HARVARD UNIVERSITY

UNDERGRADUATE THESIS

Lipschitz Constraints in Wasserstein Generative Adversarial Networks

---

Author:
Kyle SARGENT

Supervisor:
Dr. Alexander RUSH

A thesis submitted in partial fulfillment of the requirements for the degree of Bachelor of Arts with Honors

Department of Computer Science
Department of Mathematics

Acknowledgements

Thanks to Professor Rush for great advice on doing research and bearing with me through the roughest of rough drafts. I also want to thank Tasho Kaletha, Matthew Moran and Julienne Au for helping me develop my passion for math and being incredible teachers.

This thesis is dedicated to Mom and Dad.

Contents

Chapter 1

Introduction

1.1 Generative Models

When working with a collection of data, it is natural to ask how the data are distributed and according to what process they are generated. A "generative model" is some method by which a representation $\mathbb{P}_g$ of the real data generating distribution $\mathbb{P}_r$ is learned. Generative modeling can be applied to produce graphics for video games, analyze latent structure of language, find adversarial examples to make classifiers more robust, and train reinforcement learning agents. With the advent of deep learning, generative models have become richer and more expressive — and, in some cases, able to generate images or text indiscernible from the real thing. As such, building powerful and interpretable generative models is of tremendous interest for researchers in the machine learning community. We will concern ourselves here with the task of generating images, and will discuss a particularly successful generative model for doing so — a Generative Adversarial Network (GAN).

Compared to other generative models, GANs remain relatively opaque. To get an idea of just how opaque, we make use of the taxonomy of generative models in Goodfellow (2017). In estimating a data distribution, they noted that generative models use either an explicit or implicit density, and that explicit densities could be tractable (i.e. directly optimizable) or intractable (requiring the use of variational or other methods). There are models with explicit and tractable densities such as the Neural Autoregressive Density Estimator of Larochelle and Murray (2011), models with explicit, intractable densities such as the Variational Autoencoder of Kingma and Welling (2014), and finally, GANs: lacking even an explicit (much less tractable) density.

Nevertheless, when it comes to producing the most vivid, convincing images from a given distribution, the other methods mentioned above have simply fallen by the wayside. Never mind their elegant probabilistic formulations or tractable likelihoods, they simply don't stack up to the current GAN state-of-the-art (see figure 1.1).

1.2 Issues with GANs

When GAN training goes well, it can produce spectacular results. But GAN training is highly unstable. GANs can suffer from oscillations and explosions

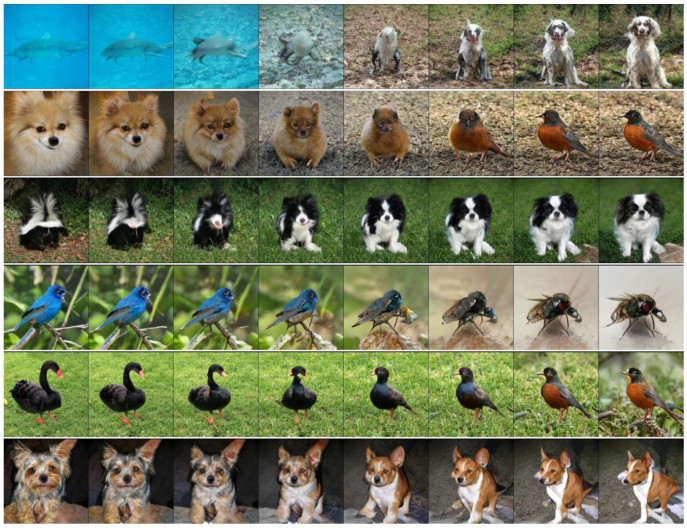

FIGURE 1.1: Samples from a recent state-of-the-art GAN Brock et al. (2018)

of the spectral norms of their weight layers during training. GANs are also prone to something called mode collapse, where the model fails to capture a mode or modes of the target distribution. One must also take care when training GANs to avoid a vanishing gradient problem: if the discriminator gets too good, too fast, the manifold on which the generated distribution lies will be classified as fake with uniformly high probability, making it impossible to learn a good direction for improvement. Indeed, many techniques for stabilizing GAN training simply amount to methods for confusing or impairing the discriminator network; for example, via the addition of random noise to training samples as in Mescheder (2018), or by re-labeling some small fraction of real examples as fake and vice-versa. Frustratingly, GANs are so opaque that most solutions to these training issues feel like ad-hoc heuristics rather than theoretically grounded modifications.

1.3 The Wasserstein GAN

Recently, the Wasserstein GAN was proposed in Arjovsky et al. (2017). The paper was a considerable step forward in the GAN field, both theoretically and practically. First, it gave some of the optimization issues in vanilla GANs a rigorous, mathematical explanation, and re-framed one's choice of GAN loss function in terms of a the loss-space topology it induced. Then, by taking this theory to its conclusion, it developed a state-of-the-art model trained on a novel loss function which demonstrated improved sample variety and training stability, and eliminated the need for careful coordination of the generator and discriminator. The mode collapse and vanishing gradient issues which had plagued GANs for years were theoretically diagnosed and convincingly addressed, though not solved outright.

The problem is that training a WGAN introduces an additional constraint to the usual GAN setup: the discriminator network must be Lipschitz continuous. The original WGAN paper chose a simple and un-optimized method for Lipschitz constraint enforcement in its discriminator. It left the question of improving this method to future work. Since then, a flurry of recent papers, namely Gulrajani et al. (2017), Miyato et al. (2018), Yuichi Yoshida (2017), and Mescheder (2018), have developed novel techniques for enforcing Lipschitz continuity. But there remains room for improvement.

1.4 The Problem and Our Work

How can we enforce Lipschitz continuity in the discriminator of a WGAN? Developing new and better methods for enforcing this constraint has proven tremendously effective in advancing the current GAN state-of-the-art.

We begin by developing a theory. To understand why Lipschitz continuity is so key, we give an extensive treatment of the topology and probability theory which grounds the Wasserstein GAN. Then, we prove some sufficient conditions for Lipschitz continuity in neural networks. Supported by this theory, we examine subsequent versions of the WGAN which have found highly optimized ways to enforce a Lipschitz constraint in the discriminator, and prove their correctness where possible.

We also conduct an empirical investigation. We have replicated the model and results of Miyato et al. (2018), currently the most successful method for WGAN Lipschitz constraint enforcement, in an open-source, modular and extendable fashion in PyTorch.[1]. We use our implementation to conduct extensive analysis of networks trained with various current methods of Lipschitz constraint enforcement and analyze the theoretical implications of the results.

Finally, motivated by these analyses, and with the intent to address the shortcomings of current methods, we experiment with some new algorithms for Lipschitz constraint enforcement, and for improving the properties of networks which are constrained to be Lipschitz.

[1] Code is available at `https://github.com/ksagit/sn_gan_pytorch`

Chapter 2

Background

2.1 Generative Adversarial Nets

2.1.1 The Standard Model

Generative Adversarial Networks (GANs) were introduced in Goodfellow et al. (2014) as a technique for building a generative model of a continuous distribution given only a collection of ground-truth samples. Two neural networks, a generator and discriminator are trained adversarially: the generator attempts to generate samples that look enough like the real data to fool the discriminator, while the discriminator tries to tell the difference between real and generated examples. The generator and discriminator are alternately trained until the generator can produce samples which approximate the data distribution.

A GAN consists of

1. The generator $g_\theta : Z \to X$, a neural network parametrized by θ which maps noise sampled from $\mathbb{P}_z$ (typically Gaussian or uniform) from a low-dimensional latent space $Z = \mathbb{R}^d$ to the high-dimensional observation space $X = \mathbb{R}^n$.

2. The discriminator $d_\omega : X \to [0, 1]$, a neural network parametrized by ω which maps elements of the observation space to the probability that the given observation was real (1) or fake (0)

The generator induces a distribution $\mathbb{P}_g$ over X. We say $g_\theta(z) \sim \mathbb{P}_g$ where $z \sim \mathbb{P}_z$. We refer to the underlying, true distribution of the data as $\mathbb{P}_d$.

The generator and discriminator are alternately trained on minibatches of real and fake data, according to an adversarial objective:

$$\min_G \max_D \mathbb{E}_{x \sim \mathbb{P}_x} \log(d_\omega(x)) + \mathbb{E}_{z \sim \mathbb{P}_z}[\log(1 - d_\omega(g_\theta(z)))] \tag{2.1}$$

In practice, better gradients for the generator are obtained by instead training it to maximize

$$\max_G \mathbb{E}_{z \sim \mathbb{P}_z}[\log(d_\omega(g_\theta(z)))] \tag{2.2}$$

Some modifications to the original GAN model have focused on changing the statistical distance or divergence between $\mathbb{P}_g$ and $\mathbb{P}_d$ that is being implicitly optimized (for example, in Arjovsky et al. (2017) and Nowozin et al.

