{\psi^*}\mathcal{M}_D$, i.e. $B_D^{\mathsf{T}} K_{DG} B_G w \neq 0$. Since $w \neq 0$ was arbitrary, this implies that $B_D^{\mathsf{T}} K_{DG} B_G$ has full column rank.

Lemma 5.5 now implies that all eigenvalues of $B^{\mathsf{T}} v'(\theta^*, \psi^*) B$ have negative real part. By Theorem 2.5, SimGD and AltGD are therefore both convergent to $\mathcal{M}_G \times \mathcal{M}_D$ near (θ^*, ψ^*) for small enough learning rates. Moreover, the rate of convergence is at least linear. □

5.4 Examples

Let us now consider some examples. For each example, we discuss the validity of the assumptions and if the training dynamics are locally convergent. It is easy to see that both Assumption I and Assumption II are true for all examples as long as we choose a strictly valid pair of activation functions (φ_1, φ_2). In the following, we therefore discuss only Assumption III and Assumption IV.

Dirac-GAN For the Dirac-GAN, we have $h(\psi) = |\psi^{\mathsf{T}} x_0|^2$, $g(\theta) = \theta$ and $\mathcal{M}_G = \{x_0\}$. As a result, $\mathcal{M}_D = \mathrm{T}_{\psi^*}\mathcal{M}_D = \{\psi \in \mathbb{R}^n \mid \psi \perp x_0\}$. Note that the special case $x_0 = 0$ implies $\mathcal{M}_D = \mathrm{T}_{\psi^*}\mathcal{M}_D = \mathbb{R}^n$. Both $\mathcal{M}_G$ and $\mathcal{M}_D$ are hence $\mathcal{C}^1$-manifolds and $\partial_w^2 h(\psi^*) = |w^{\mathsf{T}} x_0| \neq 0$ for $w \notin \mathrm{T}_{\psi^*}\mathcal{M}_D$. We also have $\partial_w g(\theta^*) = w \neq 0$ for $w \notin \mathrm{T}_{\theta^*}\mathcal{M}_G = \{0\}$. The conditions of Assumption III are hence also satisfied. However, we have supp $p_{\mathcal{D}} = \{x_0\} \neq \{\theta\} =$ supp p_θ for $\theta \neq x_0$. Assumption IV is therefore not satisfied for the multidimensional Dirac-GAN. We hence cannot apply Theorem 5.8. Indeed, as we have seen in Chapter 4, the multidimensional Dirac-GAN is not locally convergent.

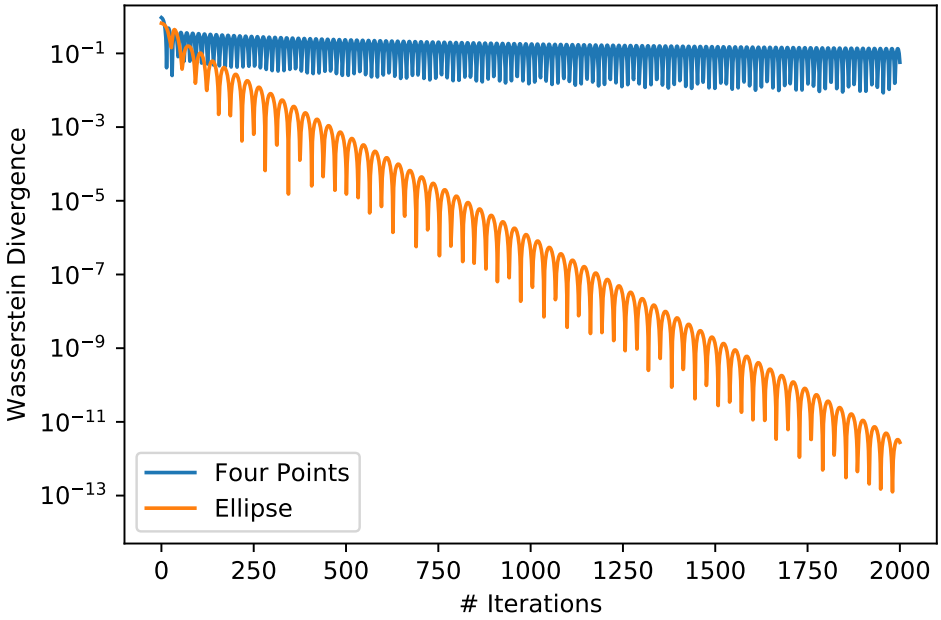


Figure 5.1: Examples. *Quantitative results for two of our examples for $h = w = 1$. The graph shows the Wasserstein divergence between the target distribution $p_{\mathcal{D}}$ and the generator distribution p_θ over the number of iterations. While the Ellipse example converges, the Four Points example does not. For this plot we use AltGD, but the results for SimGD look similar.*

Dirac-GAN with Instance Noise For the multidimensional Dirac-GAN with Gaussian instance noise, we also have $g(\theta) = \theta$ and hence $\partial_w g(\theta) = w \neq 0$ for $w \notin \mathrm{T}_{\theta^*}\mathcal{M}_G = \{0\}$ as well as $\mathcal{M}_G = \{x_0\}$. When using instance noise with standard deviation σ, h becomes

$$\begin{aligned} h(\psi) &= \mathrm{E}_{x \sim \mathcal{N}(x_0, \sigma^2 I)} \left[|\psi^\mathsf{T} x|^2 \right] \\ &= \psi^\mathsf{T} \mathrm{E}_{x \sim \mathcal{N}(x_0, \sigma^2 I)} \left[x x^\mathsf{T} \right] \psi \\ &= \psi^\mathsf{T} \left[x_0 x_0^\mathsf{T} + \sigma^2 I \right] \psi \\ &= |\psi^\mathsf{T} x_0|^2 + \sigma^2 \|\psi\|^2 \end{aligned} \tag{5.38}$$

Hence, $\mathcal{M}_D = \{0\}$ and $\partial_w^2 h(\psi^*) = |w^\mathsf{T} x_0|^2 + \sigma^2 \|w\|^2 \neq 0$ when $w \neq 0$ and $\sigma \neq 0$. Assumption III is therefore satisfied. Moreover, supp $p_{\mathcal{D}} = \mathcal{X} =$ supp p_θ for all θ, Assumption IV is hence also true. Theorem 5.8 is therefore applicable, showing that the Dirac-GAN with instance noise is linearly convergent to (θ^*, ψ^*), confirming our results from Chapter 4.

Ellipse Let us consider the Ellipse example from Chapter 1: in this example, p_0 is given by a uniform distribution on $[0, 2\pi]$, i.e. $p_0 = \mathcal{U}[0, 2\pi]$, and $p_{\mathcal{D}}$ is a uniform distribution on an ellipse with width w and height h in the two-dimensional plane. Moreover, we have

defined the generator and discriminator as

$$G_\theta(z) = \begin{pmatrix} \theta_1 \cos(z) \\ \theta_2 \sin(z) \end{pmatrix} \quad \text{and} \quad D_\psi(x) = \psi_1 x_1^2 + \psi_2 x_2^2 \tag{5.39}$$

As discussed in Chapter 1, the equilibrium point in this example is at $\theta^* = (\frac{w}{2}, \frac{h}{2})$ and $\psi^* = 0$. Assumption IV is again not satisfied. However, the proof of Lemma 5.2 still holds, since $D_{\psi^*}(x) = 0$ for all $x \in \mathcal{X}$. The Jacobian of the gradient vector field at (θ^*, ψ^*) is therefore given by

$$v'(\theta^*, \psi^*) = \begin{pmatrix} 0 & -K_{DG}^\mathsf{T} \\ K_{DG} & K_{DD} \end{pmatrix} \tag{5.40}$$

Here, K_{DD} is

$$\begin{aligned} K_{DD} &= \rho_2 \, \mathrm{E}_{x \sim p_\mathcal{D}} \left[\nabla_\psi D_{\psi^*}(x) \nabla_\psi D_{\psi^*}(x)^\mathsf{T} \right] \big|_{\psi=\psi^*} \\ &= \rho_2 \, \mathrm{E}_{z \sim \mathcal{U}[0,2\pi]} \begin{bmatrix} \theta_1^{*4} \cos^4(z) & \theta_1^{*2} \theta_2^{*2} \cos^2(z) \sin^2(z) \\ \theta_1^{*2} \theta_2^{*2} \cos^2(z) \sin^2(z) & \theta_2^{*4} \sin^4(z) \end{bmatrix} \\ &= \frac{1}{8} \rho_2 \begin{pmatrix} 3\theta_1^{*4} & \theta_1^{*2} \theta_2^{*2} \\ \theta_1^{*2} \theta_2^{*2} & 3\theta_2^{*4} \end{pmatrix} \end{aligned} \tag{5.41}$$

Hence $\det K_{DD} = \frac{1}{8} \rho_2^2 \theta_1^{*4} \theta_2^{*4} \neq 0$ if $\theta_1^*, \theta_2^* \neq 0$ and $\rho_2 \neq 0$. Similarly, K_{DG} is

$$\begin{aligned} K_{DG} &= \rho_1 \frac{\partial}{\partial \theta} \mathrm{E}_{x \sim p_\theta} \left[\nabla_\psi D_\psi(x) \right] \Big|_{\theta=\theta^*, \psi=\psi^*} \\ &= \frac{\rho_1}{2} \frac{\partial}{\partial \theta} \begin{pmatrix} \theta_1^2 \\ \theta_2^2 \end{pmatrix} \Big|_{\theta=\theta^*} \\ &= \rho_1 \begin{pmatrix} \theta_1^* & 0 \\ 0 & \theta_2^* \end{pmatrix} \end{aligned} \tag{5.42}$$

If $\theta_1^*, \theta_2^* \neq 0$, K_{DG} has therefore full column rank. Although Assumption IV is not satisfied in this example, Lemma 5.5 is still applicable, showing that the GAN training example is linearly convergent to (θ^*, ψ^*) close to the equilibrium point. This example shows that Theorem 5.8 only gives a sufficient condition for convergence and the system can still be stable if the conditions for Theorem 5.8 are not satisfied. Indeed, experimentally we find that (unregularized) GAN training converges for this example, see Figure 5.1.

Four Points We now consider the Four Points example from Chapter 1. The discriminator is again given by

$$D_\psi(x) = \psi_1 x_1^2 + \psi_2 x_2^2 \tag{5.43}$$

However, the generator is now given by the discrete distribution

$$p_\theta = \frac{1}{4} (\delta_{(-\theta_1, -\theta_2)} + \delta_{(-\theta_1, \theta_2)} + \delta_{(\theta_1, -\theta_2)} + \delta_{(\theta_1, \theta_2)}) \tag{5.44}$$

Moreover, we have defined $p_{\mathcal{D}} = p_{\theta^*}$ for some $\theta^* \in \mathbb{R}^2$. At the equilibrium, we again have $\psi^* = 0$. Note that this example is very similar to the previous example. Assumption IV is again not satisfied. However, in this case we have

$$K_{DD} = \rho_2 \begin{pmatrix} \theta_1^{*4} & \theta_1^{*2}\theta_2^{*2} \\ \theta_1^{*2}\theta_2^{*2} & \theta_2^{*4} \end{pmatrix} \tag{5.45}$$

This implies that $\det K_{DD} = 0$. For this example, Lemma 5.5 is hence not applicable. Indeed, we find that the Jacobian $v'(\theta^*, \psi^*)$ has eigenvalues on the imaginary axis for $h = w$ and is very ill-conditioned for $h \neq w$. The training dynamics are hence not convergent in this example, see Figure 5.1. In the next chapter we will see how we can handle this and similar cases.

5.5 Conclusion

In this chapter we have presented a generalized version of the GAN convergence theory by Nagarajan and Kolter [136]. While this convergence theory shows that GAN training is locally asymptotically stable in the absolutely continuous case, this condition is not satisfied for general GANs. Indeed, even our simple example from Chapter 3 does not satisfy this condition. In the next chapter we therefore use the tools from this chapter to analyze generalizations of the gradient regularizer from Chapter 3 for general GANs.

6 Regularization

The results from Chapter 5 show that GAN training is locally convergent under Assumption I, II, III and IV. Assumption I, II and III hold for simple examples of GAN training and it can be argued that they hold at least approximately for more complex cases. In contrast, Assumption IV is much stronger and does not even hold for simple examples such as the Dirac-GAN from Chapter 3 and 4.

Typically, Assumption IV holds for absolutely continuous data and generator distributions (e.g. Gaussian distributions), but does not hold for other cases. This explains why instance noise [6, 177], i.e. Gaussian noise on the data and generator distributions, helps stabilize GAN training. However, instance noise introduces additional noise into GAN training, which is not desirable. Indeed, as we have seen in Chapter 4, the amount of noise to achieve stability can be too large for a practical algorithm for high-dimensional output spaces.

In Chapter 3 and 4 we have shown that zero-centered gradient penalties have a similar effect as instance noise for the Dirac-GAN. In this chapter we generalize this result to more complex GANs. To this end, we introduce simple regularizers based on zero-centered gradient penalties and show that these regularizers make the system exponentially stable.

While gradient-based regularizers have been proposed before [64, 98, 169], these regularizers were more complicated. Moreover, none of the previous works showed the influence of gradient penalties on local convergence properties of the system. Indeed, our convergence analysis can only be extended to the regularizer by Roth et al. [169], but does not hold for the regularizers by Gulrajani et al. [64] and Kodali et al. [98]. As we have seen in Chapter 3, the latter two methods do not always converge.

6.1 Gradient Penalties

Our analysis in Chapter 3 and 4 suggests that the main effect of zero-centered gradient penalties on local stability is to penalize the discriminator for deviating from the Nash equilibrium. The simplest way to achieve this is to penalize the gradient on real data alone: when the generator distribution produces the true data distribution and the discriminator is equal to zero on the data manifold, the gradient penalty ensures that the discriminator cannot create a non-zero gradient orthogonal to the data manifold without suffering a loss in the GAN game. This leads to the following regularization term:

$$R_1(\theta, \psi) := \frac{\gamma}{2} \mathrm{E}_{\ p_{\mathcal{D}}(x)} \left[\|\nabla D_\psi(x)\|^2 \right] \tag{6.1}$$

Algorithm 4 AltGD with zero-centered Gradient Penalties

while not converged **do**
  $\theta \leftarrow \theta - h_g \nabla_\theta \mathcal{L}(\theta, \psi)$
  $\psi \leftarrow \psi + h_d \nabla_\psi (\mathcal{L}(\theta, \psi) - R_i(\theta, \psi))$
end while

Algorithm 5 SimGD with zero-centered Gradient Penalties

while not converged **do**
  $\delta_\theta \leftarrow -\nabla_\theta \mathcal{L}(\theta, \psi)$
  $\delta_\psi \leftarrow \nabla_\psi (\mathcal{L}(\theta, \psi) - R_i(\theta, \psi))$
  $\theta \leftarrow \theta + h_g v_\theta$
  $\psi \leftarrow \psi + h_d v_\psi$
end while

Note that this regularizer is a simplified version of to the regularizer derived by Roth et al. [169], which is defined as

$$\gamma \, \mathrm{E}_{x \sim p_\theta} \left[(1 - \sigma(D_\psi(x)))^2 \|\nabla_x D_\psi(x)\|^2 \right] + \gamma \, \mathrm{E}_{x \sim p_\mathcal{D}} \left[\sigma(D_\psi(x))^2 \|\nabla_x D_\psi(x)\|^2 \right] \tag{6.2}$$

However, our regularizer does not contain the additional weighting terms and penalizes the discriminator gradients only on the true data distribution.

We also consider a similar regularization term given by

$$R_2(\theta, \psi) := \frac{\gamma}{2} \mathrm{E}_{x \sim p_\theta} \left[\|\nabla D_\psi(x)\|^2 \right] \tag{6.3}$$

where we penalize the discriminator gradients on the current generator distribution instead of the true data distribution.

When we combine R_1- or R_2-regularization with Alternating Gradient Descent (AltGD), we obtain Algorithm 4. Similarly, when we use Simultaneous Gradient Descent (SimGD) instead of AltGD, we obtain Algorithm 5. Note that for the Dirac-GAN from Chapter 3, both regularizers reduce to the gradient penalty from Chapter 3 whose behavior is visualized in Figure 3.4g and Figure 3.4h.

6.2 Assumptions

In this chapter we present convergence results for the regularized GAN training dynamics for both regularization terms $R_1(\cdot)$ and $R_2(\cdot)$ under some suitable assumptions. In particular, we show that we can remove Assumption IV when we introduce the regularization terms $R_1(\cdot)$ and $R_2(\cdot)$.

As in Chapter 5, we assume variants of Assumption I, II and III, but do not assume Assumption IV. However, since we do not assume that $p_\mathcal{D}$ and p_θ have locally the same

support, we have to define the function h slightly differently now:

$$\tilde{h}(\psi) := \mathrm{E}_{x\sim p_{\mathcal{D}}}\left[|D_\psi(x)|^2 + \|\nabla_x D_\psi(x)\|^2\right] \tag{6.4}$$

Moreover, as before, we define

$$g(\theta) := \mathrm{E}_{x\sim p_\theta}\left[\nabla_\psi D_\psi(x)|_{\psi=\psi^*}\right] \tag{6.5}$$

The *reparameterization manifolds* are now defined as

$$\mathcal{M}_G := \{\theta \mid p_\theta = p_{\mathcal{D}}\} \quad \tilde{\mathcal{M}}_D := \{\psi \mid \tilde{h}(\psi) = 0\} \tag{6.6}$$

We can now state our assumptions. Assumption I becomes

Assumption I′. *We have $p_{\theta^*} = p_{\mathcal{D}}$ and $D_{\psi^*}(x) = 0$ for all $x \in \mathcal{X}$.*

Note that we now assume that $D_{\psi^*}(x) = 0$ for all $x \in \mathcal{X}$. We need this variant of Assumption I, since $D_{\psi^*}(x) = 0$ for $x \in \operatorname{supp} p_{\mathcal{D}}$ is not enough to ensure that (θ^*, ψ^*) is a stationary point of $v(\cdot)$ when Assumption IV is not satisfied. Since we only consider local convergence, we can always restrict $\mathcal{X}$ to some local neighborhood of $\operatorname{supp} p_{\mathcal{D}}$. However, for a simpler exposition, we assume $D_{\psi^*}(x) = 0$ for all $x \in \mathcal{X}$ here.

The next two assumptions are the same as Assumption II and III, but Assumption III′ uses $\tilde{h}$ from (6.4) instead of h from (5.1).

Assumption II′. *φ_1 and φ_2 are twice differentiable at 0 and (φ_1, φ_2) is strictly valid.*[1]

Assumption III′. *There are ε-balls $B_\varepsilon(\theta^*)$ and $B_\varepsilon(\psi^*)$ around θ^* and ψ^* so that $\mathcal{M}_G \cap B_\varepsilon(\theta^*)$ and $\tilde{\mathcal{M}}_D \cap B_\varepsilon(\psi^*)$ define $\mathcal{C}^1$- manifolds. Moreover, the following holds:*
(i) if $w \in \mathbb{R}^n$ is not in the tangent space of $\tilde{\mathcal{M}}_D$ at ψ^*, then $\partial_w^2 \tilde{h}(\psi^*) \neq 0$.
(ii) if $w \in \mathbb{R}^m$ is not in the tangent space of $\mathcal{M}_G$ at θ^*, then $\partial_w g(\theta^*) \neq 0$.

6.3 Linearization

To derive our convergence theory for the regularized GAN training dynamics, we have to understand the behavior of the regularized vector field $\tilde{v}_i(\cdot)$, defined by

$$\tilde{v}_i(\theta, \psi) := \begin{pmatrix} -\nabla_\theta \mathcal{L}(\theta, \psi) \\ \nabla_\psi \mathcal{L}(\theta, \psi) - \nabla_\psi R_i(\theta, \psi) \end{pmatrix} \tag{6.7}$$

Here, $R_i(\theta, \psi)$ with $i \in \{1, 2\}$ is defined as in in (6.1) and (6.3) and

$$\mathcal{L}(\theta, \psi) = \mathrm{E}_{x\sim p_\theta}\left[\varphi_1(D_\psi(x))\right] + \mathrm{E}_{x\sim p_{\mathcal{D}}}\left[\varphi_2(-D_\psi(x))\right] \tag{6.8}$$

Since, we already have derived expressions for $\nabla_\theta \mathcal{L}(\theta, \psi)$ and $\nabla_\psi \mathcal{L}(\theta, \psi)$ in Lemma 6.1, we only have to derive similar expressions for $\nabla_\psi R_1(\theta, \psi)$ and $\nabla_\psi R_2(\theta, \psi)$.

[1] See Definition 1.9 for the definition of "strictly valid".

Lemma 6.1. *The gradients $\nabla_\psi R_1(\theta,\psi)$ and $\nabla_\psi R_2(\theta,\psi)$ of the regularization terms $R_1(\cdot)$ and $R_2(\cdot)$ with respect to ψ are*

$$\nabla_\psi R_1(\theta,\psi) = \gamma \mathrm{E}_{x\sim p_\mathcal{D}}\left[\nabla_{\psi,x} D_\psi(x) \nabla_x D_\psi(x)\right] \tag{6.9}$$
$$\nabla_\psi R_2(\theta,\psi) = \gamma \mathrm{E}_{x\sim p_\theta}\left[\nabla_{\psi,x} D_\psi(x) \nabla_x D_\psi(x)\right] \tag{6.10}$$

Proof. These equations can be derived by taking the gradient of (6.1) and (6.3) with respect to ψ. □

To understand the local structure of $\tilde{v}_i(\cdot)$ near (θ^*,ψ^*), we have to understand the Jacobian $\tilde{v}_i'(\theta^*,\psi^*)$. To this end, we first derive a closed-form expression for $\nabla^2_\psi R_i(\theta^*,\psi^*)$.

Lemma 6.2. *The partial Hessians $\nabla^2_\psi R_2(\theta^*,\psi^*)$ and $\nabla^2_\psi R_1(\theta^*,\psi^*)$ of the regularization terms $R_1(\cdot)$ and $R_2(\cdot)$ with respect to ψ at (θ^*,ψ^*) are both given by*

$$\nabla^2_\psi R_i(\theta^*,\psi^*) = \gamma\, \mathrm{E}_{x\sim p_\mathcal{D}}\left[\nabla_{\psi,x} D_\psi(x) \nabla_{\psi,x} D_\psi(x)^\mathsf{T}\right]\big|_{\psi=\psi^*} \tag{6.11}$$

Moreover, both regularization terms satisfy $\nabla_{\psi,\theta} R_i(\theta^*,\psi^*) = 0$.

Proof. $\nabla^2_\psi R_i(\theta^*,\psi^*)$ can be computed by taking the derivative of (6.9) and (6.10) with respect to ψ and using that by Assumption I′ we have $D_{\psi^*}(x) = 0$ and hence also $\nabla_x D_{\psi^*}(x) = 0$ for $x \in \mathcal{X}$.

Moreover, we clearly have $\nabla_{\psi,\theta} R_1(\theta^*,\psi^*) = 0$, because $R_1(\cdot)$ does not depend on θ. To compute $\nabla_{\psi,\theta} R_2(\theta^*,\psi^*)$, we use the fact that $\nabla_x D_{\psi^*}(x) = 0$ for all $x \in \mathcal{X}$ which implies $\nabla_\psi R_2(\theta,\psi^*) = 0$ for all $\theta \in \Omega_G$ and hence also $\nabla_{\psi,\theta} R_2(\theta^*,\psi^*) = 0$. □

Using Lemma 6.2, we can derive a closed-form expression for $\tilde{v}_i'(\theta^*,\psi^*)$. As in Chapter 5, we define $\rho_1 := \varphi_1'(0) = \varphi_2'(0)$ and $\rho_2 := -\varphi_1''(0) - \varphi_2''(0)$. Note that Assumption II′ and Lemma 1.12 imply that $\rho_1 \neq 0$ and $\rho_2 > 0$.

Lemma 6.3. *Assume that (θ^*,ψ^*) satisfies Assumption I′. The Jacobian of the regularized gradient vector fields $\tilde{v}_i(\cdot)$ for $i \in \{0,1\}$ at (θ^*,ψ^*) is then*

$$\tilde{v}_i'(\theta^*,\psi^*) = \begin{pmatrix} 0 & -K_{DG}^\mathsf{T} \\ K_{DG} & -\tilde{K}_{DD} \end{pmatrix}. \tag{6.12}$$

where $\tilde{K}_{DD} = K_{DD} + L_{DD}$. The terms K_{DG}, K_{DD} and L_{DD} are given by

$$K_{DG} = \rho_1 \frac{\partial}{\partial\theta} \mathrm{E}_{x\sim p_\theta}\left[\nabla_\psi D_\psi(x)\right]\Big|_{\theta=\theta^*,\psi=\psi^*} \tag{6.13}$$
$$K_{DD} = \rho_2\, \mathrm{E}_{x\sim p_\mathcal{D}}\left[\nabla_\psi D_\psi(x) \nabla_\psi D_\psi(x)^\mathsf{T}\right]\big|_{\psi=\psi^*} \tag{6.14}$$
$$L_{DD} = \gamma\, \mathrm{E}_{x\sim p_\mathcal{D}}\left[\nabla_{\psi,x} D_\psi(x) \nabla_{\psi,x} D_\psi(x)^\mathsf{T}\right]\big|_{\psi=\psi^*} \tag{6.15}$$

Proof. The proof is almost the same as in Lemma 5.2, but we cannot use Assumption IV anymore in the proof. However, going over the proof of Lemma 5.2, we see that we can

instead use $D_{\psi^*}(x) = 0$ for all $x \in \mathcal{X}$ (Assumption I′) wherever we used Assumption IV. The expression of L_{DD} follows directly from Lemma 6.2. □

6.4 Convergence

After having analyzed the local structure of the regularized gradient vector field (6.7), we now want to apply Theorem 2.5 to show local convergence of both SimGD and AltGD for the regularized GAN training dynamics. To this end, we have to show that under Assumption I′, II′ and III′, $\mathcal{M}_G \times \tilde{\mathcal{M}}_D$ consists only of stationary points of $\tilde{v}_i(\cdot)$ and that $\tilde{v}_i'(\theta^*, \psi^*)$ has only eigenvalues with negative real-part orthogonal to $\mathrm{T}_{\theta^*}\mathcal{M}_G \times \mathrm{T}_{\psi^*}\tilde{\mathcal{M}}_D$.

Again, we first show that $\mathcal{M}_G \times \tilde{\mathcal{M}}_D$ consists only of stationary points of the regularized gradient vector fields $\tilde{v}_i(\cdot)$.

Lemma 6.4. *$\mathcal{M}_G \times \tilde{\mathcal{M}}_D$ consists only of stationary points of $\tilde{v}_i(\cdot)$ for $i \in \{1,2\}$, i.e. $\tilde{v}_i(\theta, \psi) = 0$ for all $(\theta, \psi) \in \mathcal{M}_G \times \tilde{\mathcal{M}}_D$.*

Proof. As in the proof of Lemma 5.3 we see that

$$\nabla_\psi \mathcal{L}(\theta^*, \psi^*) = \mathrm{E}_{x \sim p_\mathcal{D}} \left[\varphi_1'(0) \nabla_\psi D_\psi(x) - \varphi_2'(0) \nabla_\psi D_\psi(x) \right] = 0 \tag{6.16}$$

Now consider $\nabla_\theta \mathcal{L}(\theta, \psi)$. By Lemma 5.1 we have

$$\nabla_\theta \mathcal{L}(\theta, \psi) = \mathrm{E}_{z \sim p_0} \left[\varphi_1'(D_\psi(G_\theta(z))) \left[\frac{\partial G_\theta}{\partial \theta}(z) \right]^\mathsf{T} \cdot \nabla_x D_\psi(G_\theta(z)) \right] \tag{6.17}$$

However, for $\theta \in \mathcal{M}_G$ we have $p_\theta = p_\mathcal{D}$ and thus $G_\theta(z) \in \operatorname{supp} p_\mathcal{D}$. Moreover, for $\psi \in \tilde{\mathcal{M}}_D$ we have $\nabla_x D_\psi(x) = 0$ for all $x \in \operatorname{supp} p_\mathcal{D}$ and therefore $\nabla_x D_\psi(G_\theta(z)) = 0$. Thus, for $(\theta, \psi) \in \mathcal{M}_G \times \tilde{\mathcal{M}}_D$,

$$\nabla_\theta \mathcal{L}(\theta, \psi) = \mathrm{E}_{z \sim p_0} \left[\rho_1 \left[\frac{\partial G_\theta}{\partial \theta}(z) \right]^\mathsf{T} \cdot 0 \right] = 0 \tag{6.18}$$

Finally, consider $\nabla_\psi R_i(\theta, \psi)$. By Lemma 6.1, we have

$$\nabla_\psi R_1(\theta, \psi) = \gamma \, \mathrm{E}_{x \sim p_\mathcal{D}} \left[\nabla_{\psi,x} D_\psi(x) \nabla_x D_\psi(x) \right] \tag{6.19}$$

$$\nabla_\psi R_2(\theta, \psi) = \gamma \, \mathrm{E}_{x \sim p_\theta} \left[\nabla_{\psi,x} D_\psi(x) \nabla_x D_\psi(x) \right] \tag{6.20}$$

However, for $(\theta, \psi) \in \mathcal{M}_G \times \tilde{\mathcal{M}}_D$ we have $p_\theta = p_\mathcal{D}$ and $\nabla_x D_\psi(x) = 0$ for all $x \in \operatorname{supp} p_\mathcal{D}$. This shows that for $i \in \{1,2\}$

$$\nabla_\psi R_i(\theta, \psi) = \gamma \, \mathrm{E}_{x \sim p_\mathcal{D}} \left[\nabla_{\psi,x} D_\psi(x) \cdot 0 \right] = 0 \tag{6.21}$$

This shows the conclusion. □

Again, we can directly draw a conclusion about the interaction term K_{DG} from Lemma 6.4.

Lemma 6.5. *Let K_{DG} be defined as in Lemma 6.3. Then the following assertions hold:*

i) $K_{DG} w = 0$ *for all* $w \in \mathrm{T}_{\theta^*} \mathcal{M}_G$

ii) $K_{DG}^\mathsf{T} w = 0$ *for all* $w \in \mathrm{T}_{\psi^*} \tilde{\mathcal{M}}_D$

iii) $K_{DG} w \in (\mathrm{T}_{\psi^*} \tilde{\mathcal{M}}_D)^\perp$ *for all* $w \in \mathbb{R}^n$

iv) $K_{DG}^\mathsf{T} w \in (\mathrm{T}_{\theta^*} \mathcal{M}_G)^\perp$ *for all* $w \in \mathbb{R}^m$

Proof. The proof is identical to the proof of Lemma 5.4 except that we use Lemma 6.3 and Lemma 6.4 instead of Lemma 5.2 and Lemma 5.3. □

To prove local stability, we have to show that $\tilde{v}'(\theta^*, \psi^*)$ is well behaved when restricting it to the orthogonal complement of the tangent space of $\mathcal{M}_G \times \tilde{\mathcal{M}}_D$ at (θ^*, ψ^*). We therefore now derive results that are analogous to Lemma 5.6 and Lemma 5.7 from Chapter 5.

Lemma 6.6. *Assume that Assumption I′, II′ and III′ hold. If $w \neq 0$ is not in the tangent space of $\tilde{\mathcal{M}}_D$ at ψ^*, then $w^\mathsf{T} \tilde{K}_{DD} w > 0$.*

Proof. By Lemma 5.2, we have

$$w^\mathsf{T} K_{DD} w = \rho_2 \, \mathrm{E}_{x \sim p_\mathcal{D}} \left[\left(\nabla_\psi D_{\psi^*}(x)^\mathsf{T} w \right)^2 \right] \tag{6.22}$$

and by Lemma 6.3

$$w^\mathsf{T} L_{DD} w = \gamma \mathrm{E}_{x \sim p_\mathcal{D}} \left[\left\| \nabla_{x,\psi} D_{\psi^*}(x) w \right\|^2 \right] \tag{6.23}$$

By Assumption II, we have $\rho_2 > 0$. Hence, $w^\mathsf{T} \tilde{K}_{DD} w \geq 0$ and $w^\mathsf{T} \tilde{K}_{DD} w = 0$ implies

$$\nabla_\psi D_{\psi^*}(x) w = 0 \quad \text{and} \quad \nabla_{x,\psi} D_{\psi^*}(x) w = 0 \tag{6.24}$$

for all $x \in \operatorname{supp} p_\mathcal{D}$.

Recall that

$$\tilde{h}(\psi) := \mathrm{E}_{x \sim p_\mathcal{D}} \left[|D_\psi(x)|^2 + \|\nabla_x D_\psi(x)\|^2 \right] \tag{6.25}$$

Using the fact that $D_\psi(x) = 0$ and $\nabla_x D_\psi(x) = 0$ for $x \in \operatorname{supp} p_\mathcal{D}$, we see that the Hessian of $\tilde{h}(\psi)$ at ψ^* is

$$\nabla_\psi^2 \tilde{h}(\psi^*) = 2 \mathrm{E}_{x \sim p_\mathcal{D}} \left[\nabla_\psi D_\psi(x) \nabla_\psi D_\psi(x)^\mathsf{T} + \nabla_{\psi,x} D_\psi(x) \nabla_{\psi,x} D_\psi(x)^\mathsf{T} \right] \tag{6.26}$$

The second directional derivative $\partial_w^2 \tilde{h}(\psi)$ is therefore

$$\partial_w^2 \tilde{h}(\psi) = 2 \mathrm{E}_{x \sim p_\mathcal{D}} \left[|\nabla_\psi D_\psi(x)^\mathsf{T} w|^2 + \|\nabla_{x,\psi} D_\psi(x) w\|^2 \right] = 0 \tag{6.27}$$

By Assumption III, this can only hold if w is in the tangent space of $\tilde{\mathcal{M}}_D$ at ψ^*. □

We also need a result which is analogous to Lemma 5.7.

Lemma 6.7. *Assume that Assumption I′, II′ and III′ hold. If $w \in \mathbb{R}^n$ is not in the tangent space $\mathrm{T}_{\theta^*} \mathcal{M}_G$ of $\mathcal{M}_G$ at θ^*, then $K_{DG} w \notin \mathrm{T}_{\psi^*} \tilde{\mathcal{M}}_D$*

Proof. The proof is identical to the proof of Lemma 5.7, but we use Lemma 6.5 instead of Lemma 5.4. □

We are now ready to state our main convergence result for the regularized GAN training dynamics:

Theorem 6.8. *Assume Assumption I′, II′ and III′ hold for (θ^*, ψ^*) and that we regularize the discriminator using either $R_1(\cdot)$ or $R_2(\cdot)$. For small enough learning rates, SimGD and AltGD are then both convergent to $\mathcal{M}_G \times \tilde{\mathcal{M}}_D$ in a neighborhood of (θ^*, ψ^*). Moreover, the rate of convergence is at least linear.*

Proof. First note that by Lemma 6.4 $\mathcal{M}_G \times \tilde{\mathcal{M}}_D$ consists only of stationary points of the regularized gradient vector fields $\tilde{v}_i(\cdot)$ for $i \in \{0, 1\}$.

We now want to use Theorem 2.5 to show that both SimGD and AltGD are locally convergent to $\mathcal{M}_G \times \tilde{\mathcal{M}}_D$ for the regularized gradient vector fields $\tilde{v}_i(\cdot)$. To this end, let $\mathrm{T}_{\theta^*}\mathcal{M}_G$ and $\mathrm{T}_{\psi^*}\tilde{\mathcal{M}}_D$ denote the tangent spaces of $\mathcal{M}_G$ and $\tilde{\mathcal{M}}_D$ at θ^* and ψ^* and let $B_G \in \mathbb{R}^{n \times l_1}$ and $B_D \in \mathbb{R}^{m \times l_2}$ denote orthogonal bases of $(\mathrm{T}_{\theta^*}\mathcal{M}_G)^\perp$ and $(\mathrm{T}_{\psi^*}\tilde{\mathcal{M}}_D)^\perp$, respectively. Moreover, let

$$B := \begin{pmatrix} B_G & 0 \\ 0 & B_D \end{pmatrix} \in \mathbb{R}^{(n+m)\times(l_1+l_2)} \tag{6.28}$$

As in the proof of Theorem 5.8 we have to show that $B^\mathsf{T} \tilde{v}_i'(\theta^*, \psi^*) B$ has only eigenvalues with negative real-part.

Using Lemma 6.3, we see that $B^\mathsf{T} \tilde{v}_i'(\theta, \psi) B$ is

$$B^\mathsf{T} \tilde{v}_i'(\theta^*, \psi^*) B = \begin{pmatrix} 0 & -B_G^\mathsf{T} K_{DG}^\mathsf{T} B_D \\ B_D^\mathsf{T} K_{DG} B_G & -B_D^\mathsf{T} \tilde{K}_{DD} B_D \end{pmatrix} \tag{6.29}$$

As in the proof of Theorem 5.8 we now show that $B_D^\mathsf{T} \tilde{K}_{DD} B_D$ is positive definite and $B_D^\mathsf{T} K_{DG} B_G$ has full column rank.

To this end, first note that for any $w \in \mathbb{R}^{l_1}$, we have $B_D w \in (\mathrm{T}_{\psi^*}\tilde{\mathcal{M}}_D)^\perp$ and hence $B_D w \notin \mathrm{T}_{\psi^*}\tilde{\mathcal{M}}_D$ for $w \neq 0$. Therefore, by Lemma 6.6, we have

$$w^\mathsf{T} \left(B_D^\mathsf{T} \tilde{K}_{DD} B_D \right) w = (B_D w)^\mathsf{T} \tilde{K}_{DD} (B_D w) > 0 \tag{6.30}$$

for $w \neq 0$. As a result, we see that $B_D^\mathsf{T} \tilde{K}_{DD} B_D$ is symmetric positive definite.

Similarly, since $B_G w \notin \mathrm{T}_{\theta^*}\tilde{\mathcal{M}}_G$ for $w \neq 0$, we can apply Lemma 6.7 to obtain $K_{DG} B_G w \notin \mathrm{T}_{\psi^*}\tilde{\mathcal{M}}_D$. However, this implies that $K_{DG} B_G w$ has a non-zero component orthogonal to $\mathrm{T}_{\psi^*}\tilde{\mathcal{M}}_D$, i.e. $B_D^\mathsf{T} K_{DG} B_G w \neq 0$. Since $w \neq 0$ was arbitrary, this implies that $B_D^\mathsf{T} K_{DG} B_G$ has full column rank.

Lemma 5.5 now implies that all eigenvalues of $B^\mathsf{T} \tilde{v}_i'(\theta^*, \psi^*) B$ have negative real part. By Theorem 2.5, SimGD and AltGD are therefore both convergent to $\mathcal{M}_G \times \tilde{\mathcal{M}}_D$ near (θ^*, ψ^*) for small enough learning rates. Moreover, the rate of convergence is at least linear. □

Theorem 6.8 shows that GAN training with our gradient penalties is convergent when initialized sufficiently close to the equilibrium point. While this does not show that the

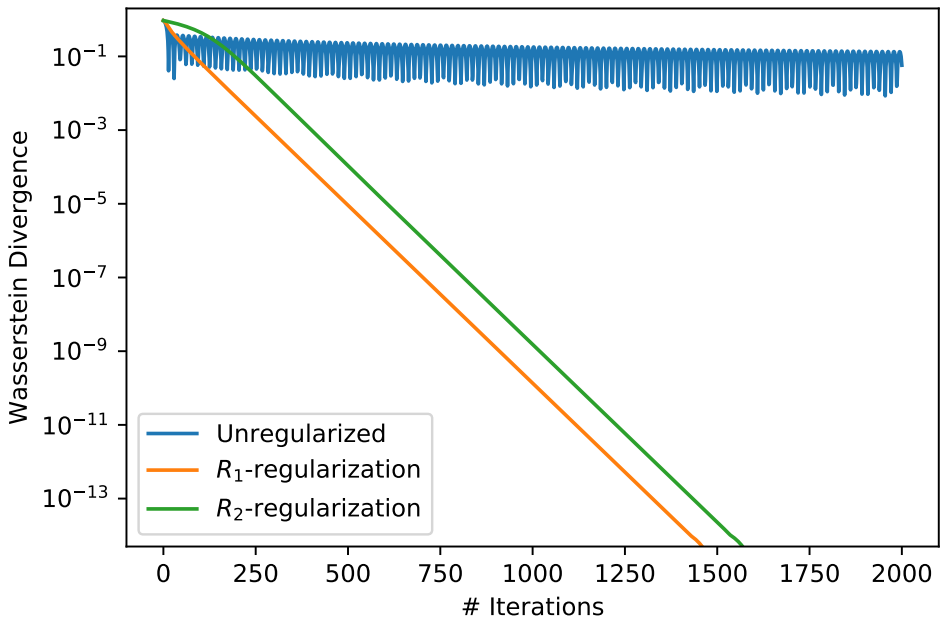


Figure 6.1: Four Points. *Quantitative results for the Four Points Example for $h = w = 1$. The graph shows the Wasserstein divergence between the target distribution p_{θ^*} and the generator distribution p_θ over the number of iterations. While unregularized GAN training does not converge, training with R_1- and R_2-regularization leads to convergence. For this plot we use AltGD, but the results for SimGD look similar.*

method is globally convergent, it at least shows that near the equilibrium the method is well-behaved.

6.5 Examples

Again, we consider some examples. More specifically, we reconsider the two non-convergent examples from Section 5.4 and discuss how zero-centered gradient penalties lead to local convergence in these cases.

Dirac-GAN For the Dirac-GAN, we have $\tilde{h}(\psi) = |\psi^\mathsf{T} x_0|^2 + \|\psi\|^2$. As a result, $\tilde{\mathcal{M}}_D = \{0\}$ and $\partial_w^2 \tilde{h}(\psi^*) = |w^\mathsf{T} x_0|^2 + \|w\|^2 \neq 0$ for $w \neq 0$. The conditions of Assumption I′, II′ and III′ are hence all satisfied and Theorem 6.8 therefore shows that the GAN training dynamics are convergent near (θ^*, ψ^*). Indeed, as we have seen in Chapter 4, the multidimensional Dirac-GAN with zero-centered gradient penalties is locally convergent.

Four Points Using Lemma 6.3 we see that in this case

$$K_{DD} = \rho_2 \begin{pmatrix} \theta_1^{*4} & \theta_1^{*2}\theta_2^{*2} \\ \theta_1^{*2}\theta_2^{*2} & \theta_2^{*4} \end{pmatrix} \tag{6.31}$$

$$L_{DD} = 4\gamma \begin{pmatrix} \theta_1^{*2} & 0 \\ 0 & \theta_2^{*2} \end{pmatrix} \tag{6.32}$$

$$K_{DG} = 2\rho_1 \begin{pmatrix} \theta_1^{*} & 0 \\ 0 & \theta_2^{*} \end{pmatrix} \tag{6.33}$$

In particular, we see that L_{DD} and K_{DG} have both full rank for $\theta_1^*, \theta_2^* \neq 0$. Lemma 6.3 and Lemma 5.5 hence imply that the Jacobian $v'(\theta^*, \psi^*)$ only has eigenvalues with negative real part. Both SimGD and AltGD are hence locally convergent near (θ^*, ψ^*). Indeed, a straightforward calculation shows that Assumption I′, II′ and III′ hold in this case and Theorem 6.8 is hence applicable. Figure 6.1 shows the Wasserstein divergence between p_θ and p_{θ^*} over the number of iterations for $h = w = 1$. As predicted theoretically, unregularized GAN training is not convergent, while R_1- and R_2-regularization lead to convergence.

In summary, we see that R_1- and R_2-regularization ensures convergence of the non-convergent examples from Section 5.4.

6.6 Extensions

In Theorem 6.8 we have proved local convergence of the regularized GAN training dynamics under Assumption I′, II′ and III′. However, practitioners often use variants of the training algorithm that do not satisfy these assumptions.

For example, in the original GAN paper, Goodfellow et al. [62] proposed a non-saturating version of GAN training which is not a zero-sum game anymore. Similarly, Wasserstein GANs (WGANs) [5] use linear activation functions $\varphi_1(t) = \varphi_2(t) = t$, which do not satisfy Assumption II′. Finally, Roth et al. [169] proposed a more complex version of our zero-centered gradient penalties and showed empirically that this regularizer stabilizes the GAN training dynamics.

However, as it turns out, our theory can be easily extended to these cases as well.

6.6.1 Nonsaturating GAN

We first discuss the nonsaturating GAN training dynamics, proposed by Goodfellow et al. [62]. In non-saturating GANs, the generator is trained to maximize

$$\mathrm{E}_{z\sim p_0}\left[\varphi_1(-D_\psi(G_\theta(z)))\right] \tag{6.34}$$

instead of minimizing

$$\mathrm{E}_{z\sim p_0}\left[\varphi_1(D_\psi(G_\theta(z)))\right] \tag{6.35}$$

Effectively, this means that $\varphi_1(t)$ is replaced with $-\varphi_1(-t)$ when training the generator. However, we have

Lemma 6.9. *When we replace φ_1 with some other differentiable function $\tilde{\varphi}_1$ such that $\varphi_1'(0) \cdot \tilde{\varphi}_1'(0) > 0$ when training the generator, the results of Theorem 6.8 still hold.*

Proof (Sketch). Almost all steps in the proof of Theorem 6.8 still hold. The only difference is that the Jacobian at (θ^*, ψ^*) is now

$$\tilde{v}_i'(\theta^*, \psi^*) = \begin{pmatrix} 0 & -\frac{\tilde{\varphi}_1'(0)}{\varphi_1'(0)} K_{DG}^\mathsf{T} \\ K_{DG} & \tilde{K}_{DD} \end{pmatrix}. \tag{6.36}$$

However, this implies that $B^\mathsf{T} \tilde{v}_i'(\theta^*, \psi^*) B$ is

$$B^\mathsf{T} \tilde{v}_i'(\theta^*, \psi^*) B = \begin{pmatrix} 0 & -\frac{\tilde{\varphi}_1'(0)}{\varphi_1'(0)} B_G^\mathsf{T} K_{DG}^\mathsf{T} B_D \\ B_D^\mathsf{T} K_{DG} B_G & -B_D^\mathsf{T} \tilde{K}_{DD} B_D \end{pmatrix} \tag{6.37}$$

Using Corollary A.4 with $h_1 = \frac{\tilde{\varphi}_1'(0)}{\varphi_1'(0)}$, we see that the eigenvalues of (6.37) are equal to the eigenvalues of

$$\begin{pmatrix} 0 & -\sqrt{\frac{\tilde{\varphi}_1'(0)}{\varphi_1'(0)}} B_G^\mathsf{T} K_{DG}^\mathsf{T} B_D \\ \sqrt{\frac{\tilde{\varphi}_1'(0)}{\varphi_1'(0)}} B_D^\mathsf{T} K_{DG} B_G & -B_D^\mathsf{T} \tilde{K}_{DD} B_D \end{pmatrix} \tag{6.38}$$

However, the matrix in (6.38) can be treated in the same way as the matrix in (6.29). The rest of the proof of Theorem 6.8 hence still applies. □

6.6.2 Linear Activations

In the proof of Theorem 6.8 we have assumed that (φ_1, φ_2) is strictly valid and therefore $\varphi_1''(0) + \varphi_2''(0) < 0$. This excludes the function $\varphi_1(t) = \varphi_2(t) = t$ which is used in WGANs. We now show that our convergence proof extends to the case where $\varphi_1(t) = \varphi_2(t) = t$ when we modify Assumption III′ as little bit.[2]

Lemma 6.10. *When we replace $\tilde{h}(\psi)$ with*

$$\tilde{h}(\psi) := \mathrm{E}_{x \sim p_\mathcal{D}} \left[\|\nabla_x D_\psi(x)\|^2 \right] \tag{6.39}$$

and $\tilde{\mathcal{M}}_D$ with $\tilde{\mathcal{M}}_D := \{\psi \mid \tilde{h}(\psi) = 0\}$ the results of Theorem 6.8 still hold for $\varphi_1(t) = \varphi_2(t) = t$.

Proof (Sketch). Almost everything in the proof of Theorem 6.8 still holds for these modified assumptions. The only thing that we have to show is that $\mathcal{M}_G \times \tilde{\mathcal{M}}_D$ still consists only of stationary points and that Lemma 6.6 still holds in this setting.

To see the former, note that by Lemma 5.1 we still have $\nabla_\theta \mathcal{L}(\theta, \psi) = 0$ for $(\theta, \psi) \in \mathcal{M}_G \times \tilde{\mathcal{M}}_D$, because we have $\nabla_x D_\psi(x) = 0$ for $\psi \in \tilde{\mathcal{M}}_D$ and $x \in \operatorname{supp} p_\mathcal{D}$. On the other hand, for $\varphi_1(t) = \varphi_2(t) = t$ we also have $\nabla_\psi \mathcal{L}(\theta, \psi) = 0$ if $\theta \in \mathcal{M}_G$, because for $\theta \in \mathcal{M}_G$

[2] We can also relax Assumption I′ to $D_{\psi^*}(x) = const.$ for all $x \in \mathcal{X}$ in this case.

the definition of $\mathcal{M}_G$ implies that $p_\theta = p_\mathcal{D}$ and hence, by Lemma 5.1,

$$\nabla_\psi \mathcal{L}(\theta, \psi) = \mathrm{E}_{x\sim p_\mathcal{D}}\left[\nabla_\psi D_\psi(x)\right] - \mathrm{E}_{x\sim p_\mathcal{D}}\left[\nabla_\psi D_\psi(x)\right] = 0 \tag{6.40}$$

To see why Lemma 6.6 still holds, first note that for $\varphi_1(t) = \varphi_2(t) = t$, we have $\rho_2 = 0$, so that, by Lemma 6.3, $K_{DD} = 0$. Hence,

$$w^\mathsf{T} \tilde{K}_{DD} w = w^\mathsf{T} L_{DD} w \tag{6.41}$$

We therefore have to show that $w^\mathsf{T} L_{DD} w \neq 0$ if w is not in the tangent space of $\tilde{\mathcal{M}}_D$. However, we have seen in the proof of Lemma 6.6 that

$$w^\mathsf{T} L_{DD} w = \gamma\, \mathrm{E}_{x\sim p_\mathcal{D}}\left[\left\|\nabla_{x,\psi} D_{\psi^*}(x) w\right\|^2\right]. \tag{6.42}$$

Hence $w^\mathsf{T} L_{DD} w = 0$ implies $\nabla_{x,\psi} D_{\psi^*}(x) w = 0$ for $x \in \operatorname{supp}\ p_\mathcal{D}$ and thus

$$\partial_w^2 \tilde{h}(\psi) = 2\,\mathrm{E}_{x\sim p_\mathcal{D}}\left[\|\nabla_{x,\psi} D_\psi(x) w\|^2\right] = 0. \tag{6.43}$$

By Assumption III, this can only be the case if w is in the tangent space of $\tilde{\mathcal{M}}_D$. This concludes the proof. □

6.6.3 Other Gradient Penalties

In Section 6.2, we have seen that both regularizers $R_1(\cdot)$ and $R_2(\cdot)$ from Section 6.1 make the GAN training dynamics locally convergent. In Section 6.1 we briefly discussed a similar, but more complex regularizer that was proposed by Roth et al. [169] who tried to find a computationally efficient approximation to instance noise. The regularizer (6.2) proposed by Roth et al. [169] is given by a linear combination of $R_1(\cdot)$ and $R_2(\cdot)$ where the weighting is adaptively chosen depending on the value of the discriminator $D_\psi(\cdot)$ at a data point x. The regularizer $R_{\text{Roth}}(\theta, \psi)$ is defined as

$$\gamma\, \mathrm{E}_{x\sim p_\theta}\left[(1 - \sigma(D_\psi(x)))^2 \|\nabla_x D_\psi(x)\|^2\right] + \gamma\, \mathrm{E}_{x\sim p_\mathcal{D}}\left[\sigma(D_\psi(x)))^2 \|\nabla_x D_\psi(x)\|^2\right] \tag{6.44}$$

Indeed, we can show that our convergence proof extends to this regularizer (and a slightly more general class of regularizers).