(2016)) or changing up the model architecture to allow for more expressivity. GAN architectures are nowadays quite varied; while their most basic formulation uses simple MLP architectures in the generator and discriminator, modern GANs often have convolutional and deconvolutional layers in their discriminator and generator respectively, as in Radford et al. (2015). Zhang et al. (2018) trained GANs which performed self-attention, while Mirza and Osindero (2014) developed a GAN to generate class-conditional samples.

2.1.2 Problems with Generative Adversarial Nets

Recall the basic setup of generative modeling: we have some true data distribution $\mathbb{P}_r$ and wish to approximate it by some distribution $\mathbb{P}_g$.

In proposition 2 of Goodfellow et al. (2014), the authors prove that in the basic GAN model, training the adversarial objective guarantees the convergence of the model distribution to the true data distribution in Jensen-Shannon divergence, which we denote JSD. In other words,

$$\text{JSD}(\mathbb{P}_{g_{\theta_i}}, \mathbb{P}_r) \overset{i \to \infty}{\to} 0$$

Unfortunately, proposition 2 provides few practical comforts due to several strong assumptions. First, it was in the setting of directly optimizing the density g, not the parameters θ_g of a neural network model. This complication introduces multiple issues: different combinations of parameters θ_g can define the same model density p_g, resulting in a non-convex optimization space. Moreover, not all possible data densities can be achieved or even closely approximated with any set of parameters θ_g, given a fixed network architecture — referred to as the issue of a model's capacity. The proof also neglects vanishing gradient issues. When gradient updates become vanishingly small, convergence of training cannot be guaranteed on a reasonable time horizon. Vanishing gradients can occur when a discriminator learns to perfectly separate the observations and then generated data, a known issue in GAN training.

The Jensen-Shannon divergence is a distance metric between probability distributions. In the same way that Euclidean space has multiple valid distance metrics, like L_1 or L_2 distance, there are multiple types of distances we can compute between different distributions over the observation space X. It happens that the Jensen-Shannon divergence, one such distance, has several properties which make it a poor choice for a GAN objective, but we do not yet have the tools to explain why. We will continue motivated by the question of how to choose a better distance to optimize between $\mathbb{P}_d$ and $\mathbb{P}_{g_\theta}$

2.2 Distances and Topologies

When attempting to train a generative model, it helps if we can think about it as minimizing $d(\mathbb{P}_{g_\theta}, \mathbb{P}_r)$ where d is a distance metric. We saw this above with the Jensen-Shannon divergence and the original GAN formulation.

Let us think about what performing backpropagation in the original GAN model means. We are implicitly minimizing the following loss function on our parameters:

$$\mathcal{L}(\theta) = \text{JSD}(\mathbb{P}_{g_\theta}, \mathbb{P}_d)$$

When this loss function is not continuous, our model may not be able to learn a good direction for improvement via backpropagation. Thus, we should think carefully about when this loss function is likely to be continuous. To do so, we have to introduce a notion of topology.

2.2.1 Mathematical Foundations

Intuitively, a topology over a set X is a way of specifying which sequences converge and which sequences do not. For our purposes, we need only consider topologies which are induced by a distance metric; in practice, there exist pathological topologies which cannot be created by any metric. From now on, we use the terms metric topology and topology interchangeably. Let us establish some basic terminology.

Definition 2.2.2: Let X have topology induced by a distance metric d. An open set is a subset $U \subseteq X$ if for every $x \in U$, there exists an $\varepsilon > 0$ such that for every $x' \in X$, $d(x', x) < \varepsilon \implies x' \in U$.

This definition basically means that every point in an open set has a small ball around it totally contained in the open set.

Definition 2.2.3: Let $f : X \to Y$. We give the following equivalent definitions.

1. f is continuous if, for every set $U \subseteq Y$ which is open in Y, $f^{-1}(U)$ is open in X.

2. f is continuous if, for every sequence $\{x_i\}_{i=1}^{\infty}$ converging to x in X, the sequence $f(x_i)$ converges to $f(x)$ in Y.

We sometimes refer to a given topology as τ, the set of all open sets in the topology. Given two topologies τ_1 and τ_2 on X, we say that τ_1 is stronger than τ_2 if $\tau_2 \subseteq \tau_1$, and strictly stronger if $\tau_2 \subsetneq \tau_1$.

We should note the following:

Lemma 2.2.4: Let d_1, d_2 be distance metrics on a space X inducing topologies τ_1, τ_2. Suppose for any x and sequence $\{x_n\}_{n=0}^{\infty}$ of points in X, we have

$$d_1(x_n, x) \overset{n\to\infty}{\to} 0 \implies d_2(x_n, x) \overset{n\to\infty}{\to} 0$$

Then $\tau_2 \subseteq \tau_1$, i.e. τ_1 is stronger than τ_2 as a topology.

Proof: See appendix.

The last lemma helps us establish some intuition. For a topology induced by some distance metric d, if the topology is very strong, that means the

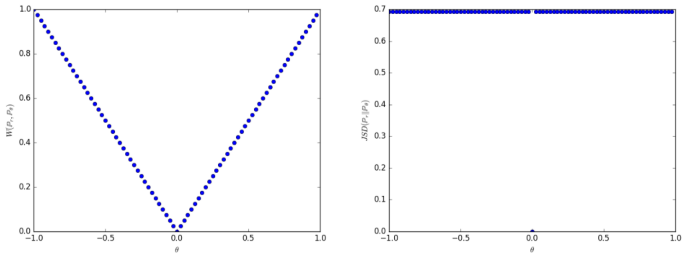

FIGURE 2.1: Wasserstein (left) vs Jensen-Shannon (right) distance between $\mathbb{P}_1, \mathbb{P}_\theta$ as a function of θ

distance that induced it is relatively discontinuous. We could have a point x in the space and a sequence $x_1, x_2, ...$ which converges to x relative to some other metric d', but which does not converge to x relative to d. Here are some examples:

Example 1: The most trivial example of a topology is the discrete topology. Suppose our distance metric is $d(x, y) = \mathbb{I}(x = y)$. That is, any two non-identical points have distance 1. This induces an extremely strong topology, because every single set is open.

Example 2: Consider the familiar L_2 distance on $\mathbb{R}^d$. This induces the so-called "standard topology" in Euclidean space. It is weaker than the discrete topology. The sequence $x_n := (\frac{1}{n}, ..., \frac{1}{n})$ converges to 0 in L_2 distance as $n \to \infty$, but not with respect to the discrete metric: $d(x_n, 0) = 1$ always.

Example 3: The following illustrative example comes from Arjovsky et al. (2017). Consider two distributions: $\mathbb{P}_0$ is a distribution over $\mathbb{R}^2$ given by $(0, Z)$ where Z is a uniform on $[0, 1]$. Define $\mathbb{P}_\theta$ as (θ, Z) where θ is some fixed parameter.

Now consider figure 2.1. Here is why topology and continuity matters: we can see that training $\mathbb{P}_\theta$ to approximate $\mathbb{P}_0$ under the Jensen-Shannon loss is impossible, as there's no gradient to descend. By contrast, a different distance metric (Wasserstein) provides a well-behaved, descendable gradient across the whole space of possible θ.

What this difference corresponds to is the relative strength of the topology induced by Jensen-Shannon. A distance which induces a weak topology, such as Wasserstein, is preferred, because it is easier for sequences to converge. We can formally verify this intuition with a lemma.

Let Θ be our space of model parameters and $\mathbb{P}(X)$ be the space of probability distributions over X. Let $\phi : \Theta \to \mathbb{P}(X)$ be a parametrization of the generator g, where we denote $\phi(\theta) = \mathbb{P}_{g_\theta}$. For a given distance d, consider the loss function $\mathcal{L}_d(\theta) \mapsto d(\mathbb{P}_{g_\theta}, \mathbb{P}_d)$. We have the following lemma:

Lemma 2.2.5: Suppose $\mathbb{P}(X)$ has the topology induced by d. Then the loss function $\mathcal{L}_d : \Theta \to \mathbb{R}_{\geq 0}$ is continuous if ϕ is continuous,

Proof: Let θ_i be a sequence converging to θ in the topology of Θ. By the triangle inequality, we have

$$d(\mathbb{P}_\theta, \mathbb{P}_d) - d(\mathbb{P}_\theta, \mathbb{P}_{\theta_i}) \leq d(\mathbb{P}_{\theta_i}, \mathbb{P}_d) \leq d(\mathbb{P}_\theta, \mathbb{P}_d) + d(\mathbb{P}_\theta, \mathbb{P}_{\theta_i})$$

Fix $\varepsilon > 0$. Since the parametrization ϕ is continuous, there exists N such that for $i \geq N$, $d(\mathbb{P}_{g_{\theta_i}}, \mathbb{P}_d) < \varepsilon/2$, completing the proof.

With these lemmas in hand, we can make a formal argument for why a weak topology is preferred. Since we are backpropagating through the loss of our parametrized model, we want the loss function $\mathcal{L}$ to be continuous. Via lemma 2.2.5, continuity of the loss function on parameters is implied by continuity of the parametrization $\theta \mapsto \mathbb{P}_\theta$. By definition 2.2.3, the parametrization ϕ is continuous only if $\phi^{-1}(U)$ is open for every open set in the space of distributions over X. But a weak topology contains fewer open sets, making this condition less restrictive, i.e. making the model parametrization more likely to be continuous.

2.2.6 A Better Statistical Distance

We have established that we should be looking for a distance metric that induces a weaker topology over the space of probability distributions over X, because that will make our model more amenable to gradient descent.

To define the Wasserstein distance, we need the following notion.

Definition 2.2.7: Let $\mathbb{P}_1$ and $\mathbb{P}_2$ be two distributions over spaces X_1, X_2. We may construct the product space $X_1 \times X_2$, and define a coupling to be a measure on this space whose marginals are the laws $\mathbb{P}_1$ and $\mathbb{P}_2$. We denote the set of couplings as $\Pi(\mathbb{P}_1, \mathbb{P}_2)$.

We can now define three distances and one divergence — the KL divergence measures differences between probability distributions, but it is not symmetric in its inputs and thus not a distance.

Definition 2.2.8: The Wasserstein (or Earth-Mover) Distance between $\mathbb{P}_1, \mathbb{P}_2$ is

$$\mathcal{W}(\mathbb{P}_\mathbb{1}, \mathbb{P}_2) = \inf_{\pi \in \Pi(\mathbb{P}_1, \mathbb{P}_2)} \mathbb{E}_{(x_1, x_2) \sim \pi} |x_1 - x_2|$$

Definition 2.2.9: The Total Variation Distance between $\mathbb{P}_1, \mathbb{P}_2$ is

$$\mathrm{TV}(\mathbb{P}_\mathbb{1}, \mathbb{P}_2) = \inf_{\pi \in \Pi(X_1, X_2)} \mathbb{E}_{(x_1, x_2) \sim \pi} \mathbb{1}[x_1 \neq x_2]$$

Definition 2.2.10: The Kullback-Leibler divergence between $\mathbb{P}_1, \mathbb{P}_2$ is

$$\mathrm{KL}(\mathbb{P}_1, \mathbb{P}_2) = \int_{x \in \mathcal{X}} P_1(x) \log\left(\frac{P_1(x)}{P_2(x)}\right) dx$$

Definition 2.2.11: The Jensen-Shannon divergence between $\mathbb{P}_1, \mathbb{P}_2$ is

$$\mathrm{KL}\left(\mathbb{P}_1, \frac{\mathbb{P}_1 + \mathbb{P}_2}{2}\right) + \mathrm{KL}\left(\mathbb{P}_2, \frac{\mathbb{P}_1 + \mathbb{P}_2}{2}\right)$$

These distances are among the most commonly used in statistical learning theory. We are now motivated to ask how the topologies they induce compare. It happens that Wasserstein is the weakest one of them all. We will prove the following theorem:

Theorem 2.2.12: Let $\mathbb{P}_i$ be a sequence of probability distributions, and $\mathbb{P}$ be a probability distribution over X, a bounded space. Then

1. $\mathrm{JSD}(P_i, P) \to 0 \implies \mathrm{TV}(\mathbb{P}_i, \mathbb{P}) \to 0$
2. $\mathrm{TV}(\mathbb{P}_i, \mathbb{P}) \to 0 \implies \mathcal{W}(\mathbb{P}_i, \mathbb{P}) \to 0$

Proof: (1) The following proof is due to Arjovsky et al. (2017). Let $\mathbb{P}_M$ denote $(\mathbb{P}_i + \mathbb{P})/2$. By the triangle inequality applied to TV, we have

$$\mathrm{TV}(\mathbb{P}_i, \mathbb{P}) \leq \mathrm{TV}(\mathbb{P}_i, \mathbb{P}_M) + \mathrm{TV}(\mathbb{P}, \mathbb{P}_M)$$

Then we just apply Pinsker's inequality:

$$\sqrt{\frac{1}{2}\mathrm{KL}(\mathbb{P}_i, \mathbb{P}_M)} + \sqrt{\frac{1}{2}\mathrm{KL}(\mathbb{P}, \mathbb{P}_M)}$$

Due to the fact that KL is nonnegative we get that this is less than

$$\sqrt{\mathrm{JSD}(\mathbb{P}_i, \mathbb{P})} + \sqrt{\mathrm{JSD}(\mathbb{P}_i, \mathbb{P})} = 2\sqrt{\mathrm{JSD}(\mathbb{P}_i, \mathbb{P})}$$

Proof: (2) Since X is bounded fix M such that $\forall x, y \in X, |x - y| < M$. Then we have

$$\begin{aligned}
\mathcal{W}(\mathbb{P}_\mathbb{1}, \mathbb{P}_2) &= \inf_{\pi \in \Pi(X_1, X_2)} \mathbb{E}_{(x_1, x_2) \sim \pi} |x_1 - x_2| \\
&\leq \inf_{\pi \in \Pi(X_1, X_2)} \mathbb{E}_{(x_1, x_2) \sim \pi} M \mathbb{1}[x_1 \neq x_2] \\
&\leq M \mathrm{TV}(\mathbb{P}_\mathbb{1}, \mathbb{P}_2)
\end{aligned}$$

Since convergence in either of the other two distances implies convergence in Wasserstein, our lemma 2.2.4 tells us that Wasserstein gives the weakest topology of the three. This seems like a good reason to investigate it further.

2.3 Kantorovich-Rubinstein Duality

We have established the weakness of the Wasserstein topology. How can we be sure this metric is even directly optimizable? We lack explicit forms of $\mathbb{P}$ and $\mathbb{P}_{g_\theta}$, so this may prove difficult. Fortunately, a classical result in analysis

yields a dual form of Wasserstein Distance which is a more tractable metric for GAN training.

We recall that

$$\mathcal{W}(\mathbb{P}_1, \mathbb{P}_2) = \inf_{\pi \in \Pi(\mathbb{P}_1, \mathbb{P}_2)} \mathbb{E}_{(x_1, x_2) \sim \pi} |x_1 - x_2|$$

To reduce this expression, we add the following term, where f ranges over maps $X \to \mathbb{R}$.

$$\inf_{\pi \in \Pi(\mathbb{P}_1, \mathbb{P}_2)} \mathbb{E}_{(x_1, x_2) \sim \pi} |x_1 - x_2| + \sup_f \mathbb{E}_{x' \sim \mathbb{P}_1}[f(x_1')] - \mathbb{E}_{x_2' \sim \mathbb{P}_2}[f(x_2') + (f(x_1) - f(x_2)]$$

We see that the term we just added is 0 if π is a valid coupling of $\mathbb{P}_1$ and $\mathbb{P}_2$ — the third expectation becomes $-f(x_1)$ because the marginal distribution of x_2 is $\mathbb{P}_2$. This resulting $-f(x_1)$ is in turn canceled by the evaluation of the second expectation, since the marginal distribution of x_1 is again $\mathbb{P}_1$.