Lemma 6.11. *When we replace the regularization terms $R_1(\theta, \psi)$ and $R_2(\theta, \psi)$ with $R_3(\theta, \psi)$ given by*

$$\mathrm{E}_{x\sim p_\theta}\left[\nu_1(D_\psi(x)) \|\nabla_x D_\psi(x)\|^2\right] + \mathrm{E}_{x\sim p_\mathcal{D}}\left[\nu_2(D_\psi(x)) \|\nabla_x D_\psi(x)\|^2\right] \tag{6.45}$$

so that $\nu_1(0) + \nu_2(0) > 0$, the results of Theorem 6.8 still hold.

Proof (Sketch). Again, we have to show that $\mathcal{M}_G \times \tilde{\mathcal{M}}_D$ still consists only of stationary points and that Lemma 6.6 still holds in this setting.

However, by using $\nabla_x D_\psi(x) = 0$ for $x \in \operatorname{supp}\ p_\mathcal{D}$ and $\psi \in \tilde{\mathcal{M}}_D$, it is easy to see that $\nabla_\psi R_3(\theta, \psi) = 0$ for all $(\theta, \psi) \in \mathcal{M}_G \times \tilde{\mathcal{M}}_D$, which implies that $\mathcal{M}_G \times \tilde{\mathcal{M}}_D$ still consists

only of stationary points.

To see why Lemma 6.6 still holds in this setting, note that (after a little bit of algebra) we still have $\nabla_{\psi,\theta} R_3(\theta^*, \psi^*) = 0$ and

$$\nabla_\psi^2 R_3(\theta^*, \psi^*) = (\nu_1(0) + \nu_2(0)) L_{DD}. \tag{6.46}$$

because $\nabla_x D_{\psi^*}(x) = 0$ for all $x \in \mathcal{X}$. The proof of Lemma 6.6 therefore still applies in this setting. The rest of the proof is identical to the proof of Theorem 6.8. □

6.7 Conclusion

In this chapter we have introduced and analyzed R_1- and R_2-regularization. These regularizers are generalizations of the zero-centered gradient penalties from Chapter 3. Surprisingly, both regularizers exhibit a similar qualitative behavior for general GANs as for the Dirac-GAN. In particular, when we use R_1- or R_2-regularization, both Simultaneous Gradient Descent and Alternating Gradient Descent become locally asymptotically stable near the equilibrium point, even when the data distribution is not absolutely continuous. In the next chapter we take this analysis even further and derive formulas for convergence rates.

7 Convergence Rates

In Chapter 6 we have shown that Generative Adversarial Networks (GANs) are locally convergent when appropriately regularized. However, so far our theory is incomplete: in Chapter 6 we could only show the existence of a learning rate h that leads to convergence, but we could not make statements about the magnitude of h nor the rate of convergence. This is problematic, since convergence could in principle be too slow for a practical algorithm. For example, as discussed in Chapter 2, this can happen for Simultaneous Gradient Descent when the Jacobian of the gradient vector field at the equilibrium point has eigenvalues with large imaginary part. In general, to learn more about the rate of convergence, we have to gain a deeper understanding of the eigenvalues of the Jacobian of the update operator.

In this chapter we therefore derive eigenvalue bounds for the Jacobian of the update operator. This will enable us to derive convergence rates for (deterministic) GANs using Simultaneous Gradient Descent (SimGD) or Alternating Gradient Descent (AltGD). This analysis is insightful, as it highlights the role of the different learning rates as well as the conditioning of the GAN training dynamics.

7.1 Eigenvalue bounds

Recall that the Jacobian at the equilibrium point (Lemma 6.3) can be written as

$$v'(\theta^*, \psi^*) = \begin{pmatrix} 0 & -K_{DG}^\mathsf{T} \\ K_{DG} & -\tilde{K}_{DD} \end{pmatrix} \tag{7.1}$$

To derive convergence rates for SimGD and AltGD, we first have to understand the eigenvalues of the Jacobian at the equilibrium point in (7.1).

In this section we therefore first state eigenvalue bounds for matrices of the form

$$J = \begin{pmatrix} 0 & -P^\mathsf{T} \\ P & -Q \end{pmatrix} \tag{7.2}$$

where $Q \in \mathbb{R}^{m \times m}$ is symmetric positive semi-definite and $P \in \mathbb{R}^{m \times n}$. Note, that the Jacobian in (7.1) is a special case of (7.2) with $P = K_{DG}$ and $Q = \tilde{K}_{DD}$. For simplicity we assume $n = m$ in this section. While this condition is not necessarily true for every GAN, it is true for all examples (e.g. the Dirac-GAN) we have considered so far.

We would like to state the bounds in terms of the singular values of Q and P. In the following, we therefore denote the minimal and maximal singular value of Q by η_{min} and η_{max}, respectively. Since Q is symmetric positive semi-definite, the singular values of Q are equal to the eigenvalues of Q. Moreover, we denote the minimal and maximal singular

values of P by s_{min} and s_{max}, respectively. As we assume $n = m$ in this chapter, the squared singular values of P are equal to the eigenvalues of PP^T.

In prior work Nagarajan and Kolter [136] proved eigenvalue bounds for matrices as in (7.2). More specifically, assuming that P has full column-rank, Nagarajan and Kolter [136] could prove that eigenvalues of (7.2) satisfy

$$\Re(\lambda) \leq -\frac{\eta_{min}}{2} \qquad \text{if } \Im(\lambda) = 0 \tag{7.3}$$

$$\Re(\lambda) \leq -\frac{\lambda_{min}(P^\mathsf{T}P)}{\eta_{max}\eta_{min} + \lambda_{min}(P^\mathsf{T}P)} \qquad \text{if } \Im(\lambda) \neq 0 \tag{7.4}$$

where $\lambda_{min}(P^\mathsf{T}P)$ denotes the minimum eigenvalue of $P^\mathsf{T}P$.

Nagarajan and Kolter [136] did not assume $n = m$. However, it turns out that when we assume $n = m$, we can derive much tighter bounds. More specifically we reformulate the problem of finding eigenvalues of (7.2) as a *quadratic eigenvalue problem* [187]. Our new eigenvalue bounds are tight in the sense that no bound that depends only on the minimum and maximum singular values of P and Q can be tighter (Lemma C.3). Our approach also yields additional insights into the spectrum of the Jacobian at equilibrium points. In particular, we derive lower bounds for $\Re(\lambda)$ as well as lower and upper bounds for $\Im(\lambda)$, all of which do not directly follow from the analysis by Nagarajan and Kolter [136]. As a result, our new bounds enable us to derive convergence bounds for SimGD and AltGD in Section 7.2 and 7.3.

We first state our main result, which we prove in Appendix C.

Theorem 7.1. *Let*

$$J = \begin{pmatrix} 0 & -P^\mathsf{T} \\ P & -Q \end{pmatrix} \in \mathbb{R}^{2n \times 2n} \tag{7.5}$$

with $P \in \mathbb{R}^{n\times n}$ and $Q \in \mathbb{R}^{n\times n}$. Moreover, assume that Q is symmetric positive semi-definite and let η_{min} and η_{max} denote the minimum and maximum singular value of Q. Similarly, let s_{min} and s_{max} denote the smallest and largest singular value of P. Then the following assertions hold:

- *If $\frac{1}{2}\eta_{max} < s_{min}$, all eigenvalues of J satisfy $\Im(\lambda) \neq 0$.*
- *If $s_{max} \leq \frac{1}{2}\eta_{min}$, all eigenvalues of J satisfy $\Im(\lambda) = 0$.*
- *All eigenvalues λ of J with $\Im(\lambda) \neq 0$ satisfy*

$$\frac{-\eta_{max}}{2} \leq \Re(\lambda) \leq \frac{-\eta_{min}}{2} \tag{7.6}$$

- *All eigenvalues λ of J with $\Im(\lambda) = 0$ satisfy*

$$-\frac{\eta_{max}}{2} - \sqrt{\frac{\eta_{max}^2}{4} - s_{min}^2} \leq \Re(\lambda) \leq -\frac{\eta_{max}}{2} + \sqrt{\frac{\eta_{max}^2}{4} - s_{min}^2} \tag{7.7}$$

- *All eigenvalues λ of J satisfy*

$$\sqrt{s_{min}^2 - \frac{\eta_{max}^2}{4}} \leq |\Im(\lambda)| \leq \sqrt{s_{max}^2 - \frac{\eta_{min}^2}{4}} \tag{7.8}$$

  Here, the lower bound in (7.8) *holds as long as the expression inside the square root is non-negative, i.e. when $\frac{1}{2}\eta_{max} < s_{min}$.*

- *All eigenvalue with $\Im(\lambda) \neq 0$ satisfy*

$$s_{min} \leq |\lambda| \leq s_{max} \tag{7.9}$$

Proof. Please see Appendix C. □

Note that our upper bound in (7.6) for $\Im(\lambda) \neq 0$ is equal to the bound (7.3) derived by Nagarajan and Kolter [136] using different techniques. The bound (7.4) in Nagarajan and Kolter [136], however, is less tight than our bound, as the next lemma shows:

Lemma 7.2. *Our bound for $\Re(\lambda)$ when $\Im(\lambda) = 0$ in* (7.7) *can be further bounded by*

$$-\eta_{max} \leq \Re(\lambda) \leq -\frac{s_{min}^2}{\eta_{max}} \tag{7.10}$$

Proof. It is easy to see, that

$$-\eta_{max} \leq -\frac{\eta_{max}}{2} - \sqrt{\frac{\eta_{max}^2}{4} - s_{min}^2} \tag{7.11}$$

Hence, $-\eta_{max} \leq \Re(\lambda)$. On the other hand, we have for all $t \geq \alpha > 0$:

$$-t + \sqrt{t^2 - \alpha^2} = \frac{(-t + \sqrt{t^2 - \alpha^2})(t + \sqrt{t^2 - \alpha^2})}{t + \sqrt{t^2 - \alpha^2}} = \frac{-\alpha^2}{t + \sqrt{t^2 - \alpha^2}} \leq \frac{-\alpha^2}{2t} \tag{7.12}$$

Inserting $t = \frac{\eta_{max}}{2}$ and $\alpha = s_{min}$ yields $\Re(\lambda) \leq -\frac{s_{min}^2}{\eta_{max}}$, hence the assertion. □

Note that this bound is always lower than the bound (7.4) by Nagarajan and Kolter [136]: according to Lemma 7.2 all eigenvalues λ of J with $\Im(\lambda) \neq 0$ satisfy

$$\Re(\lambda) \leq -\frac{s_{min}^2}{\eta_{max}} = -\frac{\eta_{min} s_{min}^2}{\eta_{min}\eta_{max}} \leq -\frac{\eta_{min} s_{min}^2}{\eta_{max}\eta_{min} + s_{min}^2} \tag{7.13}$$

Note that for $m = n$ we have $\lambda_{min}(P^\top P) = s_{min}^2$, showing that our bound is tighter than the bound in (7.4).

When we derive convergence rates for AltGD, we need a variant of Theorem 7.1 which we can prove in a similar way.

Lemma 7.3. *Let*

$$J = \begin{pmatrix} 0 & -P^\mathsf{T} \\ P & -Q - hPP^\mathsf{T} \end{pmatrix} \in \mathbb{R}^{2n \times 2n} \tag{7.14}$$

with $h > 0$ and $P, Q \in \mathbb{R}^{n \times n}$. Assume that Q is symmetric positive semi-definite and let η_{min} and η_{max} denote the minimum and maximum singular value of Q. Similarly, let s_{min} and s_{max} denote the minimum and maximum singular value of P. Then the following holds:

- *Assume that one of the following conditions is true:*

  i) $h \geq \frac{2}{s_{max}+s_{min}}$ *and* $\eta_{max} + hs_{max}^2 \leq 2s_{max}$

  ii) $h \leq \frac{2}{s_{max}+s_{min}}$ *and* $\eta_{max} + hs_{min}^2 \leq 2s_{min}$

  Then all eigenvalues of J have non-zero imaginary part.

- *All eigenvalues λ of J with non-zero imaginary part satisfy*

$$|1 + h\lambda|^2 \leq 1 - h\eta_{min} \tag{7.15}$$

Proof. Please see Appendix C. □

7.2 Simultaneous Gradient Descent

In this section we show how we can use the theory developed in Section 7.1 to derive tight convergence bounds for SimGD in GAN optimization. This analysis leads to insights into the effect of different learning rates for the generator and discriminator. Moreover, it might be useful as an inspiration to develop novel preconditioners for GAN optimization.

Let h_g and h_d denote the learning rates of the generator and discriminator, respectively. For simplicity, we also define $h := \sqrt{h_g h_d}$ and $\tau = \sqrt{h_d / h_g}$. Note that this implies $h_g = h/\tau$ and $h_d = h\,\tau$, i.e. we can regard h as the learning rate of the whole system and τ as a modifier which adjusts the learning rates for the generator and discriminator. Indeed, $\tau > 1$ implies that the discriminator has a higher learning rate than the generator and $\tau < 1$ implies that the discriminator has a lower learning rate than the generator.

As in Chapter 6, let $\mathrm{T}_{\theta^*}\mathcal{M}_G$ and $\mathrm{T}_{\psi^*}\tilde{\mathcal{M}}_D$ denote the tangent spaces of $\mathcal{M}_G$ and $\tilde{\mathcal{M}}_D$ at θ^* and ψ^*. Moreover, let $B_G \in \mathbb{R}^{n \times l_1}$ and $B_D \in \mathbb{R}^{m \times l_2}$ such that the columns of B_G and B_D define orthogonal bases of $(\mathrm{T}_{\theta^*}\mathcal{M}_G)^\perp$ and $(\mathrm{T}_{\psi^*}\tilde{\mathcal{M}}_D)^\perp$, respectively. Again, we define

$$B := \begin{pmatrix} B_G & 0 \\ 0 & B_D \end{pmatrix} \in \mathbb{R}^{(n+m) \times (l_1 + l_2)} \tag{7.16}$$

Moreover, we assume $l_1 = l_2 =: l$. To derive convergence rates for (deterministic) SimGD, we have to understand the eigenvalues of the Jacobian of the corresponding update operator orthogonal to $\mathrm{T}_{\theta^*}\mathcal{M}_G$ and $\mathrm{T}_{\psi^*}\tilde{\mathcal{M}}_D$, i.e. of $B^\mathsf{T} F'(\theta^*, \psi^*) B$.

First, we prove the following general convergence statement for SimGD.

Lemma 7.4. *Let $h := \sqrt{h_g h_d}$ and $\tau := \sqrt{\frac{h_d}{h_g}}$. Moreover, let K_{DG} and $\tilde{K}_{DD}$ be defined as in Lemma 6.3 and let*

$$J := \begin{pmatrix} 0 & -P^\mathsf{T} \\ P & -\tau Q \end{pmatrix} \tag{7.17}$$

with $P := B_D^\mathsf{T} K_{DG} B_G$ and $Q := B_D^\mathsf{T} \tilde{K}_{DD} B_D$. If all eigenvalues λ of J satisfy $|1+h\lambda| \leq \alpha < 1$, then SimGD converges in a neighborhood of (θ^*, ψ^*) with convergence rate at least α.

Proof. Recall from Chapter 6 that the update operator F for SimGD is given by

$$F(\theta, \psi) = \begin{pmatrix} \theta - h_g \nabla_\theta \mathcal{L}(\theta, \psi) \\ \psi + h_d \nabla_\psi (\mathcal{L}(\theta, \psi) - R_i(\theta, \psi)) \end{pmatrix} \tag{7.18}$$

The Jacobian of the update operator F at (θ^*, ψ^*) is hence given by

$$F'(\theta^*, \psi^*) = \begin{pmatrix} I & -h_g K_{DG}^\mathsf{T} \\ h_d K_{DG} & I - h_d \tilde{K}_{DD} \end{pmatrix} \tag{7.19}$$

As a consequence,

$$B^\mathsf{T} F'(\theta^*, \psi^*) B = I + \begin{pmatrix} 0 & -h_g P^\mathsf{T} \\ h_d P & -h_d Q \end{pmatrix} \tag{7.20}$$

with $P = B_D^\mathsf{T} K_{DG} B_G$ and $Q = B_D^\mathsf{T} \tilde{K}_{DD} B_D$.

By Corollary A.4, the eigenvalues of $B^\mathsf{T} F'(\theta^*, \psi^*) B$ are hence equal to the eigenvalues of $I + hJ$ with J as in (7.17) and $h = \sqrt{h_g h_d}$. By Theorem B.4, SimGD hence converges with convergence rate at least α, if $|1+h\lambda| \leq \alpha < 1$ for all eigenvalues λ of J. □

Note that this matrix in (7.17) is of the form (7.2) with Q replaced by τQ and we can hence apply Theorem 7.1 to it.

In the following, let $J := B^\mathsf{T} \tilde{v}_i'(\theta^*, \psi^*) B$, $Q := B_D^\mathsf{T} \tilde{K}_{DD} B_D$ and $P := B_D^\mathsf{T} K_{DG} B_G$. Moreover, let η_{min} and η_{max} denote the minimum and maximum singular value of Q. Similarly, let s_{min} and s_{max} denote the minimum and maximum singular value of P. We now come to our first result on the convergence rate of SimGD:

Lemma 7.5. *Assume that $\tau \leq \frac{2 s_{min}}{\eta_{max}}$ and $h < \frac{\tau \eta_{min}}{s_{max}^2}$. Then SimGD is convergent with rate at least*

$$\alpha_{s,1}(h, \tau) = \sqrt{1 - h\tau\eta_{min} + h^2 s_{max}^2} \tag{7.21}$$

Proof. The eigenvalues of the Jacobian of the update operator for SimGD are given by $1 + h\lambda$ where λ denote the eigenvalues of

$$J := \begin{pmatrix} 0 & -P^\mathsf{T} \\ P & -\tau Q \end{pmatrix} \tag{7.22}$$

By Theorem 7.1, all eigenvalues of J have non-zero imaginary part if $\frac{1}{2}\tau\eta_{max} < s_{min}$, i.e. if $\tau < \frac{2 s_{min}}{\eta_{max}}$. In this case,

$$|1 + h\lambda|^2 = 1 + 2h\Re(\lambda) + h^2 |\lambda|^2 \tag{7.23}$$

By Theorem 7.1, we know that $\Re(\lambda) \leq -\frac{1}{2}\tau\eta_{min}$ and $|\lambda| \leq s_{max}$. Hence,

$$|1+h\lambda|^2 \leq 1 - h\tau\eta_{min} + h^2 s_{max}^2 \tag{7.24}$$

Note that this even also holds for $\tau = \frac{2s_{min}}{\eta_{max}}$, since in this case all eigenvalues λ with $\Im(\lambda) = 0$ satisfy $\Re(\lambda) = -\frac{\tau\eta_{max}}{2}$ by Theorem 7.1 and, as a consequence, $|\lambda| = \frac{\tau\eta_{max}}{2} = s_{min}$. Therefore, we still have $\Re(\lambda) \leq -\frac{\tau\eta_{min}}{2}$ and $|\lambda| \leq s_{max}$.

The expression in (7.24) is smaller than 1 if and only if $h < \frac{\tau\eta_{min}}{s_{max}^2}$. Together with Lemma 7.4, this yields the assertion. □

Lemma 7.6. *For a fixed $\tau \leq \frac{2s_{min}}{\eta_{max}}$, the bound in* (7.21) *is minimal as a function of $h > 0$ for $h^* = \frac{\tau\eta_{min}}{2s_{max}^2}$ in which case*

$$\alpha_{s,1}(h^*, \tau) = \sqrt{1 - \left(\frac{\eta_{min}\tau}{2s_{max}}\right)^2} \tag{7.25}$$

Moreover, (7.25) *is minimal as a function of τ for $\tau^* = \frac{2s_{min}}{\eta_{max}}$ and its minimum is given by*

$$\alpha_s^* = \sqrt{1 - \left(\frac{\eta_{min}s_{min}}{\eta_{max}s_{max}}\right)^2} \tag{7.26}$$

Proof. To obtain the optimal values for h and τ, we again look at the convergence rate in (7.24). Recall that the bound in (7.24) holds if $\tau \leq \frac{2s_{min}}{\eta_{max}}$. Optimizing (7.24) in terms of h yields $h^* = \frac{\eta_{min}\tau}{2s_{max}^2}$ and

$$\alpha_{s,1}(h^*, \tau) = \sqrt{1 - \left(\frac{\eta_{min}\tau}{2s_{max}}\right)^2} \tag{7.27}$$

Optimizing this in terms of τ under the constraint $\tau \leq \frac{2s_{min}}{\eta_{max}}$ yields $\tau^* = \frac{2s_{min}}{\eta_{max}}$ and

$$\alpha_{s,1}(h^*, \tau^*) = \sqrt{1 - \left(\frac{\eta_{min}s_{min}}{\eta_{max}s_{max}}\right)^2} \tag{7.28}$$

□

So far, we have analyzed the regime $\tau \leq \frac{2s_{min}}{\eta_{max}}$ where all eigenvalues of J have non-zero imaginary part. Instead of restricting our analysis to this case, we can also try to control the real eigenvalues. We therefore now investigate the case where all eigenvalues are on the real axis.

Lemma 7.7. *Assume $\tau \geq \frac{2s_{max}}{\eta_{min}}$. Let*

$$b^- := \frac{\tau\eta_{max}}{2} - \sqrt{\frac{\tau^2\eta_{max}^2}{4} - s_{min}^2} \quad \textit{and} \quad b^+ := \frac{\tau\eta_{max}}{2} + \sqrt{\frac{\tau^2\eta_{max}^2}{4} - s_{min}^2} \tag{7.29}$$

and assume $h < \frac{2}{b^+}$. *Then SimGD is convergent with rate at least*

$$\alpha_{s,2}(h,\tau) = \begin{cases} 1 - hb^- & \text{if } h \leq \frac{2}{\tau\eta_{max}} \\ hb^+ - 1 & \text{else} \end{cases} \tag{7.30}$$

Proof. To prove the lemma, we again use Lemma 7.4.

If $\tau \geq \frac{2s_{max}}{\eta_{min}}$, i.e. $s_{max} \leq \frac{1}{2}\tau\eta_{min}$, all eigenvalues λ of J are real-valued by Theorem 7.1 and

$$-b^+ \leq \lambda \leq -b^- \tag{7.31}$$

Hence, for all eigenvalues λ of J,

$$|1 + h\lambda| \leq \max\left(|1 - hb^+|, |1 - hb^-|\right) \tag{7.32}$$

Let $r := \frac{\tau\eta_{max}}{2}$ and $s := \sqrt{r^2 - s_{min}^2}$. Then $b^- = r - s$ and $b^+ = r + s$. As a consequence,

$$(1 - hb^-)^2 = (1 - hr)^2 + h^2 s^2 + 2(1 - hr)hs \tag{7.33}$$

and similarly,

$$(1 - hb^+)^2 = (1 - hr)^2 + h^2 s^2 - 2(1 - hr)hs \tag{7.34}$$

Since $r, s \geq 0$, this implies that

$$\max\left(|1 - hb^+|, |1 - hb^-|\right) = \begin{cases} |1 - hb^-| & \text{if } h \leq \frac{1}{r} \\ |1 - hb^+| & \text{else} \end{cases} \tag{7.35}$$

Moreover, for $h \leq \frac{1}{r}$, we have

$$1 - hb^- = (1 - hr) + hs \geq 0 \tag{7.36}$$

and hence $|1 - hb^-| = 1 - hb^-$. Similarly, for $h \geq \frac{1}{r}$, we have

$$1 - hb^+ = (1 - hr) - hs \leq 0 \tag{7.37}$$

and hence $|1 - hb^+| = hb^+ - 1$.

All in all, this show

$$|1 + h\lambda| \leq \begin{cases} 1 - hb^- & \text{if } h \leq \frac{2}{\tau\eta_{max}} \\ hb^+ - 1 & \text{else} \end{cases} \tag{7.38}$$

and hence (7.30) as required. Note that $hb^+ - 1 < 1$ if and only if $h < \frac{2}{b^+}$. □

Lemma 7.8. *For a fixed* $\tau \geq \frac{2s_{max}}{\eta_{min}}$, *the bound in* (7.30) *is minimal as a function of* $h > 0$ *for*

$h^* = \frac{2}{\tau\eta_{max}}$ in which case

$$\alpha_{s,2}(h^*, \tau) = \sqrt{1 - \left(\frac{2s_{min}}{\tau\eta_{max}}\right)^2} \tag{7.39}$$

Moreover, (7.39) *is minimal as a function of τ for $\tau^* = \frac{2s_{max}}{\eta_{min}}$ and its minimum is given by*

$$\alpha_s^* = \sqrt{1 - \left(\frac{\eta_{min}s_{min}}{\eta_{max}s_{max}}\right)^2} \tag{7.40}$$

Proof. Clearly, the minimum of the bound $\alpha_{s,2}(h, \tau)$ in (7.30) is obtained for $h^* = \frac{2}{\tau\eta_{max}}$, in which case

$$\alpha_{s,2}(h^*, \tau) = 1 - hb^- = hb^+ - 1 = \sqrt{1 - \left(\frac{2s_{min}}{\tau\eta_{max}}\right)^2} \tag{7.41}$$

Minimizing this with respect to τ under the constraint $\tau \geq \frac{2s_{max}}{\eta_{min}}$ yields $\tau = \frac{2s_{max}}{\eta_{min}}$ and

$$\alpha_{s,2}(h^*, \tau^*) = \sqrt{1 - \left(\frac{\eta_{min}s_{min}}{\eta_{max}s_{max}}\right)^2} \tag{7.42}$$

□

So far, we have considered two regimes for τ: in the first regime ($\tau \leq \frac{2s_{min}}{\eta_{min}}$), all eigenvalues of J have non-zero real part. In the second regime ($\tau \geq \frac{2s_{max}}{\eta_{max}}$), all eigenvalues lie on the real axis. Interesting, when optimizing h and τ in both regimes, we obtain exactly the same convergence rate for SimGD.

This directly leads to the question if this convergence rate is optimal or if we can achieve a better convergence rate for $\frac{2s_{min}}{\eta_{max}} < \tau < \frac{2s_{max}}{\eta_{min}}$. Indeed, naively setting (7.25) equal to (7.39) and solving for τ would yield the better rate

$$\sqrt{1 - \frac{s_{min}}{s_{max}}\frac{\eta_{min}}{\eta_{max}}} \tag{7.43}$$

Unfortunately, however, this would require conflicting optimal learning rates for (7.25) and (7.39).[1] Nonetheless, we can still find good values for τ and h that yield a better convergence rate than in (7.26) and (7.40).

Lemma 7.9. *For $\tau = \frac{2s_{max}}{\sqrt{\eta_{min}\eta_{max}}}$ and $h = \sqrt{\frac{\eta_{min}}{\eta_{max}}}\frac{1}{s_{max}}$, SimGD is locally convergent with rate at least*

$$\alpha_s^{**} = \sqrt{1 - \left(\frac{s_{min}}{s_{max}}\right)^2 \frac{\eta_{min}}{\eta_{max}}} \tag{7.44}$$

[1] Nonetheless, it is easy to see that this is a lower bound (7.43) on the best achievable convergence rate for SimGD on arbitrary GANs.

This corresponds to the learning rates $h_g = \frac{\eta_{min}}{2s_{max}^2}$. *and* $h_d = \frac{2}{\eta_{max}}$ *for the generator and discriminator, respectively.*

Proof. As in Lemma 7.7, we see that for all eigenvalues λ with non-zero imaginary part (independently of τ)

$$|1 + h\lambda| \leq \sqrt{1 - h\tau\eta_{min} + h^2 s_{max}^2} \tag{7.45}$$

Similarly, as in Lemma 7.7, for all λ with zero imaginary part

$$|1 + h\lambda| \leq \begin{cases} 1 - hb^- & \text{if } h \leq \frac{2}{\tau\eta_{max}} \\ hb^+ - 1 & \text{else} \end{cases} \tag{7.46}$$

with b^-, b^+ as in Lemma 7.7. Moreover, as in Lemma 7.6, we see that the bound in (7.45) is optimal for $h_1^* = \frac{\tau\eta_{min}}{2s_{max}^2}$. Similarly, as in Lemma 7.8, we see that the bound in (7.46) is optimal for $h_2^* = \frac{2}{\tau\eta_{max}}$.

We can find a combined bound, by setting h_1^* and h_2^* equal, i.e.

$$\frac{\tau\eta_{min}}{2s_{max}^2} = \frac{2}{\tau\eta_{max}} \tag{7.47}$$

Solving for τ, yields $\tau = \frac{2s_{max}}{\sqrt{\eta_{min}\eta_{max}}}$. The bound in (7.45) becomes

$$\sqrt{1 - \frac{\eta_{min}}{\eta_{max}}} \tag{7.48}$$

Similarly, the bound in (7.46) becomes

$$\sqrt{1 - \left(\frac{s_{min}}{s_{max}}\right)^2 \frac{\eta_{min}}{\eta_{max}}} \tag{7.49}$$

The maximum of these two expression (7.48) and (7.49) is clearly given by (7.49), yielding the assertion.

To obtain the corresponding expressions for h_g and h_d, use $h_g = h\tau$ and $h_d = \frac{h}{\tau}$. □

7.3 Alternating Gradient Descent

After having analyzed SimGD, we now turn our attention towards AltGD. To this end, we first prove an analogues lemma to Lemma 7.4 that allows us to derive convergence rates for AltGD.

Lemma 7.10. *Let* $h := \sqrt{h_g h_d}$ *and* $\tau := \sqrt{\frac{h_d}{h_g}}$. *Moreover, define*

$$J := \begin{pmatrix} 0 & -P^\mathsf{T} \\ P & -\tau Q - hPP^\mathsf{T} \end{pmatrix} \tag{7.50}$$

with $P := B_D^\mathsf{T} K_{DG} B_G$ *and* $Q = B_D^\mathsf{T} \tilde{K}_{DD} B_D$. *Assume that all eigenvalues* λ *of* J *satisfy* $|1 + h\lambda| \leq \alpha < 1$. *Then AltGD converges linearly in a neighborhood of* (θ^*, ψ^*) *with convergence rate at least* α.

Proof. Recall that the update operator for the generator and discriminator are given by

$$F_G(\theta, \psi) = \begin{pmatrix} \theta - h_g \nabla_\theta \mathcal{L}(\theta, \psi) \\ \psi \end{pmatrix} \tag{7.51}$$

$$F_D(\theta, \psi) = \begin{pmatrix} \theta \\ \psi + h_d \nabla_\psi (\mathcal{L}(\theta, \psi) - R_i(\theta, \psi)) \end{pmatrix} \tag{7.52}$$

respectively.

As a result, the Jacobians are given by

$$F_G'(\theta^*, \psi^*) = \begin{pmatrix} I & -h_g K_{DG}^\mathsf{T} \\ 0 & I \end{pmatrix} \quad \text{and} \quad F_D'(\theta^*, \psi^*) = \begin{pmatrix} I & 0 \\ h_d K_{DG} & I - h_d \tilde{K}_{DD} \end{pmatrix} \tag{7.53}$$

The Jacobian of the combined update operator is hence given by

$$F'(\theta^*, \psi^*) = F_D'(\theta^*, \psi^*) \cdot F_G'(\theta^*, \psi^*) = \begin{pmatrix} I & -h_g K_{DG}^\mathsf{T} \\ h_d K_{DG} & I - h_d \tilde{K}_{DD} - h_g h_d K_{DG} K_{DG}^\mathsf{T} \end{pmatrix} \tag{7.54}$$

Lemma 6.5 shows that $K_{DG}^\mathsf{T} w \in (\mathrm{T}_{\theta^*} \mathcal{M}_G)^\perp$ for all $w \in \mathbb{R}^l$. Since $B_G B_G^\mathsf{T}$ is the projection onto $(\mathrm{T}_{\theta^*} \mathcal{M}_G)^\perp$, this shows that

$$B_D^\mathsf{T} (K_{DG} K_{DG}^\mathsf{T}) B_D = B_D^\mathsf{T} K_{DG} (B_G B_G^\mathsf{T}) K_{DG}^\mathsf{T} B_D = P P^\mathsf{T} \tag{7.55}$$

As simple calculation therefore yields

$$B^\mathsf{T} F'(\theta^*, \psi^*) B = I + \begin{pmatrix} 0 & -h_g P^\mathsf{T} \\ h_d P & -h_d Q - h_d h_g P P^\mathsf{T} \end{pmatrix} \tag{7.56}$$

Using Corollary A.4, we see that the eigenvalues of $B^\mathsf{T} F'(\theta^*, \psi^*) B$ are hence equal to the eigenvalues of $I + hJ$ with J as in (7.50). The eigenvalues are therefore $1 + h\lambda$ with λ the eigenvalues of J. By Theorem B.4, AltGD hence converges linearly with rate at least α, if $|1 + h\lambda| \leq \alpha < 1$ for all λ. □

Similarly to the previous section, we could apply Theorem 7.1 to (7.50) to obtain convergence rates for AltGD. However, it turns out that we can obtain better rates by applying Lemma 7.3 instead. The reason is that Lemma 7.3 better exploits the structure of the Jacobian in (7.50).

Lemma 7.11. *Let* $h := \sqrt{h_g h_d}$ *and* $\tau := \sqrt{\frac{h_d}{h_g}}$. *Assume that one of the following conditions hold:*

i) $h \geq \frac{2}{s_{max} + s_{min}}$ *and* $\tau \eta_{max} + h s_{max}^2 \leq 2 s_{max}$

ii) $h \leq \frac{2}{s_{max} + s_{min}}$ *and* $\tau \eta_{max} + h s_{min}^2 \leq 2 s_{min}$.

AltGD is then convergent with rate at least

$$\alpha_a(h,\tau) = \sqrt{1 - h\tau\eta_{min}} \tag{7.57}$$

Proof. This is a direct application of Lemma 7.10 and Lemma 7.3. □

Lemma 7.12. *For* $h^* = \frac{2}{s_{min}+s_{max}}$ *and* $\tau^* = \frac{2s_{min}s_{max}}{\eta_{max}(s_{min}+s_{max})}$ *the bound in* (7.57) *is optimal and the system converges with rate at least*

$$\alpha_a^* = \sqrt{1 - \frac{4s_{min}s_{max}}{(s_{min}+s_{max})^2}\frac{\eta_{min}}{\eta_{max}}} \tag{7.58}$$

This corresponds to learning rates $h_d = \frac{4s_{min}s_{max}}{\eta_{max}(s_{min}+s_{max})^2}$ *and* $h_g = \frac{\eta_{max}}{s_{min}s_{max}}$.

Proof. Consider the case $h \leq \frac{2}{s_{min}+s_{max}}$. Since we can always improve the bound (7.57) by increasing τ, we can eliminate τ by setting

$$\tau = \frac{2s_{min} - hs_{min}^2}{\eta_{max}} \tag{7.59}$$

Minimizing (7.57) hence means to maximize

$$h\tau\eta_{min} = \frac{\eta_{min}}{\eta_{max}}(2s_{min}h - h^2 s_{min}^2) \tag{7.60}$$

as a function of h. The maximum of (7.60) as a function of h without constraint is at $h = \frac{1}{s_{min}}$. However, this is bigger than $\frac{2}{s_{min}+s_{max}}$. The maximum under the constraint $h \leq \frac{2}{s_{min}+s_{max}}$ is hence at $h = \frac{2}{s_{min}+s_{max}}$.

Similarly, we see that for the case $h \geq \frac{2}{s_{min}+s_{max}}$, the maximum lies at $h = \frac{2}{s_{min}+s_{max}}$. In both cases

$$h\tau\eta_{min} = \frac{4s_{min}s_{max}}{(s_{min}+s_{max})^2}\frac{\eta_{min}}{\eta_{max}} \tag{7.61}$$

□

While the convergence rate α_a^* in (7.58) looks similar to the convergence rate α_s^{**} in (7.44), it is actually much better. Indeed, since $\frac{s_{min}}{s_{max}} \leq 1$ we have

$$\alpha_a^* = \sqrt{1 - \frac{4\frac{s_{min}}{s_{max}}}{\left(1+\frac{s_{min}}{s_{max}}\right)^2}\frac{\eta_{min}}{\eta_{max}}} \leq \sqrt{1 - \frac{s_{min}}{s_{max}}\frac{\eta_{min}}{\eta_{max}}} \leq \alpha_s^{**} \tag{7.62}$$

Indeed, α_s^{**} depends on $\frac{s_{min}}{s_{max}}$ in a quadratic way, whereas α_a^* depends on $\frac{s_{min}}{s_{max}}$ in a linear way. Moreover, as we have seen in Section 7.2, this bound is better than the best achievable rate for SimGD. This analysis hence highlights the benefits of AltGD over SimGD for training GANs.

7.4 Conclusion

In this chapter we have analyzed the eigenvalues of the Jacobian of the gradient vector field to derive convergence rates for (deterministic) GANs with finite learning rates. While these results are currently more of theoretical value, we believe that they provide important insights into the conditioning of the training dynamics. An important possible application of our theory would be, for example, to derive preconditioners that are both fast to compute and that make the system better conditioned. In addition, our analysis also highlights the benefits of AltGD over SimGD for training GANs, raising the question if we can derive even better training schemes. We hope that our analysis paves the way for further research in this direction.

8 Experimental Results

In the previous chapters we have derived a general convergence theory for Generative Adversarial Networks (GANs). In particular, in Chapter 5 we have seen that GAN training for absolutely continuous data and generator distributions is exponentially stable. However, in Chapter 3 and 4 we have derived simple counter examples showing that GAN training is not necessarily convergent when this condition is not met. Indeed, the Manifold Hypothesis [119], which is the basis for dimensionality reduction techniques, states that most real-world distributions lie in the vicinity of low dimensional manifolds. As a result, the convergence result from Chapter 5 is often not applicable to unregularized GAN training for real-world data distributions.

Fortunately, however, we have seen in Chapter 6 that it is possible to design simple regularizers that make GAN training locally convergent even when the data distribution and generator distribution lie on lower dimensional manifolds.

In this chapter we test this result experimentally. To this end, we apply our new regularizers both to low dimensional data distributions as well as real-world image distributions.[1]

8.1 Simple 2D Distributions

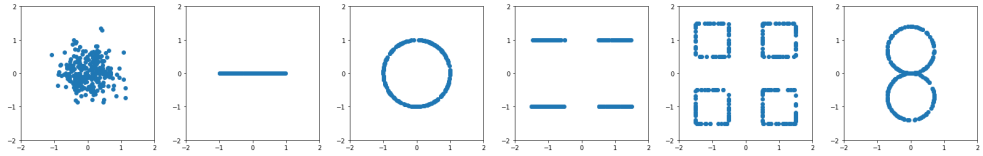
Measuring convergence for GANs is hard for high-dimensional problems, because defining a metric that can reliably determine convergence is non-trivial. We therefore first examine the behavior of different regularizers on simple 2D examples, where we can assess convergence using an estimate of the Wasserstein divergence.

8.1.1 Experimental Setup

In our first experiment, we compare unregularized GAN training [62], WGAN-GP [64] as well as GAN training with R_1- and R_2-regularization on simple 2D distributions. For a fair comparison, we run WGAN-GP both with one and five discriminator updates per generator update (as recommended by Gulrajani et al. [64]).

We test the five different training algorithms on six different 2D data distributions for six different GAN architectures. The six data distributions are visualized in Figure 8.1. All six GAN architectures consist of 4-layer fully-connected neural networks for both the generator and discriminator, where we select the number of hidden units from $\{8, 16, 32\}$ and use either Leaky-ReLU (i.e. $t \to \max(t, 0.2t)$) or Tanh-activation functions. Except for WGAN-GP, we always use the nonsaturating GAN-objective introduced by Goodfellow et al. [62]

[1] The code to reproduce the experiments presented in this chapter can be found under `https://github.com/LMescheder/gan_stability`.



(a) *(b)* *(c)* *(d)* *(e)* *(f)*

Figure 8.1: Distributions for 2D Experiments. *The four 2D data distributions on which we test the different algorithms: (a) Gaussian, (b) line, (c) circle, (d) four lines, (e) four squares, (f) eight.*

	unregularized	WGAN-GP ($n_d = 1$)	WGAN-GP ($n_d = 5$)	R_1-regularized	R_2-regularized
Gaussian	0.087	0.479	0.078	**0.049**	0.050
line	0.037	0.350	0.038	0.030	**0.028**
circle	0.059	0.307	0.067	**0.056**	0.059
four lines	0.180	0.214	0.121	0.107	**0.099**
four squares	0.232	0.373	0.159	0.157	**0.151**
eight	**0.079**	0.390	0.088	0.080	0.082
mean	0.112	0.352	0.092	0.080	**0.078**

Table 8.1: Results for 2D Experiments. *Wasserstein divergence between generator distribution and true data distribution for six different 2D distributions and five different training methods. The results are averaged over six different architectures.*

for training the generator. For WGAN-GP we use the linear generator and discriminator objectives introduced by Gulrajani et al. [64].

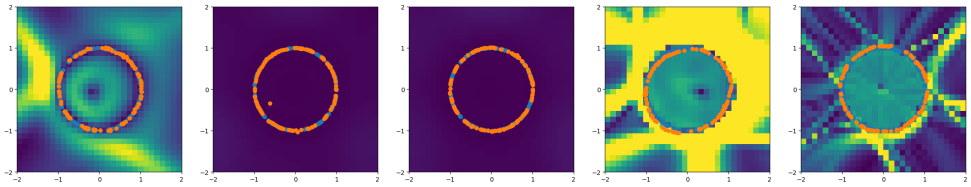
For each method, we run both Stochastic Gradient Descent (SGD) and RMSProp with 4 different learning rates: for SGD, we select the learning rate from $\{5 \cdot 10^{-3}, 10^{-2}, 2 \cdot 10^{-2}, 5 \cdot 10^{-2}\}$. For RMSProp, we select it from $\{5 \cdot 10^{-5}, 10^{-4}, 2 \cdot 10^{-4}, 5 \cdot 10^{-4}\}$. For the R_1-, R_2- and WGAN-GP-regularizers we try the regularization parameters $\gamma = 1$, $\gamma = 3$ and $\gamma = 10$.

We evaluate the results of the algorithms using the Wasserstein divergence (Section 1.1.4). We estimate the Wasserstein divergence using the Python Optimal Transport Package[2] by drawing 2048 samples from both the generator distribution and the true data distribution. We train all methods for 50k iterations and we report the Wasserstein divergence averaged over the last 10k iterations. For each method and architecture, we report the results for the hyperparameter setting which achieves the lowest Wasserstein divergence to the true data distribution.

8.1.2 Results

For each 2D-distribution and training method Table 8.1 reports the average Wasserstein divergence for the six different architectures. We see that the R_1- and R_2-regularizers perform

[2] `http://pot.readthedocs.io`



(a) unregularized *(b)* R_1 *(c)* R_2 *(d) WGAN-GP 1* *(e) WGAN-GP 5*

Figure 8.2: Qualitative Results for 2D Experiment. *Best solutions found by the different algorithms for learning a circle. The blue points are samples from the true data distribution, the orange points are samples from the generator distribution. The colored areas visualize the gradient magnitude of the equilibrium discriminator. We find that while training with* R_1*- and* R_2*-regularization converges to equilibrium discriminators that are zero in a neighborhood of the true data distribution, unregularized training and WGAN-GP converge to Energy Solutions (Appendix D).*

similarly to each other and they achieve better results than unregularized training or training with WGAN-GP. Interestingly, WGAN-GP with one discriminator update per generator update performs the worst and achieves only slightly better results than unregularized GAN training when using five discriminator update per generator update. This agrees with our results from Chapter 3 which show that WGAN-GP is not always convergent to the equilibrium point.

The best solution found by each method for the "circle"-distribution is shown in Figure 8.2. We see that the R_1- and R_2-regularizers converge to solutions for which the discriminator is zero in a neighborhood of the true data distribution. On the other hand, unregularized training and WGAN-GP converge to *Energy Solutions* where the discriminator forms a potential function for the true data distribution (Appendix D).

In summary, our results show that unregularized GAN training with neural networks can yield reasonable results for low dimensional data distributions. However, even in this setting R_1- and R_2-regularization improve the results. In contrast, WGAN-GP [64] performs much worse when we run only one discriminator update per generator update. This confirms our theoretical result that zero-centered gradient penalties are superior to WGAN-GP.

8.2 Real World Distributions

So far, we have seen that R_1- and R_2-regularization help stabilize GAN training on simple analytic examples (Chapter 3 and 4) as well as for GANs on simple 2D data distributions (Section 8.1).

To test how well the gradient penalties from Chapter 6 perform on more complicated tasks, we now compare R_1- and R_2-regularization to unregularized GAN training and WGAN-GP on a variety of challenging real-world image datasets.

8.2.1 Datasets

We first train GANs on the CelebA dataset [110] as well as four different classes of the LSUN dataset [207] at resolution 256×256. Both CelebA and LSUN consist of realistic images in a restricted domain and therefore constitute good datasets for testing training algorithms for GANs.

Afterwards, we study how well our results generalize to less restricted domains. To this end, we train conditional generative models for all 1000 classes of the ILSVRC2012 dataset (ImageNet) [171] at resolution 128×128 in a single GAN. Because of the high variability of this dataset, this is known to be a challenging task and only few prior works have managed to obtain recognizable samples on this dataset. Parallel lines of work that report results for this dataset either show a high amount of mode collapse [144, 173], report results only at a lower resolution [70] or use advanced normalization layers to stabilize the training [19, 134, 135, 209]. In contrast, our models do not use any normalization layers, but consist only of standard ResNet-blocks of convolutional and fully-connected layers. This enables us to study the influence of the training algorithm independently from the neural network architecture.[3]

Finally, to see if R_1-regularization also helps to train GANs for high-resolution image distributions, we apply our method to the CelebA-HQ dataset [88] at resolution 1024×1024. Training GANs at this resolution is a very challenging task and the only prior work [88] that reports results at this resolution uses a complicated multiresolution training schedule. In contrast, we directly train the full-resolution model end-to-end.[4]

8.2.2 Metrics

To evaluate the different training algorithms, we measure the Inception score [173] and Fréchet Inception Distance (FID) [68] between the generator distribution p_θ and true data distribution $p_\mathcal{D}$ during the training process.[5]

The Inception score is defined by

$$\text{Inception-Score}(p_\theta) = \exp\left(\mathrm{E}_{x\sim p_\theta}\left[\mathrm{KL}\left(p_{\text{IN}}(y \mid x) \,\|\, p_{\text{IN}}(y)\right)\right]\right) \tag{8.1}$$

where $p_{\text{IN}}(y \mid x)$ denotes the label distribution of an Inception Network [181] and $p_{\text{IN}}(y) = \mathrm{E}_{x\sim p_\theta}[p_{\text{IN}}(y \mid x)]$ is the corresponding marginal distribution. The idea behind the Inception score is that the Kullback-Leibler divergence in (8.1) is high if the entropy of $p_{\text{IN}}(y \mid x)$ is low and the entropy of $p_{\text{IN}}(y)$ is high. This means that the Inception score is high if all labels are predicted by the Inception Network with equal probability (high entropy of $p_{\text{IN}}(y)$), but

[3] Brock, Donahue, and Simonyan [19] report that even with spectral normalization [134] GAN training collapses in latter stages of training and R_1-regularization prevents this collapse. However, they also observe that this improved stability comes at the cost of a lower Inception score in their experiments.

[4] Current state-of-the-art methods [89, 90] for high-resolution image generation now also use R_1-regularization, reporting that this lead to considerably better results than WGAN-GP [89].

[5] For both the Inception score and FID we adapted the code from `https://github.com/mseitzer/pytorch-fid`, which uses the pretrained Inception Network from `http://download.tensorflow.org/models/image/imagenet/inception-2015-12-05.tgz`.

for one specific $x \in \mathcal{X}$ the Inception Network is very certain about the label (low entropy of $p_{\text{IN}}(y \mid x)$).

The Inception score has the drawback [10] that it does not depend on the true data distribution and is therefore not a good metric to measure convergence of training algorithms. Moreover, the Inception score is not a good measure of sample diversity (e.g. to measure mode collapse) and is inherently restricted to ImageNet.

Heusel et al. [68] therefore propose the FID as an alternative which computes a divergence between two probability distributions in feature space. To compute the FID, we first embed both the data distribution $p_{\mathcal{D}}$ and the generator distribution p_θ into the 2048-dimensional feature space of an Inception Network [181]. Afterwards, we approximate these two distributions with Gaussian distributions with the same first and second order moments. The FID is defined as the Fréchet distance[6] between these two Gaussian distributions. Using the analytic formula for the Fréchet distance between two multivariate Gaussian distributions [40], we obtain

$$\text{FID}(p_\theta, p_{\mathcal{D}}) = \|\mu_\theta - \mu_{\mathcal{D}}\|^2 + \text{tr}\left(\Sigma_{\mathcal{D}} + \Sigma_\theta - 2(\Sigma_{\mathcal{D}}\Sigma_\theta)^{1/2}\right) \tag{8.2}$$

where μ_θ, $\mu_{\mathcal{D}}$, Σ_θ, and Σ_θ denote the mean vectors and covariance matrices of the distributions obtained by embedding p_θ and $p_{\mathcal{D}}$ into the feature space of an Inception Network, respectively. As we are interested in convergence of the algorithms rather than generalization, we do not use a separate test set for computing the FID, but directly compute the FID between the training distribution and the distribution produced by the generator.

We compute both the Inception score and FID using 64k samples from the generator.

8.2.3 Experimental Setup

For the ImageNet experiment we use ResNet-architectures for the generator and discriminator, both having 26 layers in total. Both the generator and discriminator are conditioned on the labels of the input data. The architectures for the generator and discriminator are shown in Table E.3 and Table E.4 in Appendix E.1. We use pre-activation ResNet-blocks and ReLU-nonlinearities everywhere. We also multiply the output of the ResNet-blocks with 0.1. For the generator, we sample a latent variable z from a 256-dimensional standard Gaussian distribution and concatenate it with a 256 dimensional embedding of the labels on the unit sphere. The resulting 512-dimensional vector is then fed into the first fully-connected layer of the generator. The discriminator takes as input an image and outputs a 1000 dimensional vector. Depending on the label of the input, we select the corresponding index in this vector and use it as the logits for the GAN-objective.

For CelebA and LSUN, we use a similar training setup as for the ImageNet experiment, but we use slightly different architectures (Table E.1 and Table E.2). For CelebA-HQ, we use almost the same architecture as for CelebA (Table E.1 and Table E.2), but add two more levels to the generator to increase the resolution from 256×256 to 1024×1024 and decrease the number of features from 64 to 16. We modify the discriminator architecture

[6] The square root of the Fréchet distance is also called the Wasserstein-distance of order two [191].

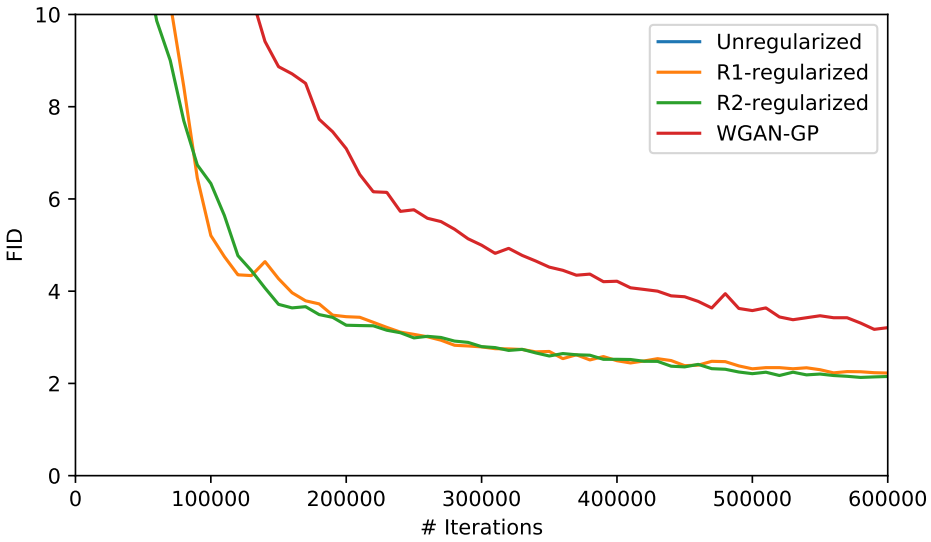


Figure 8.3: Quantitative Results CelebA. *We measure Fréchet Inception Distance (FID) between generator and data distribution over the number of iterations. Unregularized GAN training quickly collapses. The graph for unregularized GAN training is therefore outside the range of the y-axis.*

in a similar way. Except for WGAN-GP, we always use the nonsaturating GAN-objective introduced by Goodfellow et al. [62].