Since the first expectation does not depend of f, we can push the sup to the front to obtain

$$\inf_{\pi \in \Pi(\mathbb{P}_1, \mathbb{P}_2)} \sup_f \mathbb{E}_{(x_1, x_2) \sim \pi} |x_1 - x_2| + \mathbb{E}_{x' \sim \mathbb{P}_1}[f(x_1')] - \mathbb{E}_{x_2' \sim \mathbb{P}_2}[f(x_2') + (f(x_1) - f(x_2)]$$

Next we interchange inf and sup. Some technical assumptions are needed to do this, a discussion of which we leave to the appendix. We obtain

$$\sup_f \inf_{\pi \in \Pi(\mathbb{P}_1, \mathbb{P}_2)} \mathbb{E}_{(x_1, x_2) \sim \pi} |x_1 - x_2| + \mathbb{E}_{x' \sim \mathbb{P}_1}[f(x_1')] - \mathbb{E}_{x_2' \sim \mathbb{P}_2}[f(x_2') + (f(x_1) - f(x_2)]$$

Since the term $f(x_1) - f(x_2)$ does not depend on x_2', we can reorder the terms to get

$$\sup_f \mathbb{E}_{x' \sim \mathbb{P}_1}[f(x_1')] - \mathbb{E}_{x_2' \sim \mathbb{P}_2}[f(x_2')] + \inf_{\pi \in \Pi(\mathbb{P}_1, \mathbb{P}_2)} \mathbb{E}_{(x_1, x_2) \sim \pi} |x_1 - x_2| - (f(x_1) - f(x_2))$$

Let us examine second part of this term as a function of f

$$\Gamma(f) = \inf_{\pi \in \Pi(\mathbb{P}_1, \mathbb{P}_2)} \mathbb{E}_{(x_1, x_2) \sim \pi} |x_1 - x_2| - (f(x_1) - f(x_2))$$

We claim the following:

$$\Gamma(f) = \begin{cases} -\infty, & \exists x_1, x_2 : |x_1 - x_2| < |f(x_1) - f(x_2)| \\ 0, & \forall x_1, x_2 : |x_1 - x_2| \geq |f(x_1) - f(x_2)| \end{cases}$$

We relegate the formal proof of this fact to the appendix as well, but intuitively if we suppose the first case holds, we could choose a coupling π which concentrated arbitrarily large probability mass on (x_1, x_2). If the second case holds, we could instead use π concentrate probability mass on $x_1 = x_2$ and send the infimum to zero.

As it happens, the second condition in the cases above is a property of functions known as 1-Lipschitz continuity.

Definition 2.3.1: Given metric spaces X and Y, a function $f : X \to Y$ is K-Lipschitz continuous if and only if

$$\forall x_1 \forall x_2 \frac{d_Y(f(x_1), f(x_2)}{d_X(x_1, x_2)} \leq K$$

We call the smallest such K the Lipschitz constant of f.

Since the sup above can only be obtained when f is 1-Lipschitz (else $\Gamma(f)$ would send it to negative infinity), we can rewrite it as

Theorem 2.3.2: Kantorovich-Rubinstein duality:

$$\mathcal{W}(\mathbb{P}_1, \mathbb{P}_2) = \sup_{f: f \text{ is Lipschitz}} \mathbb{E}_{x' \sim \mathbb{P}_1}[f(x'_1)] - \mathbb{E}_{x'_2 \sim \mathbb{P}_2}[f(x'_2)]$$

We see that this looks quite like the original GAN adversarial loss. Recall the GAN model: We have a discriminator $d_\omega : X \to [0, 1]$ and a generator $g_\theta : Z \to X$. If we allowed our discriminator to instead take values throughout $\mathbb{R}$, we could consider a GAN model with loss

$$\mathcal{L}(\theta) = \mathbb{E}_{x \sim \mathbb{P}_d}[d_\omega(x)] - \mathbb{E}_{z \sim \mathbb{P}_z}[d_\omega(g_\theta(z)]$$

Which is the same as the Wasserstein distance between $\mathbb{P}_{g_\theta}$ and $\mathbb{P}_d$, by Kantorovich-Rubinstein. What is more, we can actually approximate this loss by taking batches of real and generated samples to approximate the expectations w.r.t the real and generated data distribution, and backpropagate through the loss to force our generator distribution $\mathbb{P}_{g_\theta}$ to converge to $\mathbb{P}_d$. Unlike Jensen-Shannon which we minimized implicitly in the original GAN model, here is a GAN objective that concretely approximates a topologically superior distance between probability distributions.

The authors of Arjovsky et al. (2017) trained a GAN on this objective and produced quality results with an impressive lack of mode collapse — see figure 2.2. Now we have the tools to explain why a WGAN can avoid mode collapse while a normal GAN (trained on Jensen-Shannon divergence) cannot. The tendency of WGANs to avoid mode collapse is due to the weakness of the Wasserstein topology — there is more likely to be a sequence of parameters θ_i which continuously converges to a particular mode under a weaker topology.

But there is a caveat: to be able to use the Kantorivich-Rubinstein form of the Wasserstein objective, we need to be able to enforce Lipschitz continuity of the discriminator. Enforcing this constraint was not the focus of Arjovsky et al. (2017), which used weight clipping in the discriminator as a quick solution.

But when it comes to which method for Lipschitz constraint enforcement is used, performance is very much is at stake. These methods have since

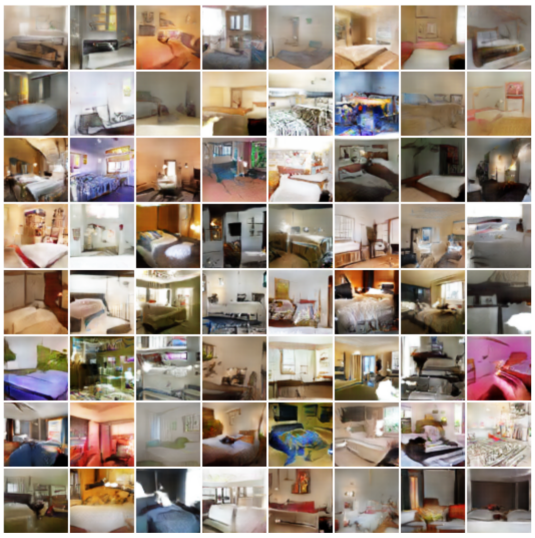

FIGURE 2.2: Samples from a WGAN trained on LSUN Bedrooms

become an active area of research, and recent models have considerably improved upon Arjovsky et al. (2017)'s baseline. To examine them, we must develop a theory for how to make a general neural network Lipschitz.

2.4 Lipschitz Constraints in Neural Networks

Neural networks are layered programs we use to learn complex relationships between different spaces. But they are also functions, and can be reasoned about as such. In the previous section, we saw that to train on the Wasserstein GAN objective effectively, we would need some way of enforcing that the discriminator, d_ω, a neural network, is Lipschitz — as a function from the observation space X to the real numbers $\mathbb{R}$.

Neural networks can be incredibly complicated, with potentially hundreds of layers. Thankfully, the Lipschitz constant of a composition of functions has the following property:

Lemma 2.4.1: Let $\|f\|_L$ denote the Lipschitz constant of a function. Then for any two Lipschitz functions f_1, f_2, we have $\|f_1 \circ f_2\|_F = \|f_1\|_F \cdot \|f_2\|_F$ since

$$|f_1(f_2(x)) - f_1(f_2(y))| \leq \|f_1\|_L |f_2(x) - f_2(y)| \leq \|f_1\|_L \|f_2\|_L |x - y|$$

To meet the conditions of Kantorovich-Rubinstein, our discriminator needs to be Lipschitz. And the compositionality of the Lipschitz constant opens the door to computing the Lipschitz constant of a general feedworward neural network — as the product of the Lipschitz constants of its layers — and regularizing or normalizing it.

There are many types of neural network layers, however. To compute the Lipschitz constant for a general neural network, we need to develop a theory of Lipschitz constants for commonly used neural network layers and

activations. Then, we can provably meet the conditions of the Kantorivich-Rubinstein.

2.4.2 Lipschitz Constants of Neural Network Layers

We will prove a series of theorems about neural network layers. First, we deal with activation functions.

Lemma 2.4.3: Let f be a continuous function $f : \mathbb{R} \to \mathbb{R}$ which is differentiable outside a finite set S. The following are equivalent:

1. Where the derivative of f exists, it is bounded by $|f'(x)| \leq K$
2. f is $K-$Lipschitz.

Proof: See appendix.

This theorem lets us classify several activation functions as Lipschitz — in particular, as 1-Lipschitz. ReLU and Hard Tanh are differentiable outside a finite set and have maximum derivative 1. and Sigmoid is differentiable everywhere with derivative also bounded by 1. Thus, these linearities will not contribute to the Lipschitz constant of the overall network.

Next, we will address the simple linear layer. First, we need the following definition.

Definition 2.4.4: The spectral norm of a matrix $A : \mathbb{R}^n \to \mathbb{R}^m$ is defined as

$$\sigma(A) \equiv \sup_{x \neq 0 \in \mathbb{R}^n} \frac{\|Ax\|_2}{\|x\|_2}$$

Intuitively, the spectral norm is a measure of how much a matrix can "blow up" a given vector. The spectral norm is also equal to the largest singular value of A. In the case where A is square and has real eigenvalues, the spectral norm is the largest eigenvalue, and we can see this by plugging in the eigenvector directly into the definition above.

It turns out that for linear maps, spectral norm and Lipschitz constant are equivalent notions.

Lemma 2.4.5: A matrix $A : \mathbb{R}^n \to \mathbb{R}^m$ has Lipschitz constant $\sigma(A)$.

Proof: See appendix.

Linear layers also have bias terms, but for any bias map $b : \mathbb{R}^n \to \mathbb{R}^m$ the Lipschitz constant is just 1, since $b(x_1) - b(x_2) = x_1 + b - (x_2 + b) = x_1 - x_2$. So the following theorem comes for free from what we've proved thus far

Theorem 2.4.6 (Linear Layers): The Lipschitz constant of a linear layer given by $Ax + b : \mathbb{R}^n \to \mathbb{R}^m$ is $\sigma(A)$.

Now we will consider the case of convolution. For our purposes, convolution refers to the convolving of a 2-dimensional square kernel over a 2-dimensional input pane. In a general convolution, the kernel discretely steps

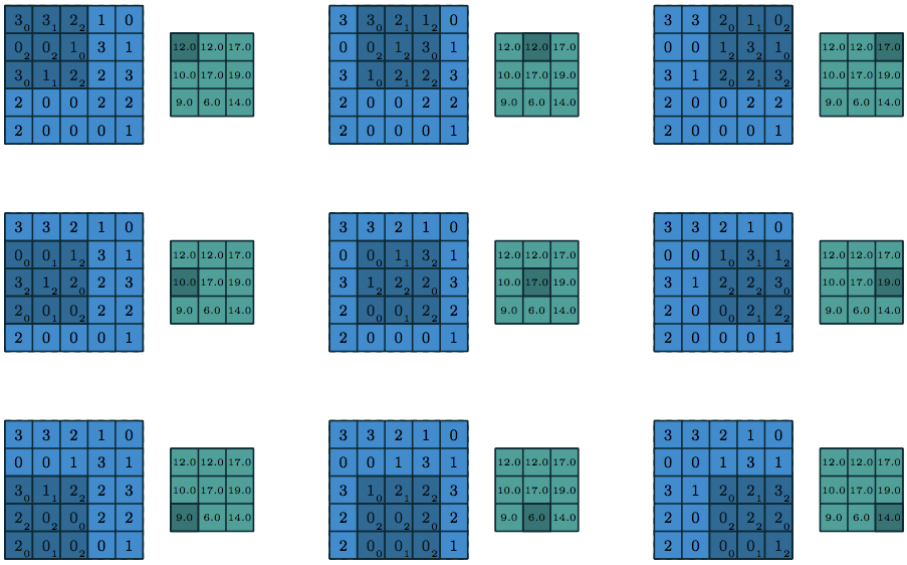

FIGURE 2.3: Convolution of 3x3-kernel over a 5x5 input pane

over the input pane; at each step, the dot product between the current area of the input pane and the kernel is computed and added to the output. Figure 2.3, due to Dumoulin and Visin (2016), is an excellent visualization of this process.

In fact, convolution of an $n \times n$ input pane by a $k \times k$ kernel can be thought of as a linear transformation by a matrix. For example, let K be a 2x2 kernel

$$K = \begin{bmatrix} k_1 & k_2 \\ k_3 & k_4 \end{bmatrix}$$

And consider a 3x3 input pane p

$$P = \begin{bmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \\ p_7 & p_8 & p_9 \end{bmatrix}$$

Convolution of P by K would yield a $2x2$ output pane, without padding. But we could also think of this as a map $C_K : \mathbb{R}^{3\times3} \to \mathbb{R}^{2\times2} \simeq T_K : \mathbb{R}^9 \to \mathbb{R}^4$. The matrix for this transformation would be given by the so-called "Toeplitz" matrix

$$T_K = \begin{bmatrix} k_1 & k_2 & 0 & k_3 & k_4 & 0 & 0 & 0 & 0 \\ 0 & k_1 & k_2 & 0 & k_3 & k_4 & 0 & 0 & 0 \\ 0 & 0 & k_1 & k_2 & 0 & k_3 & k_4 & 0 & 0 \\ 0 & 0 & 0 & k_1 & k_2 & 0 & k_3 & k_4 & 0 \end{bmatrix}$$

Thus, we can compute the Lipschitz constant for a convolution via lemma 2.4.5.

Theorem 2.4.7 (Convolutions): The Lipschitz constant of a convolution of input panes of size $h \times w$ with kernel K is given by $\sigma(T(K, h, w))$, where $T(K, h, w)$ is the corresponding Toeplitz matrix.

A general convolution also has a given number of input channels and output channels. Each pair of input and output channels has its own kernel; thus, the matrix for the transformation is a block matrix, with each block a Toeplitz matrix for an individual convolution. We have provided code for building the Toeplitz matrix for a general convolution. But constructing the Toeplitz matrix may be prohibitively expensive, especially when the number of input or output channels is large. Instead, suppose we have i input channels and o output channels and square kernels of size k; then our convolutional kernels K can be viewed as an element of $\mathbb{R}^{i\times o\times k\times k}$, or, alternatively, as a matrix R_K in $\mathbb{R}^{o\times(i\times k\times k)}$. This matrix is orders of magnitude smaller than the full block Toeplitz matrix of the convolution, and its spectral norm gives a bound on the spectral norm of the block Toeplitz matrix.

The following theorem, provided without proof, is corollary 1 of Tsuzuku et al. (2018).

Lemma 2.4.8: Let T_K be the block-Toeplitz matrix of a general convolution, and R_K be the reshaped kernel matrix from above. Then

$$\sigma(T_K) \leq \sqrt{n}\sigma(R_k)$$

Where $n = \min(k, h-k+1)\cdot\min(k, w-k+1)$.

While not the true Lipschitz constant of a convolutional layer, the approximation R_K has been used in practice, for example in Miyato et al. (2018) and Brock et al. (2018).

We must address one final point.

Lemma 2.4.9: Let

$$C_K : \mathbb{R}^{i\times h_i\times w_i} \to \mathbb{R}^{o\times h_o\times w_o}$$

Be a general convolution by a kernel K. Consider its block Toeplitz matric T_K. Then the linear map T_K^T is also a convolution, given by

$$C_{\tilde{K}}^T : \mathbb{R}^{o\times h_o\times w_o} \to \mathbb{R}^{i\times h_i\times w_i}$$

Where $\tilde{K}$ is the kernel obtained by rotating the kernel K by 180 degrees. A proof of this fact can be found in Dumoulin and Visin (2016).

We have covered the major building blocks of the neural networks typically used in GAN architectures. For additional layers and less frequently used activations such as max-pooling, interpolations, average-pooling, ELU and lRELU, consult Tsuzuku et al. (2018) for a comprehensive overview.

2.4.10 Computation of the Spectral Norm: Power Iteration

We are on the road to developing a toolkit for computing the Lipschitz constant of a general neural network. For most nonlinearities, we can appeal to lemma 2.4.3, and for linearities (linear or convolutional layers) we can use the spectral norm.

However, computing the spectral norm directly, i.e. via computing the singular value decomposition of the weight matrix, would be infeasible when

we are constantly performing weight updates and need the spectral norm in an on-line setting. Instead, we will derive a fast approximation.

To prove it, we need the Spectral Theorem for Symmetric Matrices from Axler (1997).

Theorem 2.4.11: Let $A : \mathbb{R}^n \to \mathbb{R}^n$ be a symmetric matrix. Then

1. W is diagonalizable
2. The eigenvalues of W are real
3. There exists an orthonormal basis of $\mathbb{R}^n$ given by the eigenvectors of W

Using this theorem we can derive an iterative algorithm to approximate the spectral norm of a matrix.

Theorem 2.4.12: Let $W \in \mathbb{R}^{n \times m}$ be a weight matrix. By singular value decomposition, the spectral norm of W is equal to the square root of the largest eigenvalue of $W^T W$. By the spectral theorem, take $\lambda_1 ... \lambda_n$ eigenvalues of $W^T W$ with $v_1 ... v_n$ an orthonormal eigenbasis of $\mathbb{R}^n$ ordered by size of eigenvalue, and with v_1 corresponding to the largest eigenvalue (we assume the eigenvalues are distinct). Fix $x \in \mathbb{R}^n$ such that $x^T v_1 \neq 0$ (in practice, x is initialized randomly since this happens with probability 0). Define

$$x_0 := x$$

$$x_i = \frac{(W^T W) x_{i-1}}{\|(W^T W) x_{i-1}\|_2}$$

Then we have

$$\lim_{i \to \infty} \|x_i - v_1\|_2 = 0$$

In other words, x_i converges to the largest eigenvector of $W^T W$.