For training, we use the RMSProp optimizer [69] with $\alpha = 0.99$, $\varepsilon = 10^{-8}$ (PyTorch defaults) and a learning rate of 10^{-4}. We use $\gamma = 10$ for both R_1/R_2-regularization and WGAN-GP (as recommended by Gulrajani et al. [64]). Similarly to prior work [58, 88, 204], we use an exponential moving average with decay 0.999 over the weights to produce the final model. We find that we can achieve considerably better results for all methods by performing a generator update followed by a discriminator update for the *same* latent code z instead of using different latent codes. All results in this section use this update method.[7]

For ImageNet, we use a batch size of 128 and we train the networks on two Tesla V100 GPUs for 600k iterations. For LSUN and CelebA, we use a batch size of 64 and train each model[8] for about 600k iterations on one Tesla V100 GPU. Because of memory constraints, we decrease the batch size to 32 for CelebA-HQ and train the model for about 500k iterations on two Tesla V100 GPUs. In contrast to Karras et al. [88], we train our model end-to-end during the whole course of training, i.e. we do not use progressively growing GAN-architectures (nor any of the other techniques used by Karras et al. [88] to stabilize the training).

[7] In the original publication [129] we used two different latent codes $z \sim p_0$ for the generator and discriminator update, resulting in a noisier training objective and worse final results.

[8] For LSUN and CelebA-HQ, we finetuned the pretrained models from the original publication [129] using our new update rules.

8.2.4 Results

The FID for CelebA over the number of iterations is visualized in Figure 8.3. We see that unregularized training quickly leads to mode collapse: the best FID achieved by unregularized GAN training is 29.7 after about 20k iteration, which is outside the y-range of Figure 8.3. After that, unregularized GAN training collapses. In contrast, both WGAN-GP and GAN training with R_1/R_2-regularization are stable. Moreover, the training curves of R_1- and R_2-regularization are similar to each other and, compared to WGAN-GP, both R_1- and R_2-regularization lead to better end results, confirming our theory. Some random samples of our GAN with R_1-regularization on CelebA and LSUN are shown in Figure 8.5 and 8.6, respectively. We see that our model is able to produce realistic and diverse results.

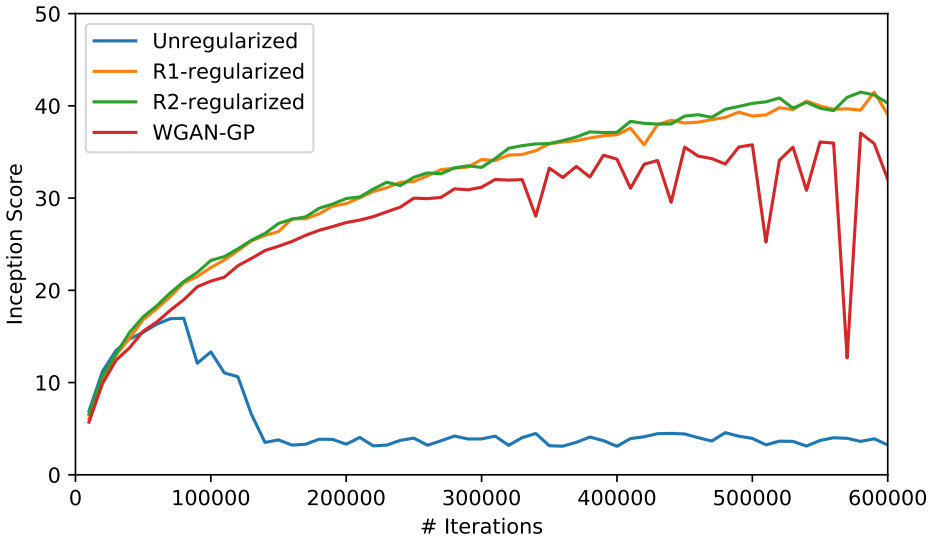
Quantitative result over the number iterations for our experiment on ImageNet are shown in Figure 8.4. Again, we see that unregularized GAN training collapses. The best Inception score and FID for unregularized GAN training on this dataset is 17.0 and 46.1 after 80k and 70k iterations, respectively. In contrast, both WGAN-GP and GAN training with R_1/R_2-regularization lead to more stable training and both the Inception score and FID keep improving as training progresses. Again GAN training with R_1/R_2-regularization achieves better end results than WGAN-GP, which is in line with our theory. In later iterations, we observe rare instabilities[9] both for WGAN-GP and R_1-regularization. While this might be due to the architectures we use for training (recall that we did not use any normalization layers), it could also be due to the fact that the realizability assumption (Assumption I′) is not satisfied. Figure 8.7 and Figure 8.8 show conditional samples for some selected ImageNet classes. While not completely photorealistic, we find that our model can produce convincing samples from all 1000 ImageNet classes. In addition, we also show some qualitative results for a conditional GAN trained on ImageNet at resolution 256×256 in Appendix F.2.

For CelebA-HQ, we find that the simple R_1-regularizer stabilizes the training, allowing our model to converge to a good solution without using a progressively growing GAN. Some random samples are shown in Figure 8.9.

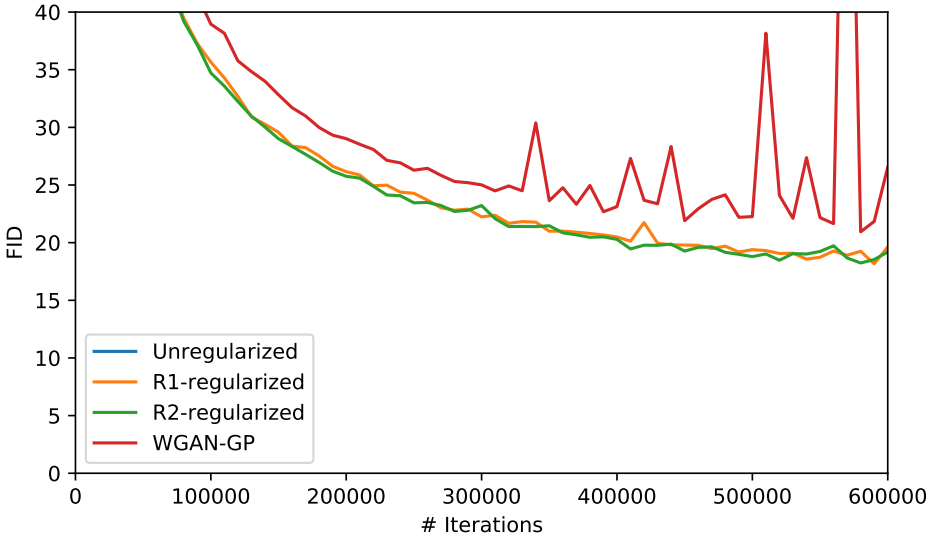
8.3 Conclusion

In this chapter we have conducted a thorough experimental evaluation of our regularizers from Chapter 6, which so far were mainly theoretically motivated. To this end, we compared our regularizers to unregularized GAN training and WGAN-GP on both simple 2D examples and high-dimensional image distributions. Our results suggest that our new regularizers also work surprisingly well for real-world examples of GAN training, which validates our theory. In particular, our regularizers enable us to train a GAN at one megapixel resolution without resorting to additional tricks such as multiresolution training.

[9] Both WGAN-GP and R_1-regularization sometimes fell into degenerate saturated solutions in later iterations. Because this happened only rarely, we simply restarted training from the previous checkpoints in these cases. Interestingly, these instabilities only happen when we train both the generator and discriminator on the same latent code as discussed in Section 8.2.3 and therefore did not occur in the experiments of the original publication [129].



(a) Inception Score



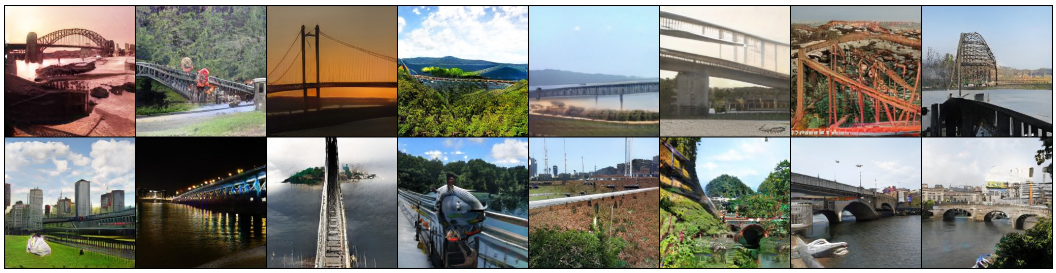
(b) FID

Figure 8.4: Quantitative Results ImageNet. *We measure the Inception score and Fréchet Inception Distance (FID) between generator and data distribution over the number of iterations.*

Figure 8.5: Samples CelebA. *Random samples for a convolutional GAN trained on the CelebA dataset [110] at resolution* 256×256.

(a) *bedroom*

(b) *church*



(c) *bridge*

(d) *tower*

Figure 8.6: Samples LSUN. *Random samples for convolutional GANs trained (separately) on four LSUN [207] classes at resolution* 256×256.

(a) *tench*

(b) *papillon*

(c) *weevil*

(d) *admiral*

(e) *lighthouse*

Figure 8.7: Conditional Samples ImageNet. *Random class conditional samples for a convolutional GAN trained on the ImageNet dataset [171] at resolution* 128×128.

(a) *home theater*

(b) *police van*

(c) *rugby ball*

(d) *pizza*

(e) *valley*

Figure 8.8: Conditional Samples ImageNet. *Random class conditional samples for a convolutional GAN trained on the ImageNet dataset [171] at resolution* 128×128.

Figure 8.9: Samples CelebA-HQ. *Random samples for a convolutional GAN trained on the CelebA-HQ dataset [88] at resolution* 1024×1024*. During the whole course of training, we directly train the full-resolution generator and discriminator end-to-end, i.e. we do not use any of the techniques described in Karras et al. [88] to stabilize the training.*

9 Limitations and further Developments

In the first part of this thesis we have seen that linearization is a useful tool for understanding the GAN training dynamics. We have designed simple examples of GAN training that we could analyze analytically and - together with Nagarajan and Kolter [136] - derived a theory for local convergence of general GANs. As we have seen, these theoretical results are useful to design new regularizers that also work very well in practice. Despite these successes, our theory has a number of limitations which hopefully will be addressed in future research.

First of all, our theory is only local. This means that while we can make statements about convergence of GAN training in some local neighborhood of the equilibrium point, our theorems do neither make statements about global convergence nor the size of the region of convergence. Of course, it can be argued that local convergence is a minimum requirement and understanding local convergence is hence a necessary first step before deriving a global theory. However, statements about the size of the region of convergence are required for a complete picture and we believe that a better theory here will yield novel insights into GAN training. Some promising research shows that a variant of Consensus Optimization [9, 55, 140] is globally convergent for a simple GAN [55], called the Linear-Quadratic-GAN [47, 136]. However, we believe that this is only a first step towards a complete theory for more complex GANs.

Second, our theory requires the realizability assumption, which holds for simple toy examples for GAN training, but does not necessarily hold for more complex systems. Again, it is important to realize that understanding convergence under the realizability assumption is a minimum requirement. Moreover, realizability might hold at least approximately for very expressive deep neural networks. However, more work needs to be done to understand what happens when the realizability assumption does not hold.

Finally, for simplicity we derived our theory for the deterministic system without noise. We deliberately focused our attention on this case, since (i) it leads to a much cleaner theory which emphasizes the structural insights over the technical details and (ii) noise can be arbitrarily reduced by increasing the batch size. The theoretical framework to analyze the noisy case is given by stochastic approximation theory [14, 167, 178]. Indeed, typical theorems[1] from stochastic approximation theory require similar assumptions as our local convergence theory. However, deriving the technical details in a principled way is beyond the scope of this thesis. A deeper analysis of the noisy case might also lead to additional insights, e.g. about properties of the stationary distributions of SimGD and AltGD near the equilibrium.

In summary, there are many opportunities to extend the results presented in this thesis. The training dynamics of GANs hence remains an interesting research topic.

[1] See Appendix B.2 for a brief introduction.

Part II

Deep Learning in Function Space

“The world is continuous, but the mind is discrete.”

David Mumford

ICM 2002 plenary talk

10 The Function Space Operator

In Part I we have analyzed the training dynamics of Generative Adversarial Networks (GANs), one of the most promising approaches to generative modeling. As we have seen, simple regularization techniques enable stable training of GANs for image distributions, even at very high resolution.

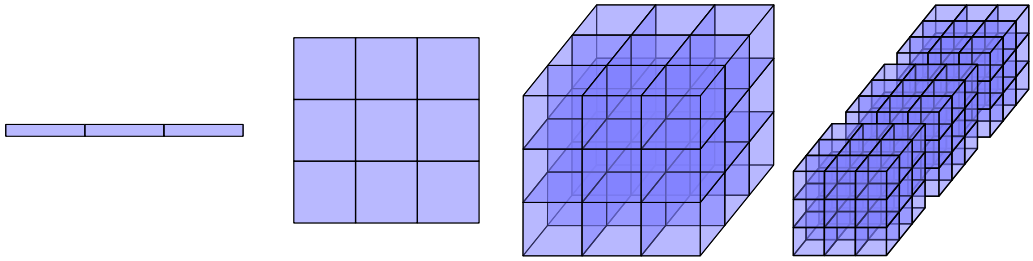
However, our world is not two-, but three-dimensional. To create useful generative models of the real-world, it is hence desirable to create models that can reason in three dimensions. The same is true for discriminative models that try to reconstruct information about the real world from 2D images or other types of input. Unfortunately, unlike in 2D, there is no canonical output representation for discriminative and generative models in 3D. The reason for this is what we call the *Curse of Discretization*, an instance of the Curse of Dimensionality: when we naively discretize an output space, the memory requirements grow exponentially with the dimensionality of space. While in two dimensions (e.g. images) this effect is not yet very severe, this changes in three dimensions where finding an expressive, yet flexible representation that works well with deep learning is a difficult task. The Curse of Discretization becomes even more severe when we want to synthesize other quantities than 3D geometry, such as texture and material properties, or would like to predict time changing geometry. We will briefly discuss these cases in Chapter 12.

Here, we propose a simple solution to the Curse of Discretization: by reinterpreting the high-dimensional output of the network as a function, we are able to avoid discretization completely during training of the neural networks. Our key insight is that a function from some space to a function space can equivalently be described as a function from a Cartesian product of spaces to a low-dimensional space. We make this statement formal by introducing the *Function Space Operator*. The Function Space Operator turns the intractable problem of learning a mapping to a function space into the tractable problem of learning a mapping from a modified input space to a low dimensional space.

In this chapter we first give a general introduction to deep learning in function space. An application to learning-based 3D reconstruction and 3D generative models can be found in Chapter 11.

10.1 The Curse of Discretization

In Chapter 1 we have defined a generative model as a neural network $G_\theta : \mathcal{Z} \to \mathcal{X}$ with parameter vector θ that takes a latent vector $z \in \mathcal{Z}$ as input and outputs an element in a high-dimensional space $\mathcal{X}$ (e.g. an image). We can make this model conditional by conditioning the model on some additional piece of information $y \in \mathcal{Y}$, e.g. a label. This leads to a model $G_\theta : \mathcal{Z} \times \mathcal{Y} \to \mathcal{X}$. The extreme case is a model that does not use any latent code $z \in \mathcal{Z}$ at all



(a) $3^1 = 3$ ***(b)*** $3^2 = 9$ ***(c)*** $3^3 = 27$ ***(d)*** $3^4 = 81$

Figure 10.1: The Curse of Discretization. *When increasing the dimensionality of $\mathcal{S}$, the number of cells in the discretization $\hat{\mathcal{S}}$ grows exponentially. Even at a very low resolution of three, this already leads to* 81 *cells if the underlying space is four-dimensional.*

Output	$\mathcal{S}$	$\mathcal{V}$	$\mathcal{X} = \mathcal{V}^{\mathcal{S}}$	$\hat{\mathcal{X}} = \mathcal{V}^{\hat{\mathcal{S}}}$
Semantic Map	$[0,1]^2$	$\mathbb{R}^k$	$[0,1]^2 \to \mathbb{R}^k$	$\mathbb{R}^{k \times N \times N}$
2D Image	$[0,1]^2$	$\mathbb{R}^3$	$[0,1]^2 \to \mathbb{R}^3$	$\mathbb{R}^{3 \times N \times N}$
3D Scalar Field	$[0,1]^3$	$\mathbb{R}$	$[0,1]^3 \to \mathbb{R}$	$\mathbb{R}^{N \times N \times N}$
3D Vector Field	$[0,1]^3$	$\mathbb{R}^3$	$[0,1]^3 \to \mathbb{R}^3$	$\mathbb{R}^{3 \times N \times N \times N}$
Time-varying 3D Scalar Field	$[0,1]^4$	$\mathbb{R}$	$[0,1]^4 \to \mathbb{R}$	$\mathbb{R}^{N \times N \times N \times N}$
Time-varying 3D Vector Field	$[0,1]^4$	$\mathbb{R}^3$	$[0,1]^4 \to \mathbb{R}^3$	$\mathbb{R}^{3 \times N \times N \times N \times N}$
3D Light Field	$[0,1]^5$	$\mathbb{R}^3$	$[0,1]^5 \to \mathbb{R}^3$	$\mathbb{R}^{3 \times N \times N \times N \times N \times N}$

Table 10.1: Output Spaces. *A selection of output spaces $\mathcal{X}$ in deep learning and the discretization $\hat{\mathcal{X}}$. Here, k denotes the number of classes for the semantic map. N denotes the spatial and temporal discretization resolution (assumed to be equal for simplicity here).*

and therefore has a deterministic output. We call such a model a *discriminative model*. This description is very general and includes many different cases such as 2D generative models, semantic segmentation, learning-based 3D reconstruction and many more (Table 10.1).

For generative models, but also many discriminative models, $\mathcal{X}$ is usually given by a high-dimensional data structure. Our first insight is that $\mathcal{X}$ can often be interpreted as a function space: while traditionally $\mathcal{X}$ is usually directly described by some (e.g. spatial) discretization $\hat{\mathcal{X}}$, it is usually better described by the set of functions $\mathcal{S} \to \mathcal{V}$ from some input domain $\mathcal{S}$ (e.g. $[0,1]^d$) to some output domain $\mathcal{V}$ (e.g. $\mathbb{R}^k$). Following common notation, we denote the space of functions from $\mathcal{S}$ to $\mathcal{V}$ by $\mathcal{V}^{\mathcal{S}}$.

Take image synthesis models as an example: while traditionally the output is usually defined as a tensor $\mathbb{R}^{3 \times N \times N}$ with spatial resolution N, this description is dependent on the discretization. Conceptually, however, we could also describe an image continuously as a function $[0,1]^2 \to \mathbb{R}^3$ that takes a 2D coordinate as input and produces a RGB-color for each of the 2D points.

However, directly outputting an arbitrary function is clearly intractable for neural networks. The common approach is therefore to discretize $\mathcal{S}$ and replace $\mathcal{X} = \mathcal{V}^{\mathcal{S}}$ with the

discretized version $\hat{\mathcal{X}} = \mathcal{V}^{\hat{\mathcal{S}}}$. Assume now that $\mathcal{S} = [0,1]^d$. If we discretize $\mathcal{S}$ in an equidistant way with resolution N in each dimension, we obtain

$$\hat{\mathcal{S}} = \left\{ \left(\frac{i_1}{N-1}, \dots, \frac{i_d}{N-1} \right) \middle| i_1, \dots, i_d \in \{0, \dots, N-1\} \right\} \tag{10.1}$$

and therefore $|\hat{\mathcal{S}}| = N^d$. We see that the size of $\hat{\mathcal{S}}$ grows exponentially with the dimensionality of $\mathcal{S}$. This is visualized in Figure 10.1, where we see that even for a coarse discretization the number of elements in $\hat{\mathcal{S}}$ becomes large as d increases.

The discretization $\hat{\mathcal{X}}$ for $\mathcal{X}$ is hence

$$\hat{\mathcal{X}} = \mathcal{V}^{\hat{\mathcal{S}}} \cong \mathcal{V}^{(N^d)} \tag{10.2}$$

We see that the dimensionality of $\hat{\mathcal{X}}$ grows exponentially with d, too. We summarize this fact in the following lemma:

Lemma 10.1. *Let $\mathcal{V}$ denote some vector space (e.g. $\mathbb{R}^k$) and $\mathcal{S} = [0,1]^d$. Let $\mathcal{X} = \mathcal{V}^{\mathcal{S}}$ and assume that we discretize $\mathcal{S}$ in an equidistant way with resolution $N \in \mathbb{N}$. Then* $\dim(\hat{\mathcal{X}}) = \dim(\mathcal{V}) \cdot N^d$.

Proof. We have seen that $\hat{\mathcal{X}} \cong \mathcal{V}^{(N^d)}$. However, $\mathcal{V}$ is a vector space of dimension $\dim(\mathcal{V})$. As a consequence, $\hat{\mathcal{X}}$ is also a vector space of dimension $\dim(\hat{\mathcal{X}}) = \dim(\mathcal{V}) \cdot N^d$. □

Discretizing $\mathcal{X}$ in this way becomes clearly intractable as d increases: even for tasks with relatively low dimensionality d such as 3D reconstruction ($d = 3$), the dimensionality of $\hat{\mathcal{X}}$ becomes a serious bottleneck for deep learning techniques (Table 10.1). This problem becomes even worse for 4D reconstruction ($d = 4$) or when we would like to predict an entire light field ($d = 5$).

We call this problem the Curse of Discretization. The Curse of Discretization is an instance of the Curse of Dimensionality. However, as we will see in the next section, we can circumvent it.

10.2 The Function Space Operator

Recall that we would like to approximate functions from $\mathcal{Z} \times \mathcal{Y}$ to some function space $\mathcal{X} = \mathcal{V}^{\mathcal{S}}$ with a neural network. As we have seen in Section 10.1, naively replacing $\mathcal{S}$ with a discretized version $\hat{\mathcal{S}}$ quickly becomes intractable when the dimensionality of $\mathcal{S}$ grows. Can we do something more clever than this?

Recall that the Curse of Discretization is an instance of the Curse of Dimensionality and that deep learning can be regarded as one of the major techniques to cope with the Curse of Dimensionality. Instead of discretizing $\mathcal{S}$, we could hence try to directly approximate the functions in $\mathcal{X} = \mathcal{V}^{\mathcal{S}}$ with neural networks.

However, while it is straightforward to approximate a single $x \in \mathcal{V}^{\mathcal{S}}$ in this way, in practice we want to approximate a function from $\mathcal{Z} \times \mathcal{Y}$ to $\mathcal{V}^{\mathcal{S}}$. Luckily, we can describe

a function $\mathcal{Z} \times \mathcal{Y} \to \mathcal{V}^{\mathcal{S}}$ equivalently as a function $\mathcal{S} \times \mathcal{Z} \times \mathcal{Y} \to \mathcal{V}$, i.e. we move $\mathcal{S}$ to the left hand side of the arrow. Formally, we have

Lemma 10.2. *For a function $f : \mathcal{Z} \times \mathcal{Y} \to \mathcal{V}^{\mathcal{S}}$, let $F(f) : \mathcal{S} \times \mathcal{Z} \times \mathcal{Y} \to \mathcal{V}$ be defined by*

$$F(f)(s,z,y) = f(z,y)(s) \tag{10.3}$$

F defines a natural[1] bijection between the space of all mappings from $\mathcal{Z} \times \mathcal{Y}$ to $\mathcal{V}^{\mathcal{S}}$ and the space of all mapping from $\mathcal{S} \times \mathcal{Z} \times \mathcal{Y}$ to $\mathcal{V}$.

Proof. To see that F is a bijection, we define an "inverse" operator G that takes a function $g : \mathcal{S} \times \mathcal{Z} \times \mathcal{Y} \to \mathcal{V}$ as input has a function $G(g) : \mathcal{Z} \times \mathcal{Y} \to \mathcal{V}^{\mathcal{S}}$ as output.

We define $G(g) : \mathcal{Z} \times \mathcal{Y} \to \mathcal{V}^{\mathcal{S}}$ by

$$G(g)(z,y)(s) := g(s,z,y) \tag{10.4}$$

To see that G is indeed the inverse to F, we calculate for $z \in \mathcal{Z}$, $y \in \mathcal{Y}$ and $s \in \mathcal{S}$

$$(G \circ F)(f)(z,y)(s) = F(f)(s,z,y) = f(z,y)(s) \tag{10.5}$$

Since this hold for all $z \in \mathcal{Z}$, $y \in \mathcal{Y}$ and $s \in \mathcal{S}$, this shows $(G \circ F)(f) = f$.

Similarly, we have for all $z \in \mathcal{Z}$, $y \in \mathcal{Y}$ and $s \in \mathcal{S}$

$$(F \circ G)(g)(s,z,y) = G(g)(z,y)(s) = g(s,z,y) \tag{10.6}$$

and therefore $(F \circ G)(g) = g$.

All in all, we see that G is the inverse to F and F is hence a bijection. □

In short, Lemma 10.2 can be written as

$$\left(\mathcal{V}^{\mathcal{S}}\right)^{\mathcal{Z} \times \mathcal{Y}} \cong \mathcal{V}^{\mathcal{S} \times \mathcal{Z} \times \mathcal{Y}} \tag{10.7}$$

Formally, this equation resembles the classical law $(a^b)^c = a^{bc}$ for real numbers a, b, c. However, in our case $\mathcal{Z}$, $\mathcal{Y}$, $\mathcal{S}$ and $\mathcal{V}$ all denote sets.

The significance of Lemma 10.2 is that it is translates the intractable problem of approximating functions in $\mathcal{Z} \times \mathcal{Y} \to \mathcal{V}^{\mathcal{S}}$ to the tractable problem of approximating functions in $\mathcal{S} \times \mathcal{Z} \times \mathcal{Y} \to \mathcal{V}$: while it is usually intractable to define a neural network that has a function $x \in \mathcal{X} = \mathcal{V}^{\mathcal{S}}$ as output, it is straightforward to approximate a function from $\mathcal{S} \times \mathcal{Z} \times \mathcal{Y}$ to $\mathcal{V} = \mathbb{R}^k$ with neural networks (assuming that k is reasonably small). We call the operator $F : \left(\mathcal{V}^{\mathcal{S}}\right)^{\mathcal{Z} \times \mathcal{Y}} \to \mathcal{V}^{\mathcal{S} \times \mathcal{Z} \times \mathcal{Y}}$ the *Function Space Operator*.

10.3 Conclusion

We have seen that both discriminative and generative models suffer from high-dimensional output spaces. Our key insight is that we can often interpret these output spaces as function

[1] The term *natural* is formally defined in the context of category theory. Please see Mac Lane [120] for details.

spaces. Naively discretizing a function space leads to the Curse of Discretization, i.e. that the dimensionality of the discretization grows exponentially with the dimensionality of the domain of the function space. Fortunately, there is a simple trick that enables us to avoid the Curse of Discretization: by introducing the Function Space Operator we can reduce the problem to an easier problem with a low-dimensional output space. In Chapter 11, we describe an application of the Function Space Operator to learning-based 3D reconstruction as well as 3D generative modeling. However, the Function Space Operator can be applied to many problems in deep learning. In particular, in Chapter 12 we give a brief overview of two applications of the Function Space Operator to texture synthesis and 4D generative modeling.

11 Occupancy Networks

In Chapter 10 we have introduced a general method to handle high-dimensional output spaces. The main insight is that these high-dimensional output spaces often arise from discretizing a function space. However, we can avoid discretization of these function spaces altogether by applying the Function Space Operator. This technique translates the difficult problem of approximating a function with a high-dimensional output space to the tractable problem of approximating a function with a low-dimensional output space.

In this chapter we present an application of the Function Space Operator to learning-based 3D reconstruction and 3D generative modeling.[1] We first discuss existing 3D representations and their limitations. Afterwards, we introduce the concept of an *Occupancy Function*, a continuous version of an occupancy grid. Using the notion of Occupancy Function, we can describe the problem of 3D generative modeling and 3D reconstruction as approximating a function with an Occupancy Function as output. This allows us to apply the Function Space Operator from Chapter 10. We call the resulting representation for 3D geometry, which uses a neural network to approximate the Occupancy Function, an *Occupancy Network*.

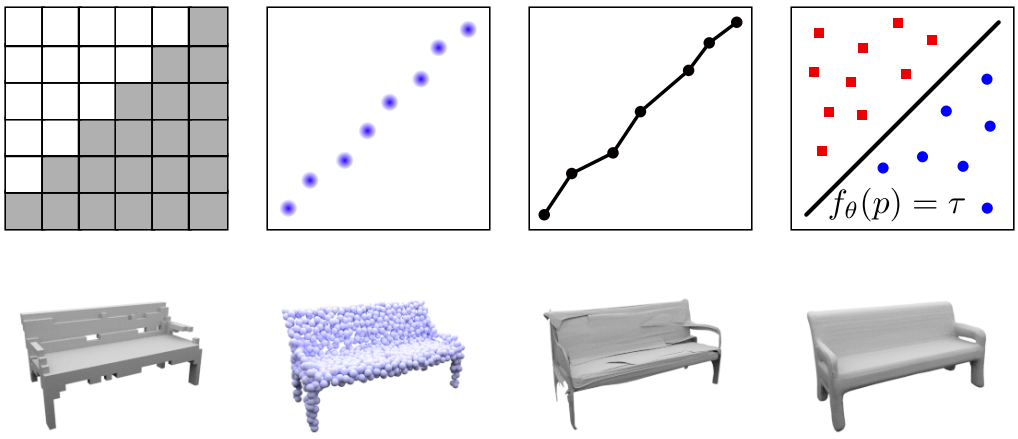
In this chapter we describe in detail how we can learn a model that infers this representation from various forms of input, such as point clouds, single images and low-resolution voxel representations. Moreover, we show how Occupancy Networks can be used in a generative context and describe a technique for extracting high-quality 3D meshes from our model at test time. Finally, we validate our approach experimentally for both generative and discriminative tasks.

11.1 Related Work

Recently, learning-based approaches for 3D reconstruction have gained popularity [17, 26, 60, 165, 198, 199]. In contrast to traditional multi-view stereo algorithms, learned models are able to encode rich prior information about the space of 3D shapes which helps to resolve ambiguities in the input.

While generative models have recently achieved remarkable successes in generating realistic high resolution images [88, 129, 195], this success has not yet been replicated in the 3D domain. In contrast to the 2D domain, the community has not yet agreed on a 3D output representation that is both memory efficient and can be efficiently inferred from data. Existing representations can be broadly categorized into three categories: voxel-based representations [17, 49, 108, 165, 179, 189, 199] , point-based representations [2, 45] and mesh representations [84, 162, 193].

[1] Also see [24, 133, 150] for concurrent work that proposes similar ideas.



(a) Voxel *(b) Point* *(c) Mesh* *(d) Ours*

Figure 11.1: Overview. *Existing 3D representations discretize the output space differently: (a) spatially in voxel representations, (b) in terms of predicted points, and (c) in terms of vertices for mesh representations. In contrast, (d) we propose to consider the continuous decision boundary of a classifier f_θ (e.g., a deep neural network) as a 3D surface which allows to extract 3D meshes at any resolution.*

In the following, we give an overview of these existing 3D representations, see Figure 11.1.

11.1.1 Voxel Representations

Due to their simplicity, voxels are the most commonly used representation for discriminative [123, 157, 176] and generative [26, 60, 165, 179, 198, 199] 3D tasks.

Early works have considered the problem of reconstructing 3D geometry from a single image using 3D Convolutional Neural Networks which operate on voxel grids [26, 188, 198]. Due to memory requirements, however, these approaches were limited to relatively small 32^3 voxel grids. While recent works [200, 201, 208] have applied 3D Convolutional Neural Networks to resolutions up to 128^3, this is only possible with shallow architectures and small batch sizes, which leads to slow training.

The problem of reconstructing 3D geometry from multiple input views has been considered in [79, 87, 151]. Ji et al. [79] and Kar, Häne, and Malik [87] encode the camera parameters together with the input images in a 3D voxel representation and apply 3D convolutions to reconstruct 3D scenes from multiple views. Paschalidou et al. [151] introduced an architecture that predicts voxel occupancies from multiple images, exploiting multi-view geometry constraints [189].

Other works have applied voxel representations to learn generative models of 3D shapes. Most of these methods are either based on Variational Autoencoders [95, 164] or Generative Adversarial Networks [62]. These two approaches were pursued in [17, 165] and [199],

respectively.

Due to the high memory requirements of voxel representations, recent works have proposed to reconstruct 3D objects in a multiresolution fashion [66, 184]. However, the resulting methods are often complicated to implement and require multiple passes over the input to generate the final 3D model. Furthermore, they are still limited to comparably small 256^3 voxel grids. For achieving sub-voxel precision, several works [32, 104, 166] have proposed to predict a Truncated Signed Distance Field (TSDF) [30] where each point in a 3D grid stores the truncated signed distance to the closest 3D surface point. However, this representation is usually much harder to learn compared to occupancy representations as the network must reason about distance functions in 3D space instead of merely classifying a voxel as occupied or not. Moreover, this representation is still limited by the resolution of the underlying 3D grid.

11.1.2 Point Representations

An interesting alternative representation of 3D geometry is given by 3D point clouds which are widely used both in the robotics and in the computer graphics communities. Qi et al. [158, 159] pioneered point clouds as a representation for discriminative deep learning tasks. They achieved permutation invariance by applying a fully-connected neural network to each point independently followed by a global pooling operation. Fan, Su, and Guibas [45] introduced point clouds as an output representation for 3D reconstruction. However, point clouds lack the connectivity structure of the underlying mesh and hence require additional post-processing [11, 21, 92, 93] steps to extract 3D geometry from the model.

11.1.3 Mesh Representations

Meshes have first been considered for discriminative 3D classification or segmentation tasks by applying convolutions on the graph spanned by the mesh's vertices and edges [18, 65, 194].

More recently, meshes have also been considered as output representation for 3D reconstruction [63, 83, 100, 193]. Unfortunately, most of these approaches are prone to generating self-intersecting meshes. Moreover, they are only able to generate meshes with simple topology [193], require a reference template from the same object class [83, 100, 162] or cannot guarantee closed surfaces [63]. Liao, Donne, and Geiger [108] propose an end-to-end learnable version of the Marching Cubes algorithm [117]. However, their approach is still limited by the memory requirements of the underlying 3D grid and hence also restricted to 32^3 voxel resolution.

11.2 Method

As we have seen in the previous section, existing representations for learning-based 3D reconstruction and 3D generative modeling suffer from discretization artifacts. In this section we propose an alternative, continuous representation which is based on our results from Chapter 10 (Figure 11.1d).

When using voxels, we represent 3D geometry as a discrete occupancy grid, i.e. we represent space as a regular 3D grid $\hat{\mathcal{S}}$. Ideally, however, we would like to reason about the occupancy not only at fixed discrete 3D locations (as in voxel representations) but at *every* possible 3D point $s \in \mathcal{S} = [0,1]^3$ where $\mathcal{S}$ describes a given bounding volume. We call the resulting function

$$x : \mathcal{S} \to \{0,1\} \tag{11.1}$$

the *Occupancy Function* of the 3D object. We can hence reinterpret the tasks of learning-based 3D reconstruction and 3D generative modeling as learning a function $\mathcal{Z} \times \mathcal{Y} \to \{0,1\}^{\mathcal{S}}$ with $\mathcal{Z}$ some latent space, $\mathcal{Y}$ a space of observations (e.g. images, point clouds, etc.) and $\mathcal{S} = [0,1]^3$. By applying the Function Space Operator from Chapter 10, we can equivalently describe this as learning a function $\mathcal{S} \times \mathcal{Z} \times \mathcal{Y} \to \{0,1\}$. While approximating a function $\mathcal{Z} \times \mathcal{Y} \to \{0,1\}^{\hat{\mathcal{S}}}$ (as in voxel representations) suffers from the Curse of Discretization, we can approximate the function $\mathcal{S} \times \mathcal{Z} \times \mathcal{Y} \to \{0,1\}$ with a neural network $f_\theta(\cdot)$ that takes a triple (s,z,y) as input and produces a real number which represents the probability of occupancy:

$$f_\theta : \mathcal{S} \times \mathcal{Z} \times \mathcal{Y} \to [0,1] \tag{11.2}$$

We call this network an Occupancy Network (ONet). Note that this network is equivalent to a neural network for binary classification, except that we are interested in the decision boundary which implicitly represents the object's surface.

In contrast to the approaches from Section 11.1, our approach leads to high resolution closed surfaces without self-intersections and does not require template meshes from the same object class as input. This idea is related to classical level set [29, 36, 149] approaches to multi-view 3D reconstruction [46, 61, 80, 99, 156, 206]. However, instead of solving a differential equation, our approach uses deep learning to obtain a more expressive representation which can be naturally integrated into an end-to-end learning pipeline.

11.3 Training

Our description in the previous section is very general and includes both the purely generative case (where $\mathcal{Y} = \{0\}$) and the purely discriminative case (where $\mathcal{Z} = \{0\}$). In this section we first focus on the discriminative case and then describe the general case. Whenever $\mathcal{Z} = \{0\}$ we omit the dependence of $f_\theta(\cdot)$ on z and write $f_\theta(s,y)$ instead of $f_\theta(s,z,y)$. Similarly, when $\mathcal{Y} = \{0\}$ we write $f_\theta(s,z)$.

Now assume that $\mathcal{Z} = \{0\}$. To learn the parameters θ of the neural network $f_\theta(s,y)$, we randomly sample points in the 3D bounding volume $\mathcal{S}$ of the object under consideration: for the i-th sample in a training batch we sample K points $s_{ij} \in \mathcal{S}$, $j = 1,\dots,K$. We then evaluate the mini-batch loss $\mathcal{L}_\mathcal{B}$ at those locations:

$$\mathcal{L}_\mathcal{B}(\theta) = \frac{1}{|\mathcal{B}|} \sum_{i=1}^{|\mathcal{B}|} \sum_{j=1}^{K} \mathcal{L}(f_\theta(s_{ij}, y_i), o_{ij}) \tag{11.3}$$

Here, y_i is the i'th observation of batch $\mathcal{B}$, $o_{ij} := x_i(s_{ij})$ denotes the true occupancy at point

s_{ij}, and $\mathcal{L}(\cdot,\cdot)$ is a cross-entropy classification loss.

The performance of our method depends on the sampling scheme that we employ for drawing the locations s_{ij} that are used for training. In Section 11.6.9 we perform a detailed ablation study comparing different sampling schemes. In practice, we found that sampling uniformly inside the bounding box of the object with an additional small padding yields the best results.

Now consider the general case where $\mathcal{Z} \neq \{0\}$. While there are different ways to train a generative model, we use a Variational Autoencoder (VAE) for learning our implicit model. To this end, we introduce an encoder network $g^{\text{vae}}_{\psi}(\cdot)$ that takes the ground truth Occupancy Function x_i as well as the observation y_i as input and predicts mean μ_i and standard deviation σ_i of a Gaussian distribution $q_{\psi}(z|x_i,y_i)$ on latent $z \in \mathcal{Z}$ as output. While it is difficult to define an encoder q_{ψ} that directly operators on the entire function x_i, we can simply encode x_i in a fixed way, e.g. by using the sampled points and ground truth occupancies $(s_{ij},o_{ij})_{j=1:K}$ as input to $g^{\text{vae}}_{\psi}(\cdot)$. Alternatively, we could subsample the surface of the object and use the resulting point cloud as input to $g^{\text{vae}}_{\psi}(\cdot)$.

During training, we optimize a lower bound [51, 95, 164] to the negative log-likelihood of the generative model:

$$\mathcal{L}^{\text{gen}}_{\mathcal{B}}(\theta,\psi) = \frac{1}{|\mathcal{B}|}\sum_{i=1}^{|\mathcal{B}|}\Big[\sum_{j=1}^{K}\mathcal{L}(f_{\theta}(s_{ij},z_i,y_i),o_{ij}) + \text{KL}\,\left(q_{\psi}(z|x_i,y_i)\,\|\,p_0(z)\right)\Big] \tag{11.4}$$

Here, KL denotes the Kullback-Leibler divergence, $p_0(z)$ is a prior distribution on the latent variable z_i (typically Gaussian) and z_i is sampled according to $q_{\psi}(z_i|x_i,y_i)$.

11.4 Inference

For extracting the isosurface corresponding to a new observation given a trained Occupancy Network, we introduce Multiresolution IsoSurface Extraction (MISE), a hierarchical isosurface extraction algorithm (Figure 11.2). By incrementally building an OctTree [77, 124, 182, 197], MISE enables us to extract high resolution meshes from the Occupancy Network without densely evaluating all points of a high-dimensional occupancy grid.

We first discretize the volumetric space at an initial resolution and evaluate the Occupancy Network $f_{\theta}(s,z,y)$ for all s in this grid. We mark all grid points s as occupied for which $f_{\theta}(s,z,y)$ is bigger or equal to some threshold[2] τ. Next, we mark all voxels as active for which at least two adjacent grid points have differing occupancy predictions. These are the voxels which would intersect the mesh if we applied the Marching Cubes algorithm at the current resolution. We subdivide all active voxels into eight subvoxels and evaluate all new grid points which are introduced to the occupancy grid through this subdivision. We repeat these steps until the desired final resolution is reached. At this final resolution, we apply the

[2]The threshold τ is the only hyperparameter of our Occupancy Network. It determines the "thickness" of the extracted 3D surface. In our experiments we cross-validate this threshold on a validation set.

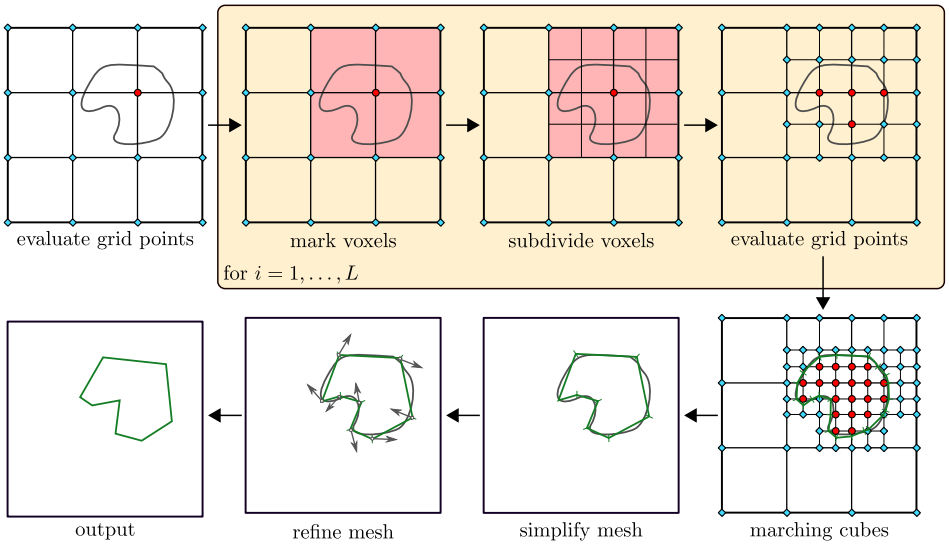


Figure 11.2: Multiresolution IsoSurface Extraction. *We first mark all points at a given resolution which have already been evaluated as either occupied (red circles) or unoccupied (cyan diamonds). We then determine all voxels that have both occupied and unoccupied corners and mark them as active (light red) and subdivide them into eight subvoxels each. Next, we evaluate all new grid points that have been introduced by the subdivision. The previous two steps are repeated until the desired output resolution is reached. Finally we extract the mesh using the Marching Cubes algorithm [117], simplify and refine the output mesh using first and second order gradient information.*

Marching Cubes algorithm [117] to extract an approximate isosurface

$$\{s \in \mathcal{S} \mid f_\theta(s,z,y) = \tau\}. \tag{11.5}$$

Our algorithm converges to the correct mesh if the occupancy grid at the initial resolution contains points from every connected component of both the interior and the exterior of the mesh. It is hence important to take an initial resolution which is high enough to satisfy this condition. In practice, we found that an initial resolution of 32^3 was sufficient in almost all cases.

The initial mesh extracted by the Marching Cubes algorithm can be further refined. In a first step, we simplify the mesh using the Fast Quadric Mesh Simplification algorithm[3] [52]. In practice, we simplify the mesh to 5,000 faces. Finally, we refine the output mesh using first and second order (i.e., gradient) information. Towards this goal, we sample random points s_k from each face of the output mesh and minimize the loss

$$\sum_{k=1}^{K} (f_\theta(s_k,z,y) - \tau)^2 + \gamma \left\| \frac{\nabla_p f_\theta(s_k,z,y)}{\|\nabla_p f_\theta(s_k,z,y)\|} - n(s_k) \right\|^2 \tag{11.6}$$

[3] `https://github.com/sp4cerat/Fast-Quadric-Mesh-Simplification`

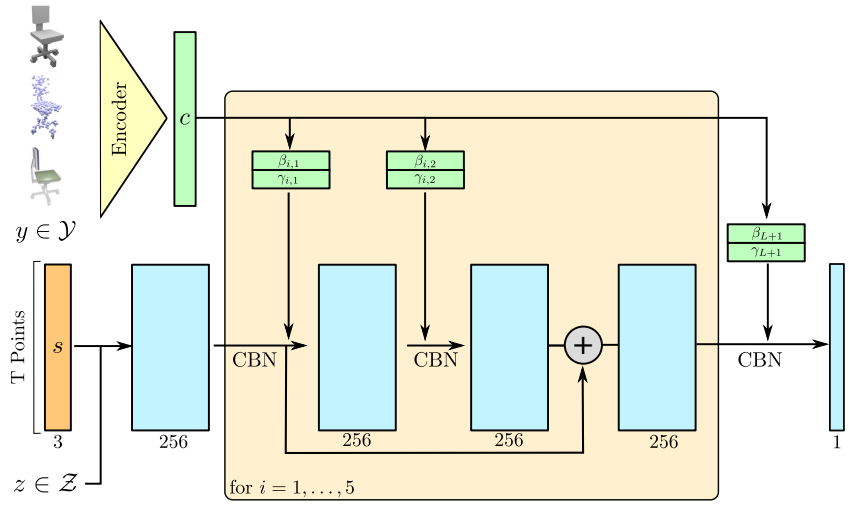


Figure 11.3: Occupancy Network Architecture. *We first compute an embedding c of the input. We then feed the input points through multiple fully-connected ResNet-blocks. In these ResNet-blocks, we use Conditional Batch Normalization to condition the network on c. Finally, we project the output of our network to one dimension using a fully-connected layer and apply the sigmoid function to obtain occupancy probabilities.*

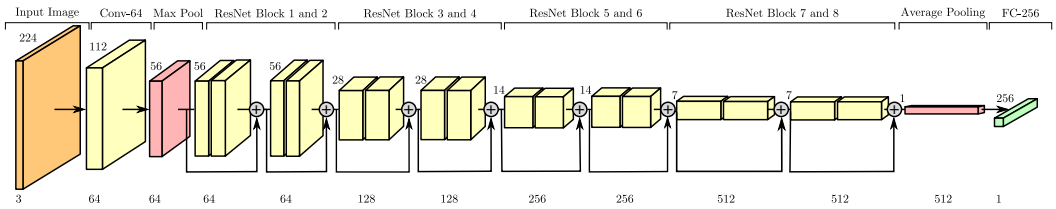
where $n(s_k)$ denotes the normal vector of the mesh at s_k. In practice, we set $\gamma = 0.01$. Minimization of the second term in (11.6) uses second order gradient information and can be efficiently implemented using Double-Backpropagation [41].

Note that this last step removes the discretization artifacts of the Marching Cubes approximation and would not be possible if we had directly predicted a voxel-based representation. In addition, our approach also allows to efficiently extract normals for all vertices of our output mesh by simply backpropagating through the Occupancy Network.

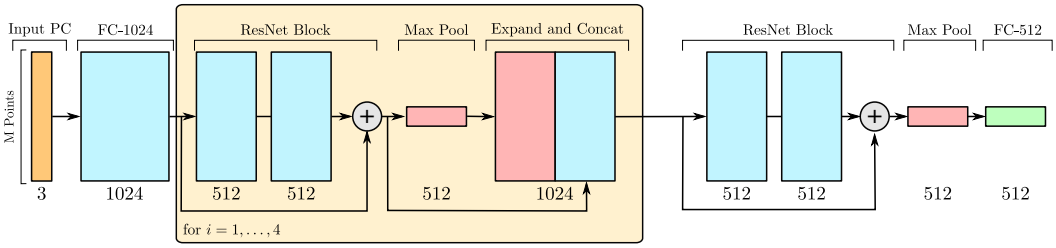
11.5 Implementation Details

We use the same basic Occupancy Network architecture (Figure 11.3) for all experiments. To this end, we first encode the condition $y \in \mathcal{Y}$ into a vector $c \in \mathbb{R}^C$ using a task-specific encoder network $g_\psi(\cdot)$. Our model then takes the output $c \in \mathbb{R}^C$ and a batch of T 3D-coordinates as input. In a generative setting, our network also takes a sampled latent code $z \sim p_0$ as input. The points are passed through a fully-connected layer to produce a 256-dimensional feature vector for each point. This feature vector is then passed through five pre-activation ResNet-blocks: each ResNet-block first applies Conditional Batch Normalization (CBN) [42, 192] to the current feature vector followed by a ReLU activation function. The output is then fed into a fully-connected layer, a second CBN layer, a ReLU activation and another fully-connected layer. The output of this operation is then added to the input of the ResNet-block.

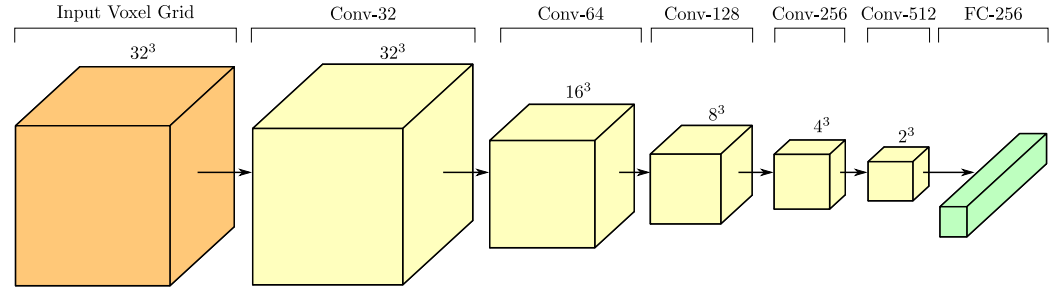
The output is finally passed through a last CBN layer and ReLU activation followed by



***(a)** Single Image 3D Reconstruction.*



***(b)** Point Cloud Completion.*



***(c)** Voxel Super-Resolution.*

Figure 11.4: Encoder Architectures. *Depending on the task, we leverage different encoder architectures: (a) For image input, we use a ResNet-18 architecture [67], which was pretrained on ImageNet. (b) For point cloud input, we use a PointNet encoder [158] with additional pooling and expansion layers. (c) For voxel input, we use a 3D Convolutional Neural Network.*

a fully-connected layer that projects the features down to one dimension. To obtain the occupancy probability, this vector can simply be passed through a sigmoid activation.

The CBN layers are implemented in the following way: first, we pass the conditional encoding c, which we obtained from the encoder network $g_\psi(\cdot)$, through fully-connected layers to obtain 256-dimensional vectors $\beta(c)$ and $\gamma(c)$. We then normalize the 256-dimensional input feature vector f_{in} using first and second order moments, multiply the output with $\gamma(c)$ and add the bias term $\beta(c)$:

$$f_{out} = \gamma(c)\frac{f_{in} - \hat{\mu}}{\sqrt{\hat{\sigma}^2 + \varepsilon}} + \beta(c), \tag{11.7}$$

Here, $\hat{\mu}$ and $\hat{\sigma}$ denote the empirical mean and standard-deviation (over the batch- and T-dimensions) of the input features f_{in} and $\varepsilon = 10^{-5}$ (the default value in PyTorch). Moreover, we compute running means over $\hat{\mu}$ and $\hat{\sigma}^2$ with momentum 0.1 during training. At test time, we replace $\hat{\mu}$ and $\hat{\sigma}^2$ with the corresponding running means.

The encoder network $g_\psi(\cdot)$ depends on the task (Figure 11.4): For single view 3D reconstruction we use a ResNet-18 architecture [67] (Figure 11.4a) which was pretrained on the ImageNet dataset [35]. However, we adjust the last fully-connected layer to project the features to a 256-dimensional embedding c. For point cloud completion we use a PointNet encoder [158] (Figure 11.4b) with five ResNet-blocks. To enable communication between points at lower layers, we also add pooling and expansion layers between the ResNet-blocks. After the ResNet-blocks, the final output is pooled using max-pooling and then projected to a 512 embedding vector using a fully-connected layer. For voxel super-resolution we use a 3D Convolutional Neural Network (Figure 11.4c) that encodes the 32^3 input into a 256-dimensional embedding vector c. We implement the encoder network $g^{\text{vae}}_\psi(\cdot)$ for our generative model as a PointNet [158] that takes a batch $(s_{ij}, o_{ij})_{j=1:K}$ of sampled points and ground truth occupancies $o_{ij} = x_i(s_{ij})$ as well as $c_i = g_\psi(y_i)$ as input and predicts mean μ_i and standard deviation σ_i of a Gaussian distribution $q_\psi(z|x_i, y_i)$ on latent $z \in \mathcal{Z}$ as output, The architecture of $g^{\text{vae}}_\psi(\cdot)$ is similar to the model shown in Figure 11.4b, but we only use four fully-connected blocks instead of the five ResNet-blocks. Moreover, we replace the last fully-connected layer with two fully-connected layers to produce both the mean μ_i and log-standard-deviation $\log \sigma_i$ of the 128 dimensional latent code z.

In all experiments, we use the Adam optimizer [96] with a learning rate of $h = 10^{-4}$ and no weight decay. For other hyperparameters of Adam we use PyTorch defaults: $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\varepsilon = 10^{-8}$. For image input, we augment the input by scaling it with a random factor between 0.75 and 1 and taking a random crop.

11.6 Experimental Results

We conduct three types of experiments to validate the proposed Occupancy Network. First, we analyze the **representation power** of Occupancy Networks by examining how well the network can reconstruct complex 3D shapes from a learned latent embedding. This gives us an upper bound on the results we can achieve when conditioning our representation on additional input. Second, we **condition** our Occupancy Networks on images, noisy

point clouds and low resolution voxel representations, and compare the performance of our method to several state-of-the-art baselines. Finally, we examine the **generative** capabilities of Occupancy Networks by adding an encoder $g_{\psi}^{\text{vae}}(\cdot)$ to our model and generating unconditional samples from this model.[4]

11.6.1 Baselines

For the single image 3D reconstruction task, we compare our approach against several state-of-the-art baselines which leverage various 3D representations: we evaluate against 3D Recurrent Reconstruction Neural Network (3D-R2N2) [26] as a voxel-based method, Point Set Generation Network (PSGN) [45] as a point-based technique and Pixel2Mesh [193] as well as AtlasNet [63] as mesh-based approaches. For point cloud inputs, we adapt 3D-R2N2 and PSGN by changing the encoder. As mesh-based baseline, we use Deep Marching Cubes (DMC) [108] which has recently reported state-of-the-art results on this task. For the voxel super-resolution task we assess the improvements with respect to the input.

In order to conduct controlled experiments and to disentangle the individual components of the different 3D reconstruction approaches, we created a PyTorch package comprising our method and several baselines [26, 45, 108, 193]. To make sure that the performance of our reimplementations of the baselines matches the performance of the original implementations, we conduct thorough comparisons to the original models. Please see Appendix E.2 for details.

11.6.2 Dataset

For our experiments we use the ShapeNet [22] subset of Choy et al. [26]. We also use the same 32^3 voxelization, image renderings and train/test split (80% and 20% of the whole dataset) as Choy et al. Moreover, we subdivide the training data into a training (70% of the whole dataset) and a validation set (10% of the whole dataset) on which we track the loss of our method and the baselines to determine when to stop training.

In order to determine if a point lies in the interior of a mesh (e.g., for measuring Intersection over Union (IoU)), we need the meshes to be watertight. We therefore use the code[5] provided by Stutz and Geiger [179], which performs TSDF-fusion on random depth renderings of the object, to create watertight versions of the meshes. We center and rescale all meshes so that they are aligned with the voxelizations from [26]. In practice, this means that we transform the meshes so that the 3D bounding box of the mesh is centered at 0 and its longest edge has a length of 1. We then sample 100k points offline in the unit cube centered at 0 with an additional small padding of 0.05 on both sides and determine if the points lie inside or outside the watertight mesh. To this end, we count the number of triangles that a ray which starts at the given point and which is parallel to the z-axis intersects. If this number is even the point lies outside the mesh, otherwise it lies inside. We save both the positions of the 100k points and their occupancies to a file. During training,

[4] The code to reproduce our experiments is available under `https://github.com/LMescheder/Occupancy-Networks`.

[5] `https://github.com/davidstutz/mesh-fusion`

we subsample 2048 points from this set (with replacement) as training data. Similarly, we also sample 100k points from the surface of the object and save them to a file. We accelerate data preprocessing using the GNU parallel tool [183]. For a fair comparison, we sample points from the surface of the watertight mesh instead of the original model as ground truth for PSGN [45], Pixel2Mesh [193] and DMC [108]. All of our evaluations are conducted with respect to these watertight meshes.

Because Choy et al. [26] only provide 32^3 voxelizations in their dataset, we compute the voxelizations for the experiment in Section 11.6.4 ourselves. To this end, we first detect all voxels that intersect the surface of the mesh and mark them as occupied. We then test for a random point in the interior of each voxel if it lies inside or outside the mesh. We mark the corresponding voxel as occupied if the first case is true. This procedure marks all voxels as occupied which intersect the mesh (or its interior).

11.6.3 Metrics

For evaluation we use the volumetric IoU, the Chamfer-L_1 distance and a Normal Consistency score.

In the following, let M_{Pred} and M_{GT} be the set of all points that are inside or on the predicted and ground truth mesh, respectively. The volumetric IoU is defined as the quotient of the volume of the two meshes' intersection and the volume of their union:

$$\text{IoU}(M_{Pred}, M_{GT}) := \frac{|M_{Pred} \cap M_{GT}|}{|M_{Pred} \cup M_{GT}|} \tag{11.8}$$

We obtain unbiased estimates of these volumes by randomly sampling 100k points from the bounding volume and determining if the points lie inside or outside M_{Pred} and M_{GT}, respectively.

We define the Chamfer-L_1 distance between the two meshes as

$$\begin{aligned} \text{Chamfer-}L_1(M_{Pred}, M_{GT}) := &\frac{1}{2\,|\partial M_{Pred}|} \int_{\partial M_{Pred}} \min_{\tilde{s} \in \partial M_{GT}} \|s - \tilde{s}\| \mathrm{d}s \\ &+ \frac{1}{2\,|\partial M_{GT}|} \int_{\partial M_{GT}} \min_{s \in \partial M_{Pred}} \|s - \tilde{s}\| \mathrm{d}\tilde{s} \end{aligned} \tag{11.9}$$

where ∂M_{Pred} and ∂M_{GT} denote the surfaces of the two meshes. Moreover, we define an Accuracy and Completeness score of M_{Pred} with respect to M_{GT}:

$$\text{Accuracy}(M_{Pred}|M_{GT}) := \frac{1}{|\partial M_{Pred}|} \int_{\partial M_{Pred}} \min_{\tilde{s} \in \partial M_{GT}} \|s - \tilde{s}\| \mathrm{d}s \tag{11.10}$$

$$\text{Completeness}(M_{Pred}|M_{GT}) := \frac{1}{|\partial M_{GT}|} \int_{\partial M_{GT}} \min_{s \in \partial M_{Pred}} \|s - \tilde{s}\| \mathrm{d}\tilde{s} \tag{11.11}$$

Note that this definition implies that *lower* Accuracy and Completeness are *better*. Moreover, note that the Chamfer-L_1 distance is just the mean of the Accuracy and Completeness scores.

Figure 11.5: Discrete vs. Continuous. *Qualitative comparison of our continuous representation (right) to voxelizations at various resolutions (left). Note how our representation encodes details which are lost in voxel-based representations.*

Similarly, we define the Normal Consistency score as

$$\begin{aligned}\text{Normal-Consistency}(M_{Pred}, M_{GT}) := &\frac{1}{2\,|\partial M_{Pred}|}\int_{\partial M_{Pred}} |n(s)^\mathsf{T} \cdot n(\text{proj}_2(s))|\mathrm{d}s \\ &+ \frac{1}{2\,|\partial M_{GT}|}\int_{\partial M_{GT}} |n(\tilde{s})^\mathsf{T} \cdot n(\text{proj}_1(\tilde{s}))|\mathrm{d}\tilde{s}\end{aligned} \tag{11.12}$$

where $n(s)$ and $n(\tilde{s})$ are the (unit) normal vectors on mesh surfaces ∂M_{Pred} and ∂M_{GT}, respectively, and $\text{proj}_2(s)$ and $\text{proj}_1(\tilde{s})$ denote the projections of s and $\tilde{s}$ onto ∂M_{GT} and ∂M_{Pred}. As a result, a *higher* Normal Consistency score is *better*.

We estimate all four quantities efficiently by sampling 100k points from the surface of both meshes and employing a KD-tree to determine the corresponding nearest neighbors from the other mesh.

11.6.4 Representation Power

In our first experiment, we investigate how well Occupancy Networks represent 3D geometry, independent of the inaccuracies of the input encoding. The question we try to answer in this experiment is whether our network can learn a memory efficient representation of 3D shapes while at the same time preserving as many details as possible. This gives us an estimate of the representational capacity of our model and an upper bound on the performance we may expect when conditioning our model on additional input. Similarly to [184], we embed each training sample in a 512-dimensional latent space and train our neural network to reconstruct the 3D shape from this embedding.

We apply our method to the training split of the "chair" category of the ShapeNet dataset. This subset is challenging to represent as it is highly varied and many models contain high-frequency details. Since we are only interested in reconstructing the training data, we do not use separate validation and test sets for this experiment.

For evaluation, we measure the volumetric IoU to the ground truth mesh. Quantitative results and a comparison to voxel representations at various resolutions are shown in Figure 11.6. We see that the Occupancy Network is able to faithfully represent the entire dataset with a high mean IoU of 0.89 while a low-resolution voxel representation is not able

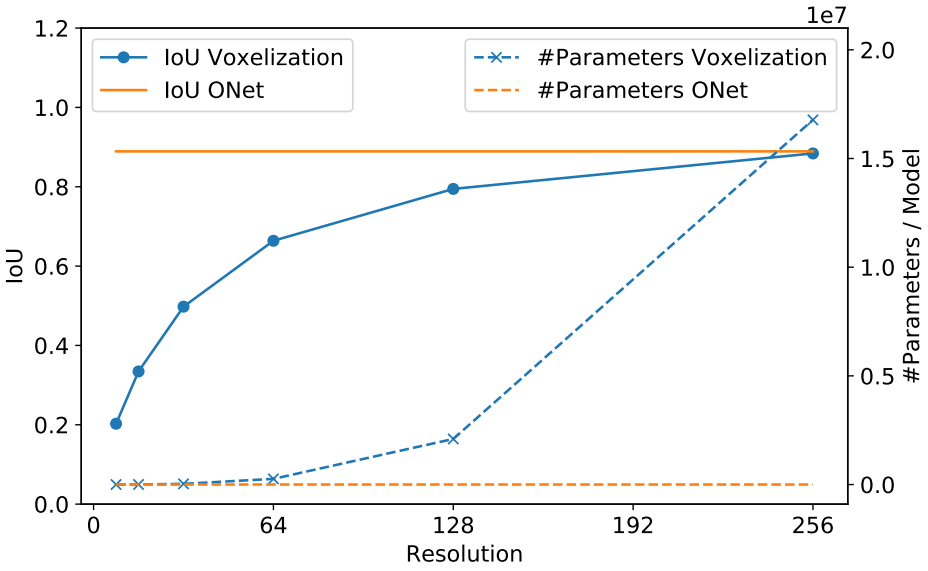


Figure 11.6: IoU vs. Resolution. *This plot shows the IoU of a voxelization to the ground truth mesh (solid blue line) in comparison to our continuous representation (solid orange line) as well as the number of parameters per model needed for the two representations (dashed lines). Note how our representation leads to larger IoU with respect to the ground truth mesh compared to a low-resolution voxel representation. At the same time, the number of parameters of a voxel representation grows cubically with the resolution, whereas the number of parameters of Occupancy Networks is independent of the resolution.*

to represent the meshes accurately. At the same time, the Occupancy Network is able to encode all 4746 training samples with as little as 6 million parameters, independently of the resolution. In contrast, the memory requirements of a voxel representation grow cubically with the spatial resolution. Qualitative results are shown in Figure 11.5. We observe that the Occupancy Network enables us to represent details of the 3D geometry which are lost in a low-resolution voxelization.

11.6.5 Single Image 3D Reconstruction

In our second experiment, we condition the Occupancy Network on an additional view of the object from a random camera location. The goal of this experiment is to evaluate how well Occupancy Functions can be inferred from complex input. While we train and test our method on the ShapeNet dataset, we also present qualitative results for the KITTI [53] and the Stanford Online Products [146] datasets. Moreover, we present a quantitative and qualitative evaluation on the Pix3D dataset [180].

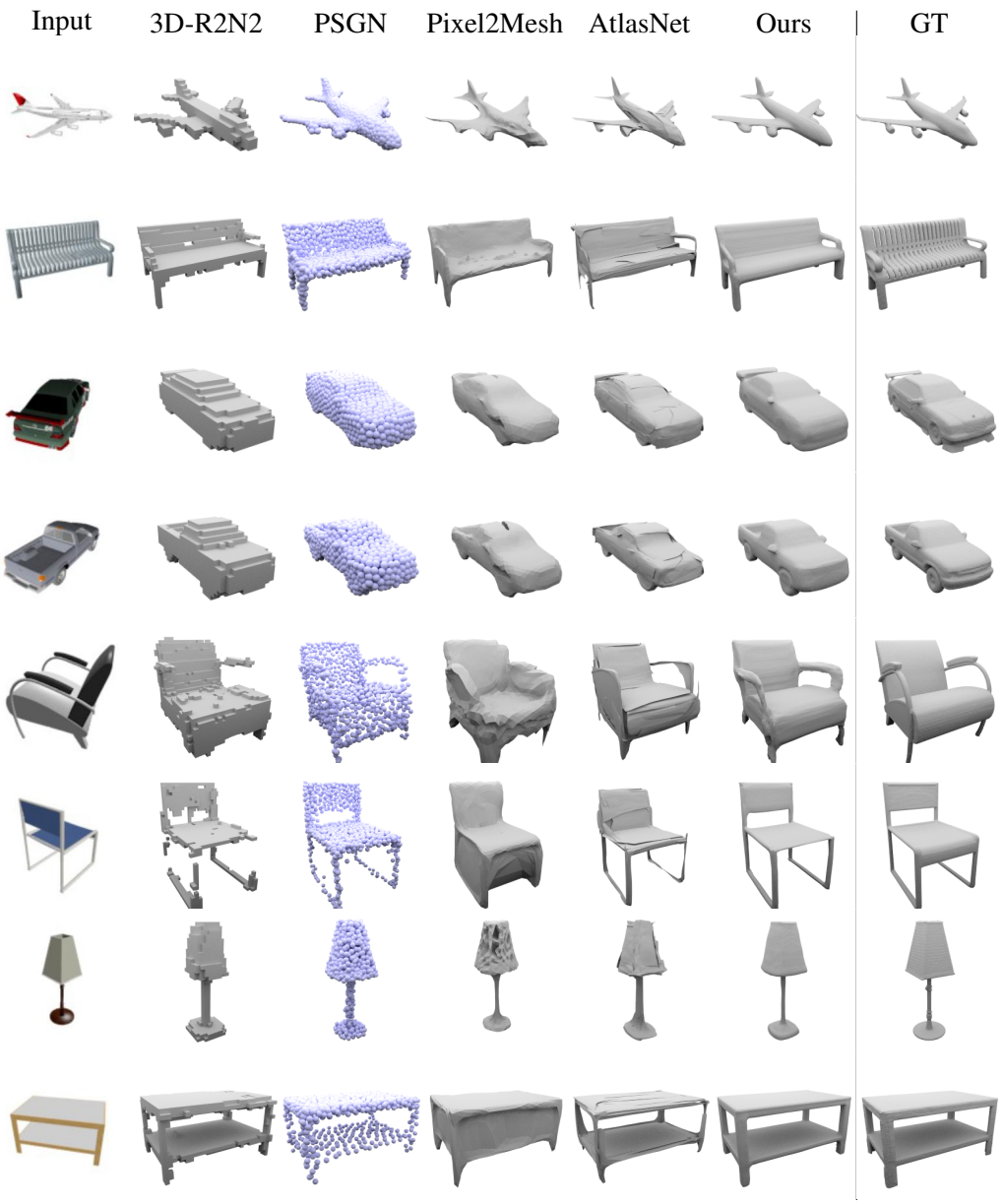


Figure 11.7: Single Image 3D Reconstruction. *The input image is shown in the first column, the other columns show the results for our method compared to various baselines and the ground truth.*

	IoU	Chamfer-L_1	Normal Consistency
3D-R2N2	49.3%	0.278	0.695
PSGN	-	0.215	-
Pixel2Mesh	48.0%	0.216	0.772
AtlasNet	-	**0.175**	0.811
Occupancy Network	**57.1%**	0.215	**0.834**

Table 11.1: Single Image 3D Reconstruction. *This table shows a numerical comparison of our approach and the baselines for single image 3D reconstruction on the ShapeNet dataset. Please see Table F.1 in Appendix F.3 for a per-category evaluation.*

ShapeNet

In this experiment, we use a ResNet-18 image encoder [67], which was pretrained on ImageNet [171]. For a fair comparison, we use the same image encoder for both 3D-R2N2 and PSGN. For PSGN we use a fully-connected decoder with 4 layers and 512 hidden units in each layer. The last layer projects the hidden representation to a 3072-dimensional vector which we reshape into 1024 3D points. As we use only a single input view, we remove the recurrent network in 3D-R2N2. We reimplemented Pixel2Mesh [193] in PyTorch, closely following the Tensorflow implementation provided by the authors. For AtlasNet [63], we use the code and pretrained model[6] from the authors.

For all methods, we track the loss and other metrics on the validation set and stop training as soon as the target metric reaches its optimum. For 3D-R2N2 and our method we use the IoU to the ground truth mesh as target metric, for PSGN and Pixel2Mesh we use the Chamfer distance to the ground truth mesh as target metric. To extract the final mesh, we use a threshold of 0.4 for 3D-R2N2 as suggested in the original publication [26]. To choose the threshold parameter τ for our method, we perform grid search on the validation set (see Section 11.6.9) and found that $\tau = 0.2$ yields a good trade-off between Accuracy and Completeness.

Qualitative results from our model and the baselines are shown in Figure 11.7. We observe that all methods are able to capture the 3D geometry of the input image. However, 3D-R2N2 produces a very coarse representation and hence lacks details. In contrast, PSGN produces a high-fidelity output, but lacks connectivity. As a result, PSGN requires additional lossy post-processing steps to produce a final mesh[7]. Pixel2Mesh is able to create compelling meshes, but often misses holes in the presence of more complicated topologies. Such topologies are frequent, for example, for the "chairs" category in the ShapeNet dataset. Similarly, AtlasNet captures the geometry well, but produces artifacts in form of self-intersections and overlapping patches. In contrast, our method is able to capture complex topologies, produces closed meshes and preserves most of the details.

Quantitative results[8] are shown in Table 11.1. We observe that our method achieves the

[6] `https://github.com/ThibaultGROUEIX/AtlasNet`

[7] See Figure E.1 in Appendix F.3 for meshing results.

[8] We later found that our numerical results improve when we do not use random crops and scaling to augment the input images (IoU: 0.593, Chamfer-L_1: 0.194, Normal Consistency: 0.840). However, for consistency

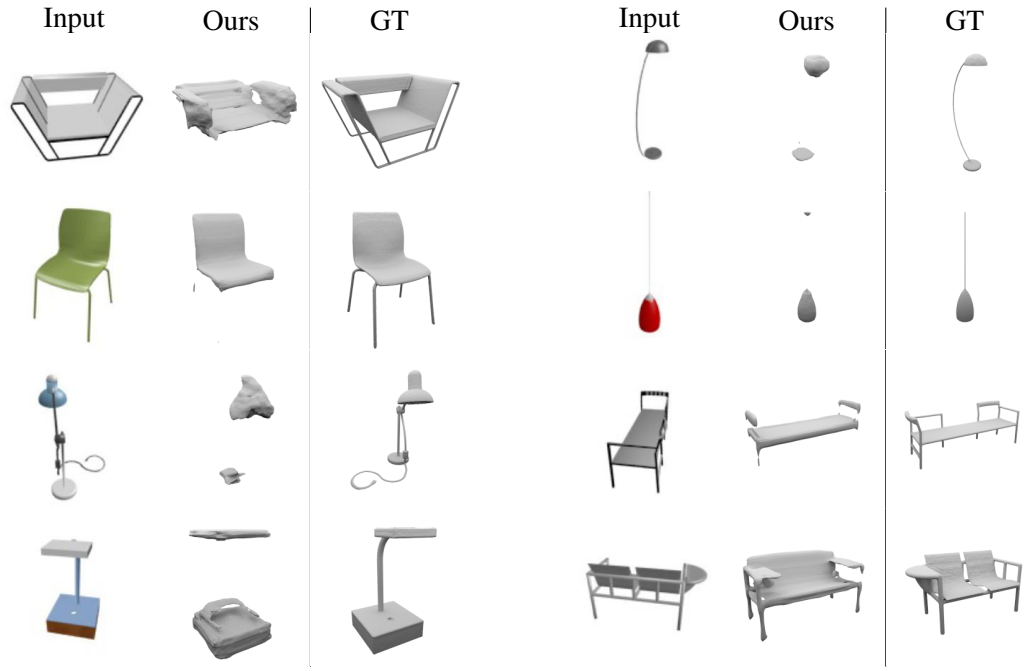


Figure 11.8: Failure Cases. *While our method generally performs well, it struggles with extremely thin object parts and objects that are very different from the objects seen during training. These kinds of objects are especially frequent for the "lamp' category of the ShapeNet dataset. The input is shown in the first column, the other columns show the results for our method compared to the ground truth.*

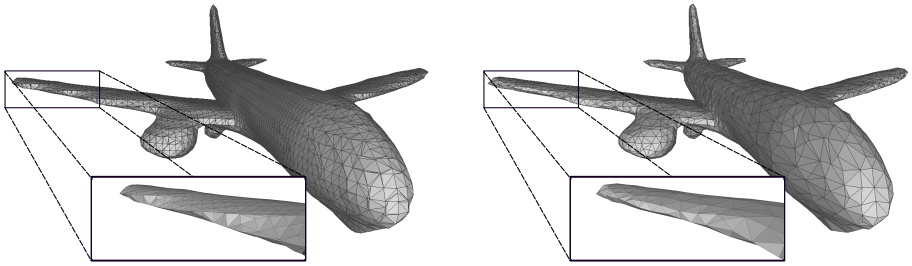
highest IoU and Normal Consistency score to the ground truth mesh. Surprisingly, while not trained with respect to Chamfer distance as PSGN, Pixel2Mesh or AtlasNet, our method also achieves good results for this metric. Note that it is not possible to evaluate the IoU for PSGN or AtlasNet, as they do not yield watertight meshes.

The inference time of our algorithm with simplification and refinement steps at resolution 128^3 is about 3s / mesh. When we remove the simplification and refinement steps, the inference time of our method reduces to 603ms / mesh with little degradation of the results. While our method is slower compared to the baselines (3D-R2N2 [26]: 11ms, PSGN [45]: 10ms, Pixel2Mesh [193]: 31ms), we can trade off accuracy and efficiency in our inference procedure by adaptively choosing the resolution: if we extract meshes at 32^3 resolution instead and omit refinement, our inference time reduces to 41ms (32ms w/o Marching Cubes).

Figure 11.9 shows results from the different stages of the MISE algorithm. We find that even without further refinement our algorithm generates high-quality meshes. Moreover, when we additionally apply simplification and refinement, the remaining discretization artifacts (e.g. on the wings of the airplane in Figure 11.9) disappear.

Some failure cases for our method are shown in Figure 11.8. While in general our method

and to ensure a fair comparison to the baselines, we report the slightly worse results with data augmentation.



(a) *After Marching Cubes* ***(b)*** *After Simplification and Refinement*

Figure 11.9: Multiresolution IsoSurface Extraction. *This figure visualizes the different stages of Multiresolution IsoSurface Extraction (MISE).*

	IoU	Chamfer-L_1	Normal Consistency
3D-R2N2	23.4%	1.888	0.489
PSGN	-	0.770	-
AtlasNet	-	**0.456**	0.752
Occupancy Network	31.9%	0.793	0.733
Occupancy Network+	**44.4%**	0.504	**0.799**

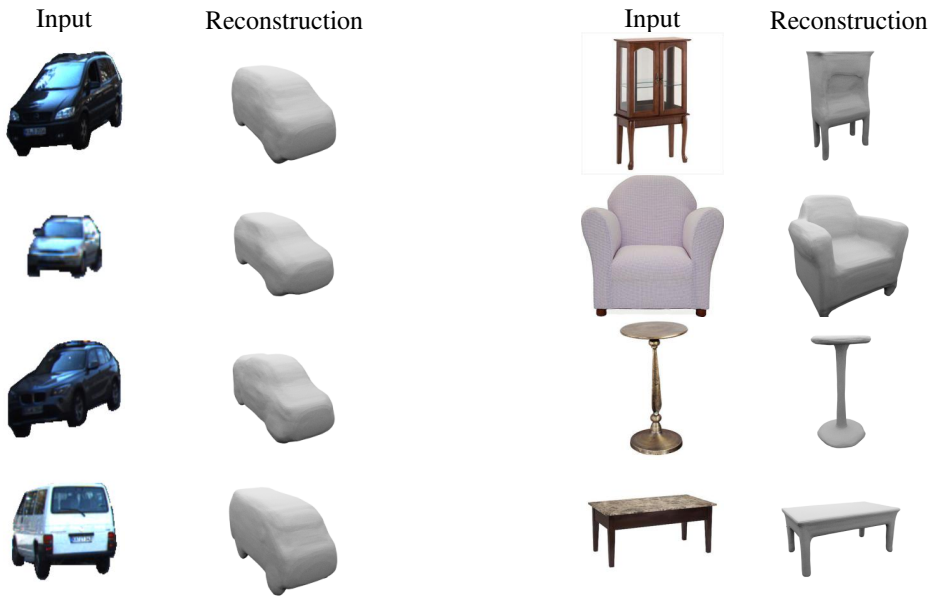
Table 11.2: Pix3D Dataset. *We evaluate the networks which were trained on ShapeNet on the Pix3D dataset [180] with ground truth masks. Occupancy Network+ was trained on our own renderings with more varied random views. Please see Table F.2 in Appendix F.3 for a per-category evaluation.*

performs well, we observe that our method sometimes has problems with very thin object parts. Objects with such parts are especially frequent in the "lamp" category of the ShapeNet dataset. Note that thin object parts are especially problematic for volumetric approaches to 3D reconstruction like our approach and 3D-R2N2 [26]. Approaches that take the metric of the 3D space into account such as PSGN, Pixel2Mesh and AtlasNet are less prone to these kinds of problems. However, these methods either only produce points clouds that have to be meshed in a postprocessing step, can only represent meshes with very simple topology or do not lead to watertight meshes.

Real Data

To test how well our model generalizes to real data, we apply our network to the KITTI [53] and Stanford Online Products datasets [146] as well as the Pix3D dataset [180]. To capture the variety in viewpoints on these datasets, we rerendered all ShapeNet objects with random camera locations and retrained our network for this task.

KITTI For the KITTI dataset, we additionally use the instance masks provided by Alhaija et al. [3] to mask and crop car regions. We then feed these images into our neural



(a) KITTI *(b) Online Products*

Figure 11.10: Real Data. *We applied our trained model to the KITTI and Stanford Online Products datasets. Despite only trained on synthetic data, our model generalizes reasonably well to real data.*

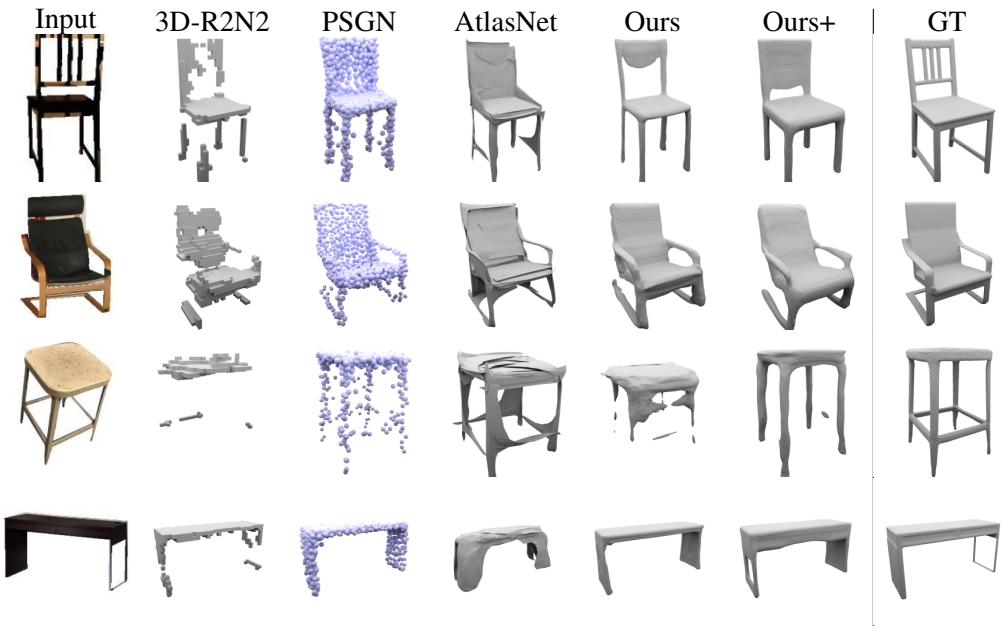


Figure 11.11: Pix3D Dataset. *While all method generalize reasonably well to Pix3D [180], they perform worse and show more artifacts than on synthetic data.*

network to predict the Occupancy Function. Some selected qualitative results are shown in Figure 11.10a. Despite only trained on synthetic data, we observe that our method is also able to generate realistic reconstructions in this challenging setting.

Online Products For the Stanford Online Products dataset, we apply the same pretrained model. Several qualitative results are shown in Figure 11.10b. Again, we observe that our method generalizes reasonably well to real images despite being trained solely on synthetic data.

Pix3D dataset We also evaluate the networks which were trained on ShapeNet on the Pix3D dataset [180] with ground truth masks. For all methods, we mask out the background and crop the image so that the object is roughly in the middle of the image. As all methods are only trained on 13 ShapeNet classes, we only use the four object classes (chair, desk, sofa, table) from Pix3D that are similar to one of those 13 ShapeNet classes.

Qualitative and quantitative results are shown in Figure 11.11 and Table 11.2, respectively. We find that all methods generalize reasonably well to real data, but achieve worse results than on synthetic data. We also find that the Occupancy Network trained on more random views (ONet+) generalizes better than Occupancy Network trained on the renderings by Choy et al. [26]. This shows that the main limiting factor on this dataset is the realism of the training dataset. We therefore expect that it is possible to further narrow the gap to synthetic data by using even more varied and higher quality renderings as well as better data augmentation strategies.

11.6.6 Point Cloud Completion

As a second conditional task, we apply our method to the problem of reconstructing the mesh from noisy point clouds. Towards this goal, we subsample 300 points from the surface of each of the (watertight) ShapeNet models and apply noise using a Gaussian distribution with zero mean and standard deviation 0.05 to the point clouds.

Again, we measure both the IoU and Chamfer-L_1 distance with respect to the ground truth mesh. The results are shown in Table 11.3. We observe that our method achieves the highest IoU and Normal Consistency score as well as the lowest Chamfer-L_1 distance. Note that all numbers are significantly better than for the single image 3D reconstruction task. This can be explained by the fact that this task is much easier for the recognition model, as there is less ambiguity and the model only has to fill in the gaps. Qualitative results are shown Figure F.12 and Figure F.12 in Appendix F.3.

11.6.7 Voxel Super-Resolution

As a final conditional task, we apply Occupancy Networks to 3D super-resolution [175]. Here, the task is to reconstruct a high-resolution mesh from a coarse 32^3 voxelization of this mesh.

	IoU	Chamfer-L_1	Normal Consistency
3D-R2N2	56.5%	0.169	0.719
PSGN	-	0.144	-
DMC	67.4%	0.117	0.848
Occupancy Network	**77.8%**	**0.079**	**0.895**

Table 11.3: 3D Reconstruction from Point Clouds. *This table shows a numerical comparison of our approach with respect to the baselines for 3D reconstruction from point clouds on the ShapeNet dataset. We measure IoU, Chamfer-L_1 distance and Normal Consistency score with respect to the ground truth mesh. Please see Table F.3 in Appendix F.3 for a per-category evaluation.*

	IoU	Chamfer-L_1	Normal Consistency
Input	63.1%	0.136	0.810
Occupancy Network	**70.3%**	**0.109**	**0.879**

Table 11.4: Voxel Super-Resolution. *This table shows a numerical comparison of the output of our approach in comparison to the input on the ShapeNet dataset.*

The results are shown in Table 11.4. We observe that our model considerably improves IoU, Chamfer-L_1 distance and Normal Consistency score compared to the coarse input mesh. Please see Figure F.14 in Appendix F.3 for qualitative results.

11.6.8 Unconditional Mesh Generation

Finally, we apply our Occupancy Network to unconditional mesh generation, training it separately on four categories of the ShapeNet dataset in an unsupervised fashion. Our goal is to explore how well our model can represent the latent space of 3D models. Some samples are shown in Figure 11.12. Indeed, we find that our model can generate compelling new models. In Figure F.15, F.16, F.17 and F.17 in Appendix F.3 we show interpolations in latent space for our models, showing that our models learn a meaningful latent space of 3D objects.

11.6.9 Ablation Study

In this section we test how the various components of our model affect its performance on the single-image 3D reconstruction task.

Effect of Sampling Strategy First, we examine how the sampling strategy affects the performance of our final model. We try three different sampling strategies: (i) sampling 2048 points uniformly in the bounding volume of the ground truth mesh (uniform sampling), (ii) sampling 1024 points inside and 1024 points outside mesh (equal inside/outside) and (iii) sampling 1024 points uniformly and 1024 points on the surface of the mesh plus some

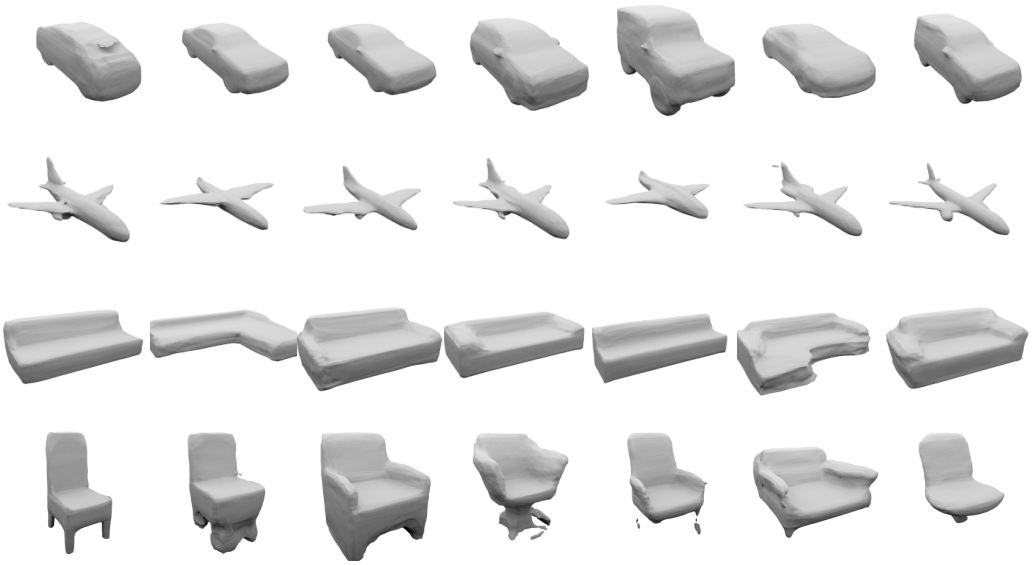


Figure 11.12: Unconditional 3D Samples. *Random samples of our unsupervised models trained on the categories "car", "airplane", "sofa" and "chair" of the ShapeNet dataset. We see that our models are able to capture the distribution of 3D objects and produce compelling new samples.*

Gaussian noise with standard deviation 0.1 (uniform + near surface). We also examine the effect of the number of sampling points by decreasing this number from 2048 to 64.

The results are shown in Table 11.5a. To our surprise, we find that uniform, the simplest sampling strategy, works best. We explain this by the fact that other sampling strategies introduce bias to the model: for example, when sampling an equal number of points inside and outside the mesh, we implicitly tell the model that every object has a volume of 0.5. Indeed, when using this sampling strategy, we observe thickening artifacts in the model's output. Moreover, we find that reducing the number of sampling points from 2048 to 64 still leads to good performance, although the model does not perform as well as a model trained with 2048 sampling points.

Effect of Architecture To test the effect of the various components of our architecture, we test two variations: (i) we remove the Conditional Batch Normalization and replace it with a linear layer in the beginning of the network that projects the encoding of the input to the required hidden dimension and (ii) we remove all ResNet-blocks in the decoder and replace them with linear blocks. The results are presented in Table 11.5b. We find that both components are helpful to achieve good performance.

Effect of the Threshold Parameter To understand the effect of the threshold parameter τ, we examine its effect on various metrics on a validation set. To this end, we vary τ from $\tau = 0.1$ to $\tau = 0.6$ and use MISE to extract the corresponding meshes. For each value

	IoU	Chamfer-L_1	Normal Consistency
uniform sampling (2048)	**57.1%**	**0.215**	0.834
uniform sampling (64)	55.4%	0.256	0.829
equal inside/outside	47.5%	0.291	**0.835**
uniform + near surface	53.6%	0.254	0.822

(a) *Influence of Sampling Strategy*

	IoU	Chamfer-L_1	Normal Consistency
full model	**57.1%**	**0.215**	**0.834**
no ResNet-blocks	55.9%	0.243	0.831
no CBN	52.2%	0.301	0.806

(b) *Influence of Network Architecture*

Table 11.5: Ablation Study. *When we vary the sampling strategy, we observe that uniform sampling in the bounding volume performs best. Similarly, when we vary the architecture, we find that our ResNet-architecture with Conditional Batch Normalization yields the best results.*

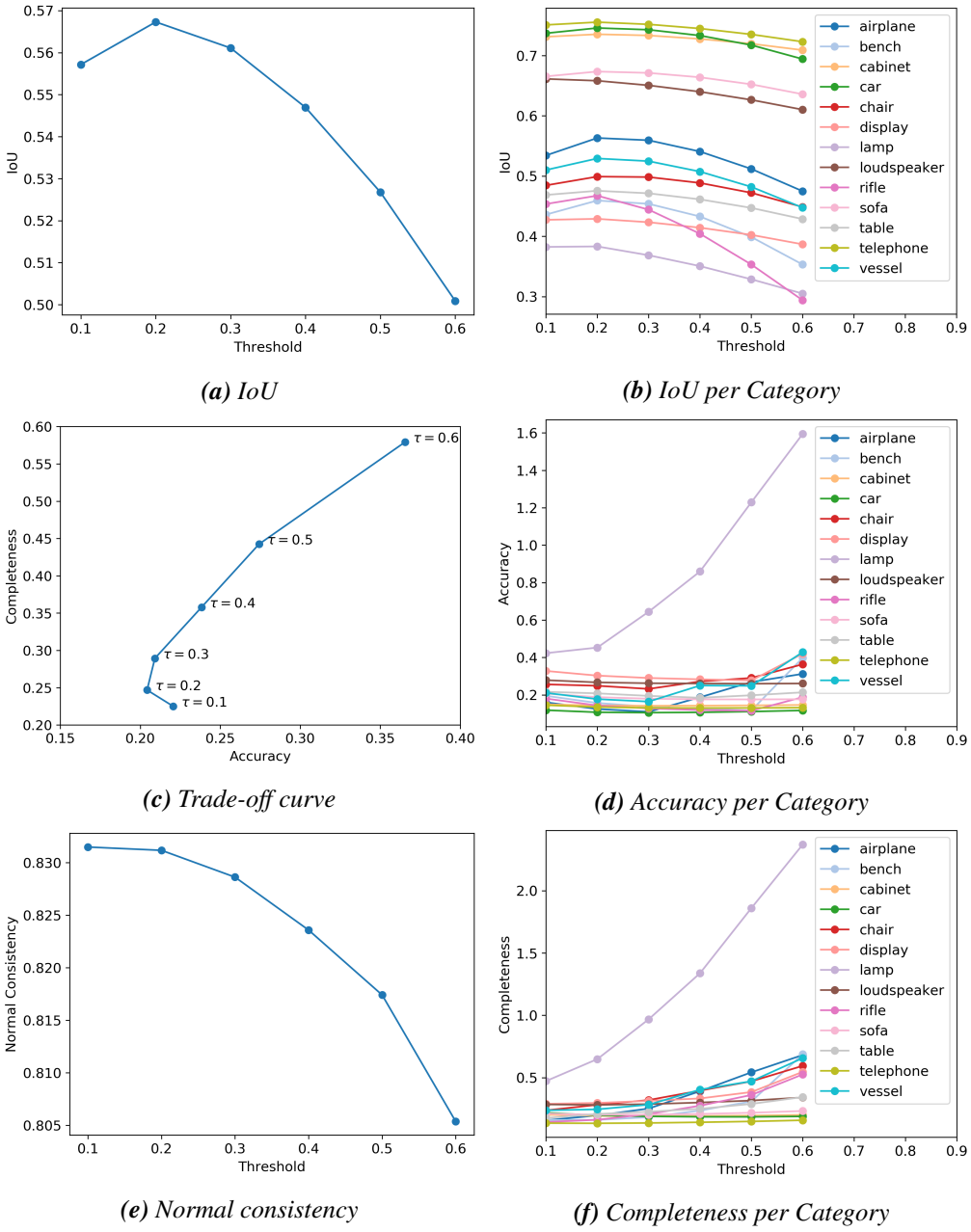
of τ, we measure the volumetric IoU, the Completeness, the Accuracy and the Normal Consistency.

The results are shown in Figure 11.13. We observe that IoU improves as we decrease τ until we reach the critical level of $\tau \approx 0.2$. For smaller τ, the IoU becomes worse again. As expected, we observe that the Completeness score becomes better with smaller values of τ. Surprisingly, we find that the Accuracy score also improves with smaller τ until it reaches its optimum at $\tau \approx 0.2$. This is counterintuitive, because a small value of τ could a priori lead to spurious geometries which should deteriorate the Accuracy score. We explain this counterintuitive observation by the fact that both the Accuracy and the Completeness are better when the gap between the two meshes is smaller. Although we might add some spurious geometry for some objects for small τ, the meshes are closer to each other which leads to better Accuracy and Completeness scores. This effect dominates the Accuracy until we reach $\tau \approx 0.2$. For smaller τ the Accuracy becomes worse again because spurious geometry appears. We also observe that some categories are more difficult to learn. In particular the "lamp" category is very challenging for our method due to very fine geometry.

All in all, we see that the selection of the threshold parameter is a critical component of our method and a threshold parameter of $\tau \approx 0.2$ yields the best performance.

11.7 Conclusion

In this chapter we have introduced Occupancy Networks, an application of the Function Space Operator to learning-based 3D reconstruction and 3D generative modeling. In contrast to existing representations for 3D geometry, Occupancy Networks are not constrained by the discretization of the 3D space and can hence be used to represent realistic high-resolution



***(a)** IoU*

***(b)** IoU per Category*

***(c)** Trade-off curve*

***(d)** Accuracy per Category*

***(e)** Normal consistency*

***(f)** Completeness per Category*

Figure 11.13: Threshold Parameter. *Effect of threshold parameter on IoU, Accuracy score, Completeness score and Normal Consistency score for the single-view 3D reconstruction experiment.*

meshes. Our experiments demonstrate that Occupancy Networks are very expressive and can be used effectively both for supervised and unsupervised learning. We hence believe that Occupancy Networks are a useful tool which can be applied to a wide variety of 3D tasks.

12 Extensions

In Chapter 10, we have introduced the Function Space Operator as a way of circumventing the Curse of Discretization. In Chapter 11 we applied this concept to the challenging tasks of learning-based 3D reconstruction and 3D generative modeling, yielding a type of model that we call Occupancy Network. However, the Function Space Operator is a much more general concept and can be applied to arbitrary deep learning tasks that can be described as having a function as output.

In this chapter we introduce two extensions of Occupancy Networks that illustrate the usefulness of the Function Space Operator for other applications. First, we show how we can extend Occupancy Networks to also predict texture information. Afterwards, we show how Occupancy Networks can be extended to 4D reconstruction.

While both of these extensions are an application of the Function Space Operator, they also require novel insights. In this thesis we only give a brief overview of the methods and refer interested readers to the respective publications [141, 145] for details.

12.1 Texture Fields

In Occupancy Networks we predict the geometry of 3D objects from a sampled latent code $z \in \mathcal{Z}$ and / or from some observation $y \in \mathcal{Y}$ such as a single view of the object.

While this approach produces plausible 3D geometry, there is clearly something missing: in the real world 3D objects have materials. Assuming that these materials are Lambertian, we can model them as a texture consisting of RGB-color values.

Historically, texture is often represented using a UV-mapping which maps every location on the surface to an image. Unfortunately, however, this representation requires a template model for the 3D shape and is thus hard to integrate into a deep learning pipeline. Recent deep learning approaches have hence resorted to a voxel representation where texture can be easily modeled by assigning a color value to each voxel.

When using colored voxels, we effectively learn a function $\mathcal{Z} \times \mathcal{Y} \rightarrow \mathcal{V}^{\hat{\mathcal{S}}}$, where $\mathcal{Y}$ is an observation space (e.g. single views of the object), $\mathcal{V} = \mathbb{R}^3$ represents the RGB-color values and $\hat{\mathcal{S}}$ is the discretized version of $\mathcal{S} = [0,1]^3$. However, as we have seen Chapter 10, we can circumvent the costly discretization step of $\mathcal{S}$ by applying the Function Space Operator: instead of learning a function $\mathcal{Z} \times \mathcal{Y} \rightarrow \mathcal{V}^{\hat{\mathcal{S}}}$, we can learn a function $\mathcal{S} \times \mathcal{Z} \times \mathcal{Y} \rightarrow \mathcal{V}$. We call the resulting model a Texture Field.

Our model [145] is visualized in Figure 12.1. Note that we also condition on the shape of the 3D object. Conditioning on the shape is not a limitation: we can always use our model from Chapter 11 to first reconstruct or generate the 3D shape before applying the Texture Field to it. We supervise our model with ground truth RGB images and depth maps. This

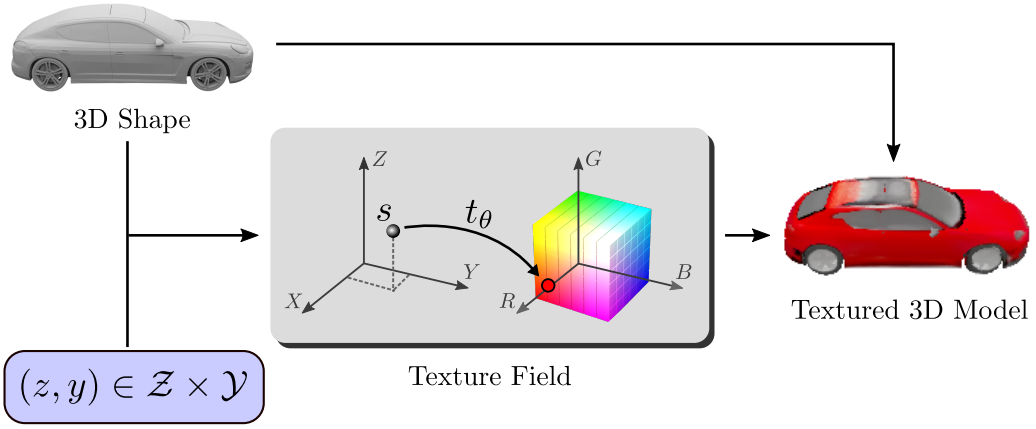


Figure 12.1: Texture Fields. *Texture Fields take a 3D shape and of an object and a tuple* $(z,y) \in \mathcal{Z} \times \mathcal{Y}$ *as input and output a continuous function* t_θ *which maps any 3D point* s *to a color value, enabling the prediction of accurately textured 3D models.*

way, we can train our generative model both as a Variational Autoencoder (VAE) and (by introducing an additional image discriminator) as a Generative Adversarial Network (GAN).

For details on the method, architectures and training see Oechsle et al. [145]. Here, we only show qualitative results for our (purely) generative model in Figure 12.3. One useful application of our generative model is to transfer texture from one model to another as visualized in Figure 12.4. The results in Figure 12.4 show that our model learns a useful latent texture representation that is disentangled from the shape of the object.

12.2 Occupancy Flow

While Occupancy Networks and Texture Fields allow to reconstruct and generate realistic static 3D objects, the real world is in motion. In principle, time-varying geometry can be modeled by learning a function $\mathcal{Z} \times \mathcal{Y} \rightarrow \mathcal{V}^{\mathcal{S}}$ where $\mathcal{S} = [0,1]^4$ is a space-time volume and $\mathcal{V} = [0,1]$ denotes occupancy probability. Again, we see that we can apply the Function Space Operator to turn this difficult problem into the tractable problem of approximating a function $\mathcal{S} \times \mathcal{Z} \times \mathcal{Y} \rightarrow \mathcal{V}$ with a neural network.

However, representing time-varying geometry this way has the drawback of not providing temporal correspondences: while the representation $\mathcal{S} \times \mathcal{Z} \times \mathcal{Y} \rightarrow \mathcal{V}$ allows to extract 3D meshes for every point in time, the model does not encode any information about 3D motion and therefore cannot reason about correspondences across time.

To circumvent this problem, we propose to learn two neural networks, the Occupancy Network $f_\theta : \mathcal{S}_0 \times \mathcal{Z} \times \mathcal{Y} \rightarrow \mathcal{V}_{occ}$ and the Velocity Network $v_\theta : \mathcal{S} \times \mathcal{Z} \times \mathcal{Y} \rightarrow \mathcal{V}_{vel}$ with $\mathcal{S}_0 = [0,1]^3$, $\mathcal{S} = [0,1]^4$, $\mathcal{V}_{occ} = [0,1]$ and $\mathcal{V}_{vel} = \mathbb{R}^3$, visualized in Figure 12.2. This results in spatio-temporal model which we call Occupancy Flow (OFlow). While the first network encodes the geometry of the object at $t = 0$, the second one describes a time-varying vector field, which encodes 3D motion. As we show in the main publication [141], these two

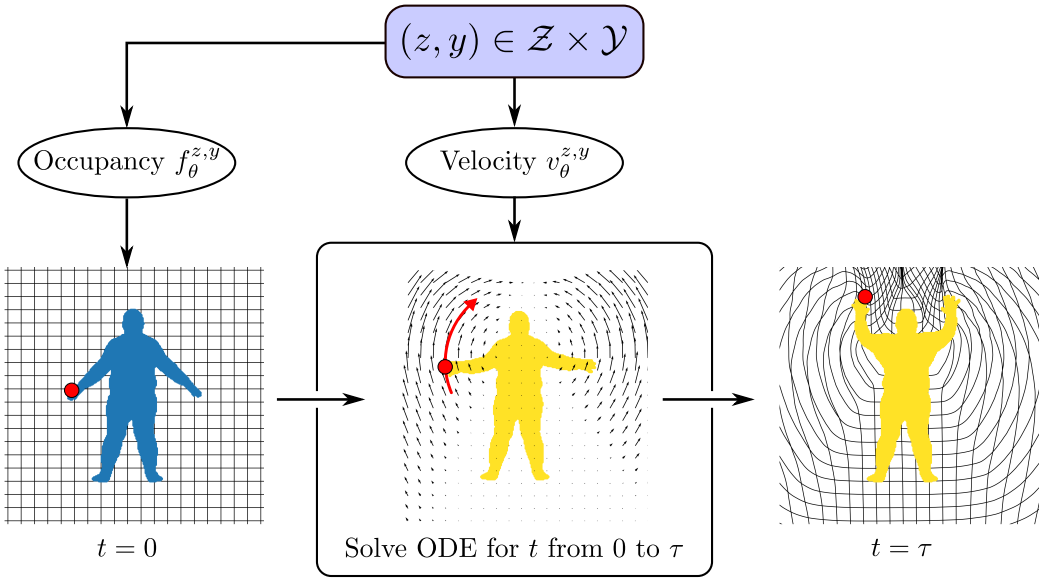


Figure 12.2: Occupancy Flow. *In Occupancy Flow, we use an Occupancy Network $f_\theta^{z,y}$ to predict a mesh at time $t = 0$. We then propagate points on the surface of the mesh at $t = 0$ forward in time by integrating an input-dependent vector field $v_\theta^{z,y}$.*

functions together define a time-varying Occupancy Function $\mathcal{S} \times \mathcal{Z} \times \mathcal{Y} \to \mathcal{V}_{occ}$ that can be trained in the same way as Occupancy Networks, optionally making use of additional temporal correspondences between points during training. During inference we can use the Occupancy Network $f_\theta(\cdot)$ to extract a mesh at $t = 0$ and then use the Velocity Network $v_\theta(\cdot)$ to propagate the vertices of mesh forward in time. This way, we obtain dense temporal correspondences for our results.