Proof: Since v_i are an eigenbasis of $\mathbb{R}^n$, write $x = a_1 v_1 + ... + a_n v_n$. Then

$$\|(W^T W) x_{i-1}\|_2 = \sqrt{a_1^2(\lambda_1^{2i}) + ... + a_n^2(\lambda_n^{2i})}$$

So we can write

$$x_i = \sum_{k=1}^{n} \frac{a_k \lambda_k^i v_k}{\sqrt{a_1^2(\lambda_1^{2i}) + ... + a_n^2(\lambda_n^{2i})}}$$

Since the eigenvalues are distinct and λ_1 is the largest, all terms of this sum but the first go to zero. We have

$$\lim_{i \to \infty} x_i = \frac{a_1 \lambda_1^i v_1}{\sqrt{a_1^2(\lambda_1^{2i}) + ... + a_n^2(\lambda_n^{2i})}} = v_1$$

We can use the power iteration method to compute the spectral norm of a general matrix A by computing x_i out to a suitably large i and then calculating

$$\sigma(A) \approx \sqrt{A^T A x_i / \|x_i\|_2}$$

2.5 Summary

We began by introducing the original GAN model. After developing some background in topology, we examined the original GAN objective — an implicit minimization of Jensen-Shannon divergence — and analyzed the shortcomings of using Jensen-Shannon as our distance in terms of the topology it induces on the space of probability distributions over the observation space X.

Motivated by the shortcomings of Jensen-Shannon, we introduced several other commonly used statistical distances and picked the one which induced the weakest topology: Wasserstein distance. Then, we derived the Kantorovich-Rubinstein dual form of the Wasserstein distance, which we then wrote down as a concrete adversarial GAN objective, where the discriminator was constrained to be 1-Lipschitz.

Motivated by the question of how to enforce the 1-Lipschitz constraint in the discriminator of our new adverarial objective, we proved that a feedforward neural network had Lipschitz constant equal to the product of the Lipschitz constants of its layers, and undertook a systematic investigation of the Lipschitz constants of general types of neural network components: activations, convolutions, and linear layers. Due to the importance of the spectral norm in computing the Lipschitz constant of convolutions and linear layers, we derived a practical algorithm for its approximate computation.

We now have the tools to discuss and analyze recent work, which has approached the question of how to enforce Lipschitz continuity in the discriminator using many of the facts and procedures we have outlined.

Chapter 3

Related Work

3.1 Lipschitz Continuity Enforcement in GANs

Lipschitz continuity enforcement for the WGAN is an active area of research. But it has been shown that GANs trained under losses other than the WGAN loss still benefit from explicit enforcement of a Lipschitz constraint in the discriminator. Lipschitz continuity constraints can simply be applied to stabilize training, as in Brock et al. (2018) which used them in both the generator and discriminator, or to provably guard a network against adversarial examples as in Tsuzuku et al. (2018). As a direct consequence of the perfect discrimination theorems of Arjovsky and Bottou (2017), Lipschitz constraints can also be shown to prevent vanishing gradients, even in a general GAN, when the real and generated data manifolds are totally separable. Though we are motivated here by the problem of Lipschitz constraints in the context of the WGAN theoretical framework, we note that fast, scalable and effective methods for enforcing Lipschitz continuity in general GANs are in high demand.

We can separate Lipschitz enforcement techniques into two categories: those which directly and provably enforce a Lipschitz constraint, and those which add a regularization term to the discriminator's loss, indirectly punishing networks which are not $K-$Lipschitz for suitably small K.

3.2 Indirect Methods

The indirect method of Yuichi Yoshida (2017) is equivalent to using the sum of the spectral norms of the network's weight layers as an explicit regularization term, while the method of Gulrajani et al. (2017) regularizes deviations in in the norm of the gradient of the discriminator from 1. The R_1 and R_2 gradient penalty of Mescheder (2018) also regularize the Lipschitz constant indirectly by penalizing the gradient of the discriminator and provide some attractive theoretical guarantees for WGAN convergence, though the more recent work of Brock et al. (2018) has found them to be too restrictive in practice.

3.3 Direct Methods

Weight-clipping, as presented in Arjovsky et al. (2017), is a direct constraint. For a proof of this, see Aappendix B. The spectral normalization algorithm of Miyato et al. (2018) also directly constrains the Lipschitz constant of the discriminator — this time, by rescaling the weight layers to have spectral norm 1 before each forward pass. For direct techniques involving spectral norm reparametrization, such as Miyato et al. (2018), Brock et al. (2018), and this work, the spectral norm and first few singular values are typically approximated via power iteration, which we derived in background section 2.4

Though not explicitly intended to bound the spectral norm, the popular weight normalization algorithm of Salimans and Kingma (2016) can be shown to do so directly. However, Miyato et al. (2018) demonstrated the tendency of this algorithm to collapse the spectra of the discriminator.

Surprisingly, indirect methods can be combined with direct methods to yield superior results, despite the fact that a direct constraint should theoretically already guarantee Lipschitz continuity. For example, the best model of Miyato et al. (2018) employs both the two-sided gradient penalty of Gulrajani et al. (2017) and their own novel direct Lipschitz constraint, achieving superior results to the models which used either technique in isolation.

3.4 Our Contribution

First, we analyze the model of Miyato et al. (2018) and theoretically derive some worst-case scenarios for its Lipschitz constant. Then, we present an algorithm for the fast computation of the correct spectral norm of a convolution, which does not rely on the bound of Tsuzuku et al. (2018). Thus, unlike in Miyato et al. (2018), we can directly compute the Lipschitz constant of our network. Using this knowledge, we experiment with a variety of discriminator networks of different Lipschitz constants and discover a trade-off between Lipschitz constant and model capacity – in particular, noting the poor performance of a GAN whose discriminator is 1-Lipschitz.

Then, we experiment with two extensions of the baseline algorithm for spectral normalization designed to mitigate the rank collapse issues which are common to several methods of enforcing Lipschitz constraints. We give their performance on the Inception Score metric of Salimans et al. (2016), and analyze the spectra of their weights to determine the effectiveness of the extension.

Chapter 4

Models

4.1 Techniques for Enforcing a Lipschitz Constraint

Arjovsky et al. (2017) used weight clipping to enforce Lipschitz continuity of the critic network, but noted this was "clearly a terrible way to enforce a Lipschitz constraint." Since then, the question of how to enforce Lipschitz continuity of the critic network has been answered in a few ways.

4.1.1 Spectral Normalization (SN)

The power iteration method (see section 4 of background) can be used to approximate the dominant singular value of a matrix. Miyato et al. (2018) introduced the notion of reparametrized linear and convolutional layers which, given a weight W, keep a running estimate of the left-eigenvector u of W^TW corresponding to the dominant singular value. For convolutional layers, the reshaped kernel matrix R_K (refer to background section 2.4) is used for W instead of the full Toeplitz matrix T_k.

During each forward pass of their network, the approximation is updated for some number of power method iterations through the procedure

(1) $v_i \leftarrow \frac{W_i^T u_{i-1}}{\|W_i^T u_{i-1}\|_2}$

(2) $u_i \leftarrow \frac{W_i v_i}{\|W_i v_i\|_2}$

Due to the slow rate of change of the weight matrices during gradient descent, only one iteration of the power method after each forward pass was needed to update the estimates of u and σ — making spectral normalization quite computationally cheap. The full algorithm is given in algorithm 1.

To prove a model trained in this fashion is K-Lipschitz, we simply note that since the linear weight layers have spectral norm 1 and the convolutional layers have spectral norm $\sqrt{n}$ given by the bound in lemma 2.4.8, a feedforward network trained in this manner has Lipschitz constant $\sqrt{n}^c$, where c is the number of convolutional layers, and we have not used nonlinearities with derivative greater than 1.

Algorithm 1 Spectral Normalization (Miyato et al., 2018)

For each weight matrix W_i in the network, initialize u_i randomly
for Each forward call during training of layer i with weight W_i **do**
  (Update spectral parameters)
  for number of power method iterations Ip **do**
    $v_i \leftarrow \frac{W_i^T u_{i-1}}{\|W_i^T u_{i-1}\|_2}$
    $u_i \leftarrow \frac{W_i v_i}{\|W_i v_i\|_2}$
    $\sigma_i \leftarrow u_i^T W_i v_i$
  end for
  (Store the normalized weight matrix)
  $W_i \leftarrow W_i / \sigma_i$
end for

4.1.2 SN with a Toeplitz-normalized Discriminator

In designing their discriminator to be Lipschitz, the authors of Miyato et al. (2018) used a theorem of Tsuzuku et al. (2018) to bound the Lipschitz constant of a convolutional layer. We recall it here as the following lemma from the background:

Lemma 2.4.8: Let T_K be the block-Toeplitz matrix of a general convolution, and R_K be the reshaped kernel matrix. Then

$$\sigma(T_K) \leq \sqrt{n}\sigma(R_K)$$

The problem with using a bound for the spectral norm of the convolution is that the Lipschitz constant is multiplicative. The discriminator of Miyato et al. (2018) has 10 convolutional layers. If $k = 2$ and h, w are sufficiently large, then $n = 4$, and we could potentially be dealing with a discriminator which is $2^{10} = 1024-$Lipschitz! Indeed, the fact that Miyato et al. (2018)'s class-conditional model was improved by adding the gradient penalty of Gulrajani et al. (2017) suggests that in some cases the model required more Lipschitz conditioning than was obtained by the reparametrization using the reshaped kernel matrix.

Due to this multiplicative effect of the error bound, it would be better if we could use the actual Toeplitz matrix to do power iteration. We know that computing the spectral norm of the full Toeplitz matrix and normalizing by that can be prohibitively expensive — for example, a standard convolutional layer in the ResNet discriminator of Gulrajani et al. (2017) trained on the CIFAR-10 dataset of Krizhevsky (2009) might have 256 input channels, 256 output channels, input height 32 and width 32. With a 3x3 kernel, this yields a full block Toeplitz matrix in $\mathbb{R}^{246016 \times 262144}$.

Instead, we can use the transpose convolution to perform implicit power iteration of the Toeplitz matrix. Recall lemma 2.4.9, which says that given a convolution

$$C_K : \mathbb{R}^{i \times h_i \times w_i} \to \mathbb{R}^{o \times h_o \times w_o}$$

The transpose convolution is given by

$$C_{\tilde{k}}^T : \mathbb{R}^{o \times h_o \times w_o} \to \mathbb{R}^{i \times h_i \times w_i}$$

Where $\tilde{K}$ is the rotation of the kernel K by 180 degrees. Moreover, we have that the transpose convolution corresponds to multiplication by T_K^T, where T_K is the Toeplitz matrix of C_K.

Using this, we can do implicit power iteration without having to multiply by the full Toeplitz matrix. Initializing u randomly in $\mathbb{R}^{o \times h_o \times w_o}$, we can compute successive approximations as before

(1) $v_i \leftarrow \frac{C_{\tilde{k}}^T(u_{i-1})}{\|C_{\tilde{k}}^T(u_{i-1})\|_2}$

(2) $u_i \leftarrow \frac{C_k^T(v_i)}{\|C_k(v_i)\|_2}$

And we are guaranteed that $u_i \odot C(v_i) \to \sigma(T_K)$. With this slight modification of the model of Miyato et al. (2018), we can construct a discriminator which is provably 1-Lipschitz up to the error bound of the power iteration method.

4.1.3 Gradient Penalty

Recall lemma 2.4.3, which says that Lipschitz continuity is equivalent to bounded gradient in the case where f is differentiable almost everywhere. The gradient penalty of Gulrajani et al. (2017) directly regularizes differences in the norm of the gradient of the discriminator f from 1, sampling from $\mathbb{P}_{\tilde{d}}$ of points interpolated between real and generated examples.

$$R = \lambda \mathbb{E}_{x \sim P_{\tilde{d}}}[(\|\nabla f(x)\|_2 - 1)^2]$$

In this way, f is constrained to be $1-$Lipschitz. Increasing λ tightens the bound on the Lipschitz constant.

We train a model with this penalty, but this extension is not novel — we include it for completeness of our analysis.

4.2 Improving Networks with an Enforced Lipschitz Constraint

Some methods for enforcing a Lipschitz constraint can be too restrictive. Consider the following normalizations

1. Weight clipping (WC) to a fixed interval $[-c, c]$ as in Arjovsky et al. (2017)

2. The weight normalization (WN) of Salimans and Kingma (2016), setting the l_2 norm of the row of each weight matrix to 1

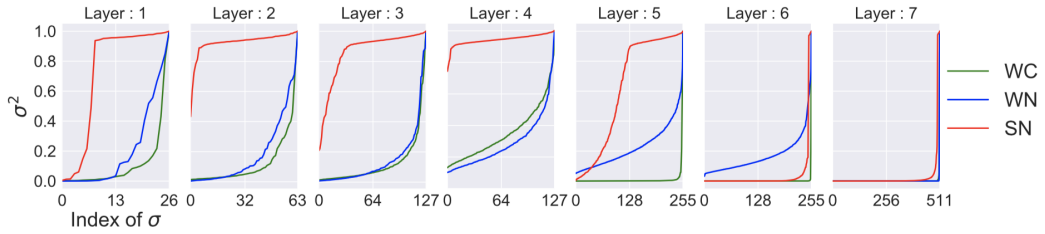

FIGURE 4.1: Singular value plots of weight matrices of the discriminator in Miyato et al. (2018) trained with different weight normalizations

3. Spectral normalization (SN)

We make the following observation: all three normalizations are sufficient to enforce Lipschitz continuity of the discriminator. For proofs of (1) and (2), consult lemma 2.4 of appendix A.

Miyato et al. (2018) investigated these normalizations and identified the first two methods as impairing the performance of the discriminator by collapsing its rank. Figure 4.1 gives the singular values of the weight matrices of their discriminator trained with the different normalizations. Observe the notable rank collapse in the (WC) and (WN) algorithm. By contrast, (SN) has well-behaved spectra.

In short, not all normalizations which enforce Lipschitz continuity are created equally; some introduce undesirable side effects such as the spectral collapse we see in figure 4.1. We continue motivated by the question of how to avoid rank collapse when generally enforcing Lipschitz continuity.

4.2.1 Quantifying Rank Collapse

For a weight matrix in $W^{n\times m}$, we introduce the following metric to approximately quantify rank collapse.

$$r(W) = \frac{\|W\|_F}{m\sigma(W)}$$

$r(W)$ represents the area under the curve of a given spectral plot, for example 4.1 layer 1, normalized so that $\max(r(W)) = 1$. Intuitively, r can be thought of as a "spectral Gini coefficient," measuring how unequal the distribution of spectral values is — if the largest spectral value dominates the others, $r(W)$ will be small, and if the distribution of spectra is even, $r(W)$ will be close to 1.

We see that $r(W_{SN}) > r(W_{WN})$ and $r(W_{SN}) > r(W_{WC})$ in every layer of the discriminator of Miyato et al. (2018). In other words, spectral normalization does the best job of enforcing spectral equality. $r(W)$ can be used to quantify whether the discriminator uses all features of the input space. But it can also be used to quantify the degree of mode collapse in the generator: if $r(W)$ is low for a generator's weight matrices, the generated samples

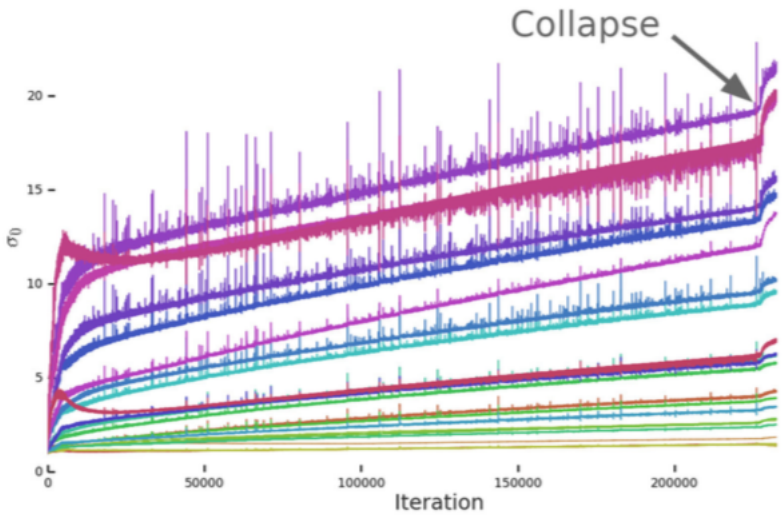

FIGURE 4.2: Oscillations of the spectral norm of discriminator weights in the model of Brock et al. (2018)

will tend to lie on a low dimensional manifold, corresponding to a collapsed generator. If $r(W)$ is high, sample variety ought to be greater.