Again, we refer the interested reader to the main publication [141] for details. Here, we only show some qualitative results of our generative model in Figure 12.5 and Figure 12.6. In Figure 12.5a we show generated results when keeping the latent code for the shape fixed and interpolating the latent code for the motion. Similarly, in Figure 12.5b we show results when keeping the latent code for the motion fixed and interpolating the latent code for the shape. All in all, the results in Figure 12.5 show that we can learn a useful disentangled representation of shape and motion. Similarly to Texture Fields, we can use this insight to transfer the motion from one object to another, which is visualized in Figure 12.6.

12.3 Conclusion

In this chapter we have described the core ideas behind two extensions of Occupancy Networks: Texture Fields and Occupancy Flow. While Texture Fields enable us to also represent texture of 3D geometry, Occupancy Flow extends Occupancy Networks to 4D reconstruction and 4D generative modeling. Both extensions can be seen as an application of the Function Space Operator and hence illustrate the usefulness of the concept.

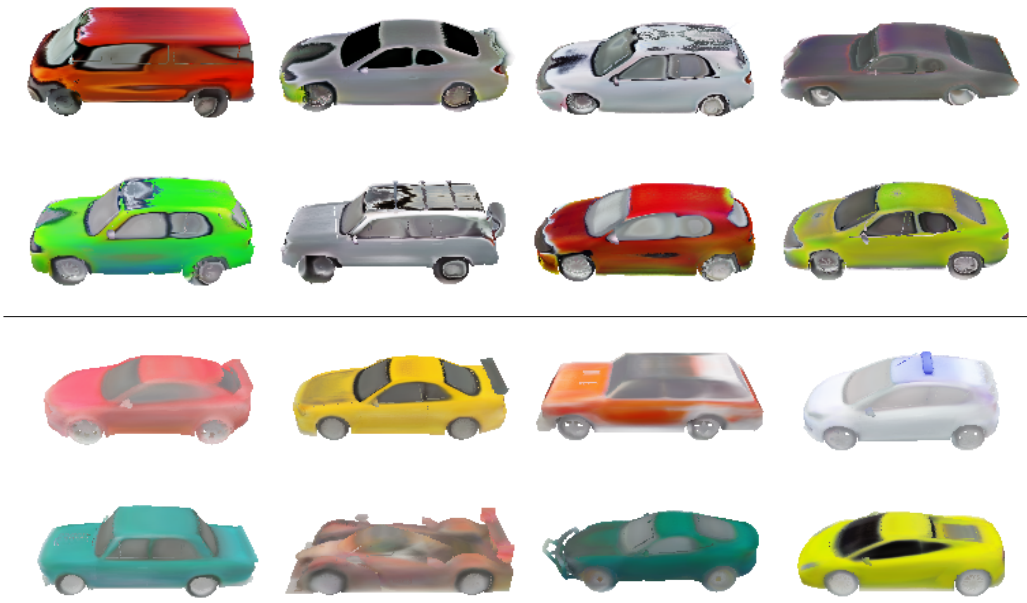


Figure 12.3: Generative Texture Model. *Textures generated using the GAN (top 2 rows) and VAE (bottom 2 rows) models.*

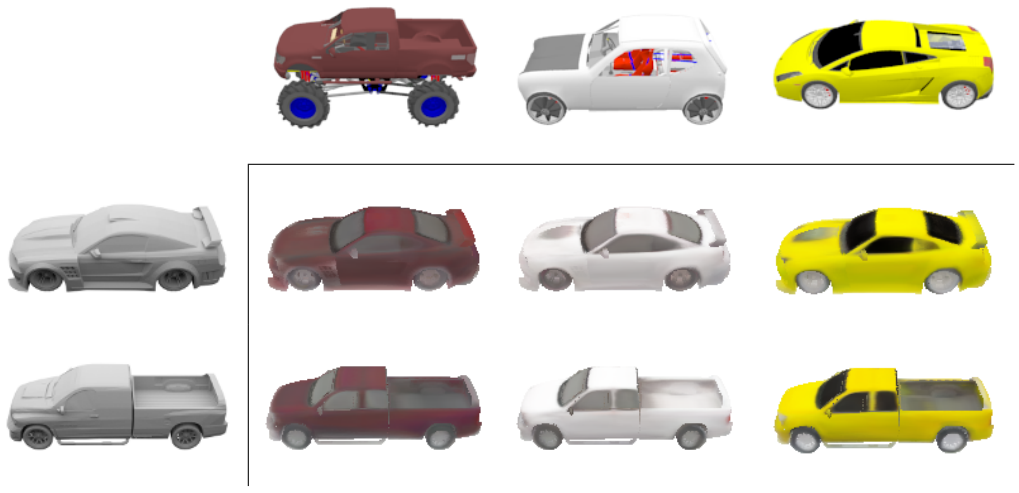
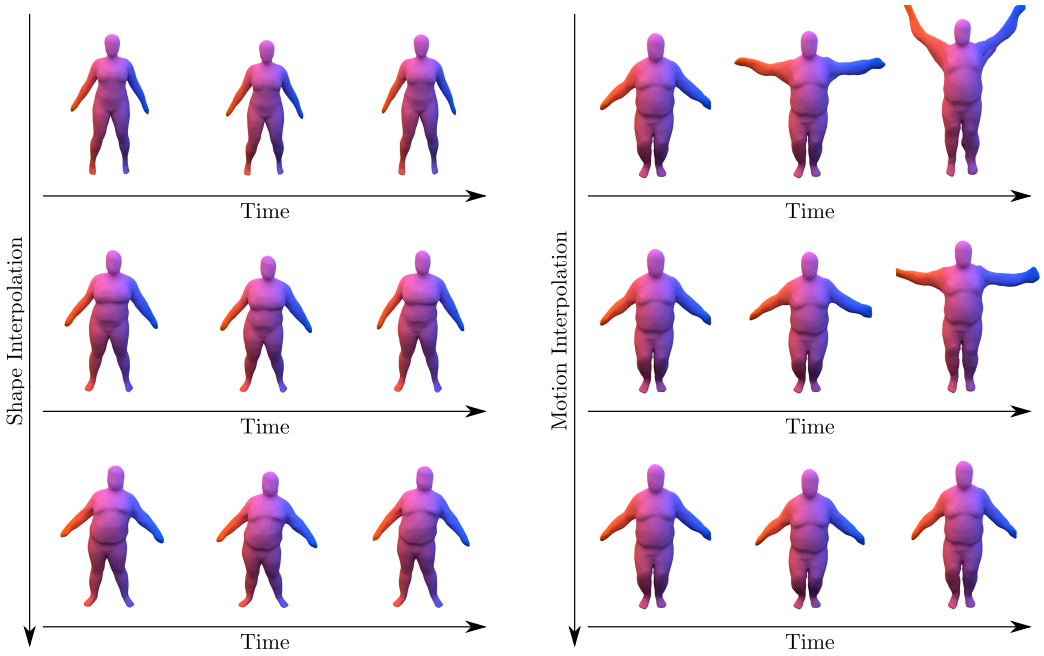


Figure 12.4: Texture Transfer. *Our VAE-based generative model transfers appearance information (top) to the target models (left).*



(a) Shape Interpolation *(b) Motion Interpolation*

Figure 12.5: Generative Motion Model. *We see that Occupancy Flow is able to learn a meaningful latent representation of both shape and motion.*

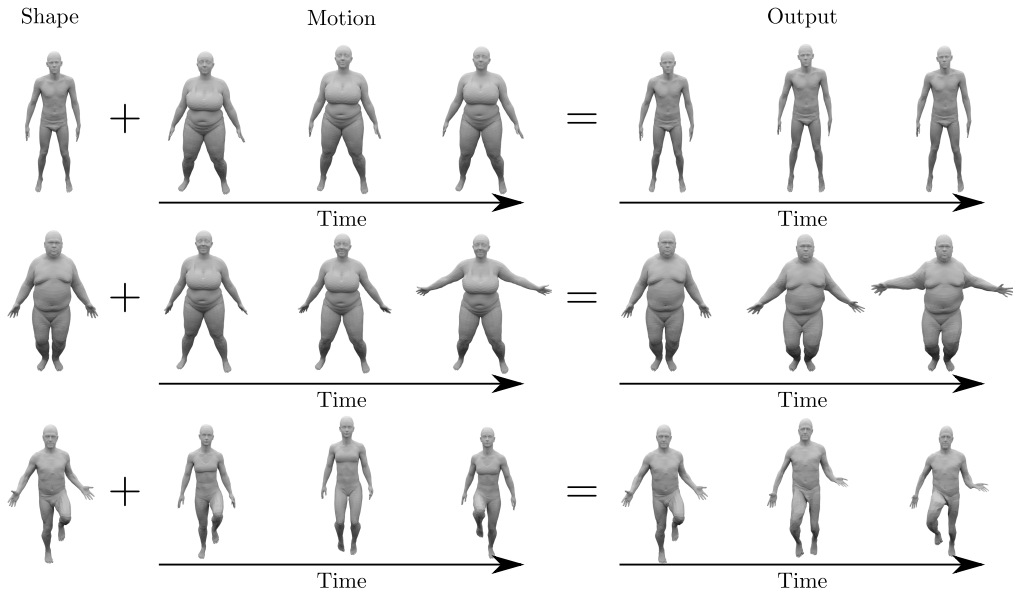


Figure 12.6: Motion Transfer. *We take a start shape (first column) and encode the motion from another sequence (second column) to transfer this motion to the shape (third column).*

13 Limitations and further Developments

As we have seen, the Function Space Operator is a useful tool to enable deep learning for high-dimensional output spaces. Consequently, the 3D deep learning community has quickly adopted the concept and proposed different models that are based on this idea [8, 25, 57, 74, 97, 130, 132, 141, 145, 150, 172, 174, 196]. However, deep learning in function space does not solve all problems and can also result in new challenges.

The first challenge is to find efficient neural network architectures for implicit representations: while voxel representations allow for highly specialized architectures, such as 3D convolutions [26, 123, 157, 176], finding these architectures is much harder for implicit representations. Instead, in implicit representations we predict the quantity of interest (e.g. occupancy) solely based on a single $s \in \mathcal{S}$ (e.g. 3D location). One possible approach for designing better architectures is to think about the information flow from the input in $\mathcal{Y}$: for example, it is possible to leverage camera information [172, 196] for single-image-3D reconstruction to condition the network on local instead of global features. However, other approaches are conceivable: for example, it might be possible to design neural network architectures that directly operate on the implicit representation, similarly to 3D Convolutional Neural Networks.

The second challenge is to find efficient inference algorithms for implicit representations. In contrast to neural networks that directly output point clouds, meshes or voxels, implicit representations require a postprocessing step to extract geometry. In Chapter 11 we proposed a simple inference algorithm based on building an OctTree [130]. Moreover, in Chapter 12 we briefly discussed Occupancy Flow [141], an extension of Occupancy Networks for 4D reconstruction that only has to extract a mesh once during inference. However, mesh extraction still requires up to 3s and is therefore not fast enough for real-time applications. In addition to these mesh extraction algorithms, other inference algorithms are possible: for instance, Park et al. [150] skip the mesh extraction step altogether and use volumetric rendering to directly obtain renderings from their implicit representation.

Finally, current implicit 3D representations still have to be supervised from ground truth 3D data. Unfortunately, this data is usually hard or expensive to acquire and it would be desirable to design methods that remove this restriction. One approach is to design a differentiable renderer to enable training from RGB images. Existing work designs differentiable renderers either for meshes [23, 56, 91, 112, 115, 212] or voxel representations [114, 139, 152, 188]. However, there is little work that investigates differentiable rendering for implicit representations. Only recently[1] Liu et al. [113] and Niemeyer et al. [142] proposed to learn implicit representations from RGB and RGB-D images However, both Liu et al. [113] and Niemeyer et al. [142] still require silhouette and camera information.

[1] The author of this thesis contributed to the latter paper [142].

Conclusion

In this thesis we have revisited generative models from two perspectives.

First, we have analyzed the stability of Generative Adversarial Networks (GANs). We have derived a theory for local convergence of the GAN training dynamics and used it to derive regularizers for GAN training. Our results can be summarized as follows: we found that (i) GAN training suffers from eigenvalues of the Jacobian close to the imaginary axis, (ii) naive GAN optimization is not always convergent, (iii) zero-centered gradient penalties can stabilize training and (iv) our regularizers enable us to train high-resolution GANs without the need for multiresolution training. While we did not answer all theoretical and practical questions about GAN training, we hope that our research inspires more theoretical work into the training dynamics of GANs and related algorithms. Moreover, we believe that our results are useful to derive novel algorithms for GAN training.

In the second part of the thesis, we have analyzed the expressiveness of generative and certain discriminative models. Here, we focused on the problem of high-dimensional output spaces, which are especially problematic for generative models. To tackle the emerging issues, we proposed the Function Space Operator, a mechanism that is applicable to a large variety of domains whose output space can be described as a function space. Our experiments show that this simple technique enables an approach to 3D generative and discriminative deep learning that does not require any discretization during training, called Occupancy Networks. We experimentally found that (i) Occupancy Networks are able to represent high-dimensional 3D geometry, (ii) implicit representations can be efficiently learned from data such as 2D images and (iii) can be used to learn a generative model in 3D. Moreover, we briefly described two extensions that apply the Function Space Operator to 4D generative modeling as well as a generative model of texture.

All in all, we found that both theoretical and experimental tools are useful to scale up generative models to complex high-dimensional spaces. Indeed, recent state-of-the-art approaches for generative modeling [89] have already adopted some of the ideas presented in this thesis. While there is much more work to be done, we believe that the research presented in this thesis will help develop the next generation of deep generative models.

Appendix

A Linear Algebra

In this chapter we first summarize two well-known results about the determinant of matrices. The first result is a lemma about the determinant of block matrices. The second result, also known as *Matrix Determinant Lemma*, gives a formula for the determinant of the sum of an invertible matrix and the matrix product of two low-rank matrices. For completeness, we also provide short proofs of these results. Afterwards, we derive a lemma about the eigenvalues of certain matrices that is useful when considering the effect of preconditioners and learning rates in GAN training (see Chapter 7). Finally, we state a lemma about the operator norm of a matrix which will be useful in Appendix B.

Lemma A.1 (Determinant of Block Matrices)**.** *Let*

$$M = \begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix} \in \mathbb{C}^{(n+m)\times(n+m)} \tag{A.1}$$

with $M_{11} \in \mathbb{C}^{n\times n}$, $M_{12} \in \mathbb{C}^{n\times m}$, $M_{21} \in \mathbb{C}^{m\times n}$ and $M_{22} \in \mathbb{C}^{m\times m}$. Moreover, assume that M_{11} is invertible. Then

$$\det(M) = \det(M_{11}) \det(M_{22} - M_{21} M_{11}^{-1} M_{12}) \tag{A.2}$$

Proof. Regard

$$\det(M) = \det \begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix} \tag{A.3}$$

Multiplying the first row with $-M_{21}M_{11}^{-1}$ from the left and adding it to the second row yields

$$\det(M) = \det \begin{pmatrix} M_{11} & M_{12} \\ 0 & M_{22} - M_{21} M_{11}^{-1} M_{12} \end{pmatrix} = \det(M_{11}) \det(M_{22} - M_{21} M_{11}^{-1} M_{12}) \tag{A.4}$$

□

Lemma A.2 (Matrix Determinant Lemma)**.** *For $A \in \mathbb{C}^{n\times n}$ invertible and $U, V \in \mathbb{C}^{n\times m}$ we have*

$$\det(A + UV^\mathsf{T}) = \det(A) \det(I + V^\mathsf{T} A^{-1} U) \tag{A.5}$$

Proof. Let

$$M := \begin{pmatrix} A & U \\ -V^\mathsf{T} & I \end{pmatrix} \tag{A.6}$$

Using Lemma A.1, we obtain

$$\det(M) = \det(I) \det(A + UV^\mathsf{T}) = \det(A + UV^\mathsf{T}) \tag{A.7}$$

On the other hand, again by Lemma A.1,

$$\det(M) = \det(A)\det(I + V^{\mathsf{T}} A^{-1} U) \tag{A.8}$$

and hence the assertion. □

The next lemma allows us to examine training algorithms for Generative Adversarial Networks (GANs) where we use different learning rates for the generator and discriminator. Moreover, this lemma can also be used to analyze the effect of preconditioners on GAN training.

Lemma A.3. *Let $H_1 \in \mathbb{R}^{n\times n}$ and $H_2 \in \mathbb{R}^{m\times m}$ denote symmetric positive definite matrices. The eigenvalues of the matrix*

$$\begin{pmatrix} H_1 J_{11} & H_1 J_{12} \\ H_2 J_{21} & H_2 J_{22} \end{pmatrix} \in \mathbb{R}^{(n+m)\times(n+m)} \tag{A.9}$$

are equal to the eigenvalues of the matrix

$$\begin{pmatrix} H_1^{1/2} J_{11} H_1^{1/2} & H_1^{1/2} J_{12} H_2^{1/2} \\ H_2^{1/2} J_{21} H_1^{1/2} & H_2^{1/2} J_{22} H_2^{1/2} \end{pmatrix} \tag{A.10}$$

Proof. Let

$$J := \begin{pmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{pmatrix} \qquad \text{and} \qquad H := \begin{pmatrix} H_1 & 0 \\ 0 & H_2 \end{pmatrix} \tag{A.11}$$

Then

$$\begin{pmatrix} H_1 J_{11} & H_1 J_{12} \\ H_2 J_{21} & H_2 J_{22} \end{pmatrix} = HJ \tag{A.12}$$

and

$$\begin{pmatrix} H_1^{1/2} J_{11} H_1^{1/2} & H_1^{1/2} J_{12} H_2^{1/2} \\ H_2^{1/2} J_{21} H_1^{1/2} & H_2^{1/2} J_{22} H_2^{1/2} \end{pmatrix} = H^{1/2} J H^{1/2} \tag{A.13}$$

Since, $HJ = H^{1/2}(H^{1/2} J H^{1/2}) H^{-1/2}$, the assertion follows. □

Corollary A.4. *Let $h_1, h_2 > 0$. The eigenvalues of the matrix*

$$\begin{pmatrix} h_1 J_{11} & h_1 J_{12} \\ h_2 J_{21} & h_2 J_{22} \end{pmatrix} \tag{A.14}$$

are equal to the eigenvalues of the matrix

$$\begin{pmatrix} h_1 J_{11} & \sqrt{h_1 h_2} J_{12} \\ \sqrt{h_1 h_2} & h_2 J_{22} \end{pmatrix} \tag{A.15}$$

Proof. This is a consequence of Lemma A.3 when we set $H_1 = h_1 I$ and $H_2 = h_2 I$. □

Recall that the *operator norm* of a matrix $M \in \mathbb{C}^{n\times n}$ is defined by

$$\|M\| := \sup_{\substack{w\in\mathbb{C}^n \\ \|w\|=1}} \|Mw\| \tag{A.16}$$

Furthermore, for an invertible matrix $Q \in \mathbb{C}^{n\times n}$ and $w \in \mathbb{C}^n$, we define

$$\|w\|_Q := \|Q^{-1}w\| \quad \text{and} \quad \|M\|_Q := \|Q^{-1}MQ\| \tag{A.17}$$

A simple calculation shows that

$$\|Mw\|_Q = \|Q^{-1}MQQ^{-1}w\| \le \|Q^{-1}MQ\|\|Q^{-1}w\| = \|M\|_Q\|w\|_Q \tag{A.18}$$

We have[1]

Lemma A.5. *Let $M \in \mathbb{C}^{n\times n}$. Assume that λ_{max} is the eigenvalue of M with the biggest absolute value. For every $\varepsilon > 0$ there is an invertible matrix $Q \in \mathbb{C}^{n\times n}$, such that*

$$\|M\|_Q < |\lambda_{max}| + \varepsilon \tag{A.19}$$

Proof. First assume that all eigenvalues of M are distinct. Then there is $Q \in \mathbb{C}^{n\times n}$ such that $Q^{-1}MQ$ is a diagonal matrix and the diagonal entries of $Q^{-1}MQ$ are the eigenvalues of M. This shows that for every $w \in \mathbb{C}^n$ with $\|w\| = 1$

$$\|Q^{-1}MQw\|^2 = \sum_{i=1}^{n} |\lambda_i w_i|^2 \le |\lambda_{\max}|^2 \sum_{i=1}^{n} |w_i|^2 = |\lambda_{\max}|^2 \tag{A.20}$$

where λ_i denote the eigenvalues of M. This shows that $\|M\|_Q \le |\lambda_{\max}|$.

Consider now the case where the eigenvalues of M may not be distinct. While we cannot diagonalize M, we can find an invertible $P = (p_1, \ldots, p_n) \in \mathbb{C}^{n\times n}$ such that $P^{-1}MP$ is lower triangular and the diagonal entries of $P^{-1}MP$ are the eigenvalues of M. Then,

$$M \cdot p_i = \lambda_i p_i + \sum_{j>i} \beta_{j,i} p_j \tag{A.21}$$

for some $\beta_{j,i} \in \mathbb{C}$. We now set $q_i = \varepsilon^{-i} p_i$ for $\varepsilon \in (0,1)$ and $Q = (q_1, \ldots, q_n) \in \mathbb{C}^{n\times n}$. We have

$$M \cdot q_i = \lambda_i q_i + \sum_{j>i} \varepsilon^{j-i} \beta_{j,i} q_j \tag{A.22}$$

Hence, for $w \in \mathbb{C}^n$, $\|w\| = 1$, using the Cauchy-Schwarz inequality,

$$\|Q^{-1}MQw\|^2 = \sum_{i=1}^{n} |\lambda_i w_i + \sum_{j<i} \varepsilon^{i-j} \beta_{i,j} w_j|^2 \le |\lambda_{max}| + \varepsilon \sum_{i,j} |\beta_{i,j}|^2 \tag{A.23}$$

For $\varepsilon > 0$ small enough this yields the assertion. □

[1] See also Bertsekas [12], Proposition A 15.

B Discrete Dynamical Systems

In this chapter we derive some basic results from the theory of discrete nonlinear dynamical systems. For a similar description of the theory of continuous nonlinear dynamical systems see for example Khalil [94] and Nagarajan and Kolter [136].

B.1 Deterministic Operators

In this section we consider continuously differentiable operators $F : \Omega \to \Omega$ acting on an open set $\Omega \subset \mathbb{R}^n$. A fixed point of F is a point $\omega^* \in \Omega$ such that $F(\omega^*) = \omega^*$. We are interested in stability and convergence of the fixed point iteration $F^{(k)}(\omega)$ near the fixed point. To this end, we first have to define what we mean by stability and local convergence.

Definition B.1. *Let $\omega^* \in \Omega$ be a fixed point of a continuously differentiable operator $F : \Omega \to \Omega$. We call ω^**

- stable *if for every $\varepsilon > 0$ there is $\delta > 0$ such that $\|\omega - \omega^*\| < \delta$ implies $\|F^{(k)}(\omega) - \omega^*\| < \varepsilon$ for all $k \in \mathbb{N}$.*
- asymptotically stable *if it is stable and there is $\delta > 0$ such that $\|\omega - \omega^*\| < \delta$ implies that $F^{(k)}(\omega)$ converges to ω^**
- exponentially stable *if there is $\alpha \in [0,1)$, $\delta > 0$ and $C > 0$ such that $\|\omega - \omega^*\| < \delta$ implies*

$$\|F^{(k)}(\omega) - \omega^*\| < C\|\omega - \omega^*\|\alpha^k \tag{B.1}$$

for all $k \in \mathbb{N}$.

If ω^* is asymptotically stable fixed point of F, we call the algorithm obtained by iteratively applying F *locally convergent* to ω^*. If ω^* is exponentially stable, we call the corresponding algorithm *linearly convergent*. Moreover, if ω^* is exponentially stable, we call the infimum of all α so that (B.1) holds for some $C > 0$ the *convergence rate* of the fixed point iteration.

As it turns out, local convergence of fixed point iterations can be analyzed by examining the spectrum of the Jacobian of the fixed point operator. We have the following central Theorem:[1]

Theorem B.2. *Let $F : \Omega \to \Omega$ denote a $\mathcal{C}^1$-mapping on an open subset Ω of $\mathbb{R}^n$ and $\omega^* \in \Omega$ be a fixed point of F. Assume that the absolute values of the eigenvalues of the Jacobian $F'(\omega^*)$ are all smaller than one. Then the fixed point iteration $F^{(k)}(\omega)$ is locally convergent to ω^*. Moreover, the rate of convergence is at least linear with convergence rate $|\lambda_{max}|$ where λ_{max} denotes the eigenvalue of $F'(\omega^*)$ with the largest absolute value.*

[1] See also Bertsekas [12], Proposition 4.4.1

Proof. Let $J := F'(\omega^*)$ and $Q \in \mathbb{C}^{n\times n}$ denote an invertible matrix. Moreover, let $\|\cdot\|_Q$ be defined as in (A.17) and let $\varepsilon > 0$. By the definition of the Jacobian $F'(\omega^*)$ there is $\delta > 0$ such that for $\|\omega - \omega^*\|_Q < \delta$ we have

$$\|F(\omega) - \omega^*\|_Q \leq \|J(\omega - \omega^*)\|_Q + \varepsilon\|\omega - \omega^*\|_Q \tag{B.2}$$

By Lemma A.5 we can choose Q such that

$$\|F(\omega) - \omega^*\|_Q \leq (\lambda_{\max} + 2\varepsilon)\|\omega - \omega^*\|_Q \tag{B.3}$$

Since $\varepsilon > 0$ was arbitrary, we can choose ε such that $\lambda_{\max} + 2\varepsilon < 1$. This implies $\|F(\omega) - \omega^*\|_Q < \delta$ and we therefore recursively obtain

$$\|F^{(k)}(\omega) - \omega^*\|_Q \leq (\lambda_{\max} + 2\varepsilon)^k\|\omega - \omega^*\|_Q \tag{B.4}$$

Since $\|\cdot\|_Q$ and $\|\cdot\|$ are equivalent norms on $\mathbb{R}^n$, there is $C > 0$ and $\tilde{\delta}$ such that

$$\|F^{(k)}(\omega) - \omega^*\| \leq C(\lambda_{\max} + 2\varepsilon)^k\|\omega - \omega^*\| \tag{B.5}$$

for $\|\omega - \omega^*\| < \tilde{\delta}$, showing that ω^* is exponentially stable. □

As it turns out, the converse of Theorem B.2 is also true, showing that the condition in Theorem B.2 is both necessary and sufficient for ω^* being an exponentially stable fixed point of F.

Lemma B.3. *If ω^* is an exponentially stable fixed point of the $\mathcal{C}^1$-mapping $F : \Omega \to \Omega$, then all eigenvalues of $F'(\omega^*)$ have absolute value smaller than one.*

Proof. Let $J := F'(\omega^*)$. First note that the mapping $\omega \to F^{(k)}(\omega)$ obtained by applying F k-times to ω is differentiable and has a fixed point at ω^*. Moreover, by the multivariate chain rule, the Jacobian of $F^{(k)}$ at ω^* is J^k.

This means that there is a function g_k with

$$F^{(k)}(\omega) = \omega^* + J^k(\omega - \omega^*) + g_k(\omega - \omega^*) \tag{B.6}$$

and

$$\lim_{\omega\to\omega^*} \frac{\|g_k(\omega - \omega^*)\|}{\|\omega - \omega^*\|} = 0 \tag{B.7}$$

For $t > 0$, consider $\omega = \omega^* + tw$. Together with (B.6) this yields

$$t\|J^k w\| \leq \|F^{(k)}(\omega^* + tw) - \omega^*\| + \|g_k(t\,w)\| \tag{B.8}$$

Since ω^* is an exponentially stable fixed point of F, there is $C > 0$ and $\alpha < 1$ such that for t small enough

$$t\|J^k w\| \leq tC\|w\|\alpha^k + \|g_k(t\,w)\| \tag{B.9}$$

Dividing by t on both sides yields

$$\|J^k w\| \leq C\|w\|\alpha^k + \frac{1}{t}\|g_k(t\,w)\| \tag{B.10}$$

Since this holds for all $t > 0$ small enough, we can consider the case $t \to 0$. In this case the remainder $\frac{1}{t}\|g_k(tw)\|$ vanishes, yielding $\|J^k w\| \leq C\|w\|\alpha^k$ and therefore $\lim_{k\to\infty}\|J^k w\| = 0$ for all $w \in \mathbb{R}^n$. However, this can only hold for arbitrary $w \in \mathbb{R}^n$ if all eigenvalues of J have absolute value smaller than one. □

For deep neural networks, we usually do not have a single fixed point but a submanifold of equivalent fixed points. We therefore need a generalization of Theorem B.2 that takes submanifolds $\mathcal{M}$ of fixed points into account. In the following, let $\mathrm{T}_{\omega^*}\mathcal{M}$ denote the tangent space of $\mathcal{M}$ at ω^* and $(\mathrm{T}_{\omega^*}\mathcal{M})^\perp$ its orthogonal complement.

We now prove the following generalization of Theorem B.2:

Theorem B.4. *Assume that $F : \Omega \to \Omega \subseteq \mathbb{R}^n$ is a $\mathcal{C}^1$-mapping and let $\mathcal{M} \subseteq \Omega$ denote an $(n-l)$-dimensional $\mathcal{C}^1$-submanifold of Ω so that $F(\omega) = \omega$ for all $\omega \in \mathcal{M}$. Let $\omega^* \in \mathcal{M}$ and assume that $B = (b_1, \ldots, b_l) \in \mathbb{R}^{n\times l}$ is a basis of $(\mathrm{T}_{\omega^*}\mathcal{M})^\perp$. Let $J := B^\mathsf{T} F'(\omega^*)B$. Then the fixed point iteration defined by F is locally convergent to $\mathcal{M}$ with linear convergence rate in a neighborhood of ω^*. Moreover, the convergence rate is $|\lambda_{\max}|$ with $\lambda_{\max}$ the eigenvalue of J with largest absolute value.*

In the remainder of this section, we prove Theorem B.4. To this end, we first reduce Theorem B.4 to the case where $\mathcal{M} = \{0\}^l \times \mathbb{R}^{n-l}$:

Lemma B.5. *Let $\mathcal{M} \subseteq \Omega$ denote an $(n-l)$-dimensional $\mathcal{C}^1$-submanifold of $\mathbb{R}^n$ and let $\omega^* \in \mathcal{M}$. Assume that $B = (b_1, \ldots, b_l) \in \mathbb{R}^{n\times l}$ and $\tilde{B} = (\tilde{b}_1, \ldots, \tilde{b}_{n-l}) \in \mathbb{R}^{n\times(n-l)}$ are orthogonal bases of $(\mathrm{T}_{\omega^*}\mathcal{M})^\perp$ and $\mathrm{T}_{\omega^*}\mathcal{M}$, respectively. Then there is a neighborhood $U \subseteq \Omega$ of ω^*, a neighborhood $V \subseteq \mathbb{R}^n$ of 0 and a $\mathcal{C}^1$-diffeomorphism $\phi : U \to V$ such that*

- $\phi(U \cap \mathcal{M}) = V \cap (\{0\}^l \times \mathbb{R}^{n-l})$
- $\phi(\omega^*) = 0$
- $\phi'(\omega^*) = (B, \tilde{B})^\mathsf{T} \in \mathbb{R}^{n\times n}$

Proof. Since $\mathcal{M} \subseteq \Omega$ is an $(n-l)$-dimensional $\mathcal{C}^1$-submanifold of $\mathbb{R}^n$, there is (by definition) a neighborhood U_0 of ω^*, a neighborhood V_0 of 0 in $\mathbb{R}^n$ and a $\mathcal{C}^1$-diffeomorphism $\phi_0 : U_0 \to V_0$ such that $\phi_0(U_0 \cap \mathcal{M}) = V_0 \cap (\{0\}^l \times \mathbb{R}^{n-l})$ and $\phi_0(\omega^*) = 0$. We therefore only have to show that we can modify ϕ_0 into ϕ so that additionally $\phi'(\omega^*) = (B, \tilde{B})^\mathsf{T} \in \mathbb{R}^{n\times n}$.

To this end, define

$$Q := (B, \tilde{B})^\mathsf{T} \cdot \left(\phi_0'(\omega^*)\right)^{-1} \in \mathbb{R}^{n\times n} \tag{B.11}$$

and let $U := U_0$ and $V := Q(V_0)$. Moreover, define $\phi : U \to V$ by

$$\phi(\omega) := Q \cdot \phi_0(\omega) \tag{B.12}$$

Since Q is an invertible matrix, ϕ is a $\mathcal{C}^1$-diffeomorphism and we have $\phi(\omega^*) = 0$. Moreover,

$$\begin{aligned} Q(\{0\}^l \times \mathbb{R}^{n-l}) &= (B, \tilde{B})^\mathsf{T} \cdot \left(\phi_0'(\omega^*)\right)^{-1} (\{0\}^l \times \mathbb{R}^{n-l}) \\ &= (B, \tilde{B})^\mathsf{T} (\mathrm{T}_{\omega^*} \mathcal{M}) \\ &= \{0\}^l \times \mathbb{R}^{n-l} \end{aligned} \tag{B.13}$$

and therefore

$$\begin{aligned} \phi(U \cap \mathcal{M}) &= Q(V_0 \cap (\{0\}^l \times \mathbb{R}^{n-l})) \\ &= V \cap Q(\{0\}^l \times \mathbb{R}^{n-l}) \\ &= V \cap (\{0\}^l \times \mathbb{R}^{n-l}) \end{aligned} \tag{B.14}$$

Finally, we have

$$\phi'(\omega^*) = (B, \tilde{B})^\mathsf{T} \cdot \left(\phi_0'((\omega^*)\right)^{-1} \phi_0'(\omega^*) = (B, \tilde{B})^\mathsf{T} \tag{B.15}$$

showing the assertion. □

The next lemma is a discrete version of Theorem A.4 from Nagarajan and Kolter [136] and we prove it in a similar way.

Lemma B.6. *Let $F : \Omega_1 \to \Omega_2$ define a $\mathcal{C}^1$-mapping that maps an open neighborhood Ω_1 of 0 to another open neighborhood Ω_2 of 0. Assume that $F(0, \omega_2) = (0, \omega_2)$ for all $(0, \omega_2) \in \Omega_1$ where $\omega_2 \in \mathbb{R}^{n-l}$. Moreover, assume that all eigenvalues of*

$$J := \frac{\partial}{\partial \omega_1} F(\omega_1, 0)\bigg|_{\omega_1 = 0} \in \mathbb{R}^{l \times l} \tag{B.16}$$

have absolute value smaller than one. Then there is an open neighborhood $U_1 \subseteq \Omega_1$ so that for all $\omega \in U_1$ we have $F^{(k)}(\omega) \in \Omega_1 \cap \Omega_2$ and $F^{(k)}(\omega)$ is locally convergent to $\mathcal{M} := \{(0, \omega_2) \in \Omega_2 \mid \omega_2 \in \mathbb{R}^{n-l}\}$ with linear convergence rate . Moreover, the convergence rate is $|\lambda_{\max}|$ with $\lambda_{\max}$ the eigenvalue of J with largest absolute value.

Proof. In the following, we write $F(\omega_1, \omega_2) = (F_1(\omega_1, \omega_2), F_2(\omega_1, \omega_2))$, so that the fixed point iteration can be written as

$$\omega_1^{(k+1)} = F_1(\omega_1^{(k)}, \omega_2^{(k)}) \tag{B.17}$$

$$\omega_2^{(k+1)} = F_2(\omega_1^{(k)}, \omega_2^{(k)}). \tag{B.18}$$

To prove the assertion, it suffices to show that there are open neighborhoods $U_1 \subseteq \Omega_1$ and $U_2 \subseteq \Omega_1 \cap \Omega_2$ of 0 such that for $\omega^{(0)} \in U_1$ we have $\omega^{(k)} \in U_2$ for all $k \in \mathbb{N}$ and

$$\|\omega_1^{(k)}\| \leq C \|\omega_1^{(0)}\| \alpha^k \tag{B.19}$$

for some $\alpha \in [0, 1)$.

We first examine the behavior of F_1 near $(0,0)$. To this end, we develop F_1 into a Taylor-series

$$F_1(\omega_1,\omega_2) = J\omega_1 + g_1(\omega_1,\omega_2) \tag{B.20}$$

For $Q \in \mathbb{C}^{l\times l}$, let the norm $\|\cdot\|_Q$ on $\mathbb{R}^l$ be defined as in (A.17). We first show that for any $c>0$ we have $\|g_1(\omega_1,\omega_2)\|_Q \le c\|\omega_1\|_Q$ sufficiently close to $(0,0)$: because $F_1(0,\omega_2)=0$ for all ω_2 close to 0, $g_1(\omega_1,\omega_2)$ must be of the form $g_1(\omega_1,\omega_2)=h_1(\omega_1,\omega_2)\omega_1$ with $h_1(0,0)=0$. This shows that for every $c>0$ there is $\delta>0$ such that for $\|\omega_1\|_Q<\delta$ and $\|\omega_2\|<\delta$ we have $\|g_1(\omega_1,\omega_2)\|_Q \le c\|\omega_1\|_Q$.

Let $\varepsilon>0$. According to Lemma A.5 we can select $Q\in\mathbb{C}^{l\times l}$ such that

$$\|J\omega_1\|_Q < (|\lambda_{\max}|+\varepsilon)\|\omega_1\|_Q \tag{B.21}$$

for $\omega_1\in\mathbb{R}^l$ where $|\lambda_{\max}|$ denotes the eigenvalue of J with the largest absolute value.

Hence, for $\|\omega_1\|_Q<\delta$ and $\|\omega_2\|<\delta$,

$$\|F_1(\omega_1,\omega_2)\|_Q \le \|J\omega_1\|_Q + \|g_1(\omega_1,\omega_2)\|_Q < (|\lambda_{\max}|+\varepsilon+c)\|\omega_1\|_Q \tag{B.22}$$

Because we can make $c+\varepsilon$ as small as we want, this shows that

$$\|\omega_1^{(k)}\|_Q \le \alpha^k \|\omega_1^{(0)}\|_Q < \delta \tag{B.23}$$

for some $\alpha\in[0,1)$, if $\|\omega_1^{(0)}\|_Q<\delta$ and $\|\omega_2^{(l)}\|<\delta$ for all $l=0,\dots,k-1$ with $\delta>0$ sufficiently small. We therefore have to show that the iterates $\omega_2^{(k)}$ stay in a neighborhood of 0, i.e. $\|\omega_2^{(k)}\|<\delta$, when the starting iterates $\omega_1^{(0)}$ and $\omega_2^{(0)}$ are initialized sufficiently close to 0.

To show this, we develop F_2 into a Taylor-series around 0:

$$F_2(\omega_1,\omega_2) = \omega_2 + g_2(\omega_1,\omega_2). \tag{B.24}$$

Again, we see that g_2 must be of the form $g_2(\omega_1,\omega_2)=h_2(\omega_1,\omega_2)\omega_1$, showing that $\|g_2(\omega_1,\omega_2)\|\le c'\|\omega_1\|_Q$ for some fixed constant $c'>0$ if $\|\omega_1\|_Q<\delta$ and $\|\omega_2\|<\delta$ (note that in general $h_2(0,0)\neq 0$). We therefore have

$$\begin{aligned}
\|\omega_2^{(k)}-\omega_2^{(0)}\| &\le \sum_{l=0}^{k-1}\|g_2(\omega_1^{(l)},\omega_2^{(l)})\| \\
&\le \sum_{l=0}^{k-1} c'\|\omega_1^{(l)}\|_Q \\
&\le \sum_{l=0}^{k-1} c'\alpha^l\|\omega_1^{(0)}\|_Q \\
&\le \frac{c'}{1-\alpha}\|\omega_1^{(0)}\|_Q
\end{aligned} \tag{B.25}$$

Hence, if we initialize $\omega^{(0)}$ so that $\|\omega_1^{(0)}\|_Q < \min(1,\frac{1-\alpha}{2c'})\delta$ and $\|\omega_2^{(0)}\|<\frac{\delta}{2}$, we have

$\|\omega_1^{(k)}\|_Q < \delta$ and $\|\omega_2^{(k)}\| < \delta$ for all $k \in \mathbb{N}$. Moreover, we further have by (B.23) and because $\|\cdot\|$ and $\|\cdot\|_Q$ define equivalent norms on $\mathbb{R}^l$

$$\|\omega_1^{(k)}\| \leq C\alpha^k \|\omega_1^{(0)}\| \tag{B.26}$$

for some $C > 0$ and all $k \in \mathbb{N}$. This concludes the proof. □

We are now ready to prove Theorem B.4:

Theorem B.4. Let $\phi : U \to V$ be defined as in Lemma B.5 and $F_\phi : V \cap \phi(F^{-1}(U)) \to V$ be defined by $F_\phi := \phi \circ F \circ \phi^{-1}$. It suffices to show that F_ϕ satisfies the condition of Lemma B.6.

First of all, for $\omega_2 \in \mathbb{R}^{n-l}$ we have $\phi^{-1}(0,\omega_2) \in \mathcal{M}$ if $(0,\omega_2) \in V$. Since $F(\omega) = \omega$ for all $\omega \in \mathcal{M}$, this shows that

$$F_\phi(0,\omega_2) = \phi\left(F\left(\phi^{-1}(0,\omega_2)\right)\right) = \phi\left(\phi^{-1}(0,\omega_2)\right) = (0,\omega_2) \tag{B.27}$$

Moreover, since $\phi'(\omega^*) = (B,\tilde{B})^\mathsf{T}$,

$$\begin{aligned} F'_\phi(0,0) &= \phi'(\omega^*) \cdot F'(\omega^*) \cdot (\phi'(\omega^*))^{-1} \\ &= \begin{pmatrix} B^\mathsf{T} \\ \tilde{B}^\mathsf{T} \end{pmatrix} F'(\omega^*) \begin{pmatrix} B & \tilde{B} \end{pmatrix} \\ &= \begin{pmatrix} B^\mathsf{T}F'(\omega^*)B & B^\mathsf{T}F'(\omega^*)\tilde{B} \\ \tilde{B}^\mathsf{T}F'(\omega^*)B & \tilde{B}^\mathsf{T}F'(\omega^*)\tilde{B} \end{pmatrix} \end{aligned} \tag{B.28}$$

Hence,

$$\frac{\partial}{\partial \omega_1} F_\phi(\omega_1,0)\bigg|_{\omega_1=0} = B^\mathsf{T}F'(\omega^*)B = J \tag{B.29}$$

This shows that F_ϕ satisfies the assumption of Lemma B.6. Because for ω sufficiently close to ω^*

$$F^{(k)} = \phi^{-1} \circ F_\phi^{(k)} \circ \phi \tag{B.30}$$

this yields the assertion. □

B.2 Stochastic Operators

In this thesis we focus on deterministic dynamical systems. However, many results from the theory of deterministic dynamical systems can be extended to stochastic systems with uniformly bounded noise in a straightforward way. Importantly, to achieve robustness under noise, we usually require linear convergence in the deterministic case. In this section we show that systems that are linearly convergent to a fixed point in the deterministic case are convergent in the stochastic case when the system is appropriately annealed (e.g. by annealing the learning rate) and the noise is uniformly bounded. For more information please see e.g. Spall [178] and the seminal work by Robbins and Monro [167].

Assume that $F_{\mathcal{B}}(\omega)$ is a stochastic operator depending the (random) batch $\mathcal{B}$. Let $F(\omega) := \mathrm{E}_{\mathcal{B}}[F_{\mathcal{B}}(\omega)]$. Consider the iteration

$$\omega^{(k+1)} = F_{\mathcal{B}}(\omega^{(k)}) = F(\omega^{(k)}) + \varepsilon^{(k)} \tag{B.31}$$

with $\varepsilon^{(k)} := F_{\mathcal{B}}(\omega^{(k)}) - F(\omega^{(k)})$. Note that by definition $\mathrm{E}_{\mathcal{B}}\left[\varepsilon^{(k)}\right] = 0$. Moreover, we have

$$F'(\omega) = \mathrm{E}_{\mathcal{B}}\left[F'_{\mathcal{B}}(\omega)\right] \tag{B.32}$$

For simplicity we assume that ω^* is an isolated fixed points of $F : \Omega \to \Omega$ in this section.[2]

In the following we assume that $\|\cdot\|_Q$ is a norm as in (A.17) so that there is $\delta > 0$ and $0 \leq \alpha < 1$ with

$$\|F(\omega) - \omega^*\|_Q \leq \alpha \|\omega - \omega^*\|_Q \tag{B.33}$$

for all $\|\omega - \omega^*\|_Q < \delta$. Note that by Lemma A.5 such a norm $\|\cdot\|_Q$ always exists if all eigenvalues of $F'(\omega^*)$ have absolute value smaller than one (cf. the proof of Theorem B.2).

We now come to our first stability result of stochastic dynamical systems near fixed points.

Lemma B.7. *Assume that ω^* is a fixed point of F and there is $Q \in \mathbb{C}^{n\times n}$ as well as $\delta > 0$ such that*

$$\|F(\omega) - \omega^*\|_Q \leq \alpha \|\omega - \omega^*\|_Q \tag{B.34}$$

for $\|\omega - \omega^*\|_Q < \delta$. Moreover, assume that $\|\varepsilon^{(k)}\| < (1-\alpha)\delta$ for all $k \in \mathbb{N}$ and $\|\omega^{(0)} - \omega^*\|_Q < \delta$. We then have $\|\omega^{(k)} - \omega^*\|_Q < \delta$ for all $k \in \mathbb{N}$.

Proof. We prove the lemma via induction over k. By assumption $\|\omega^{(0)}\|_Q < \delta$.

Assume $\|\omega^{(k)} - \omega^*\|_Q < \delta$. Then

$$\begin{aligned} \|\omega^{(k+1)} - \omega^*\|_Q &\leq \|F(\omega^{(k)}) - \omega^*\|_Q + \|\varepsilon^{(k)}\|_Q \\ &\leq \alpha \|\omega^{(k)} - \omega^*\| + (1-\alpha)\delta \\ &\leq \alpha\delta + (1-\alpha)\delta \\ &= \delta \end{aligned} \tag{B.35}$$

Induction over k now yields the assertion. □

Note that the requirements in Lemma B.7 on $\varepsilon^{(k)}$ are very weak and $\varepsilon^{(k)}$ could even be a sequence chosen in an adversarial way. This demonstrates the strength of linear convergence on local stability.

We know consider the annealed operator

$$F_{\mathcal{B},\beta}(\omega) := \beta F_{\mathcal{B}}(\omega) + (1-\beta)\omega \tag{B.36}$$

[2] Establishing stochastic convergence results towards manifolds of equilibrium points is indeed much harder than in the deterministic case, since random movements along the tangent space of the manifold can have a nontrivial influence on the dynamics in this case. We therefore would need much stronger assumptions (e.g. compactness of $\mathcal{M}$) to establish convergence.

for some $\beta > 0$ and similarly

$$F_\beta(\omega) := \mathrm{E}_{\mathcal{B}}\left[F_{\mathcal{B},\beta}(\omega)\right] = \beta F(\omega) + (1-\beta)\omega \tag{B.37}$$

Consider the iteration

$$\omega^{(k+1)} = F_{\mathcal{B},\beta_k}(\omega^{(k)}) = F_{\beta_k}(\omega^{(k)}) + \beta_k \varepsilon^{(k)} \tag{B.38}$$

In the following, we need a lemma about the convergence of infinite sums.

Lemma B.8. *Let $a_{k,l} \in \mathbb{R}$ define a sequence of sequences such that $\lim_{l\to\infty} a_{k,l} = a_k \in \mathbb{R}$. Moreover assume that $|a_{k,l}| \le b_k$ with $\sum_{k=1}^{\infty} b_k < \infty$. Then*

$$\lim_{l\to\infty} \sum_{k=1}^{\infty} a_{k,l} = \sum_{k=1}^{\infty} a_k \tag{B.39}$$

Proof. For $\varepsilon > 0$ choose $m \in \mathbb{N}$ such that

$$\sum_{k=m+1}^{\infty} b_k < \varepsilon \tag{B.40}$$

Then

$$|\sum_{k=1}^{\infty} a_{k,l} - \sum_{k=1}^{\infty} a_k| \le \sum_{k=1}^{m} |a_{k,l} - a_k| + 2\sum_{k=m+1}^{\infty} b_k < \sum_{k=1}^{m} |a_{k,l} - a_k| + 2\varepsilon \tag{B.41}$$

Since $\lim_{l\to\infty} a_{k,l} = a_k$ we can choose $l_0 \in \mathbb{N}$ such that $|a_{k,l} - a_k| < \frac{\varepsilon}{m}$ for $l \ge l_0$ and $k = 1,\dots,m$. Then

$$|\sum_{k=1}^{\infty} a_{k,l} - \sum_{k=1}^{\infty} a_k| < 3\varepsilon \tag{B.42}$$

which shows the assertion. □

We now come to our main stochastic convergence theorem.

Theorem B.9. *Assume that ω^* is a fixed point of F and there is $Q \in \mathbb{C}^{n\times n}$ as well as $\delta > 0$ such that*

$$\|F(\omega) - \omega^*\|_Q \le \alpha \|\omega - \omega^*\|_Q \tag{B.43}$$

for $\|\omega - \omega^*\|_Q < \delta$. Assume that β_k is a sequence such that $0 \le \beta_k \le 1$ and

$$\sum_{k=0}^{\infty} \beta_k = \infty \quad \text{and} \quad \sum_{k=0}^{\infty} \beta_k^2 < \infty \tag{B.44}$$

Moreover, assume that $\|\varepsilon^{(k)}\|_Q < (1-\alpha)\delta$ for all $k \in \mathbb{N}$ and that all $\varepsilon^{(k)}$ are uncorrelated with $\mathrm{E}\left[\|\varepsilon^{(k)}\|_Q^2\right] \le \sigma^2$. Let $\omega^{(0)}$ such that $\|\omega^{(0)} - \omega^*\|_Q < \delta$. Then $\omega^{(k)}$ converges in L_2 to ω^*, i.e.

$$\lim_{k\to\infty} \mathrm{E}\ \|\omega^{(k)} - \omega^*\|_Q^2 = 0 \tag{B.45}$$

Proof. First, note that

$$\begin{aligned} \|F_{\beta_k}(\omega) - \omega^*\|_Q &= \|\beta_k(F(\omega) - \omega^*) + (1-\beta_k)(\omega - \omega^*)\|_Q \\ &\leq (\alpha\beta_k + (1-\beta_k))\|\omega - \omega^*\|_Q \\ &= \alpha_k \|\omega - \omega^*\|_Q \end{aligned} \tag{B.46}$$

with $\alpha_k := \alpha\beta_k + (1-\beta_k) = 1 - \beta_k(1-\alpha) \leq 1$.

As in the proof of Lemma B.7 we therefore obtain for $\|\omega^{(k)} - \omega^*\|_Q < \delta$

$$\begin{aligned} \|\omega^{(k+1)} - \omega^*\|_Q &\leq \|F_{\beta_k}(\omega^{(k)}) - \omega^*\|_Q + \beta_k \|\varepsilon^{(k)}\|_Q \\ &< (\alpha\beta_k + (1-\beta_k))\delta + \beta_k(1-\alpha)\delta \\ &= \delta \end{aligned} \tag{B.47}$$

Inductively, we hence see that $\|\omega^{(k)} - \omega^*\|_Q < \delta$ for all $k \in \mathbb{N}$.

Using (B.46) and the fact that $\omega^{(k)}$ and $\varepsilon^{(k)}$ are uncorrelated, we obtain

$$\begin{aligned} \mathrm{E}\left[\|\omega^{(k+1)} - \omega^*\|_Q^2\right] &= \mathrm{E}\left[\|F_{\beta_k}(\omega^{(k)}) - \omega^* + \beta_k\varepsilon^{(k)}\|_Q^2\right] \\ &\leq \alpha_k^2 \,\mathrm{E}\left[\|\omega^{(k)} - \omega^*\|_Q^2\right] + \beta_k^2\sigma^2 \end{aligned} \tag{B.48}$$

Hence, recursively,

$$\mathrm{E}\left[\|\omega^{(k+1)} - \omega^*\|_Q^2\right] \leq \sum_{l=0}^{k} \left(\prod_{m=l+1}^{k} \alpha_m\right)^2 \beta_l^2\sigma^2 + \left(\prod_{m=0}^{k} \alpha_m\right)^2 \|\omega^{(0)} - \omega^*\|_Q^2 \tag{B.49}$$

However,

$$\prod_{m=l+1}^{k} \alpha_m = \exp\left(\sum_{m=l+1}^{k} \log(1-\beta_m(1-\alpha))\right) \leq \exp\left(-\sum_{m=l+1}^{k} \beta_m(1-\alpha)\right) \tag{B.50}$$

which converges to zero as k goes to infinity by assumption. Lemma B.8 now yields the assertion, because

$$\left(\prod_{m=l+1}^{k} \alpha_m\right)^2 \beta_l^2 \leq \beta_l^2 \tag{B.51}$$

and $\sum_l \beta_l^2$ converges by assumption. □

A typical example of a sequence β_k that satisfies the assumption of Theorem B.9 is $\beta_k = \frac{\beta_0}{k+1}$. Indeed, it is possible to prove a convergence rate of $\mathcal{O}(k^{-1/2})$ in this case [178].

C Eigenvalue bounds

We will now derive the theory necessary to prove Theorem 7.1. To this end, we reformulate the problem of finding eigenvalues of J as a *quadratic eigenvalue problem* [187].

Lemma C.1. *Let*

$$J = \begin{pmatrix} 0 & -P^\mathsf{T} \\ P & -Q \end{pmatrix} \in \mathbb{R}^{(n+m)\times(n+m)} \tag{C.1}$$

with $P \in \mathbb{R}^{m\times n}$ and $Q \in \mathbb{R}^{m\times m}$. Then the characteristic polynomial of J can be written as

$$\chi(\lambda) = \lambda^{n-m} \det(\lambda^2 I + \lambda Q + PP^\mathsf{T}) \tag{C.2}$$

Proof. Using Lemma A.1 about the determinant of block matrices, we obtain

$$\begin{aligned} \det(\lambda I - J) = \det \begin{pmatrix} \lambda I & P^\mathsf{T} \\ -P & \lambda I + Q \end{pmatrix} &= \det(\lambda I) \det\left(\lambda I + Q + P(\lambda I)^{-1} P^\mathsf{T}\right) \\ &= \lambda^{n-m} \det\left(\lambda^2 I + \lambda Q + PP^\mathsf{T}\right) \end{aligned} \tag{C.3}$$

□

Lemma C.1 shows that we can analyze the eigenvalues of J by analyzing the quadratic eigenvalue problem in (C.2). To this end, the following lemma is helpful:

Lemma C.2. *Let $\lambda \in \mathbb{C}$ be a solution to the quadratic eigenvalue problem*

$$\det(\lambda^2 I + \lambda A_1 + A_2) = 0 \tag{C.4}$$

with $A_1, A_2 \in \mathbb{C}^{m\times m}$ Hermitian. Then there is $w \in \mathbb{C}^m$, $\|w\| = 1$, such that λ is of the form

$$\lambda = \frac{-w^H A_1 w}{2} \pm \sqrt{\frac{(w^H A_1 w)^2}{4} - w^H A_2 w} \tag{C.5}$$

Proof. Because of (C.4), the matrix $\lambda^2 I + A_1\lambda + A_2$ is singular. As a result, there exist $w \in \mathbb{C}^n$, $\|w\| = 1$ such that

$$\lambda^2 w + \lambda A_1 w + A_2 w = 0 \tag{C.6}$$

Multiplying with w^H from the left yields

$$\lambda^2 + w^H A_1 w\, \lambda + w^H A_2 w = 0 \tag{C.7}$$

Note that $w^H A_1 w, w^H A_2 w \in \mathbb{R}$, because both A_1 and A_2 are Hermitian by assumption. Solving this equations for λ yields the assertion. □

We are now have our tools ready to prove Theorem 7.1. To this end, we will use Lemma C.1 together with Lemma C.2 to obtain expressions of the eigenvalues for which we can compute bounds.

Theorem 7.1. *Let*

$$J = \begin{pmatrix} 0 & -P^{\mathsf{T}} \\ P & -Q \end{pmatrix} \in \mathbb{R}^{2n \times 2n} \tag{7.5}$$

with $P \in \mathbb{R}^{n \times n}$ and $Q \in \mathbb{R}^{n \times n}$. Moreover, assume that Q is symmetric positive semi-definite and let η_{min} and η_{max} denote the minimum and maximum singular value of Q. Similarly, let s_{min} and s_{max} denote the smallest and largest singular value of P. Then the following assertions hold:

- *If $\frac{1}{2}\eta_{max} < s_{min}$, all eigenvalues of J satisfy $\Im(\lambda) \neq 0$.*
- *If $s_{max} \leq \frac{1}{2}\eta_{min}$, all eigenvalues of J satisfy $\Im(\lambda) = 0$.*
- *All eigenvalues λ of J with $\Im(\lambda) \neq 0$ satisfy*

$$\frac{-\eta_{max}}{2} \leq \Re(\lambda) \leq \frac{-\eta_{min}}{2} \tag{7.6}$$

- *All eigenvalues λ of J with $\Im(\lambda) = 0$ satisfy*

$$-\frac{\eta_{max}}{2} - \sqrt{\frac{\eta_{max}^2}{4} - s_{min}^2} \leq \Re(\lambda) \leq -\frac{\eta_{max}}{2} + \sqrt{\frac{\eta_{max}^2}{4} - s_{min}^2} \tag{7.7}$$

- *All eigenvalues λ of J satisfy*

$$\sqrt{s_{min}^2 - \frac{\eta_{max}^2}{4}} \leq |\Im(\lambda)| \leq \sqrt{s_{max}^2 - \frac{\eta_{min}^2}{4}} \tag{7.8}$$

  Here, the lower bound in (7.8) *holds as long as the expression inside the square root is non-negative, i.e. when $\frac{1}{2}\eta_{max} < s_{min}$.*

- *All eigenvalue with $\Im(\lambda) \neq 0$ satisfy*

$$s_{min} \leq |\lambda| \leq s_{max} \tag{7.9}$$

Proof. By Lemma C.1, the characteristic polynomial of J is given by

$$\chi(\lambda) = \det(\lambda^2 I + \lambda Q + PP^{\mathsf{T}}) \tag{C.8}$$

Employing Lemma C.2 we further see that every non-zero eigenvalue λ of J satisfies

$$\lambda = \frac{-w^H Q w}{2} \pm \sqrt{\frac{(w^H Q w)^2}{4} - \|P^{\mathsf{T}} w\|^2} \tag{C.9}$$

for some $w \in \mathbb{C}^n$ with $\|w\| = 1$. There are two cases to consider.

In the first case $\frac{1}{4}(w^H Q w)^2 < \|P^\mathsf{T} w\|^2$. In this case $\Re(\lambda) = -\frac{1}{2} w^H Q w$ and

$$\Im(\lambda) = \pm\sqrt{\|P^\mathsf{T} w\|^2 - \frac{(w^H Q w)^2}{4}} \tag{C.10}$$

which directly shows

$$-\frac{\eta_{max}}{2} \leq \Re(\lambda) \leq -\frac{\eta_{min}}{2} \tag{C.11}$$

$$\sqrt{s_{min}^2 - \frac{\eta_{max}^2}{4}} \leq |\Im(\lambda)| \leq \sqrt{s_{max}^2 - \frac{\eta_{min}^2}{4}} \tag{C.12}$$

as long as the expressions in the square roots are non-negative. We also see that

$$|\Re(\lambda)|^2 + |\Im(\lambda)|^2 = \frac{1}{4} w^H Q w + \|P^\mathsf{T} w\|^2 - \frac{1}{4} w^H Q w = \|P^\mathsf{T} w\|^2 \tag{C.13}$$

and thus

$$s_{min}^2 \leq |\Re(\lambda)|^2 + |\Im(\lambda)|^2 \leq s_{max}^2 \tag{C.14}$$

In the second case $\frac{1}{4}(w^H Q w)^2 \geq \|P^\mathsf{T} w\|^2$. In this case $\Im(\lambda) = 0$ and $\Re(\lambda)$ is of the form

$$\Re(\lambda) = -\frac{w^H Q w}{2} \pm \sqrt{\frac{(w^H Q w)^2}{4} - \|P^\mathsf{T} w\|^2} \tag{C.15}$$

However, it is easy to see (e.g. by calculating the first derivative) that the functions $g_1(t) := -t + \sqrt{t^2 - \alpha^2}$ is increasing in t and decreasing in α for every $t \geq \alpha > 0$. Similarly, $g_2(t) := -t - \sqrt{t^2 - \alpha^2}$ is decreasing in t and increasing in α for every $t \geq \alpha > 0$.

This directly yields the bounds

$$-\frac{\eta_{max}}{2} - \sqrt{\frac{\eta_{max}^2}{4} - s_{min}^2} \leq \Re(\lambda) \leq -\frac{\eta_{max}}{2} + \sqrt{\frac{\eta_{max}^2}{4} - s_{min}^2} \tag{C.16}$$

as long as the expressions in the square roots are non-negative.

Moreover, note that the first case $\frac{1}{4}(w^H Q w)^2 < \|P^\mathsf{T} w\|^2$ cannot arise when $\frac{1}{2}\eta_{min} \geq s_{max}$. Similarly, the second case $\frac{1}{4}(w^H Q w)^2 \geq \|P^\mathsf{T} w\|^2$ cannot arise when $\frac{1}{2}\eta_{max} < s_{min}$.

□

The bounds in Theorem 7.1 are tight in the sense that no bound that only depends only on η_{min}, η_{max}, s_{min} and s_{max} can be tighter. This is a consequence of the next lemma:

Lemma C.3. *Let J be defined as in* (7.5) *with*

$$P := \begin{pmatrix} s_1 & 0 & 0 & 0 \\ 0 & s_1 & 0 & 0 \\ 0 & 0 & s_2 & 0 \\ 0 & 0 & 0 & s_2 \end{pmatrix} \qquad \text{and} \qquad Q := \begin{pmatrix} \eta_1 & 0 & 0 & 0 \\ 0 & \eta_2 & 0 & 0 \\ 0 & 0 & \eta_1 & 0 \\ 0 & 0 & 0 & \eta_2 \end{pmatrix} \tag{C.17}$$

for $\eta_1, \eta_2, s_1, s_2 \geq 0$. The eigenvalues of J are then

$$\left\{ -\frac{\eta_i}{2} \pm \sqrt{\frac{\eta_i^2}{4} - s_j^2} \;\middle|\; i,j \in \{1,2\}, \right\} \tag{C.18}$$

In particular, all applicable bounds in Theorem 7.1 are tight for J.

Proof. By Lemma C.1, the characteristic polynomial of J is

$$\chi(\lambda) = \det(\lambda^2 I + \lambda Q + PP^\mathsf{T}) = \prod_{i,j \in \{1,2\}} (\lambda^2 + \eta_i \lambda + s_j^2) \tag{C.19}$$

Setting $\chi(\lambda)$ to zero yields the assertion. □

To prove convergence rates for Alternating Gradient Descent in Chapter 7, we need the following variant of Theorem 7.1:

Lemma 7.3. *Let*

$$J = \begin{pmatrix} 0 & -P^\mathsf{T} \\ P & -Q - hPP^\mathsf{T} \end{pmatrix} \in \mathbb{R}^{2n \times 2n} \tag{7.14}$$

with $h > 0$ and $P, Q \in \mathbb{R}^{n \times n}$. Assume that Q is symmetric positive semi-definite and let η_{min} and η_{max} denote the minimum and maximum singular value of Q. Similarly, let s_{min} and s_{max} denote the minimum and maximum singular value of P. Then the following holds:

- *Assume that one of the following conditions is true:*

  i) $h \geq \frac{2}{s_{max}+s_{min}}$ *and* $\eta_{max} + hs_{max}^2 \leq 2s_{max}$

  ii) $h \leq \frac{2}{s_{max}+s_{min}}$ *and* $\eta_{max} + hs_{min}^2 \leq 2s_{min}$

  Then all eigenvalues of J have non-zero imaginary part.

- *All eigenvalues λ of J with non-zero imaginary part satisfy*

$$|1 + h\lambda|^2 \leq 1 - h\eta_{min} \tag{7.15}$$

Proof. By Lemma C.1, the characteristic polynomial of J is

$$\det(\lambda^2 I + \lambda(Q + hPP^\mathsf{T}) + PP^\mathsf{T}) \tag{C.20}$$

By Lemma C.2, the eigenvalues λ of J can be written as

$$\lambda = -\frac{w^H(Q + hPP^\mathsf{T})w}{2} \pm \sqrt{\frac{(w^H(Q + hPP^\mathsf{T})w)^2}{4} - \|P^\mathsf{T}w\|^2} \tag{C.21}$$

Similarly as in the proof of Theorem 7.1, we see that all eigenvalues of J are imaginary if

$$\eta_{max} + hs^2 \leq 2s \tag{C.22}$$

for every $s \in [s_{min}, s_{max}]$. However, maximizing $hs^2 - 2s$ over s in $s \in [s_{min}, s_{max}]$ yields $s = s_{min}$ if $h \leq \frac{2}{s_{max}+s_{min}}$ and $s = s_{max}$ else.

Moreover, in this case,

$$\Re(\mu) = -\frac{w^H(Q + hPP^\mathsf{T})w}{2} \tag{C.23}$$

$$\Im(\mu) = \pm\sqrt{\|P^\mathsf{T}w\|^2 - \frac{(w^H(Q + hPP^\mathsf{T})w)^2}{4}} \tag{C.24}$$

Hence,

$$|\mu|^2 = \Re(\mu)^2 + \Im(\mu)^2 = \|P^\mathsf{T}w\|^2 \tag{C.25}$$

and thus

$$|1 + h\mu|^2 = 1 + 2h\Re(\mu) + h^2|\mu|^2 = 1 - hw^HQw \leq 1 - h\eta_{min} \tag{C.26}$$

□

D Energy Solutions

In this chapter we briefly discuss a stable kind of equilibrium that can occur for unregularized GAN training. However, this kind of equilibrium requires a more expressive discriminator and can also be ill-conditioned. For technical reasons, we assume that supp $p_{\mathcal{D}}$ defines a $\mathcal{C}^1$-manifold in this section.

Energy Solutions are solutions where the discriminator forms a potential function for the true data distribution. Such solutions (θ^*, ψ^*) satisfy the following property:

Assumption I″. *We have $p_{\theta^*} = p_{\mathcal{D}}$, $D_{\psi^*}(x) = 0$, $\nabla_x D_{\psi^*}(x) = 0$ and $w^\mathsf{T} \nabla_x^2 D_{\psi^*}(x) w > 0$ for all $x \in$ supp $p_{\mathcal{D}}$ and w not in the tangent space of* supp $p_{\mathcal{D}}$ *at x.*

We also need a modified version of Assumption III which ensures certain regularity properties of the reparameterization manifolds $\mathcal{M}_G$ and $\tilde{\mathcal{M}}_D$ near the equilibrium (θ^*, ψ^*). To formulate Assumption III″, we need

$$\tilde{g}(\psi) := \nabla_\theta \, \mathrm{E}_{x \sim p_\theta} \left[D_\psi(x) \right] \big|_{\theta = \theta^*} \tag{D.1}$$

Assumption III″. *There are ε-balls $\mathcal{B}_\varepsilon(\theta^*)$ and $\mathcal{B}_\varepsilon(\psi^*)$ around θ^* and ψ^* so that $\mathcal{M}_G \cap \mathcal{B}_\varepsilon(\theta^*)$ and $\tilde{\mathcal{M}}_D \cap \mathcal{B}_\varepsilon(\psi^*)$ define $\mathcal{C}^1$- manifolds. Moreover, the following holds:*

(i) if $w \in \mathbb{R}^m$ is not in the tangent space of $\tilde{\mathcal{M}}_D$ at ψ^*, then $\partial_w \tilde{g}(\psi^*) \neq 0$.

(ii) if $w \in \mathbb{R}^n$ is not in the tangent space of $\mathcal{M}_G$ at θ^*, then there is a latent code $z \in \mathcal{Z}$ so that $\frac{\partial}{\partial \theta} G_\theta(z) w|_{\theta = \theta^*}$ is not in the tangent space of supp $p_{\mathcal{D}}$ *at $G_{\theta^*}(z) \in$* supp $p_{\mathcal{D}}$.

The first part of Assumption III″ implies that the generator gradients become nonzero whenever the discriminator moves away from an equilibrium discriminator. The second part of Assumption III″ means that every time the generator leaves the equilibrium, it pushes some data point away from supp $p_{\mathcal{D}}$, i.e. the generator is not simply redistributing mass on supp $p_{\mathcal{D}}$.

In Theorem D.3 we show that Energy Solutions lead to local convergence of the unregularized GAN training dynamics. For the proof, we first need a generalization of Lemma 5.2:

Lemma D.1. *Assume that (θ^*, ψ^*) satisfies Assumption I″. The Jacobian of the gradient vector field $v(\theta, \psi)$ at (θ^*, ψ^*) is then given by*

$$v'(\theta^*, \psi^*) = \begin{pmatrix} -K_{GG} & -K_{DG}^\mathsf{T} \\ K_{DG} & -K_{DD} \end{pmatrix}. \tag{D.2}$$

The terms K_{DD} and K_{DG} are given by

$$K_{GG} = \rho_1 \, \mathrm{E}_{z\sim p_0} \left[\left[\frac{\partial G_\theta}{\partial \theta}(z) \right]^\mathsf{T} \nabla_x^2 D_{\psi^*}(G_\theta(z)) \frac{\partial G_\theta}{\partial \theta}(z) \right] \Bigg|_{\theta=\theta^*} \tag{D.3}$$

$$K_{DD} = \rho_2 \, \mathrm{E}_{x\sim p_\mathcal{D}} \left[\nabla_\psi D_\psi(x) \nabla_\psi D_\psi(x)^\mathsf{T} \right] \big|_{\psi=\psi^*} \tag{D.4}$$

$$K_{DG} = \rho_1 \, \frac{\partial}{\partial \theta} \mathrm{E}_{x\sim p_\theta} \left[\nabla_\psi D_\psi(x) \right] \Big|_{\theta=\theta^*,\psi=\psi^*} \tag{D.5}$$

Proof. Almost all parts of the proof of Lemma 5.2 are still valid, but Assumption IV is no longer true. In the proof of Lemma 5.2 we used Assumption IV to show $\nabla_\theta^2 \mathcal{L}(\theta^*, \psi^*) = 0$ and $\nabla_{\psi,\theta}^2 \mathcal{L}(\theta^*, \psi^*) = K_{DG}$.

However, by Assumption I″ we have $\nabla_x D_\psi(x) = 0$ for $x \in \operatorname{supp} p_\mathcal{D}$. We can therefore calculate $\nabla_{\psi,\theta}^2 \mathcal{L}(\theta^*, \psi^*)$ by taking the derivative of (5.7) with respect to ψ and exploit that all terms that contain $\nabla_x D_{\psi^*}(x)$ for $x \in \operatorname{supp} p_\mathcal{D}$ vanish. Using (5.7), we see that

$$\begin{aligned} \nabla_{\theta,\psi}^2 \mathcal{L}(\theta, \psi) = \mathrm{E}_{z\sim p_0} \Big[& \varphi_1''(D_\psi(G_\theta(z))) \left[\frac{\partial G_\theta}{\partial \theta}(z) \right]^\mathsf{T} \cdot \nabla_x D_\psi(G_\theta(z)) \nabla_\psi D_\psi(G_\theta(z))^\mathsf{T} \\ & + \varphi_1'(D_\psi(G_\theta(z))) \left[\frac{\partial G_\theta}{\partial \theta}(z) \right]^\mathsf{T} \cdot \nabla_{x,\psi} D_\psi(G_\theta(z)) \Big] \end{aligned} \tag{D.6}$$

Hence,

$$\begin{aligned} \nabla_{\theta,\psi}^2 \mathcal{L}(\theta^*, \psi^*) &= \mathrm{E}_{z\sim p_0} \left[\rho_1 \left[\frac{\partial G_\theta}{\partial \theta}(z) \right]^\mathsf{T} \cdot \nabla_{x,\psi} D_\psi(G_\theta(z)) \right] \Bigg|_{\theta=\theta^*,\psi=\psi^*} \\ &= \rho_1 \nabla_\theta \, \mathrm{E}_{z\sim p_0} \left[\frac{\partial D_\psi}{\partial \psi}(G_\theta(z)) \right] \Bigg|_{\theta=\theta^*,\psi=\psi^*} \end{aligned} \tag{D.7}$$

This shows that

$$\nabla_{\psi,\theta}^2 \mathcal{L}(\theta^*, \psi^*) = \nabla_{\theta,\psi}^2 \mathcal{L}(\theta^*, \psi^*)^\mathsf{T} = \rho_1 \, \frac{\partial}{\partial \theta} \mathrm{E}_{x\sim p_\theta} \left[\nabla_\psi D_\psi(x) \right] \Big|_{\theta=\theta^*,\psi=\psi^*} \tag{D.8}$$

Similarly, to compute $\nabla_\theta^2 \mathcal{L}(\theta^*, \psi^*)$ we take the partial Jacobian of (5.7) with respect to θ and evaluate at (θ^*, ψ^*). Here, we again exploit that all terms that contain $\nabla_x D_{\psi^*}(x) = 0$ vanish, showing $\nabla_\theta^2 \mathcal{L}(\theta^*, \psi^*) = K_{GG}$. □

We also need a simple generalization of Lemma 5.5.

Lemma D.2. *Assume $J \in \mathbb{R}^{(n+m)\times(n+m)}$ is of the following form:*

$$J = \begin{pmatrix} -Q_1 & -P^\mathsf{T} \\ P & -Q_2 \end{pmatrix} \tag{D.9}$$

where $Q_1 \in \mathbb{R}^{n\times n}$ is a symmetric positive semi-definite matrix and $Q_2 \in \mathbb{R}^{m\times m}$ is a symmetric

positive definite matrix and $P \in \mathbb{R}^{m\times n}$ has full column rank. Then all eigenvalues λ of J satisfy $\Re(\lambda) < 0$.

Proof. The proof is almost identical to the proof of Lemma 5.5, except that now

$$\Re(\lambda) = \frac{\lambda + \bar{\lambda}}{2} = -w_1 Q_1 w_1 - w_2 Q_2 w_2 \leq -w_2 Q_2 w_2 \tag{D.10}$$

The rest of the proof remains the same. □

We are now ready to formulate our convergence result for Energy Solutions:

Theorem D.3. *Assume Assumption I′′, II′ and III′′ hold for (θ^*, ψ^*). For small enough learning rates, Simultaneous Gradient Descent (SimGD) and Alternating Gradient Descent (AltGD) for the (unregularized) gradient vector field $v(\cdot)$ are both convergent to $\mathcal{M}_G \times \tilde{\mathcal{M}}_D$ in a neighborhood of (θ^*, ψ^*). Moreover, the rate of convergence is at least linear.*

Proof (Sketch). The proof is similar to the proof of Theorem 5.8.

First, note that $\mathcal{M}_G \times \tilde{\mathcal{M}}_D$ still only consists of equilibrium points. Next, we consider

$$B^\mathsf{T} v'(\theta^*, \psi^*) B = \begin{pmatrix} -B_G^\mathsf{T} K_{GG} B_G & -B_G^\mathsf{T} K_{DG}^\mathsf{T} B_D \\ B_D^\mathsf{T} K_{DG} B_G & -B_D^\mathsf{T} K_{DD} B_D \end{pmatrix} \tag{D.11}$$

For w not in the tangent space of $\mathcal{M}_G$ at θ^*, we have $w^\mathsf{T} K_{GG} w > 0$. This can be shown using Lemma D.1, Assumption I′′ and the second part of Assumption III′′. This implies that $B_G^\mathsf{T} K_{GG} B_G$ is positive definite.

Moreover, we need to show that for w not in the tangent space of $\tilde{\mathcal{M}}_D$ at ψ^*, we have $K_{DG}^\mathsf{T} w \notin \mathrm{T}_\theta^* \mathcal{M}_G$. This can be shown by applying the first part of Assumption III′′. As a result, we see that $B_G^\mathsf{T} K_{DG}^\mathsf{T} B_D$ has full column rank.

The rest of the proof is the same as the proof of Theorem 5.8 except that we use Lemma D.2 instead of Lemma 5.5. □

Note that Energy Solutions are only possible, if the discriminator is able to satisfy Assumption I′′. This is not the case for the Dirac-GAN from Chapter 3. However, if we use a quadratic discriminator instead, there are also Energy Solutions to the unregularized GAN training dynamics for the Dirac-GAN. To see this, we can parameterize $D_\psi(x)$ as

$$D_\psi(x) := \psi_1 x^2 + \psi_2 x. \tag{D.12}$$

It is easy to check that the Dirac-GAN with a discriminator as in (D.12) indeed has Energy Solutions: every (θ, ψ) with $\theta = 0$ and $\psi_2 = 0$ defines an equilibrium point of the Dirac-GAN and the GAN training dynamics are locally convergent near this point if $\psi_1 > 0$. Note however, that even though all equilibria with $\psi_1 > 0$ are points of attraction for the *continuous* GAN training dynamics, they may not be attractors for the *discretized system* when ψ_1 is large and the learning rate h is fixed. In general, the conditioning of Energy Solutions depends on the condition numbers of the Hessians $\nabla_x^2 D_{\psi^*}(x)$ at all $x \in \operatorname{supp} p_\mathcal{D}$. Indeed, the presence of ill-conditioned Energy Solutions might be one possible explanation

why Wasserstein GAN with Gradient Penalties often works well in practice although it is not even locally convergent for the Dirac-GAN.

E Additional Implementation Details

E.1 GAN Architectures

For CelebA and LSUN, we use Convolutional Neural Networks for both the generator (Table E.1) and discriminator (Table E.2). For CelebA-HQ, we use almost the same architecture as for CelebA, but add two more levels to the generator to increase the resolution from 256×256 to 1024×1024 and decrease the number of features from 64 to 16. We modify the discriminator architecture in a similar way.

For the ImageNet experiment, we use ResNet-architectures for the generator and discriminator, both having 26 layers in total. Both the generator and discriminator are conditioned on the labels of the input data. The architectures for the generator and discriminator are shown in Table E.3 and Table E.4.

Layer	Output Size	Filter
Fully-Connected	$1024 \cdot 4 \cdot 4$	$512 \rightarrow 1024 \cdot 4 \cdot 4$
Reshape	$1024 \times 4 \times 4$	-
ResNet-Block	$1024 \times 4 \times 4$	$1024 \rightarrow 1024 \rightarrow 1024$
NN-Upsampling	$1024 \times 8 \times 8$	-
ResNet-Block	$1024 \times 8 \times 8$	$1024 \rightarrow 1024 \rightarrow 1024$
NN-Upsampling	$1024 \times 16 \times 16$	-
ResNet-Block	$512 \times 16 \times 16$	$1024 \rightarrow 512 \rightarrow 512$
NN-Upsampling	$512 \times 32 \times 32$	-
ResNet-Block	$256 \times 32 \times 32$	$512 \rightarrow 256 \rightarrow 256$
NN-Upsampling	$256 \times 64 \times 64$	-
ResNet-Block	$128 \times 64 \times 64$	$256 \rightarrow 128 \rightarrow 128$
NN-Upsampling	$128 \times 128 \times 128$	-
ResNet-Block	$64 \times 128 \times 128$	$128 \rightarrow 64 \rightarrow 64$
NN-Upsampling	$64 \times 256 \times 256$	-
ResNet-Block	$64 \times 256 \times 256$	$64 \rightarrow 64 \rightarrow 64$
Conv2D	$3 \times 256 \times 256$	$64 \rightarrow 3$

Table E.1: Generator Architecture. *Generator architecture for CelebA- and LSUN-experiments.*

Layer	Output Size	Filter
Conv2D	$64 \times 256 \times 256$	$3 \rightarrow 64$
ResNet-Block	$64 \times 256 \times 256$	$64 \rightarrow 64 \rightarrow 64$
Avg-Pool2D	$64 \times 128 \times 128$	-
ResNet-Block	$128 \times 128 \times 128$	$64 \rightarrow 64 \rightarrow 128$
Avg-Pool2D	$128 \times 64 \times 64$	-
ResNet-Block	$256 \times 64 \times 64$	$128 \rightarrow 128 \rightarrow 256$
Avg-Pool2D	$256 \times 32 \times 32$	-
ResNet-Block	$512 \times 32 \times 32$	$256 \rightarrow 256 \rightarrow 512$
Avg-Pool2D	$512 \times 16 \times 16$	-
ResNet-Block	$1024 \times 16 \times 16$	$512 \rightarrow 512 \rightarrow 1024$
Avg-Pool2D	$1024 \times 8 \times 8$	-
ResNet-Block	$1024 \times 8 \times 8$	$1024 \rightarrow 1024 \rightarrow 1024$
Avg-Pool2D	$1024 \times 4 \times 4$	-
Fully-Connected	$1024 \cdot 4 \cdot 4$	$1024 \cdot 4 \cdot 4 \rightarrow 1$

Table E.2: Discriminator Architecture. *Discriminator architecture for CelebA- and LSUN-experiments.*

Layer	Output Size	Filter
Fully-Connected	$1024 \cdot 4 \cdot 4$	$512 \rightarrow 1024 \cdot 4 \cdot 4$
Reshape	$1024 \times 4 \times 4$	-
ResNet-Block	$1024 \times 4 \times 4$	$1024 \rightarrow 1024 \rightarrow 1024$
ResNet-Block	$1024 \times 4 \times 4$	$1024 \rightarrow 1024 \rightarrow 1024$
NN-Upsampling	$1024 \times 8 \times 8$	-
ResNet-Block	$1024 \times 8 \times 8$	$1024 \rightarrow 1024 \rightarrow 1024$
ResNet-Block	$1024 \times 8 \times 8$	$1024 \rightarrow 1024 \rightarrow 1024$
NN-Upsampling	$1024 \times 16 \times 16$	-
ResNet-Block	$512 \times 16 \times 16$	$1024 \rightarrow 512 \rightarrow 512$
ResNet-Block	$512 \times 16 \times 16$	$512 \rightarrow 512 \rightarrow 512$
NN-Upsampling	$512 \times 32 \times 32$	-
ResNet-Block	$256 \times 32 \times 32$	$512 \rightarrow 256 \rightarrow 256$
ResNet-Block	$256 \times 32 \times 32$	$256 \rightarrow 256 \rightarrow 256$
NN-Upsampling	$256 \times 64 \times 64$	-
ResNet-Block	$128 \times 64 \times 64$	$256 \rightarrow 128 \rightarrow 128$
ResNet-Block	$128 \times 64 \times 64$	$128 \rightarrow 128 \rightarrow 128$
NN-Upsampling	$128 \times 128 \times 128$	-
ResNet-Block	$64 \times 128 \times 128$	$128 \rightarrow 64 \rightarrow 64$
ResNet-Block	$64 \times 128 \times 128$	$64 \rightarrow 64 \rightarrow 64$
Conv2D	$3 \times 128 \times 128$	$64 \rightarrow 3$

Table E.3: Generator Architecture. *Generator architecture for ImageNet-experiment.*

Layer	Output Size	Filter
Conv2D	$64 \times 128 \times 128$	$3 \rightarrow 64$
ResNet-Block	$64 \times 128 \times 128$	$64 \rightarrow 64 \rightarrow 64$
ResNet-Block	$128 \times 128 \times 128$	$64 \rightarrow 64 \rightarrow 128$
Avg-Pool2D	$128 \times 64 \times 64$	-
ResNet-Block	$128 \times 64 \times 64$	$128 \rightarrow 128 \rightarrow 128$
ResNet-Block	$256 \times 64 \times 64$	$128 \rightarrow 128 \rightarrow 256$
Avg-Pool2D	$256 \times 32 \times 32$	-
ResNet-Block	$256 \times 32 \times 32$	$256 \rightarrow 256 \rightarrow 256$
ResNet-Block	$512 \times 32 \times 32$	$256 \rightarrow 256 \rightarrow 512$
Avg-Pool2D	$512 \times 16 \times 16$	-
ResNet-Block	$512 \times 16 \times 16$	$512 \rightarrow 512 \rightarrow 512$
ResNet-Block	$1024 \times 16 \times 16$	$512 \rightarrow 512 \rightarrow 1024$
Avg-Pool2D	$1024 \times 8 \times 8$	-
ResNet-Block	$1024 \times 8 \times 8$	$1024 \rightarrow 1024 \rightarrow 1024$
ResNet-Block	$1024 \times 8 \times 8$	$1024 \rightarrow 1024 \rightarrow 1024$
Avg-Pool2D	$1024 \times 4 \times 4$	-
ResNet-Block	$1024 \times 4 \times 4$	$1024 \rightarrow 1024 \rightarrow 1024$
ResNet-Block	$1024 \times 4 \times 4$	$1024 \rightarrow 1024 \rightarrow 1024$
Fully-Connected	$1024 \cdot 4 \cdot 4$	$1024 \cdot 4 \cdot 4 \rightarrow 1000$

Table E.4: Discriminator Architecture. *Discriminator architecture for ImageNet-experiment.*

E.2 Occupancy Network Baselines

In order to conduct controlled experiments and to disentangle the individual components of the different 3D reconstruction approaches, we created a PyTorch package comprising our method and several baselines [26, 45, 108, 193]. To make sure that the performance of our reimplementations of the baselines matches the performance of the original implementations, we conduct thorough comparisons to the original models.

3D-R2N2

The 3D Recurrent Reconstruction Neural Network (3D-R2N2) [26] is a method to map one or multiple images of an object to its 3D shape. The images are encoded as a low-dimensional vector and subsequently used as input to a 3D-LSTM network [72] to combine features from multiple views. Finally, the decoder uses the LSTM network's hidden states to produce a voxel representation of the object.

In our experiments, we use a ResNet-18 as encoder and a decoder consisting of one fully-connected and four transposed convolutional layers with ReLU activations. As we do not use multiple input views for 3D reconstruction, we do not utilize the proposed 3D-LSTM module. A quantitative comparison of the authors' implementation and our version can be found in Table E.5.

Point Set Generation Networks

Point Set Generation Network (PSGN) [45] is a point-based 3D object reconstruction model that takes a single image as input and outputs 1024 3D point coordinates.

The comparison of results from our architecture for PSGN and the deterministic two-branch version proposed in the original publication can be found in Table E.6. The former uses a ResNet-18 as encoder and four fully-connected layers with a hidden dimension of 512 and ReLU activations as decoder. The latter consists of seven blocks of multiple convolutional layers as encoder, and four blocks of transposed and ordinary convolutional layers as decoder where connections to the respective mirrored encoder blocks are introduced similar to U-Net [168]. We refer to the original publication [45] for more details. We train the models for 100 hours on a Nvidia GTX 1080 GPU with a batch size of 64.

As indicated in the table, the ResNet-18-based version achieves slightly better Accuracy, Completeness and Chamfer-L_1 distance. We suspect that the higher scores are caused by using a more powerful ResNet-18 as the encoder.

Our experiments suggest that PSGN is good at reconstructing 3D geometry. Unfortunately, however, PSGN lacks connectivity information and hence requires additional postprocessing steps to obtain the final mesh. In our experiments we find that reconstructing the final mesh from the output of PSGN is indeed very challenging. To reconstruct the mesh, we tried both Screened Poisson Surface Reconstruction (SPSR) [93] and Ball Pivoting (BP) [11]. Unfortunately, however, we were not able to find a set of hyperparameters for SPSR that is able to produce reasonable meshes for all input classes. We believe that this is partly due to the fact that PSGN does not output any normals. As SPSR requires normals, we therefore

have to estimate the normals in a preprocessing step using a moving window. Ball Pivoting works better in our experiments and some qualitative results are shown in Figure E.1. While Ball Pivoting succeeds in generating reasonable meshes from the output of PSGN, the meshes are very rough and have a lot of missing parts. In particular, the meshes generated by Ball Pivoting are not watertight.

E.2.1 Pixel2Mesh

Pixel2Mesh [193] is a mesh-based approach to reconstruct 3D shapes from single images. The initial start mesh of an ellipsoid is progressively deformed to match the object's shape by extracting perceptual features from the input image. The model consists of three feature pooling and mesh deformation blocks, each containing 12 (first two) or 13 (latter) graph convolution layers with ResNet-inspired [67] skip connections and ReLU activations. In the latter two blocks, the perceptual and hidden features from the last graph convolution layer are concatenated before an unpooling operation is applied. This way, the number of vertices is increased proportionally to the number of edges in the mesh. The final outputs of the network are locations for 156, 628, and 2466 vertices, each output corresponding to one block.

In our experiments, we adhere to the implementation provided in the official GitHub repository[1] which differs slightly from details given in the paper [193]. We compare the results from the authors' pretrained model on data they provide as well as our implementation on our ground truth data in different settings. While our ground truth point clouds are a fixed number of uniformly sampled points from the CAD models, the Pixel2Mesh authors performed Poisson-disk sampling which results in different numbers of points for each CAD model. In addition, they multiplied the vertex coordinates by a factor of 0.57 and inverted the y and z axes. To provide a fair comparison, we conduct our experiments using a template ellipsoid adjusted to our data by inverting the transformation and use a high number of target points (8,000). We train our model for several days on a Nvidia Titan X GPU with batch size 12.

The quantitative results can be seen in Table E.7. We observe that we can almost completely reproduce the results from the pretrained model with our reimplementation. In addition, we find that qualitative results and failure cases are very similar as well. For the car class, we note that while the car meshes of our implementation sometimes have cavities on the roof, the ones from the pretrained model show the same artifacts on the bottom. In our experience, the results depend on a carefully selected ellipsoid alignment as well as number of ground truth points. In contrast to Pixel2Mesh, the Occupancy Network is not directly trained on Chamfer loss and, surprisingly, yet achieves almost the same Chamfer-L_1 distance. In addition, our proposed method outperforms the Pixel2Mesh method significantly in both, Intersection over Union (IoU) and Normal Consistency score.

[1] `https://github.com/nywang16/Pixel2Mesh`

E.2.2 AtlasNet

We evaluate AtlasNet [63] using the code and pretrained model with 25 parameterizations provided by the authors[2]. As AtlasNet uses a slightly different test/train split as our code[3], we test AtlasNet on the intersection of the test splits of [63] and [26]. Because the data was normalized differently in [63] than in our framework, we use the ground truth point clouds provided by [63] to renormalize the output of [63] to the unit cube.

E.2.3 Deep Marching Cubes

We apply Deep Marching Cubes (DMC) [108] as a baseline for mesh-based point cloud completion. Given a point cloud, the method predicts mesh objects in an explicit representation consisting of occupancy probabilities and vertex displacement variables on a 3D grid. The first part of the pipeline distills point cloud information into a 3D grid by extracting point features and performing 3D grid pooling. Then, a U-Net-based [168] architecture with 3D convolutional layers parameterizes the mapping between the extracted grid features and the predicted mesh representation. The full architecture is trained in an end-to-end fashion using four different losses. The first loss penalizes the mean Euclidean distance between each point of the ground truth point cloud and its closest mesh face. A second loss (the weight prior) encourages the occupancy probabilities at the boundary of the scene to be small. The absolute difference between a pair of adjacent occupancy probabilities is penalized to enforce smoothness. Furthermore, an additional loss preserves smoothness between adjacent faces.

In the original implementation these loss functions are implemented as a CFFI CUDA extension for PyTorch since these loss functions are computationally very expensive. We adapted the original implementation in PyTorch 0.3.0 to our framework in PyTorch 1.0 without algorithmic changes. Unfortunately, the CFFI CUDA extensions in [108] are not compatible with PyTorch versions $> 0.3.0$, so that we transferred the original extensions to C++ / CUDA extensions for PyTorch 1.0.

To ensure a fair comparison, we use point clouds consisting of 300 instead of 3,000 points as input, in contrast to the original implementation. However, during training, we still apply the point-to-mesh loss to the full point cloud with 3,000 points following [108]. We observe that the default weighting of the weight prior 10 as chosen in the original implementation causes artifacts in form of rectangular shapes for flat 3D objects, thus we reduce the weight to 5.

[2] `https://github.com/ThibaultGROUEIX/AtlasNet`

[3] Almost all samples in the test split from [63] are also in the test split from [26], but not the other way around.

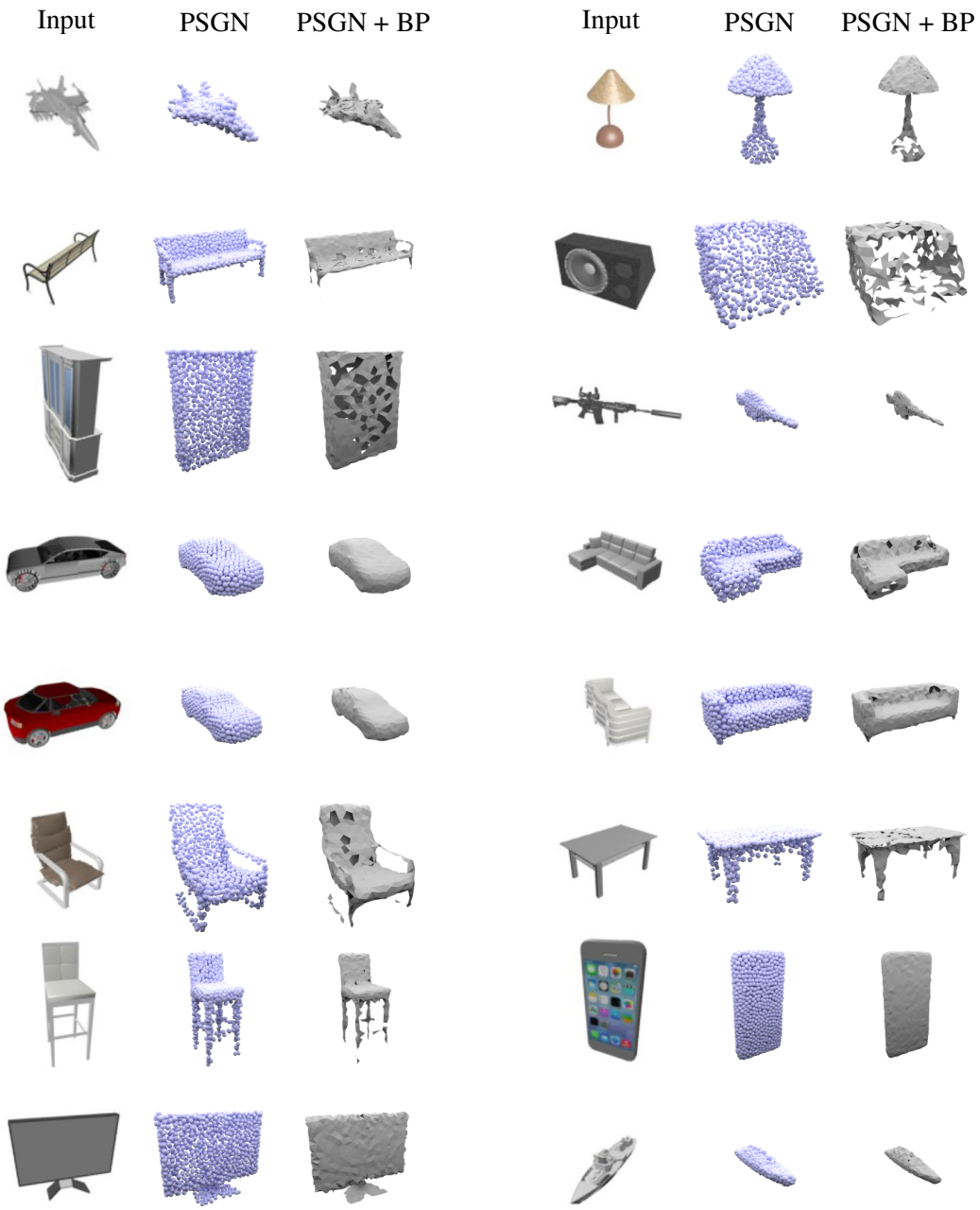


Figure E.1: PSGN Meshing Results. *To obtain a mesh for the output of PSGN [45], we apply the Ball Pivoting (BP) algorithm [11]. We find that meshing the output of PSGN directly is non-trivial: While the combination of PSGN and BP allows to obtain coarse meshes from single images, the output has many missing parts and the method does not lead to closed meshes.*

	3D-R2N2 (authors)			3D-R2N2 (ResNet18)
	1 view	3 views	5 views	1 view
category				
airplane	51.3%	54.9%	56.1%	**60.6%**
bench	42.1%	50.2%	52.7%	**56.2%**
cabinet	71.6%	76.3%	**77.2%**	76.4%
car	79.8%	82.9%	83.6%	**84.6%**
chair	46.6%	53.3%	**55.0%**	53.2%
display	46.8%	54.5%	**56.5%**	52.8%
lamp	38.1%	41.5%	**42.1%**	38.9%
loudspeaker	66.2%	70.8%	**71.7%**	68.5%
rifle	54.4%	59.3%	**60.0%**	59.5%
sofa	62.8%	69.0%	**70.6%**	**70.6%**
table	51.3%	56.4%	**58.0%**	57.1%
telephone	66.1%	73.2%	**75.4%**	74.0%
vessel	51.3%	59.6%	**61.0%**	60.6%
mean	56.0%	61.7%	**63.1%**	62.5%

Table E.5: 3D-R2N2 Reimplementation. *This table shows the IoU reported in the original 3D-R2N2 paper and our reimplementation with respect to the* voxelized *ground truth mesh. In contrast to other IoU values reported in this thesis, in this table we compare with the* 32^3 *voxelization of the ground truth meshes and not the high resolution meshes themselves.*

category	Accuracy PSGN (2 branch)	Accuracy PSGN (ResNet18)	Accuracy ONet	Completeness PSGN (2 branch)	Completeness PSGN (ResNet18)	Completeness ONet	Chamfer-L_1 PSGN (2 branch)	Chamfer-L_1 PSGN (ResNet18)	Chamfer-L_1 ONet
airplane	0.113	**0.101**	0.133	0.191	0.173	**0.161**	0.152	**0.137**	0.147
bench	0.161	**0.133**	0.154	0.262	0.230	**0.156**	0.212	0.181	**0.155**
cabinet	0.154	**0.138**	0.150	0.307	0.291	**0.184**	0.231	0.215	**0.167**
car	0.126	0.117	**0.116**	0.235	0.221	**0.203**	0.181	0.169	**0.159**
chair	0.233	**0.187**	0.223	0.333	0.307	**0.233**	0.283	0.247	**0.228**
display	0.258	**0.235**	0.281	0.351	0.333	**0.275**	0.304	0.284	**0.278**
lamp	0.301	**0.275**	0.402	0.372	**0.354**	0.557	0.337	**0.314**	0.479
loudspeaker	0.253	**0.245**	0.285	0.403	0.386	**0.315**	0.328	0.316	**0.300**
rifle	0.113	**0.110**	0.148	0.159	0.158	**0.134**	0.136	**0.134**	0.141
sofa	0.206	**0.171**	0.194	0.307	0.278	**0.195**	0.257	0.224	**0.194**
table	0.165	**0.152**	0.189	0.307	0.293	**0.189**	0.236	0.222	**0.189**
telephone	0.140	**0.122**	0.149	0.219	0.201	**0.130**	0.179	0.161	**0.140**
vessel	0.186	**0.153**	0.197	0.249	**0.223**	0.238	0.217	**0.188**	0.218
mean	0.185	**0.164**	0.202	0.284	0.265	**0.228**	0.235	**0.215**	**0.215**

Table E.6: PSGN Reimplementation. *Comparison of our architecture for PSGN and the two branch architecture proposed in the original publication.*

category	IoU			Chamfer-L_1			Normal Consistency		
	Pixel2Mesh (authors)	Pixel2Mesh (reimpl.)	ONet	Pixel2Mesh (authors)	Pixel2Mesh (reimpl.)	ONet	Pixel2Mesh (authors)	Pixel2Mesh (reimpl.)	ONet
airplane	40.0%	42.0%	**57.1%**	0.191	0.187	**0.147**	0.775	0.759	**0.840**
bench	27.2%	32.3%	**48.5%**	0.198	0.201	**0.155**	0.736	0.732	**0.813**
cabinet	60.9%	66.4%	**73.3%**	0.228	0.196	**0.167**	0.828	0.834	**0.879**
car	54.4%	55.2%	**73.7%**	0.190	0.180	**0.159**	0.762	0.756	**0.852**
chair	37.1%	39.6%	**50.1%**	0.270	0.265	**0.228**	0.745	0.746	**0.823**
display	44.0%	**49.0%**	47.1%	0.245	**0.239**	0.278	0.840	0.830	**0.854**
lamp	30.4%	32.3%	**37.1%**	0.315	**0.308**	0.479	0.688	0.666	**0.731**
loudspeaker	56.3%	59.9%	**64.7%**	0.306	**0.285**	0.300	0.796	0.782	**0.832**
rifle	34.1%	40.2%	**47.4%**	0.167	0.164	**0.141**	0.715	0.718	**0.766**
sofa	54.4%	61.3%	**68.0%**	0.233	0.212	**0.194**	0.806	0.820	**0.863**
table	32.3%	39.5%	**50.6%**	0.229	0.218	**0.189**	0.785	0.784	**0.858**
telephone	62.5%	66.1%	**72.0%**	0.156	0.149	**0.140**	0.912	0.907	**0.935**
vessel	35.7%	39.7%	**53.0%**	0.221	**0.212**	0.218	0.716	0.699	**0.794**
mean	43.8%	48.0%	**57.1%**	0.227	0.216	**0.215**	0.777	0.772	**0.834**

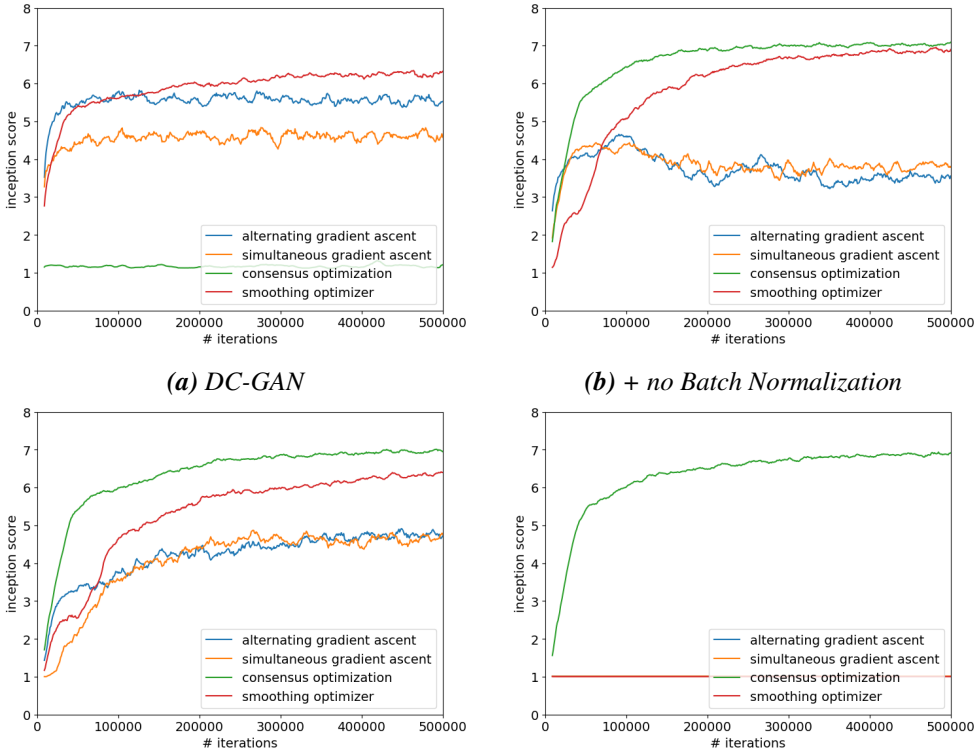
Table E.7: Pixel2Mesh Reimplementation. *Comparison of our implementation of Pixel2Mesh, the pretrained Pixel2Mesh network provided by the authors and our method. While both methods achieve a similar Chamfer-L_1 distance, our method still performs significantly better regarding IoU and Normal Consistency score. Note that our method does not directly optimize the Chamfer distance during training whereas Pixel2Mesh does.*

F Additional Experimental Results

F.1 Consensus Optimization

CIFAR-10 A quantitative comparison of Consensus Optimization to Alternating Gradient Descent (AltGD) and Simultaneous Gradient Descent (SimGD) is shown in Figure F.1. We see that in three out of four configurations, Consensus Optimization achieves a higher Inception score than SimGD and AltGD. The smoothing optimizer in Figure F.1 is similar to Consensus Optimization, where we apply the regularizer only to the discriminator. While is also often works, it usually achieves a lower Inception score and fails in the scenario where we use a Jensen-Shannon objective. This is also the case for an architecture where we add an additional fully-connected layer at the last layer (Figure F.2): while Consensus Optimization works well here, the smoothing optimizer fails to converge. Figure F.3 shows qualitative results on CIFAR-10 when trained with different divergence functions. Interestingly, Consensus Optimization converges both for the Jensen-Shannon divergence and the Indicator divergence, where normal SimGD and AltGD do not converge. Finally, Figure F.4, Figure F.6 and Figure F.5 show the effects of the hyperparameters on AltGD and Consensus Optimization. We see that the learning rate and regularization parameter are critical hyperparameters for Consensus Optimization. As a rule of thumb, the learning rate should be between 10^{-5} and 10^{-4} and the regularization parameter should be between 1 and 10.

CelebA In this experiment, we again use a DC-GAN-like architecture [160] without Batch Normalization in the generator or the discriminator. Moreover, we additionally use a constant number of filters in each layer and add additional ResNet-layers. A qualitative comparison between AltGD and Consensus Optimization on CelebA is shown in Figure F.7. We clearly see that AltGD leads to mode collapse and Consensus Optimization yields better results.



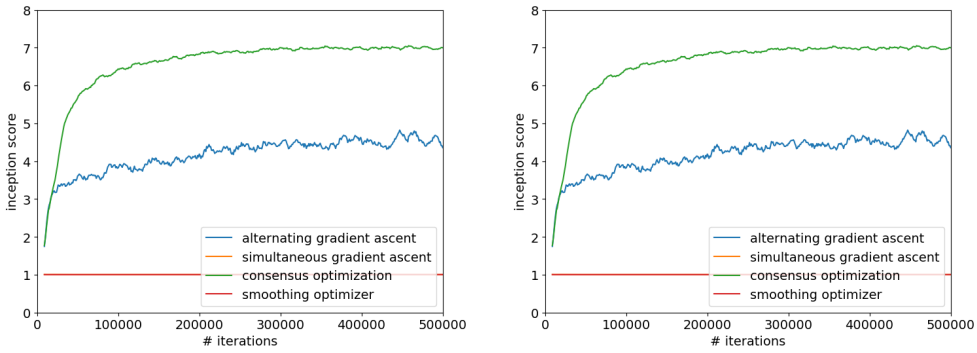
(a) *DC-GAN*

(b) *+ no Batch Normalization*

(c) *+ constant number of filters in each layer*

(d) *+ Jensen-Shannon objective*

Figure F.1: Inception Score. *Inception score over the number of iterations for the results in Figure 2.7. To investigate if Consensus Optimization can alternatively be interpreted as a way of smoothing the discriminator, we also conducted experiments where we removed the regularization term from the generator loss but keep it for the discriminator loss. We call the corresponding algorithm* smoothing optimizer. *While the smoothing optimizer trains a DC-GAN with Batch Normalization successfully where Consensus Optimization fails, it usually performs worse than Consensus Optimization.*

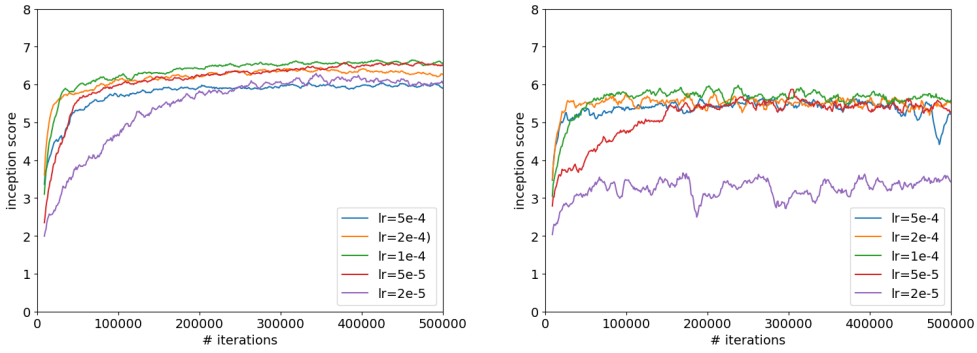


(a) *+ constant number of filters in each layer* ***(b)*** *+ Jensen-Shannon objective*

Figure F.2: Inception Score. *Inception score over the number of iterations for training the architecture from Figure F.1 with an additional fully-connected layer in the discriminator on CIFAR-10. We observe similar results as in Figure F.1, but the smoothing optimizer fails for the architecture with a constant number of filters in each layer where Consensus Optimization succeeds.*

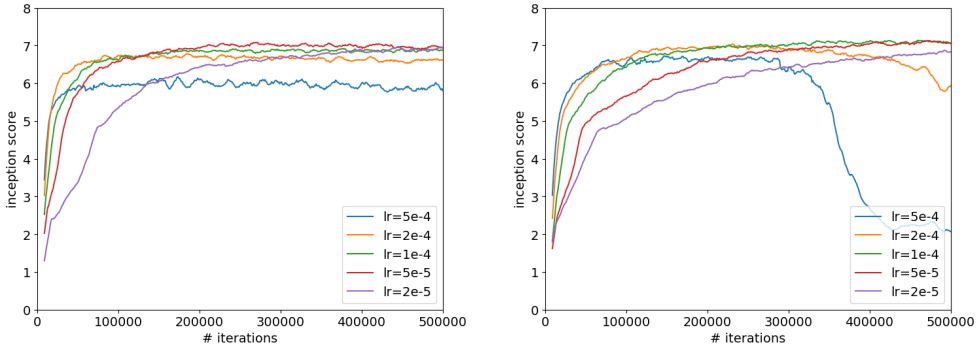
(a) *Standard GAN objective* ***(b)*** *Jensen-Shannon* ***(c)*** *Indicator divergence*

Figure F.3: Divergences. *We can use Consensus Optimization to train models with different generator objectives corresponding to different divergence functions: all samples were generated by a DC-GAN model without Batch Normalization and with a constant number of filters in each layer. We see that Consensus Optimization can be used to train GANs with a large variety of divergence functions.*



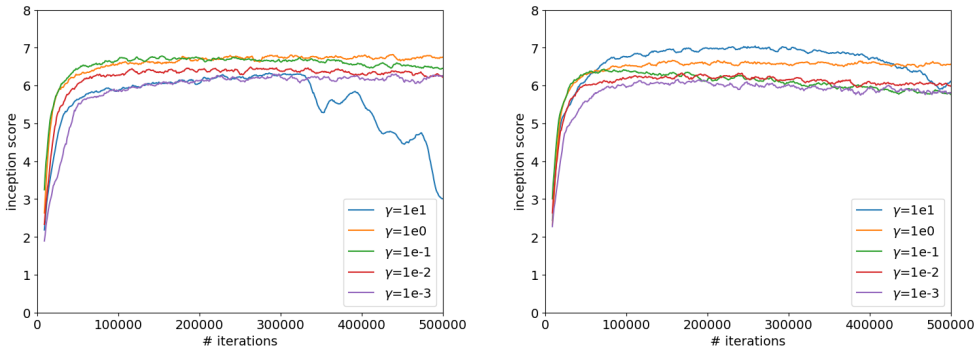
(a) *3 convolutional layers* ***(b)*** *4 convolutional layers*

Figure F.4: Learning Rate. *Effect of the learning rate on training a DC-GAN architecture with AltGD. We use RMSProp [69] as an optimizer for all experiments.*



(a) *3 convolutional layers* ***(b)*** *4 convolutional layers*

Figure F.5: Learning Rate. *Effect of the learning rate on Consensus Optimization. We use the RMSProp optimizer [69] for all experiments with a regularization parameter* $\gamma = 0.1$ *for the architecture with* 3 *convolutional layers and* $\gamma = 10$ *for the architecture with* 4 *convolutional layers.*



(a) *3 convolutional layers* ***(b)*** *4 convolutional layers*

Figure F.6: Regularization Parameter. *Effect of the regularization parameter γ on Consensus Optimization. We use the RMSProp optimizer [69] with a learning rate of $2 \cdot 10^{-4}$ for all experiments.*

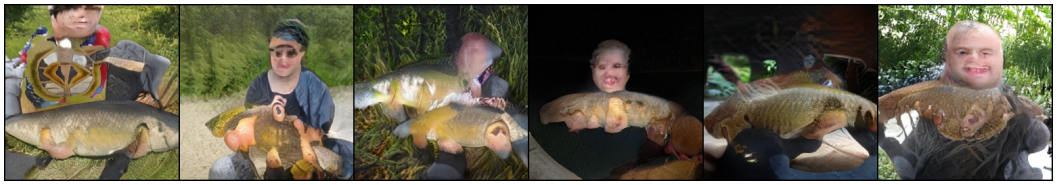
(a) *SimGD* ***(b)*** *AltGD* ***(c)*** *ConOpt*

Figure F.7: Results on CelebA. *Samples generated from a model where both the generator and discriminator are based on a Deep Convolutional GAN (DC-GAN) [160]. However, we use no Batch Normalization, a constant number of filters in each layer and additional ResNet-layers. We see that only our method results in visually compelling results. While SimGD completely fails to train the model, AltGD results in a bad solution. Although AltGD does better than SimGD on this example, there is a significant amount of mode collapse and the results keep changing from iteration to iteration.*

F.2 High Resolution Image Synthesis

Figure F.8 and Figure F.9 show samples for a conditional GAN trained on ImageNet at resolution 256×256. The architecture is similar to the one for the ImageNet experiment from Chapter 8, but we use one more level to increase the resolution from 128×128 to 256×256 and decrease the number of features from 64 to 32. Moreover, we use 2048 instead of 1024 features in the top-most layer for the generator and the lower layer of the discriminator. We train our model on four Tesla V100 GPUs for about 750k iterations and a batch size of 192.

We observe that R_1-regularization stabilizes the training even at this resolution, allowing us to obtain realistic high-resolution samples on this challenging task. Indeed, even in later iterations of training, we did not experience any instabilities. The model reaches an Inception score of 45.0 and a FID of 32.0.



(a) tench

(b) papillon

(c) weevil

(d) admiral

(e) lighthouse

(f) castle

Figure F.8: High-Resolution Samples ImageNet. *Random class conditional samples for a GAN trained on the ImageNet dataset [171] at resolution* 256×256.

(a) home theater

(b) police van

(c) rugby ball

(d) ski

(e) pizza

(f) valley

Figure F.9: High-Resolution Samples ImageNet. *Random class conditional samples for a GAN trained on the ImageNet dataset [171] at resolution* 256×256.

F.3 Occupancy Networks

Additional qualitative results for the single image 3D reconstruction task are shown in Figure F.10 and Figure F.11. While all methods are able to reconstruct the 3D geometry, we observe that our method can better capture high frequencies details than other methods. Per-category quantitative results for the single-image 3D reconstruction experiment are shown in Table F.1. Moreover, Table F.2 contains per-category results on the Pix3D dataset.

Qualitative results for point cloud completion are shown in Figure F.12 and Figure F.13. We observe that our method is able to reconstruct high frequency details from the sparse input point clouds. The full quantitative results for point cloud completion are shown in Table F.3. Qualitative results for the voxel super-resolution task are shown in Figure F.14.

Figure F.15, F.16, F.17 and F.18 show interpolations in latent space for our unconditional model. We find that our model learns a meaningful representation of the input data.

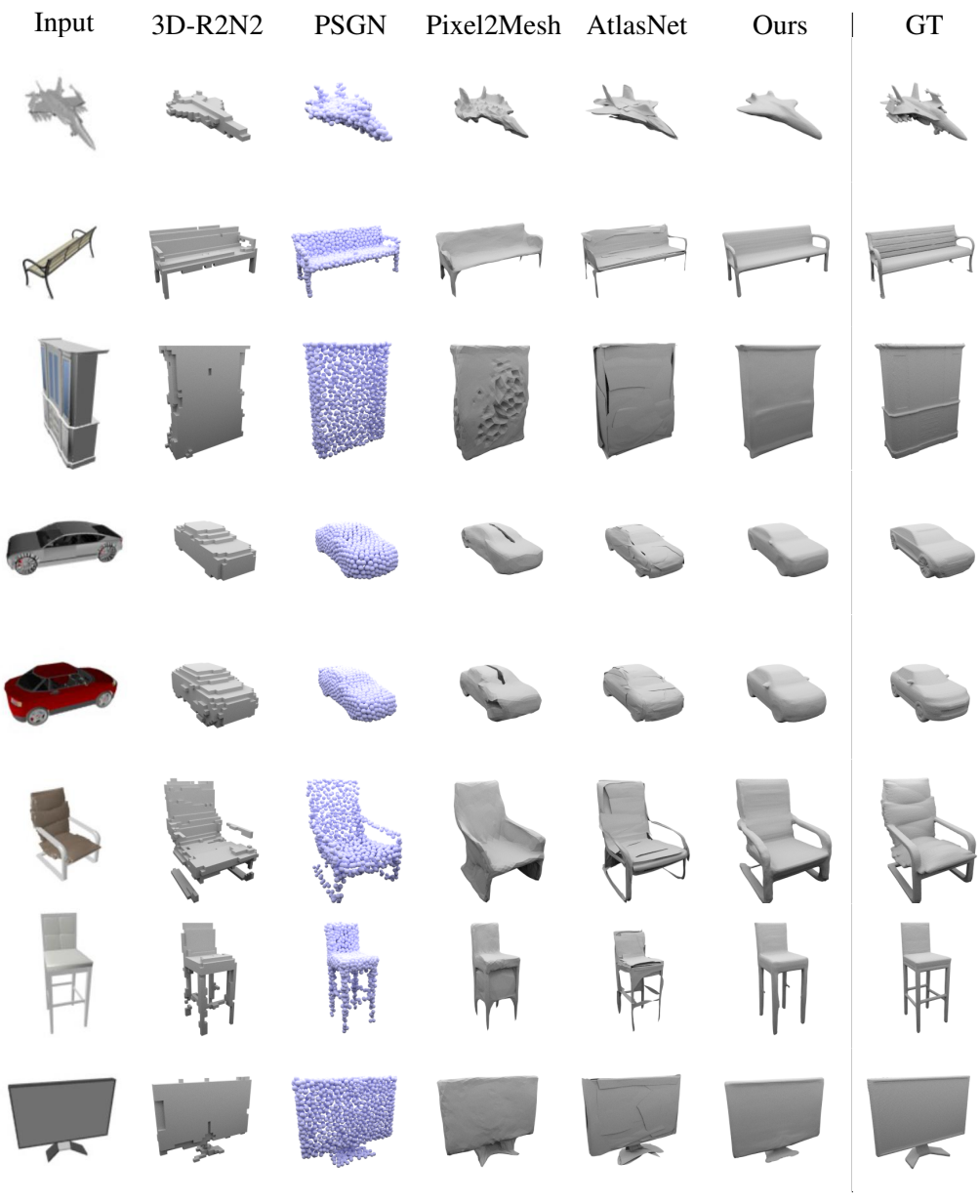


Figure F.10: Single Image 3D Reconstruction. *The input image is shown in the first column, the other columns show the results for our method compared to various baselines.*

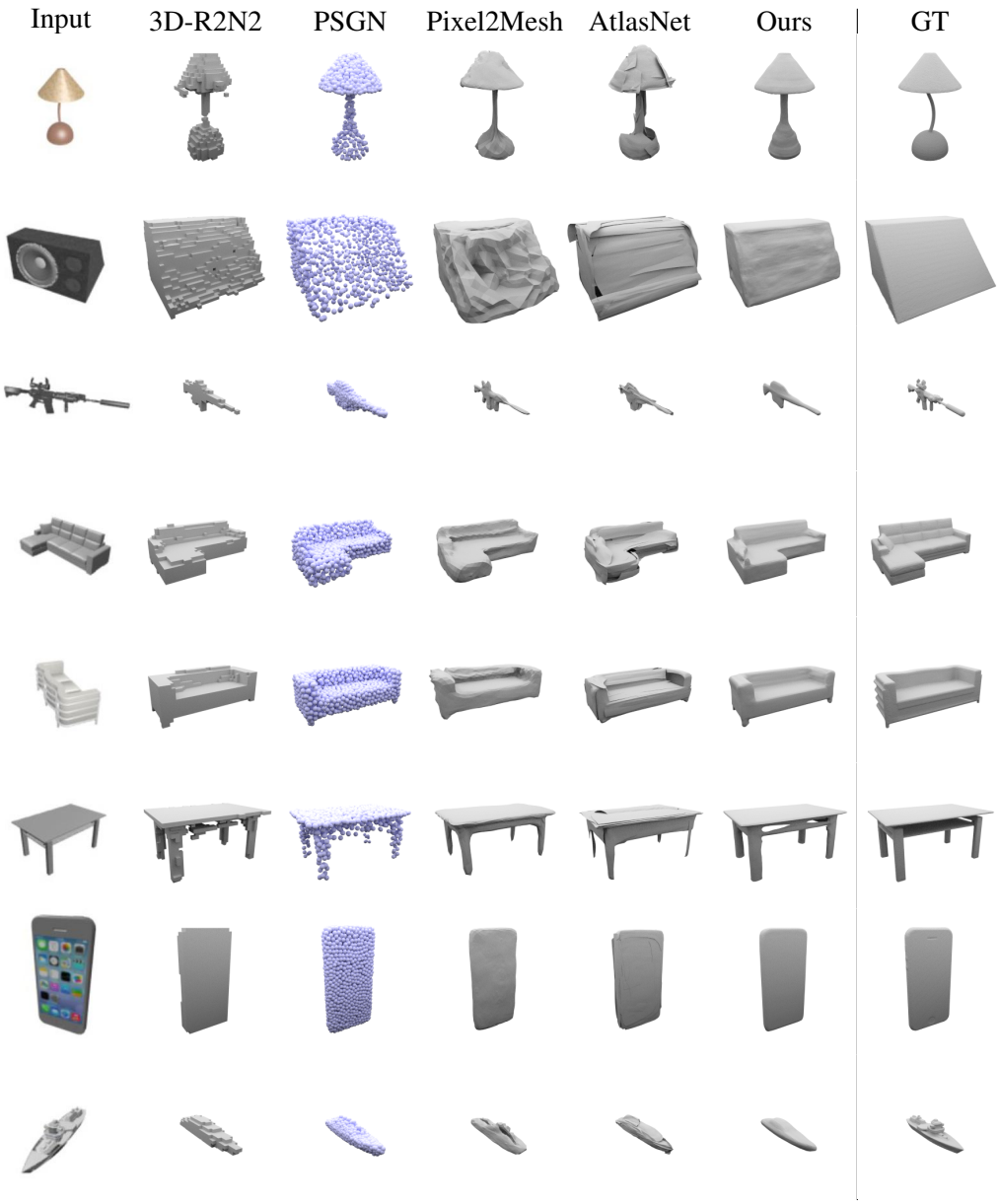


Figure F.11: Single Image 3D Reconstruction. *The input image is shown in the first column, the other columns show the results for our method compared to various baselines.*

	IoU					Chamfer-L_1					Normal Consistency				
category	3D-R2N2	PSGN	Pixel2Mesh	AtlasNet	ONet	3D-R2N2	PSGN	Pixel2Mesh	AtlasNet	ONet	3D-R2N2	PSGN	Pixel2Mesh	AtlasNet	ONet
airplane	42.6%	-	42.0%	-	**57.1%**	0.227	0.137	0.187	**0.104**	0.147	0.629	-	0.759	0.836	**0.840**
bench	37.3%	-	32.3%	-	**48.5%**	0.194	0.181	0.201	**0.138**	0.155	0.678	-	0.732	0.779	**0.813**
cabinet	66.7%	-	66.4%	-	**73.3%**	0.217	0.215	0.196	0.175	**0.167**	0.782	-	0.834	0.850	**0.879**
car	66.1%	-	55.2%	-	**73.7%**	0.213	0.169	0.180	**0.141**	0.159	0.714	-	0.756	0.836	**0.852**
chair	43.9%	-	39.6%	-	**50.1%**	0.270	0.247	0.265	**0.209**	0.228	0.663	-	0.746	0.791	**0.823**
display	44.0%	-	**49.0%**	-	47.1%	0.314	0.284	0.239	**0.198**	0.278	0.720	-	0.830	**0.858**	0.854
lamp	28.1%	-	32.3%	-	**37.1%**	0.778	0.314	0.308	**0.305**	0.479	0.560	-	0.666	0.694	**0.731**
loudspeaker	61.1%	-	59.9%	-	**64.7%**	0.318	0.316	0.285	**0.245**	0.300	0.711	-	0.782	0.825	**0.832**
rifle	37.5%	-	40.2%	-	**47.4%**	0.183	0.134	0.164	**0.115**	0.141	0.670	-	0.718	0.725	**0.766**
sofa	62.6%	-	61.3%	-	**68.0%**	0.229	0.224	0.212	**0.177**	0.194	0.731	-	0.820	0.840	**0.863**
table	42.0%	-	39.5%	-	**50.6%**	0.239	0.222	0.218	0.190	**0.189**	0.732	-	0.784	0.832	**0.858**
telephone	61.1%	-	66.1%	-	**72.0%**	0.195	0.161	0.149	**0.128**	0.140	0.817	-	0.907	0.923	**0.935**
vessel	48.2%	-	39.7%	-	**53.0%**	0.238	0.188	0.212	**0.151**	0.218	0.629	-	0.699	0.756	**0.794**
mean	49.3%	-	48.0%	-	**57.1%**	0.278	0.215	0.216	**0.175**	0.215	0.695	-	0.772	0.811	**0.834**

Table F.1: Single Image 3D Reconstruction. *This table shows a numerical comparison of our approach and the baselines for single image 3D reconstruction on the ShapeNet dataset. We measure the Intersection over Union (IoU), Chamfer-L_1 distance and Normal Consistency score for various methods with respect to the ground truth mesh. Note that in contrast to prior work, we compute the IoU with respect to the high-resolution mesh and not a coarse voxel representation. All methods apart from AtlasNet [63] are evaluated on the test split by Choy et al. [26]. Since AtlasNet uses a pretrained model, we evaluate it on the intersection of the test splits from [26] and [63].*

	IoU					Chamfer-L_1					Normal Consistency				
category	3D-R2N2	PSGN	AtlasNet	ONet	ONet+	3D-R2N2	PSGN	AtlasNet	ONet	ONet+	3D-R2N2	PSGN	AtlasNet	ONet	ONet+
chair	14.6%	-	-	22.4%	**31.9%**	2.572	0.750	**0.392**	0.628	0.531	0.401	-	0.731	0.715	**0.747**
desk	17.0%	-	-	26.8%	**41.4%**	1.776	0.760	**0.509**	1.047	0.568	0.478	-	0.717	0.682	**0.780**
sofa	48.3%	-	-	58.0%	**69.9%**	0.537	0.592	**0.367**	0.427	0.413	0.645	-	0.792	0.817	**0.861**
table	13.7%	-	-	20.4%	**34.5%**	2.665	0.979	0.556	1.071	**0.502**	0.431	-	0.768	0.720	**0.809**
mean	23.4%	-	-	31.9%	**44.4%**	1.888	0.770	**0.456**	0.793	0.504	0.489	-	0.752	0.733	**0.799**

Table F.2: Pix3D dataset. *We evaluate the networks which were trained on ShapeNet on the Pix3D dataset [180] with ground truth masks. ONet+ was trained on our own renderings with more varied random views.*

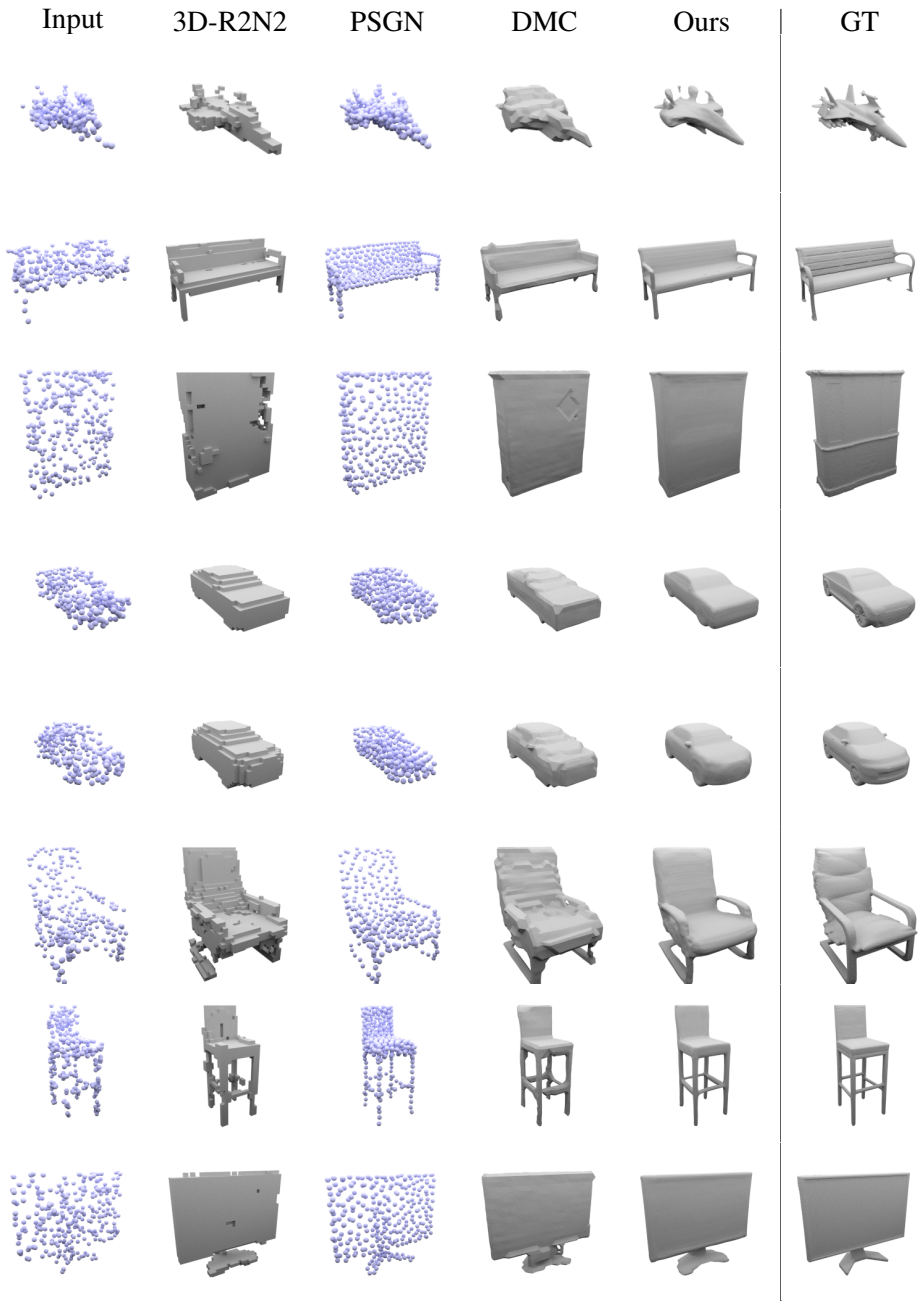


Figure F.12: Point Cloud Completion. *The input point cloud is shown in the first column, the other columns show the results for our method compared to various baselines.*

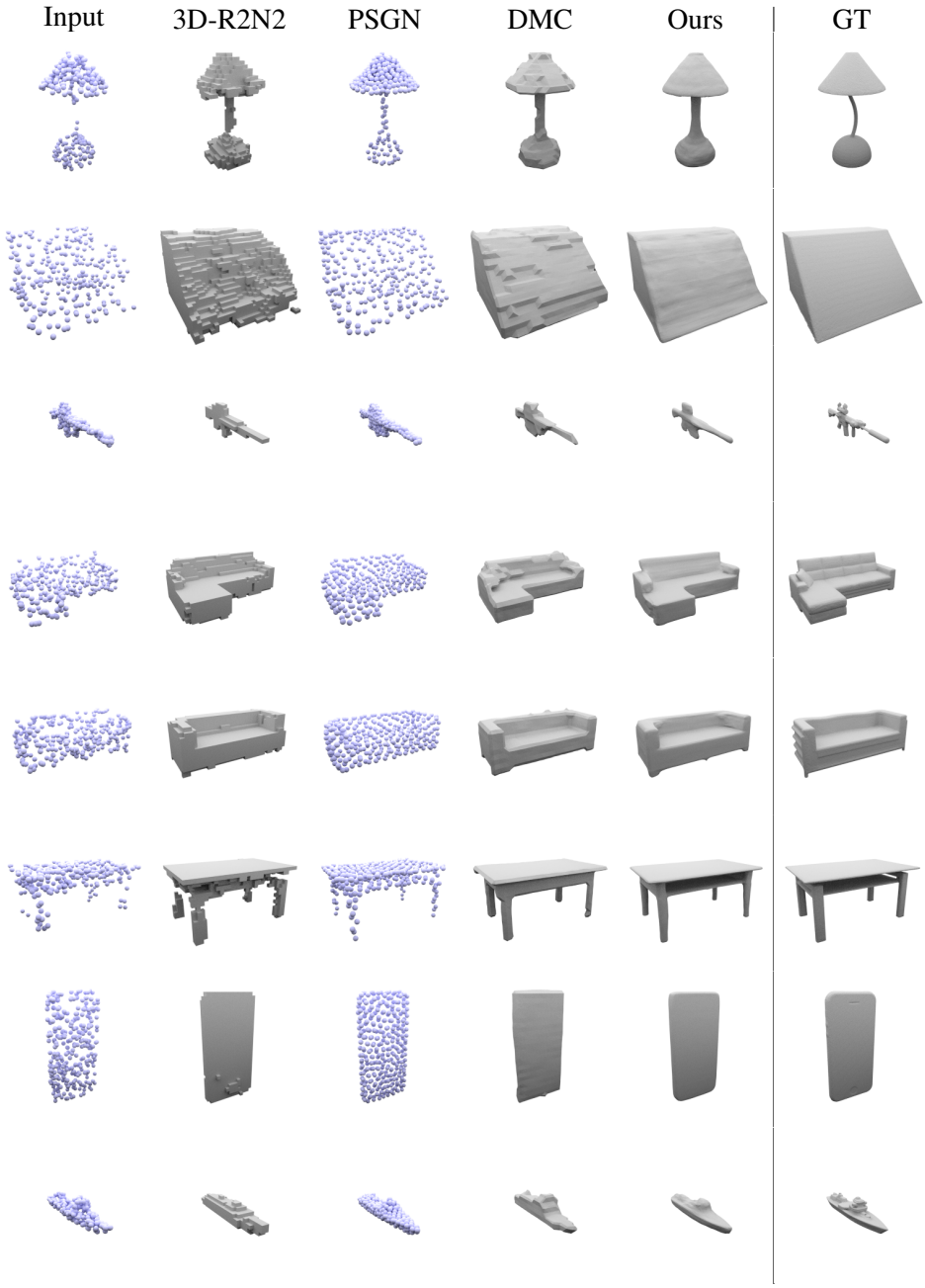


Figure F.13: Point Cloud Completion. *The input point cloud is shown in the first column, the other columns show the results for our method compared to various baselines.*

	IoU				Chamfer-L_1				Normal Consistency			
category	3D-R2N2	PSGN	DMC	ONet	3D-R2N2	PSGN	DMC	ONet	3D-R2N2	PSGN	DMC	ONet
airplane	46.4%	-	57.0%	**76.0%**	0.145	0.100	0.106	**0.056**	0.668	-	0.814	**0.897**
bench	44.1%	-	53.6%	**71.6%**	0.145	0.125	0.102	**0.059**	0.695	-	0.808	**0.878**
cabinet	71.8%	-	80.2%	**86.7%**	0.167	0.162	0.114	**0.073**	0.790	-	0.887	**0.916**
car	68.1%	-	73.1%	**83.5%**	0.197	0.130	0.166	**0.098**	0.719	-	0.825	**0.875**
chair	52.3%	-	64.2%	**73.6%**	0.181	0.165	0.121	**0.089**	0.676	-	0.842	**0.890**
display	60.3%	-	75.3%	**81.7%**	0.167	0.158	0.096	**0.076**	0.755	-	0.902	**0.926**
lamp	35.6%	-	48.4%	**56.6%**	0.231	0.202	0.170	**0.135**	0.601	-	0.777	**0.813**
loudspeaker	70.5%	-	78.9%	**82.8%**	0.200	0.201	0.151	**0.116**	0.741	-	0.882	**0.898**
rifle	41.9%	-	58.0%	**69.4%**	0.145	0.087	0.083	**0.060**	0.707	-	0.780	**0.864**
sofa	70.4%	-	80.0%	**87.2%**	0.158	0.144	0.103	**0.069**	0.760	-	0.886	**0.928**
table	48.3%	-	62.4%	**75.9%**	0.172	0.163	0.103	**0.071**	0.742	-	0.874	**0.917**
telephone	69.9%	-	84.8%	**91.5%**	0.128	0.109	0.062	**0.041**	0.851	-	0.949	**0.970**
vessel	55.4%	-	60.3%	**74.8%**	0.163	0.126	0.150	**0.085**	0.647	-	0.798	**0.859**
mean	56.5%	-	67.4%	**77.8%**	0.169	0.144	0.117	**0.079**	0.719	-	0.848	**0.895**

Table F.3: Point Cloud Completion. *This table shows a per-category numerical comparison of our approach and the baselines for point cloud completion on the ShapeNet dataset. We measure the IoU, Chamfer-L_1 distance and Normal Consistency score for various methods with respect to the ground truth mesh. Note that in contrast to prior work, we compute the IoU with respect to the high-resolution mesh and not a coarse voxel representation. Due to missing connectivity information, IoU and the Normal Consistency score cannot be calculated for PSGN.*

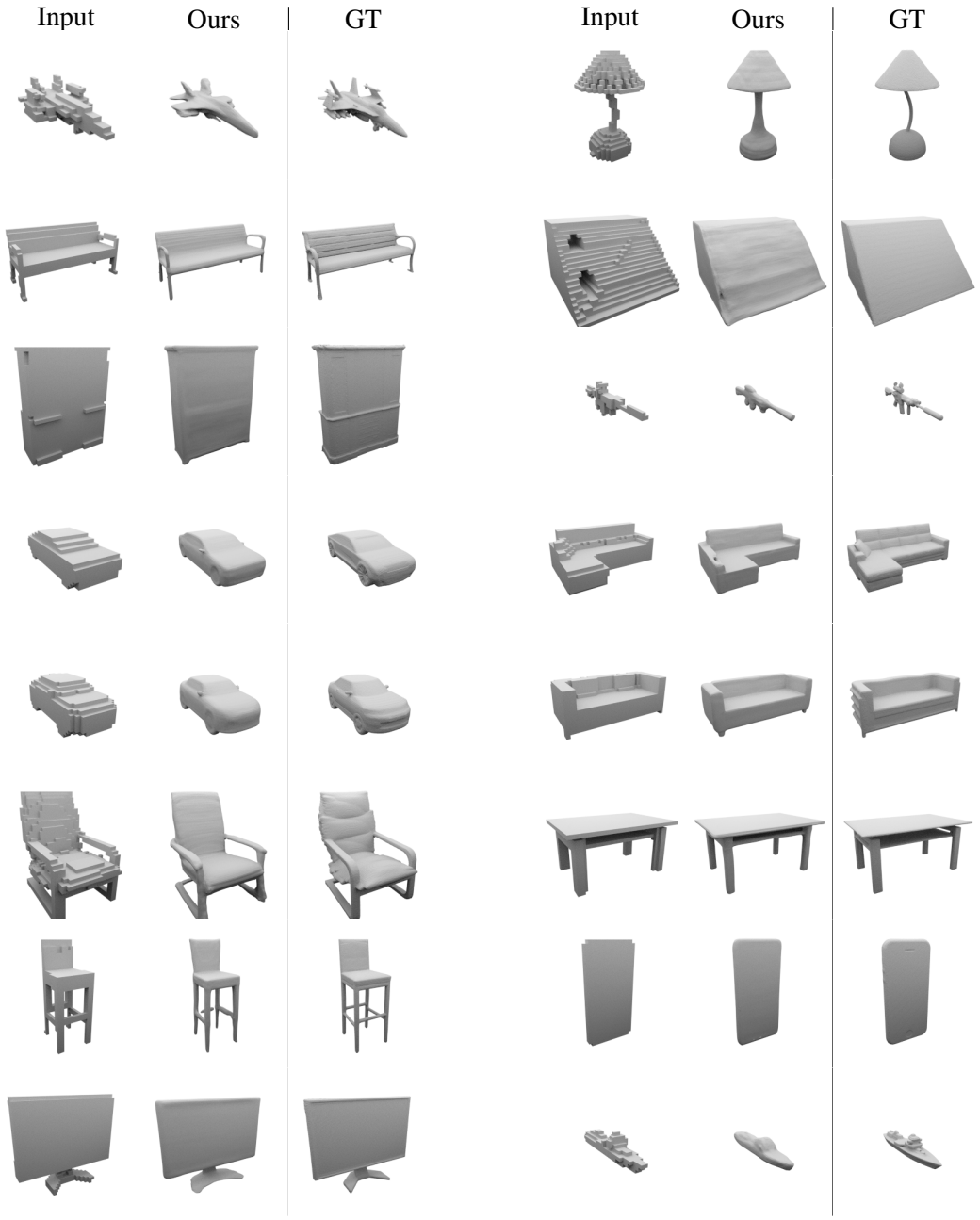


Figure F.14: Voxel Super-Resolution. *The input is shown in the first column, the other columns show the results for our method compared to the ground truth.*

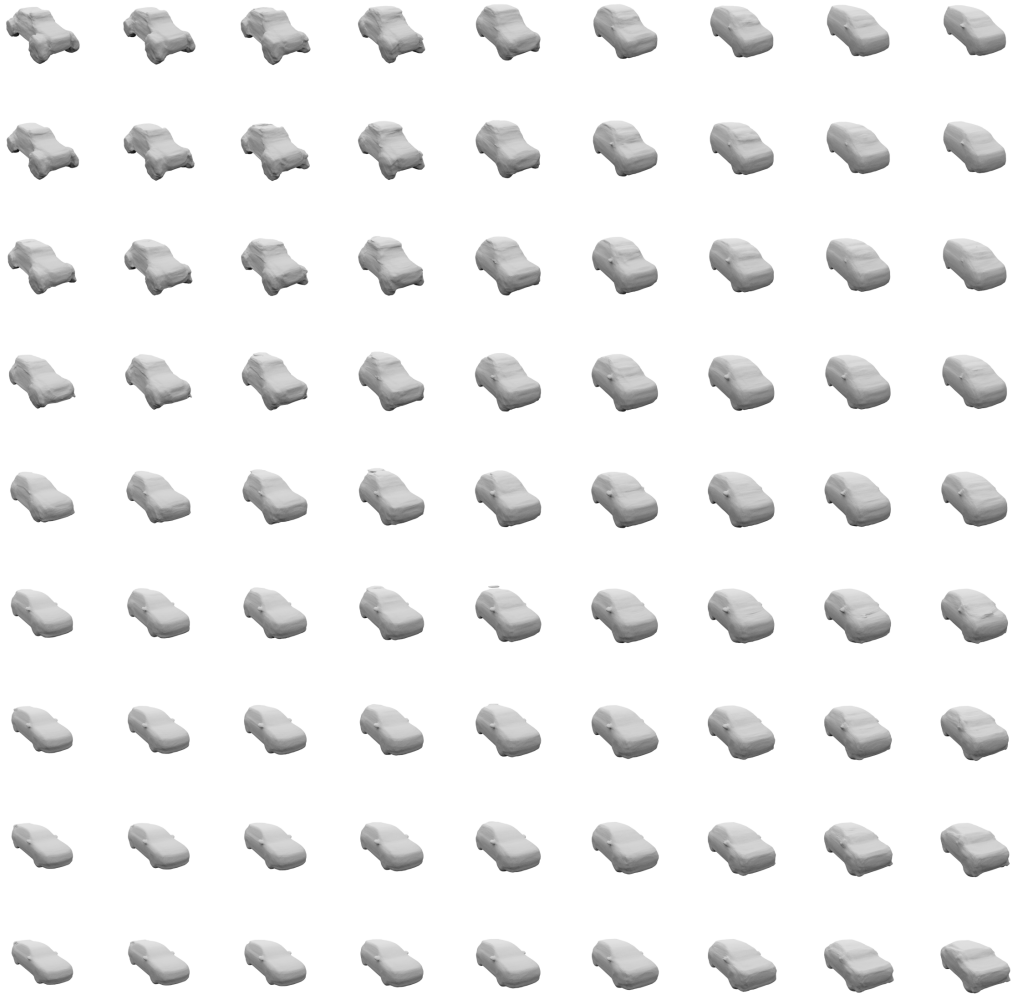


Figure F.15: Unconditional Model. *Interpolations in latent space for "car" category of the ShapeNet dataset.*

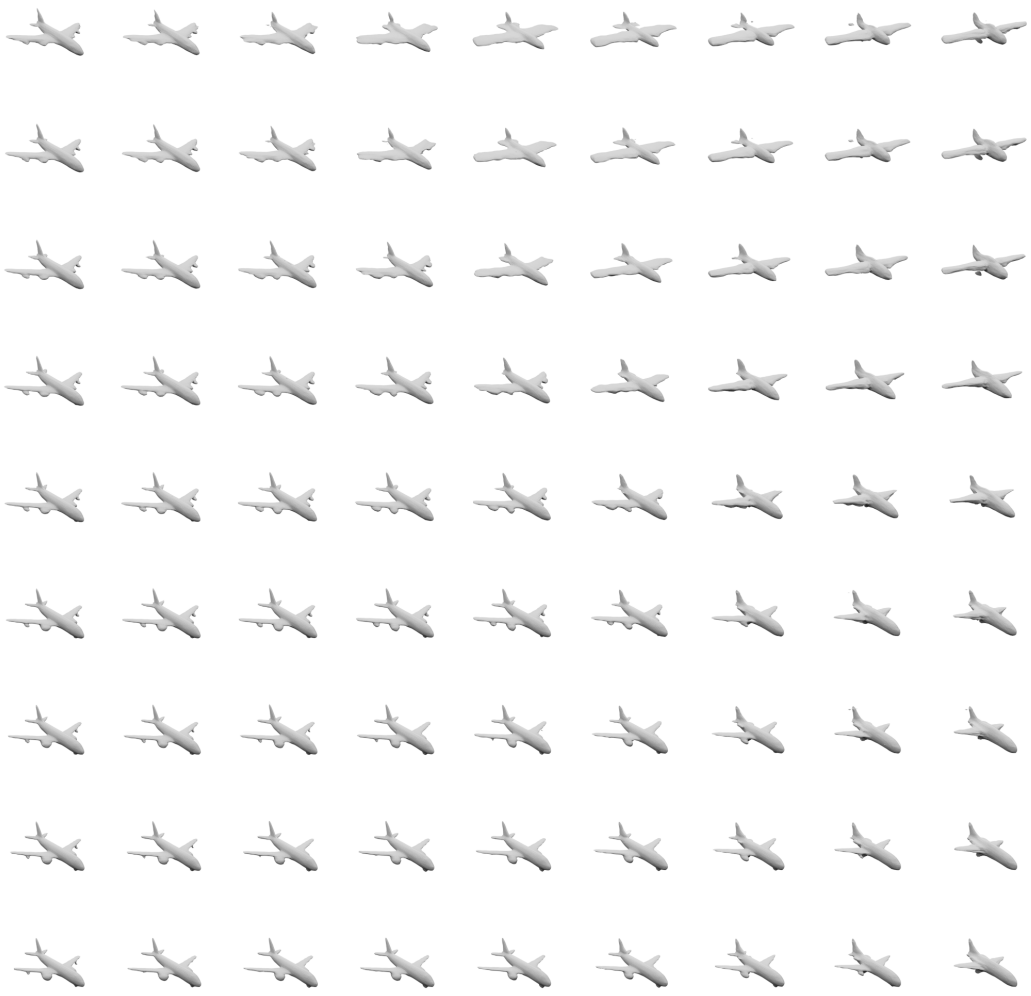


Figure F.16: Unconditional Model. *Interpolations in latent space for "airplane" category of the ShapeNet dataset.*

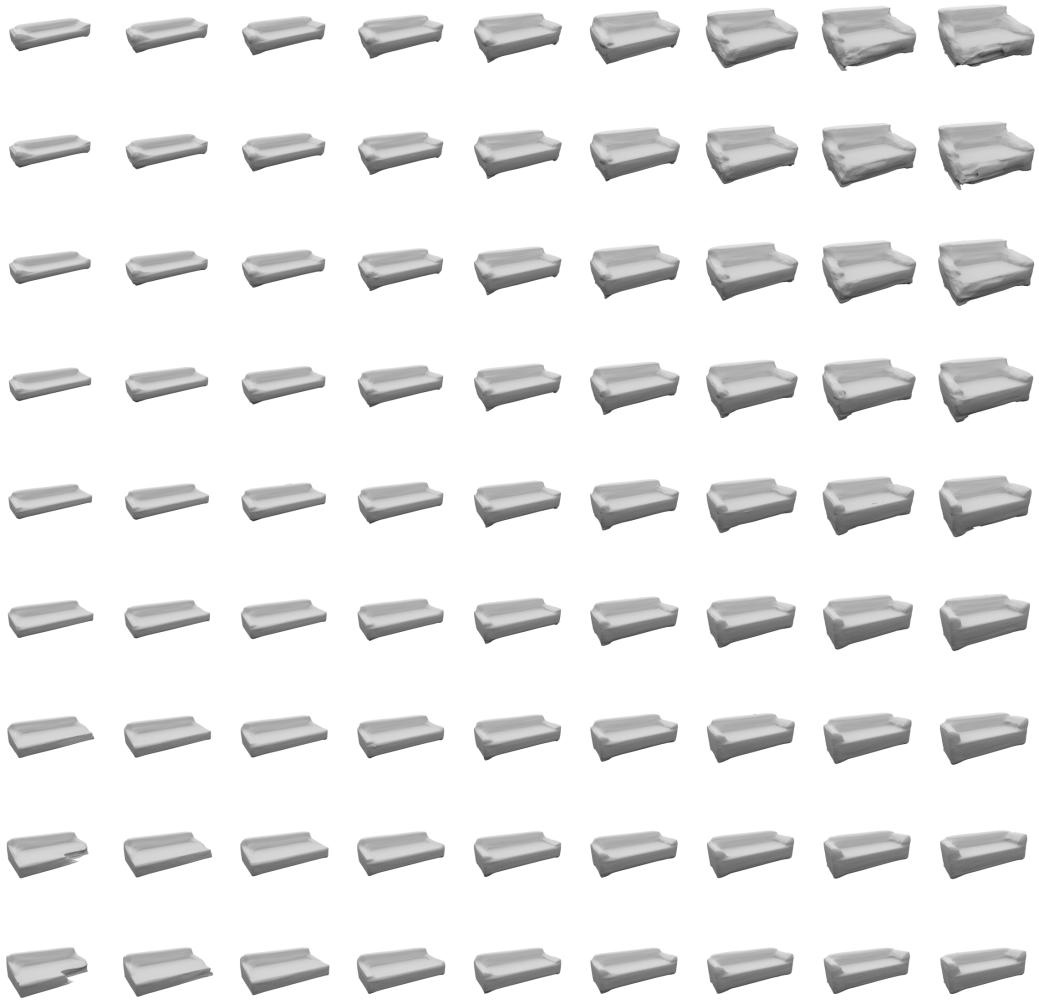


Figure F.17: Unconditional Model. *Interpolations in latent space for "sofa" category of the ShapeNet dataset.*

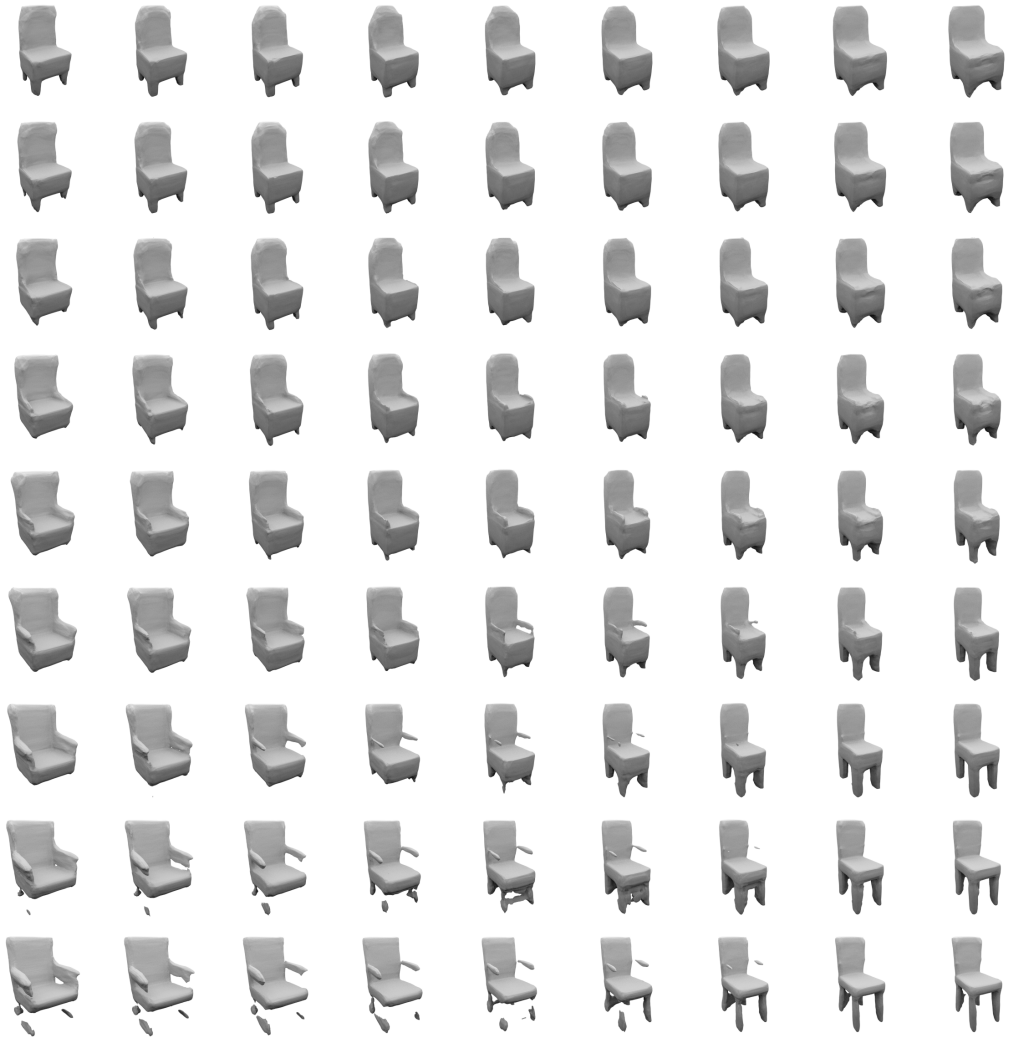


Figure F.18: Unconditional Model. *Interpolations in latent space for "chair" category of the ShapeNet dataset.*

G Credits

Many results in this thesis were developed in collaborative projects [128, 129, 130, 141, 142]. This is particularly true for the work presented in Part II.

The initial idea to represent a mesh as the decision boundary of a classifier (Chapter 11) was suggested to me by Michael Oechsle and Andreas Geiger. Michael Oechsle and Michael Niemeyer helped with the implementation for the experiments in Chapter 11 and with proofreading the paper [130]. More specifically, Michael Oechsle helped to adapt the original DMC implementation [108] in PyTorch 0.3.0 to our framework in PyTorch 1.0. Moreover, Michael Niemeyer reimplemented the Pixel2Mesh [193] baseline and generated the data to create Figure 11.10. Figure 11.1 was mostly designed by Andreas Geiger. Figure 11.4 was designed by Michael Niemeyer and me.

The ideas presented in Chapter 12 were developed jointly by Michael Oechsle, Michael Niemeyer, Andreas Geiger and me. For the Texture Fields project [145], most of the implementation was done by Michael Oechsle. The results in Figure 12.3 and 12.4 were generated by Michael Oechsle. Similarly, Michael Niemeyer did most of the implementation of Occupancy Flow [141] and generated the results in Figure 12.5 and 12.6. Figure 12.1 was created by Michael Oechsle, Andreas Geiger and me. Similarly, Figure 12.2 was designed by Michael Niemeyer and me.

Andreas Geiger was involved in the discussions and proofreading of the paper drafts for all projects [128, 129, 130, 141, 142]. Moreover, Sebastian Nowozin was involved in the discussions and proofreading of three of the five projects [128, 129, 130].

In general, most of the content presented in this thesis is the result of a collaborative effort. Assigning single ideas and implementations to individuals is therefore only a very rough approximation. All projects greatly benefited from the input of all team members and would not have been possible in the same way without this input.

H Publications

- Y. Liao, K. Schwarz, **L. Mescheder**, and A. Geiger. "Towards unsupervised learning of generative models for 3D controllable image synthesis". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2020
- M. Niemeyer, **L. Mescheder**, M. Oechsle, and A. Geiger. "Differentiable volumetric rendering: Learning implicit 3D representations without 3D supervision". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2020
- M. Niemeyer, **L. Mescheder**, M. Oechsle, and A. Geiger. "Occupancy Flow: 4D Reconstruction by Learning Particle Dynamics". In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2019
- M. Oechsle, **L. Mescheder**, M. Niemeyer, T. Strauss, and A. Geiger. "Texture Fields: Learning Texture Representations in Function Space". In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2019
- **L. Mescheder**, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. "Occupancy Networks: Learning 3D Reconstruction in Function Space". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2019
- **L. Mescheder**, A. Geiger, and S. Nowozin. "Which Training Methods for GANs do actually Converge?" In: *Proc. of the International Conf. on Machine learning (ICML)*. 2018
- H. A. Alhaija, S. K. Mustikovela, **L. Mescheder**, A. Geiger, and C. Rother. "Augmented Reality Meets Computer Vision: Efficient Data Generation for Urban Driving Scenes". In: *International Journal of Computer Vision* 126.9 (2018), pp. 961–972
- **L. Mescheder**, S. Nowozin, and A. Geiger. "The Numerics of GANs". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017
- **L. Mescheder**, S. Nowozin, and A. Geiger. "Adversarial Variational Bayes: Unifying Variational Autoencoders and Generative Adversarial Networks". In: *Proc. of the International Conf. on Machine learning (ICML)*. 2017
- H. A. Alhaija, S. K. Mustikovela, **L. Mescheder**, A. Geiger, and C. Rother. "Augmented Reality Meets Deep Learning for Car Instance Segmentation in Urban Scenes". In: *Proc. of the British Machine Vision Conf. (BMVC)*. 2017
- **L. Mescheder**, S. Nowozin, and A. Geiger. *Probabilistic Duality for Parallel Gibbs Sampling without Graph Coloring*. 2016. arXiv: `1611.06684`

Abbreviations

3D-R2N2	3D Recurrent Reconstruction Neural Network
AltGD	Alternating Gradient Descent
BP	Ball Pivoting
CAD	Computer Aided Design
CBN	Conditional Batch Normalization
CelebA	Large-scale CelebFaces Attributes Dataset
CFFI	C Foreign Function Interface
CNN	Convolutional Neural Network
ConOpt	Consensus Optimization
CR	Critical Regularization
CUDA	Compute Unified Device Architecture
DC-GAN	Deep Convolutional GAN
DMC	Deep Marching Cubes
DRAGAN	Deep Regret Analytic Generative Adversarial Networks
FID	Fréchet Inception Distance
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
IoU	Intersection over Union
Leaky-ReLU	Leaky Rectified Linear Unit
LSTM	Long short-term memory
LSUN	Large Scene Understanding Dataset
MISE	Multiresolution IsoSurface Extraction
MSE	Mean Squared Error
OFlow	Occupancy Flow
ONet	Occupancy Network

PSGN	Point Set Generation Network
ReLU	Rectified Linear Unit
ResNet	Residual Network
SGD	Stochastic Gradient Descent
SimGD	Simultaneous Gradient Descent
SPSR	Screened Poisson Surface Reconstruction
TSDF	Truncated Signed Distance Field
VAE	Variational Autoencoder
WGAN-GP	Wasserstein GAN with Gradient Penalties
WGAN	Wasserstein GAN

Glossary

3D Recurrent Reconstruction Neural Network (3D-R2N2) Method for learning-based 3D reconstruction using a 3D-Convolutional Neural Network. Introduced by Choy et al. [26]. 142, 146–150, 197, 201, 214, 215, 218, 219, 231

Accuracy score Evaluation metric for 3D meshes. Defined as the mean distance of points from the predicted mesh to the ground truth mesh. 143, 147, 154, 155, 197, 233, *see also:* Chamfer-L_1 distance

Adam Optimizer introduced by Kingma and Ba [96] that extends RMSProp by introducing an additional momentum term. 141

Alternating Gradient Descent (AltGD) Method for finding equilibrium points in smooth two-player games by applying alternating gradient updates on the cost functions of the two players. 18, 22, 23, 26, 27, 29–31, 35–39, 43, 44, 46, 50–52, 54, 56, 60, 68, 69, 74, 77–79, 84, 87, 89–91, 94–97, 103–106, 121, 186, 191, 205, 208, 209, 231, 234

AtlasNet Single view 3D reconstruction method proposed by Groueix et al. [63] based on deforming a collection of small patches. 142, 146–150, 199, 214–216

Ball Pivoting (BP) Algorithm introduced by Bernardini et al. [11] for reconstructing a mesh from a point cloud. 197, 198, 200, 231

Batch Normalization Technique introduced by Ioffe and Szegedy [75] to accelerate and stabilize training of deep neural networks. 35, 36, 38, 139, 153, 154, 205–207, 209, 231, 234

CelebA-HQ High resolution dataset by Karras, Laine, and Aila [89] based on CelebA. 110–113, 119, 193

Chamfer distance Distance between two sets in a metric space. Defined as the mean squared distance from a point on one mesh to the other mesh and vice versa. 147, 148, 198, 203, *see also:* Chamfer-L_1 distance

Chamfer-L_1 distance Distance between two sets in a metric space. Defined as the mean distance from a point on one mesh to the other mesh and vice versa. Average of Accuracy score and Completeness score. 143, 147, 151, 152, 197, 198, 203, 216, 220, *see also:* Chamfer distance

CIFAR-10 Low-resolution (32×32) dataset of 60,000 images belonging to 10 different classes introduced by Krizhevsky and Hinton [102]. 35, 36, 205, 207

Completeness score Evaluation metric for 3D meshes. Defined as the mean distance of points from the ground truth mesh to the predicted mesh. 143, 147, 154, 155, 197, 233, *see also:* Chamfer-L_1 distance

Compute Unified Device Architecture (CUDA) Parallel computing platform and application programming interface by Nvidia for using the GPU for general purpose computing. 199, 231

Computer Aided Design (CAD) Approach for creating and modifying designs using computers. 198, 231, 238

Conditional Batch Normalization (CBN) Technique to condition a neural network on additional input by making the learned moments in Batch Normalization conditional, used e.g. by Dumoulin, Shlens, and Kudlur [42] and Vries et al. [192]. 139, 141, 153, 154, 231

Consensus Optimization (ConOpt) Method for finding equilibrium points in smooth two-player games when the Jacobian has purely imaginary eigenvalues or eigenvalues with large imaginary part. Introduced by Mescheder, Nowozin, and Geiger [128]. 23, 30–38, 46, 52, 53, 121, 205–209, 231

Convolutional Neural Network (CNN) Type of neural network that uses convolutional layer. 134, 140, 141, 163, 193, 231, 233

Curse of Dimensionality Umbrella term stating that many problems in machine learning become harder in higher dimensions. 127, 129, 234

Curse of Discretization Instance of the Curse of Dimensionality discussed in Chapter 10 that occurs when discretizing function spaces with high-dimensional domain. 127, 129, 131, 136, 157, 235

Deep Convolutional GAN (DC-GAN) GAN architecture introduced by Radford, Metz, and Chintala [160]. One of the first architectures that was able to produce realistic images. 35, 36, 38, 205–209, 231

Deep Marching Cubes (DMC) Technique for learning-based 3D reconstruction that makes the Marching Cubes algorithm differentiable. Introduced by Liao, Donne, and Geiger [108]. 142, 143, 199, 218, 219, 227, 231

Deep Regret Analytic Generative Adversarial Networks (DRAGAN) Type of GAN introduced by [98] that uses a similar regularizer like WGAN-GP. 51, 55, 231

Dirac-GAN Minimal GAN for which neither Simultaneous Gradient Descent nor Alternating Gradient Descent converges. Introduced by Mescheder, Geiger, and Nowozin [129] and described in detail in Chapter 3 and 4. 4, 39–43, 45, 47, 48, 50–56, 59, 60, 62, 63, 65–69, 71, 73, 78, 79, 83, 84, 90, 94, 95, 191, 192

Energy Solution Stable equilibrium for unregularized GAN training where discriminator forms a potential function for data distribution. See Appendix D for details. 71, 109, 189, 191

Fast Quadric Mesh Simplification Mesh simplification algorithm by Garland and Heckbert [52] which approximates surface errors using quadric matrices. Implementation: https://github.com/sp4cerat/Fast-Quadric-Mesh-Simplification 138

Fréchet Inception Distance (FID) Metric proposed by Heusel et al. [68] to evaluate Generative Adversarial Networks. The metric is defined as the Fréchet Distance between two Gaussian distributions that approximate the distributions obtained by applying an inception network to the true data distribution and the generator distribution. Lower is better. 110–114, 210, 231, 235

Function Space Operator Operator introduced in Chapter 10 that defines a bijection between the space of functions $\mathcal{Z} \times \mathcal{Y} \to \mathcal{V}^{\mathcal{S}}$ and the space of functions from $\mathcal{S} \times \mathcal{Z} \times \mathcal{Y} \to \mathcal{V}$. Approximating the latter kind of function is often easier and does not suffer from the Curse of Discretization. 4, 127, 130, 131, 133, 136, 154, 157–159, 163, 165

Generative Adversarial Network (GAN) Generative model introduced by Goodfellow et al. [62] based on training a generator and discriminator in an adversarial fashion. v, vii, xxi, 2–4, 9, 13–18, 22–27, 31, 34, 37, 39–43, 45–47, 50, 53–57, 59, 60, 62, 66, 68, 69, 71, 73, 74, 80, 81, 83–85, 87, 89–91, 93–95, 98, 102, 105–107, 109–113, 115–119, 121, 127, 134, 158, 165, 169, 170, 189, 191, 207, 210–212, 231, 234–236, 238

ImageNet Hierarchical image database. Dataset for ImageNet Large Scale Visual Recognition Challenge. `http://www.image-net.org` 110–113, 140, 141, 147, 193, 195, 196, 210

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) Popular challenge for image classification and object detection [171]. 110, 231, 235, *see also:* ImageNet

Inception Network Convolutional neural network architecture proposed by Szegedy et al. [181]. Used for computing the Inception score and Fréchet Inception Distance. 110, 111

Inception score Score proposed by Salimans et al. [173] to evaluate Generative Adversarial Networks. The score is defined as the Kullback-Leibler divergence between the output of an inception network conditioned on a generated image and the marginal label distribution and applying the exponential function to this value. Higher is better. 35, 36, 110, 111, 113, 114, 205, 210, 235

Indicator divergence Divergence between probability measures p_1 and p_2 which is 0 if $p_1 = p_2$ and ∞ otherwise. 10, 11, 205

Intersection over Union (IoU) Metric to compare 3D meshes by dividing the volume of their intersection by the volume of their union. Score always lies between zero (worst) and one (best). 142–145, 147, 148, 151, 152, 154, 155, 198, 201, 203, 216, 220, 231

KITTI Dataset for autonomous driving introduced by Geiger et al. [53]. `http://www.cvlibs.net/datasets/kitti` 145, 149, 150

Kullback-Leibler divergence Probabilistic divergence between probability distributions. xxiii, 10, 14, 110, 137, 235

Large Scene Understanding Dataset (LSUN) Dataset for scene understanding with different scene categories by Yu et al. [207]. 110–113, 116, 193, 194, 231

Large-scale CelebFaces Attributes Dataset (CelebA) Dataset of celebrity faces with attributes by Liu et al. [110]. 110–113, 115, 193, 194, 231, 233

Leaky Rectified Linear Unit (Leaky-ReLU) Variant of ReLU activation function for neural networks defined by $t \mapsto \max(t, \alpha t)$. for some $\alpha > 0$. 107, 231

Least-Squares-GAN GAN proposed by Mao et al. [121] that uses $\varphi_1(t) = \varphi_2(t) = -(1+t)^2$ as activation functions. 16

Long short-term memory (LSTM) Type of RNN-cell with memory introduced by Hochreiter and Schmidhuber [72] to tackle the vanishing gradient problem. 197, 231

Manifold Hypothesis Hypothesis that most real-world distributions lie in the vicinity of a low dimensional submanifold of the ambient space. Basis for manifold learning and dimensionality reduction techniques, see e.g. Ma and Fu [119]. 107

Marching Cubes Algorithm for extracting an isosurface from a discrete grid of values. Introduced by Lorensen and Cline [117]. 135, 137–139, 148, 234

Mean Squared Error (MSE) Error metric defined as the mean squared difference between two vectors. 68, 69, 231

Multiresolution IsoSurface Extraction (MISE) Method introduced by Mescheder et al. [130] to extract an isosurface from a continuous function by incrementally building an OctTree. 137, 148, 153, 231

Neural Network divergence Probabilistic divergence defined by a neural network discriminator (see Section 1.1.5). The concept was proposed by Arora et al. [7]. 10, 15

Normal Consistency score Measure to assess how consistent the normals of two meshes are. Defined as the mean dot product of the normals on one mesh and the normals at the closest points on the other other mesh and vice versa. 143, 144, 147, 148, 151, 152, 154, 155, 198, 203, 216, 220

Occupancy Flow (OFlow) Representation for time-varying 3D geometry introduced by Niemeyer et al. [141] based on a static Occupancy Network at a $t = 0$ and a time-varying vector field. 158, 159, 161, 163, 227, 231, 238

Occupancy Function Representation of 3D geometry as a function that assigns one to all points inside the object and zero to the points outside. 3, 133, 136, 137, 145, 151, 159

Occupancy Network (ONet) Representation for 3D geometry introduced by Mescheder et al. [130] based on representing 3D geometry as the decision boundary of a deep learning classifier. xxi, 4, 133, 136, 137, 139, 141, 142, 144, 145, 149, 151, 152, 154, 156–159, 163, 165, 198, 217, 231, 237

OctTree Representation of space where voxels of interest are recursively subdivided into 8 subvoxels. 137, 163, 236

Pix3D Dataset of pixel-level aligned image-shape pairs. Introduced by Sun et al. [180]. `http://pix3d.csail.mit.edu` 145, 149–151, 213, 217

Pixel2Mesh Single view 3D reconstruction method proposed by Wang et al. [193] based on deforming a template ellipsoid using local features. 142, 143, 146–149, 198, 203, 214, 215, 227

Point Set Generation Network (PSGN) Technique for learning-based 3D reconstruction that represents geometry as a point cloud. Introduced by Fan, Su, and Guibas [45]. 142, 143, 146–150, 197, 198, 200, 214, 215, 218–220, 232

PointNet Permutation-invariant neural network architecture for point clouds. Proposed by Qi et al. [158]. 140, 141

Python Optimal Transport Package Python library to efficiently estimate the Wasserstein divergence and other quantities based on optimal transport. `http://pot.readthedocs.io` 108

PyTorch Deep learning framework [153] that is based on dynamic computation graphs. `https://pytorch.org/` 32, 112, 141, 142, 147, 199, 227

Rectified Linear Unit (ReLU) Activation function for neural networks defined by $t \mapsto \max(t, 0)$. 34, 71, 111, 139, 197, 198, 232, 236

Residual Network (ResNet) Type of neural network where the input of each block is added to the output which leads to more stable training. Introduced by He et al. [67]. 110, 111, 139–141, 147, 153, 154, 193, 197, 198, 205, 209, 232

RMSProp Optimizer introduced by Hinton, Srivastava, and Swersky [69] that normalizes the gradients by dividing them with a historical average of their norms. 34, 108, 112, 208, 209, 233

Screened Poisson Surface Reconstruction (SPSR) Variant of Poisson Surface Reconstruction [92] which explicitly incorporates an interpolation constraint. Introduced by Kazhdan and Hoppe [93]. 197, 232

ShapeNet Repository of 3D CAD models. Introduced by Chang et al. [22]. xviii, 142, 144, 145, 147–149, 151–153, 216, 217, 220, 222–225

Simultaneous Gradient Descent (SimGD) Method for finding equilibria in smooth two-player games by simultaneously performing gradient descent on the cost functions of the two players. 18, 22, 23, 25–31, 34, 35, 37–39, 42, 43, 47, 50, 52, 54, 56, 60, 68, 69, 74, 77–79, 84, 87, 89–91, 94–96, 98, 99, 101–103, 105, 106, 121, 191, 205, 209, 232, 234

Stanford Online Products Dataset of online products introduced by Oh Song et al. [146] `http://cvgl.stanford.edu/projects/lifted_struct` 145, 149–151

Stochastic Gradient Descent (SGD) Stochastic version of gradient descent where the gradient is estimated using mini-batches. 108, 232

StyleGAN High-resolution Generative Adversarial Network introduced by Karras, Laine, and Aila [89] that conditions the generator architecture on the latent code $z \in \mathcal{Z}$ using Adaptive Instance Normalization. 2

Tensorflow Deep learning framework [1] that is based on static computation graphs. `https://www.tensorflow.org/` 32

Texture Field Continuous representation of texture proposed by Oechsle et al. [145]. 157–159, 227

Truncated Signed Distance Field (TSDF) Function that assigns the signed distance with respect to to an object to every point in space and which is truncated to a certain maximum absolute value. 135, 142, 232

Variational Autoencoder (VAE) Generative model introduced by Kingma and Welling [95] and Rezende, Mohamed, and Wierstra [164] based on maximum-likelihood training with approximate variational inference. 9, 134, 137, 158, 160, 232

Velocity Network Continuous representation of a 3D motion field proposed by Niemeyer et al. [141]. Used in Occupancy Flow. 158, 159

Wasserstein divergence Divergence between probability measures based on optimal transport, which - among others - serves as the inspiration for Wasserstein GANs. Considered, e.g., by Gini [59], Kantorovich and Rubinstein [85], Kantorovich [86], and Vaserstein [190]. 10, 14, 15, 50, 79, 90, 91, 107, 108, 237, 238

Wasserstein GAN (WGAN) Variant of GAN introduced by Arjovsky, Chintala, and Bottou [5] which is inspired by the Wasserstein divergence. Forces the discriminator to be Lipschitz. 42, 46, 50, 51, 53, 91, 92, 232, 238

Wasserstein GAN with Gradient Penalties (WGAN-GP) Variant of WGAN introduced by Gulrajani et al. [64] based on penalizing the discriminator for gradients whose norm is not one. 46, 51–53, 55, 107–110, 112, 113, 192, 232, 234

Bibliography

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. A. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zhang. *TensorFlow: A system for large-scale machine learning*. 2016. arXiv: 1605.08695.

[2] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. J. Guibas. "Learning Representations and Generative Models for 3D Point Clouds". In: *Proc. of the International Conf. on Machine learning (ICML)*. 2018.

[3] H. A. Alhaija, S. K. Mustikovela, L. Mescheder, A. Geiger, and C. Rother. "Augmented Reality Meets Deep Learning for Car Instance Segmentation in Urban Scenes". In: *Proc. of the British Machine Vision Conf. (BMVC)*. 2017.

[4] H. A. Alhaija, S. K. Mustikovela, L. Mescheder, A. Geiger, and C. Rother. "Augmented Reality Meets Computer Vision: Efficient Data Generation for Urban Driving Scenes". In: *International Journal of Computer Vision* 126.9 (2018), pp. 961–972.

[5] M. Arjovsky, S. Chintala, and L. Bottou. "Wasserstein generative adversarial networks". In: *Proc. of the International Conf. on Machine learning (ICML)*. 2017.

[6] M. Arjovsky and L. Bottou. "Towards Principled Methods for Training Generative Adversarial Networks". In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2017.

[7] S. Arora, R. Ge, Y. Liang, T. Ma, and Y. Zhang. "Generalization and Equilibrium in Generative Adversarial Nets (GANs)". In: *Proc. of the International Conf. on Machine learning (ICML)*. 2017.

[8] M. Atzmon, N. Haim, L. Yariv, O. Israelov, H. Maron, and Y. Lipman. "Controlling Neural Level Sets". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[9] D. Balduzzi, S. Racanière, J. Martens, J. N. Foerster, K. Tuyls, and T. Graepel. "The Mechanics of n-Player Differentiable Games". In: *Proc. of the International Conf. on Machine learning (ICML)*. 2018.

[10] S. Barratt and R. Sharma. *A Note on the Inception Score*. 2018. arXiv: 1801.01973.

[11] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. "The ball-pivoting algorithm for surface reconstruction". In: *IEEE Trans. on Visualization and Computer Graphics (VCG)* 5.4 (1999), pp. 349–359.

[12] D. P. Bertsekas. *Nonlinear programming*. Athena scientific Belmont, 1999.

[13] F. Bogo, A. Kanazawa, C. Lassner, P. V. Gehler, J. Romero, and M. J. Black. “Keep It SMPL: Automatic Estimation of 3D Human Pose and Shape from a Single Image”. In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2016.

[14] V. S. Borkar and S. P. Meyn. “The ODE method for convergence of stochastic approximation and reinforcement learning”. In: *SIAM Journal on Control and Optimization* 38.2 (2000), pp. 447–469.

[15] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge Univ. Press, 2004.

[16] L. M. Bregman. “The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming”. In: *USSR computational mathematics and mathematical physics* 7.3 (1967), pp. 200–217.

[17] A. Brock, T. Lim, J. M. Ritchie, and N. Weston. “Generative and Discriminative Voxel Modeling with Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems (NeurIPS) Workshops*. 2016.

[18] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. “Geometric Deep Learning: Going beyond Euclidean data”. In: *Signal Processing Magazine* 34.4 (2017), pp. 18–42.

[19] A. Brock, J. Donahue, and K. Simonyan. “Large Scale GAN Training for High Fidelity Natural Image Synthesis”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2019.

[20] A. Brøndsted and R. T. Rockafellar. “On the subdifferentiability of convex functions”. In: *Proceedings of the American Mathematical Society* 16.4 (1965), pp. 605–611.

[21] F. Calakli and G. Taubin. “SSD: Smooth Signed Distance Surface Reconstruction”. In: *Computer Graphics Forum* 30.7 (2011), pp. 1993–2002.

[22] A. X. Chang, T. A. Funkhouser, L. J. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. *ShapeNet: An Information-Rich 3D Model Repository*. 2015. arXiv: `1512.03012`.

[23] W. Chen, H. Ling, J. Gao, E. Smith, J. Lehtinen, A. Jacobson, and S. Fidler. “Learning to Predict 3D Objects with an Interpolation-based Differentiable Renderer”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[24] Z. Chen and H. Zhang. “Learning Implicit Fields for Generative Shape Modeling”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2019).

[25] Z. Chen and H. Zhang. “Learning Implicit Fields for Generative Shape Modeling”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[26] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. “3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction”. In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2016.

[27] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo. “Stargan: Unified generative adversarial networks for multi-domain image-to-image translation”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[28] Y. Choi, Y. Uh, J. Yoo, and J.-W. Ha. “StarGAN v2: Diverse image synthesis for multiple domains”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[29] D. Cremers, M. Rousson, and R. Deriche. “A review of statistical approaches to level set segmentation: integrating color, texture, motion and shape”. In: *International Journal of Computer Vision* 72.2 (2007), pp. 195–215.

[30] B. Curless and M. Levoy. “A Volumetric Method for Building Complex Models from Range Images”. In: *ACM Trans. on Graphics (SIGGRAPH)*. 1996.

[31] R. Dahl, M. Norouzi, and J. Shlens. “Pixel recursive super resolution”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2017.

[32] A. Dai, C. R. Qi, and M. Nießner. “Shape Completion using 3D-Encoder-Predictor CNNs and Shape Synthesis”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[33] I. Danihelka, B. Lakshminarayanan, B. Uria, D. Wierstra, and P. Dayan. *Comparison of Maximum Likelihood and GAN-based training of Real NVPs*. 2017. arXiv: 1705.05263.

[34] C. Daskalakis, A. Ilyas, V. Syrgkanis, and H. Zeng. “Training GANs with Optimism”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2018.

[35] J. Deng, W. Dong, R. Socher, L.-j. Li, K. Li, and L. Fei-fei. “ImageNet: A large-scale hierarchical image database”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2009.

[36] A. Dervieux and F. Thomasset. “A finite element method for the simulation of a Rayleigh-Taylor instability”. In: *Approximation methods for Navier-Stokes problems*. Springer, 1980, pp. 145–158.

[37] L. Dinh, D. Krueger, and Y. Bengio. “NICE: Non-linear Independent Components Estimation”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2015.

[38] L. Dinh, R. Pascanu, S. Bengio, and Y. Bengio. “Sharp minima can generalize for deep nets”. In: *Proc. of the International Conf. on Machine learning (ICML)*. 2017.

[39] L. Dinh, J. Sohl-Dickstein, and S. Bengio. “Density estimation using Real NVP”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2017.

[40] D. Dowson and B. Landau. “The Fréchet distance between multivariate normal distributions”. In: *Journal of Multivariate Analysis* 12.3 (1982), pp. 450–455.

[41] H. Drucker and Y. Le Cun. “Improving generalization performance using double backpropagation”. In: *IEEE Trans. on Neural Networks* 3.6 (1992), pp. 991–997.

[42] V. Dumoulin, J. Shlens, and M. Kudlur. “A Learned Representation For Artistic Style”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2017.

[43] A. El-Nouby, S. Sharma, H. Schulz, D. Hjelm, L. E. Asri, S. E. Kahou, Y. Bengio, and G. W. Taylor. "Tell, draw, and repeat: Generating and modifying images based on continual linguistic instruction". In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2019.

[44] S. A. Eslami, D. J. Rezende, F. Besse, F. Viola, A. S. Morcos, M. Garnelo, A. Ruderman, A. A. Rusu, I. Danihelka, K. Gregor, et al. "Neural scene representation and rendering". In: *Science* 360.6394 (2018), pp. 1204–1210.

[45] H. Fan, H. Su, and L. J. Guibas. "A Point Set Generation Network for 3D Object Reconstruction from a Single Image". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[46] O. Faugeras and R. Keriven. "Level set methods and the stereo problem". In: *International Conference on Scale-Space Theories in Computer Vision*. 1997.

[47] S. Feizi, C. Suh, F. Xia, and D. Tse. *Understanding GANs: the LQG Setting*. 2017. arXiv: 1710.10793.

[48] M. Fréchet. "Sur les ensembles de fonctions et les opérations linéaires". In: *CR Acad. Sci. Paris* 144 (1907), pp. 1414–1416.

[49] M. Gadelha, S. Maji, and R. Wang. "3D Shape Induction from 2D Views of Multiple Objects". In: *Proc. of the International Conf. on 3D Vision (3DV)*. 2017.

[50] Y. Ganin and V. S. Lempitsky. "Unsupervised Domain Adaptation by Backpropagation". In: *Proc. of the International Conf. on Machine learning (ICML)*. 2015.

[51] M. Garnelo, D. Rosenbaum, C. Maddison, T. Ramalho, D. Saxton, M. Shanahan, Y. W. Teh, D. J. Rezende, and S. M. A. Eslami. "Conditional Neural Processes". In: *Proc. of the International Conf. on Machine learning (ICML)*. 2018.

[52] M. Garland and P. S. Heckbert. "Simplifying surfaces with color and texture using quadric error metrics". In: *Visualization'98. Proceedings*. IEEE. 1998, pp. 263–269.

[53] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. "Vision meets Robotics: The KITTI Dataset". In: *International Journal of Robotics Research (IJRR)* 32.11 (2013), pp. 1231–1237.

[54] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel. "ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness". In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2019.

[55] I. Gemp and S. Mahadevan. *Global Convergence to the Equilibrium of GANs using Variational Inequalities*. 2018. arXiv: 1808.01531.

[56] K. Genova, F. Cole, A. Maschinot, A. Sarna, D. Vlasic, and W. T. Freeman. "Unsupervised Training for 3D Morphable Model Regression". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[57] K. Genova, F. Cole, D. Vlasic, A. Sarna, W. T. Freeman, and T. Funkhouser. "Learning Shape Templates with Structured Implicit Functions". In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2019.

[58] G. Gidel, H. Berard, P. Vincent, and S. Lacoste-Julien. “A Variational Inequality Perspective on Generative Adversarial Nets”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2019.

[59] C. Gini. “Sulla misura della concentrazione e della variabilità dei caratteri”. In: *Atti del Reale Istituto veneto di scienze, lettere ed arti* 73 (1914), pp. 1203–1248.

[60] R. Girdhar, D. F. Fouhey, M. Rodriguez, and A. Gupta. “Learning a Predictable and Generative Vector Representation for Objects”. In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2016.

[61] B. Goldluecke and M. Magnor. “Space-time isosurface evolution for temporally coherent 3D reconstruction”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2004.

[62] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. “Generative Adversarial Nets”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2014.

[63] T. Groueix, M. Fisher, V. G. Kim, B. Russell, and M. Aubry. “AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[64] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. “Improved Training of Wasserstein GANs”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[65] K. Guo, D. Zou, and X. Chen. “3D Mesh Labeling via Deep Convolutional Neural Networks”. In: *ACM Trans. on Graphics (SIGGRAPH)*. 2015.

[66] C. Häne, S. Tulsiani, and J. Malik. “Hierarchical Surface Prediction for 3D Object Reconstruction”. In: *Proc. of the International Conf. on 3D Vision (3DV)*. 2017.

[67] K. He, X. Zhang, S. Ren, and J. Sun. “Deep residual learning for image recognition”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[68] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. “GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[69] G. Hinton, N. Srivastava, and K. Swersky. “Lecture 6a overview of mini-batch gradient descent”. In: *Coursera Lecture slides* (2012). URL: `https://class.coursera.org/neuralnets-2012-001/lecture`.

[70] R. D. Hjelm, A. P. Jacob, A. Trischler, G. Che, K. Cho, and Y. Bengio. “Boundary Seeking GANs”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2018.

[71] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio. “Learning deep representations by mutual information estimation and maximization”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2019.

[72] S. Hochreiter and J. Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9.8 (1997), pp. 1735–1780.

[73] S. Huang, C. Lin, S. Chen, Y. Wu, P. Hsu, and S. Lai. “AugGAN: Cross Domain Adaptation with GAN-Based Data Augmentation”. In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2018.

[74] Z. Huang, T. Li, W. Chen, Y. Zhao, J. Xing, C. LeGendre, L. Luo, C. Ma, and H. Li. “Deep Volumetric Video From Very Sparse Multi-view Performance Capture”. In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2018.

[75] S. Ioffe and C. Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: *Proc. of the International Conf. on Machine learning (ICML)*. 2015.

[76] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. “Image-to-image translation with conditional adversarial networks”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[77] C. L. Jackins and S. L. Tanimoto. “Oct-trees and their use in representing three-dimensional objects”. In: *Computer Graphics and Image Processing* 14.3 (1980), pp. 249–270.

[78] J. L. W. V. Jensen et al. “Sur les fonctions convexes et les inégalités entre les valeurs moyennes”. In: *Acta mathematica* 30 (1906), pp. 175–193.

[79] M. Ji, J. Gall, H. Zheng, Y. Liu, and L. Fang. “SurfaceNet: An End-to-end 3D Neural Network for Multiview Stereopsis”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2017.

[80] H. Jin, D. Cremers, A. J. Yezzi, and S. Soatto. “Shedding light on stereoscopic segmentation”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2004.

[81] L. Juvela, B. Bollepalli, J. Yamagishi, and P. Alku. “Waveform generation for text-to-speech synthesis using pitch-synchronous multi-scale generative adversarial networks”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019.

[82] S. Kakutani. “Concrete representation of abstract (M)-spaces (A characterization of the space of continuous functions)”. In: *Annals of Mathematics* (1941), pp. 994–1024.

[83] A. Kanazawa, M. J. Black, D. W. Jacobs, and J. Malik. “End-to-end Recovery of Human Shape and Pose”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[84] A. Kanazawa, S. Tulsiani, A. A. Efros, and J. Malik. “Learning Category-Specific Mesh Reconstruction from Image Collections”. In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2018.

[85] L. V. Kantorovich and G. S. Rubinstein. “On a space of completely additive functions”. In: *Vestnik Leningrad. Univ* 13.7 (1958), pp. 52–59.

[86] L. V. Kantorovich. “On the translocation of masses”. In: *Dokl. Akad. Nauk. USSR (NS)*. Vol. 37. 1942, pp. 199–201.

[87] A. Kar, C. Häne, and J. Malik. “Learning a Multi-View Stereo Machine”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[88] T. Karras, T. Aila, S. Laine, and J. Lehtinen. “Progressive Growing of GANs for Improved Quality, Stability, and Variation”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2018.

[89] T. Karras, S. Laine, and T. Aila. “A style-based generator architecture for generative adversarial networks”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[90] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila. “Analyzing and improving the image quality of StyleGAN”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[91] H. Kato, Y. Ushiku, and T. Harada. “Neural 3D Mesh Renderer”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[92] M. M. Kazhdan, M. Bolitho, and H. Hoppe. “Poisson surface reconstruction”. In: *Eurographics Symposium on Geometry Processing (SGP)*. 2006.

[93] M. M. Kazhdan and H. Hoppe. “Screened poisson surface reconstruction”. In: *ACM Trans. on Graphics (SIGGRAPH)* 32.3 (2013), p. 29.

[94] H. K. Khalil. “Nonlinear Systems”. In: *Prentice-Hall, New Jersey* 2.5 (1996), pp. 5–1.

[95] D. P. Kingma and M. Welling. “Auto-Encoding Variational Bayes”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2014.

[96] D. P. Kingma and J. Ba. “Adam: A Method for Stochastic Optimization”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2015.

[97] A. Kirillov, Y. Wu, K. He, and R. Girshick. “Pointrend: Image segmentation as rendering”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[98] N. Kodali, J. Abernethy, J. Hays, and Z. Kira. *On Convergence and Stability of GANs*. 2017. arXiv: `1705.07215`.

[99] K. Kolev, T. Brox, and D. Cremers. “Robust variational segmentation of 3D objects from multiple views”. In: *Joint Pattern Recognition Symposium*. 2006.

[100] C. Kong, C.-H. Lin, and S. Lucey. “Using Locally Corresponding CAD Models for Dense 3D Reconstructions From a Single Image”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[101] G. Korpelevich. “Extragradient method for finding saddle points and other problems”. In: *Matekon* 13.4 (1977), pp. 35–49.

[102] A. Krizhevsky and G. Hinton. *Learning multiple layers of features from tiny images*. Tech. rep. University of Toronto, 2009.

[103] T. Kynkäänniemi, T. Karras, S. Laine, J. Lehtinen, and T. Aila. “Improved Precision and Recall Metric for Assessing Generative Models”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[104] L. Ladicky, O. Saurer, S. Jeong, F. Maninchedda, and M. Pollefeys. “From Point Clouds to Mesh Using Regression”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2017.

[105] Y. LeCun, Y. Bengio, and G. Hinton. “Deep learning”. In: *Nature* 521.7553 (2015), pp. 436–444.

[106] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al. “Photo-realistic single image super-resolution using a generative adversarial network”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[107] K. Lee, G. Ros, J. Li, and A. Gaidon. “SPIGAN: Privileged Adversarial Learning from Simulation”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2019.

[108] Y. Liao, S. Donne, and A. Geiger. “Deep Marching Cubes: Learning Explicit Surface Representations”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[109] Y. Liao, K. Schwarz, L. Mescheder, and A. Geiger. “Towards unsupervised learning of generative models for 3D controllable image synthesis”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[110] Z. Liu, P. Luo, X. Wang, and X. Tang. “Deep Learning Face Attributes in the Wild”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2015.

[111] M.-Y. Liu, X. Huang, A. Mallya, T. Karras, T. Aila, J. Lehtinen, and J. Kautz. “Few-shot unsupervised image-to-image translation”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2019.

[112] S. Liu, W. Chen, T. Li, and H. Li. “Soft Rasterizer: Differentiable Rendering for Unsupervised Single-View Mesh Reconstruction”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2019.

[113] S. Liu, S. Saito, W. Chen, and H. Li. “Learning to Infer Implicit Surfaces without 3D Supervision”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[114] S. Lombardi, T. Simon, J. Saragih, G. Schwartz, A. Lehrmann, and Y. Sheikh. “Neural Volumes: Learning Dynamic Renderable Volumes from Images”. In: *ACM Trans. on Graphics (SIGGRAPH)*. 2019.

[115] M. M. Loper and M. J. Black. “OpenDR: An Approximate Differentiable Renderer”. In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2014.

[116] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. “SMPL: A skinned multi-person linear model”. In: *ACM Trans. on Graphics (SIGGRAPH)* 34.6 (2015), pp. 1–16.

[117] W. E. Lorensen and H. E. Cline. “Marching Cubes: A High Resolution 3D Surface Construction Algorithm”. In: *ACM Trans. on Graphics (SIGGRAPH)*. 1987.

[118] G. Louppe, J. Hermans, and K. Cranmer. “Adversarial Variational Optimization of Non-Differentiable Simulators”. In: *Conference on Artificial Intelligence and Statistics (AISTATS)*. 2019.

[119] Y. Ma and Y. Fu. *Manifold learning theory and applications*. CRC press, 2011.

[120] S. Mac Lane. *Categories for the working mathematician*. Vol. 5. Springer Science & Business Media, 2013.

[121] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. P. Smolley. “Least squares generative adversarial networks”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2017.

[122] A. Markoff. “On mean values and exterior densities.” In: *Matematiceskij sbornik* 46.1 (1938), pp. 165–191.

[123] D. Maturana and S. Scherer. “VoxNet: A 3D Convolutional Neural Network for real-time object recognition”. In: *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*. 2015.

[124] D. Meagher. “Geometric modeling using octree encoding”. In: *Computer graphics and image processing* 19.2 (1982), pp. 129–147.

[125] P. Mertikopoulos, B. Lecouat, H. Zenati, C.-S. Foo, V. Chandrasekhar, and G. Piliouras. “Optimistic mirror descent in saddle-point problems: Going the extra (gradient) mile”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2019.

[126] L. Mescheder, S. Nowozin, and A. Geiger. *Probabilistic Duality for Parallel Gibbs Sampling without Graph Coloring*. 2016. arXiv: 1611.06684.

[127] L. Mescheder, S. Nowozin, and A. Geiger. “Adversarial Variational Bayes: Unifying Variational Autoencoders and Generative Adversarial Networks”. In: *Proc. of the International Conf. on Machine learning (ICML)*. 2017.

[128] L. Mescheder, S. Nowozin, and A. Geiger. “The Numerics of GANs”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[129] L. Mescheder, A. Geiger, and S. Nowozin. “Which Training Methods for GANs do actually Converge?” In: *Proc. of the International Conf. on Machine learning (ICML)*. 2018.

[130] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. “Occupancy Networks: Learning 3D Reconstruction in Function Space”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[131] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein. “Unrolled Generative Adversarial Networks”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2017.

[132] M. Michalkiewicz, J. K. Pontes, D. Jack, M. Baktashmotlagh, and A. Eriksson. "Implicit Surface Representations As Layers in Neural Networks". In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2019.

[133] M. Michalkiewicz, J. K. Pontes, D. Jack, M. Baktashmotlagh, and A. P. Eriksson. *Deep Level Sets: Implicit Surface Representations for 3D Shape Inference*. 2019. arXiv: 1901.06802.

[134] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. "Spectral Normalization for Generative Adversarial Networks". In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2018.

[135] T. Miyato and M. Koyama. "cGANs with Projection Discriminator". In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2018.

[136] V. Nagarajan and J. Z. Kolter. "Gradient descent GAN optimization is locally stable". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[137] X. Nguyen, M. J. Wainwright, and M. I. Jordan. "Estimating divergence functionals and the likelihood ratio by convex risk minimization". In: *IEEE Transactions on Information Theory* 56.11 (2010), pp. 5847–5861.

[138] A. Nguyen, J. Clune, Y. Bengio, A. Dosovitskiy, and J. Yosinski. "Plug & Play Generative Networks: Conditional Iterative Generation of Images in Latent Space". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[139] T. Nguyen-Phuoc, C. Li, S. Balaban, and Y. Yang. "RenderNet: A deep convolutional network for differentiable rendering from 3D shapes". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[140] W. Nie and A. Patel. "Towards a Better Understanding and Regularization of GAN Training Dynamics". In: *Proc. of the Conference on Uncertainty in Artificial Intelligence (UAI)*. 2019.

[141] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger. "Occupancy Flow: 4D Reconstruction by Learning Particle Dynamics". In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2019.

[142] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger. "Differentiable volumetric rendering: Learning implicit 3D representations without 3D supervision". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[143] S. Nowozin, B. Cseke, and R. Tomioka. "f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2016.

[144] A. Odena, C. Olah, and J. Shlens. "Conditional Image Synthesis with Auxiliary Classifier GANs". In: *Proc. of the International Conf. on Machine learning (ICML)*. 2017.

[145] M. Oechsle, L. Mescheder, M. Niemeyer, T. Strauss, and A. Geiger. "Texture Fields: Learning Texture Representations in Function Space". In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2019.

[146] H. Oh Song, Y. Xiang, S. Jegelka, and S. Savarese. “Deep metric learning via lifted structured feature embedding”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[147] A. van den Oord, N. Kalchbrenner, L. Espeholt, K. Kavukcuoglu, O. Vinyals, and A. Graves. “Conditional Image Generation with PixelCNN Decoders”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2016.

[148] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. “Pixel Recurrent Neural Networks”. In: *Proc. of the International Conf. on Machine learning (ICML)*. 2016.

[149] S. Osher and J. A. Sethian. “Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations”. In: *Journal of Computational Physics* 79.1 (1988), pp. 12–49.

[150] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. “DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[151] D. Paschalidou, A. O. Ulusoy, C. Schmitt, L. van Gool, and A. Geiger. “RayNet: Learning Volumetric 3D Reconstruction with Ray Potentials”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[152] D. Paschalidou, A. O. Ulusoy, and A. Geiger. “Superquadrics Revisited: Learning 3D Shape Parsing beyond Cuboids”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[153] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[154] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. “Context encoders: Feature learning by inpainting”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[155] X. B. Peng, A. Kanazawa, S. Toyer, P. Abbeel, and S. Levine. “Variational Discriminator Bottleneck: Improving Imitation Learning, Inverse RL, and GANs by Constraining Information Flow”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2019.

[156] J.-P. Pons, R. Keriven, and O. Faugeras. “Modelling dynamic scenes by registering multi-view image sequences”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2005.

[157] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. Guibas. “Volumetric and Multi-View CNNs for Object Classification on 3D Data”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[158] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. "PointNet: Deep learning on point sets for 3D classification and segmentation". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[159] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[160] A. Radford, L. Metz, and S. Chintala. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2016.

[161] A. Rakhlin and K. Sridharan. "Online Learning with Predictable Sequences". In: *Proc. of the Conference on Computational Learning Theory (COLT)*. 2013.

[162] A. Ranjan, T. Bolkart, S. Sanyal, and M. J. Black. "Generating 3D faces using Convolutional Mesh Autoencoders". In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2018.

[163] L. J. Ratliff, S. Burden, and S. S. Sastry. "Characterization and computation of local Nash equilibria in continuous games". In: *Annual Allerton Conference on Communication, Control, and Computing*. 2013.

[164] D. J. Rezende, S. Mohamed, and D. Wierstra. "Stochastic Backpropagation and Approximate Inference in Deep Generative Models". In: *Proc. of the International Conf. on Machine learning (ICML)*. 2014.

[165] D. J. Rezende, S. M. A. Eslami, S. Mohamed, P. Battaglia, M. Jaderberg, and N. Heess. "Unsupervised Learning of 3D Structure from Images". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2016.

[166] G. Riegler, A. O. Ulusoy, H. Bischof, and A. Geiger. "OctNetFusion: Learning Depth Fusion from Data". In: *Proc. of the International Conf. on 3D Vision (3DV)*. 2017.

[167] H. Robbins and S. Monro. "A stochastic approximation method". In: *The Annals of Mathematical Statistics* (1951), pp. 400–407.

[168] O. Ronneberger, P. Fischer, and T. Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. 2015.

[169] K. Roth, A. Lucchi, S. Nowozin, and T. Hofmann. "Stabilizing Training of Generative Adversarial Networks through Regularization". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[170] W. Rudin. *Real and complex analysis*. Tata McGraw-hill education, 2006.

[171] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision* 115.3 (2015), pp. 211–252.

[172] S. Saito, Z. Huang, R. Natsume, S. Morishima, A. Kanazawa, and H. Li. “PIFu: Pixel-Aligned Implicit Function for High-Resolution Clothed Human Digitization”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2019.

[173] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. “Improved Techniques for Training GANs”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2016.

[174] V. Sitzmann, M. Zollhöfer, and G. Wetzstein. “Scene Representation Networks: Continuous 3D-Structure-Aware Neural Scene Representations”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[175] E. Smith, S. Fujimoto, and D. Meger. “Multi-View Silhouette and Depth Decomposition for High Resolution 3D Object Representation”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[176] S. Song and J. Xiao. “Deep Sliding Shapes for Amodal 3D Object Detection in RGB-D Images”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[177] C. K. Sønderby, J. Caballero, L. Theis, W. Shi, and F. Huszár. “Amortised MAP Inference for Image Super-resolution”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2017.

[178] J. C. Spall. *Introduction to stochastic search and optimization: estimation, simulation, and control*. Vol. 65. John Wiley & Sons, 2005.

[179] D. Stutz and A. Geiger. “Learning 3D Shape Completion from Laser Scan Data with Weak Supervision”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[180] X. Sun, J. Wu, X. Zhang, Z. Zhang, C. Zhang, T. Xue, J. B. Tenenbaum, and W. T. Freeman. “Pix3D: Dataset and Methods for Single-Image 3D Shape Modeling”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[181] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. “Rethinking the inception architecture for computer vision”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[182] R. Szeliski. “Rapid octree construction from image sequences”. In: *CVGIP: Image understanding* 58.1 (1993), pp. 23–32.

[183] O. Tange. “GNU Parallel - The Command-Line Power Tool”. In: *;login: The USENIX Magazine* 36.1 (2011), pp. 42–47. URL: `http://www.gnu.org/s/parallel`.

[184] M. Tatarchenko, A. Dosovitskiy, and T. Brox. “Octree Generating Networks: Efficient Convolutional Architectures for High-resolution 3D Outputs”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2017.

[185] H. Thanh-Tung, T. Tran, and S. Venkatesh. “Improving Generalization and Stability of Generative Adversarial Networks”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2019.

[186] L. Theis, A. van den Oord, and M. Bethge. “A note on the evaluation of generative models”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2016.

[187] F. Tisseur and K. Meerbergen. “The quadratic eigenvalue problem”. In: *SIAM review* 43.2 (2001), pp. 235–286.

[188] S. Tulsiani, T. Zhou, A. A. Efros, and J. Malik. “Multi-view supervision for single-view reconstruction via differentiable ray consistency”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[189] A. O. Ulusoy, A. Geiger, and M. J. Black. “Towards Probabilistic Volumetric Reconstruction using Ray Potentials”. In: *Proc. of the International Conf. on 3D Vision (3DV)*. 2015.

[190] L. N. Vaserstein. “Markov processes over denumerable products of spaces, describing large systems of automata”. In: *Problemy Peredachi Informatsii* 5.3 (1969), pp. 64–72.

[191] C. Villani. *Optimal transport: old and new*. Vol. 338. Springer Science & Business Media, 2008.

[192] H. de Vries, F. Strub, J. Mary, H. Larochelle, O. Pietquin, and A. C. Courville. “Modulating early visual processing by language”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[193] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang. “Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images”. In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2018.

[194] P. Wang, Y. Gan, P. Shui, F. Yu, Y. Zhang, S. Chen, and Z. Sun. “3D shape segmentation via shape fully convolutional networks”. In: *Computers & Graphics* 70 (2018), pp. 128–139.

[195] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro. “High-Resolution Image Synthesis and Semantic Manipulation With Conditional GANs”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[196] W. Wang, X. Qiangeng, D. Ceylan, R. Mech, and U. Neumann. “DISN: Deep Implicit Surface Network for High-quality Single-view 3D Reconstruction”. In: *Advances in Neural Information Processing Systems (NeurIPS)* (2019).

[197] K.-Y. Wong and R. Cipolla. “Structure and motion from silhouettes”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2001.

[198] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. “3D ShapeNets: A deep representation for volumetric shapes”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2015.

[199] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum. “Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2016.

[200] J. Wu, Y. Wang, T. Xue, X. Sun, B. Freeman, and J. Tenenbaum. “MarrNet: 3D shape reconstruction via 2.5D sketches”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[201] J. Wu, C. Zhang, X. Zhang, Z. Zhang, W. T. Freeman, and J. B. Tenenbaum. “Learning Shape Priors for Single-View 3D Completion and Reconstruction”. In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2018.

[202] A. K. Yadav, S. Shah, Z. Xu, D. W. Jacobs, and T. Goldstein. “Stabilizing Adversarial Nets with Prediction Methods”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2018.

[203] C. Yang, X. Lu, Z. Lin, E. Shechtman, O. Wang, and H. Li. “High-resolution image inpainting using multi-scale neural patch synthesis”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[204] Y. Yazici, C. Foo, S. Winkler, K. Yap, G. Piliouras, and V. Chandrasekhar. “The Unusual Effectiveness of Averaging in GAN Training”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2019.

[205] R. A. Yeh, C. Chen, T. Yian Lim, A. G. Schwing, M. Hasegawa-Johnson, and M. N. Do. “Semantic image inpainting with deep generative models”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[206] A. Yezzi and S. Soatto. “Stereoscopic segmentation”. In: *International Journal of Computer Vision* 53.1 (2003), pp. 31–43.

[207] F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao. *LSUN: Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop*. 2015. arXiv: 1506.03365.

[208] X. Zhang, Z. Zhang, C. Zhang, J. B. Tenenbaum, W. T. Freeman, and J. Wu. “Learning to Reconstruct Shapes from Unseen Classes”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[209] H. Zhang, I. J. Goodfellow, D. N. Metaxas, and A. Odena. “Self-Attention Generative Adversarial Networks”. In: *Proc. of the International Conf. on Machine learning (ICML)*. 2019.

[210] Z. Zheng, X. Yang, Z. Yu, L. Zheng, Y. Yang, and J. Kautz. “Joint discriminative and generative learning for person re-identification”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[211] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. “Unpaired image-to-image translation using cycle-consistent adversarial networks”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2017.

[212] J. Zienkiewicz, A. J. Davison, and S. Leutenegger. “Real-time height map fusion using differentiable rendering”. In: *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*. 2016.