4.2.2 Direct Regularization of $r(W)$ in the Generator

Though slightly optimistic, we might hope that adding

$$-\lambda \sum_{i=1}^{n} r(W_i)$$

to the loss of the generator could ameliorate mode collapse by directly penalizing low rank weight matrices.

4.2.3 Spectral Gradient Clamping

The following extension is motivated by the results of Brock et al. (2018). In a systematic investigation of GAN training, the authors determined that the spectral norm σ_0 of the weight layers of the discriminator was at least proximately connected to the stability of training — the spectral norms of the discriminator weights would rise slowly and then explode at collapse. An illustration of this is given in figure 4.2. The authors additionally found the ratio $\frac{\sigma_0}{\sigma_1}$ of the first singular value over the second singular value to be a good proxy for training stability.

We were motivated to seek a gradient transformation that could help condition the spectra to improve the $r(W)$ metric above and to stabilize σ_0/σ_1.

Clipping gradients, as proposed in Pascanu et al. (2012), can improve the stability and convergence speed of general neural networks. Moreover, the addition of gradient penalties of Mescheder (2018) appear to improve the

stability and results of WGAN training, as does the gradient penalty of Gulrajani et al. (2017). As such, it seems that modifications or penalties on the gradients of d could improve the performance of d.

As a replacement for spectral normalization, Brock et al. (2018) experimented with spectral clamping: setting $\sigma_0 = \min(\sigma_0, r \cdot \sigma_1)$ where $r > 1$ is some fixed ratio. We experimented with spectral gradient clamping on top of the baseline spectral normalization model. Spectral gradient clipping follows essentially the same principle as spectral clipping, just applied to its gradient.

The gradient of W is a matrix with the same shape as W, which we denote W'. As in spectral normalization, we maintain running estimates u_0, v_0 of the first left- and right- singular vectors of the gradient of W. We also maintain an estimate of the first two singular values and of the second left- and right-singular vectors u_1, v_1, which we compute as

(1) $\sigma_0 \leftarrow u_0^T W' v_0$

(2) $v_1 \leftarrow \frac{(W' - \sigma_0 u v^T) u_1}{\|(W' - \sigma_0 u v^T) u_1\|_2}$

(3) $u_1 \leftarrow \frac{(W' - \sigma_0 u v^T)^T v_1}{\|(W' - \sigma_0 u v^T)^T v_1\|_2}$

(4) $\sigma_1 \leftarrow u_1^T (W' - \sigma_0 u v^T) v_1$

The matrix $W' - \sigma_0 u v^T$ can be be thought of as the matrix W' with the first singular value deleted from the singular value decomposition. In this way, steps 2 and 3 actually constitute a separate iteration of the power method, for the purpose of finding the second singular value (which is in this case the largest singular value of the matrix $W - \sigma_0 u v^T$.

After each backward pass, we simply clamp the gradients so that σ_0 is at most $r \cdot \sigma_1$, where r is some fixed hyperparameter greater than 1.

$$W' \leftarrow W' - \min(0, \sigma_0 - r \cdot \sigma_1) u_0 v_0^T$$

We experimented with spectral gradient clamping in both the generator and discriminator, with various settings of r.

Chapter 5

Experiments

Having built several GAN models, we will use them to generate images and compare generated samples. We will also investigate the spectra of the weight layers to determine the degree of mode collapse and the effectiveness of our extensions.

5.1 Dataset

When training a GAN to generate images, we need a collection of ground-truth samples for the generator to try and emulate. For our experiments, we use the CIFAR-10 dataset of Krizhevsky (2009). CIFAR-10 is a dataset of 60,000 32x32x3, images in 10 classes — for example, horses, cars, boats. There are 6000 examples in each class, split into a train set of 50,000 examples and a test set of 10,000.

For preprocessing, we transformed the color channels of each image to lie in $[-1, 1]$ and added uniform instance noise $\varepsilon \sim \mathcal{U}[0, \frac{1}{128}]$ to each image. Figure 5.1 gives some examples from the CIFAR10 dataset.

5.2 Architecture

Our baseline model architecture replicates that of Miyato et al. (2018), which in turn replicated the architectures of Gulrajani et al. (2017). The architecture of our discriminator is given by figure 5.2, and the generator is in 5.3.

Both networks are modeled after the ResNet of He et al. (2015), which consist of a series of compositions of ResBlocks. A ResBlock is a neural network layer consisting of two convolutions in series with ReLU nonlinearities added to a convolution which bypasses the rest of the layer. We use Batch normalization before our nonlinearities only in the generator. A diagram of a a ResBlock is given in figure 5.4.

5.3 Training

We tested several GAN loss functions: standard, hinge, and WGAN, and found the hinge loss to perform best in practice. The hinge loss is given by

$$\mathcal{L}(g_\theta) = \mathbb{E}_{z \sim \mathbb{P}_z}[-d_\omega(g_\theta(z))]$$

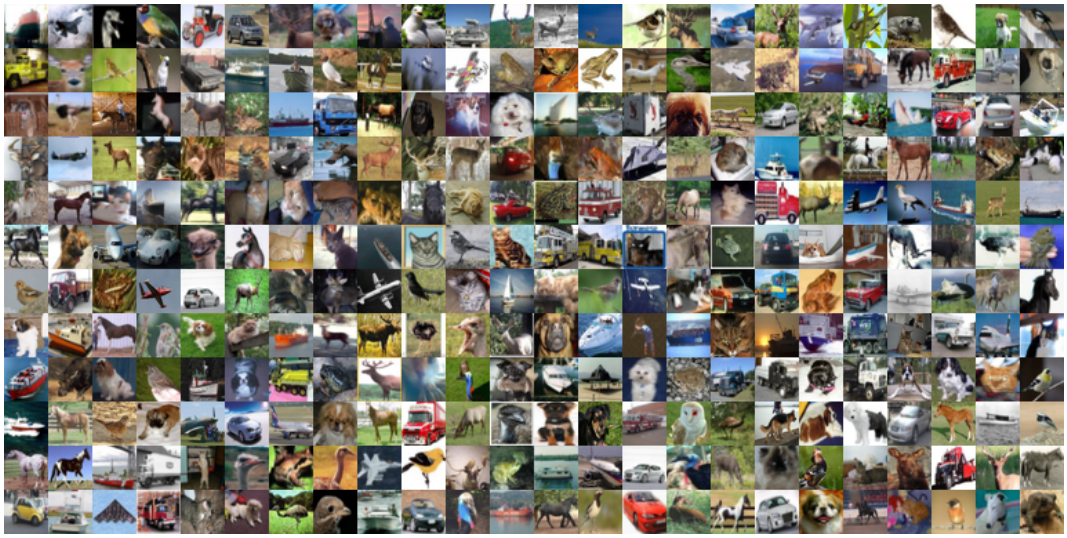

FIGURE 5.1: Random samples from CIFAR10

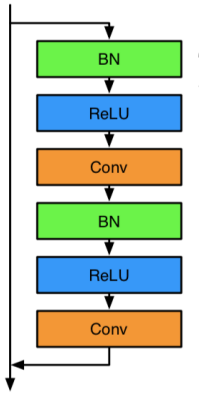

FIGURE 5.2: A ResBlock (Diagram due to Miyato et al. (2018))

Discriminator				
Layer	Kern-Size	Downsample	In-Channels	Out-Channels
ResBlock	3	True	3	128
ResBlock	3	True	128	128
ResBlock	3	False	128	128
ResBlock	3	False	128	128
ReLU	-	-	-	-
SumPool	-	-	-	-
Linear	-	-	128	1

FIGURE 5.3: Discriminator Architecture

Generator				
Layer	Kern-Size	Upsample	In-Channels	Out-Channels
Linear	-	-	128	16*256
ResBlock	3	True	256	256
ResBlock	3	True	256	256
ResBlock	3	True	256	256
BatchNorm	-	-	-	-
Conv2d	3	-	256	3
Tanh	-	-	-	-

FIGURE 5.4: Generator Architecture

$$\mathcal{L}(d_\omega) = \mathbb{E}_{z\sim\mathbb{P}_z}[\rho(1 + d_\omega(g_\theta(x)))] + \mathbb{E}_{x\sim\mathbb{P}_d}[\rho(1 - d_\omega(x))]$$

Where ρ is a ReLU nonlinearity. For every update of the generator objective, we performed five updates of the discriminator. When updating the discriminator, we sampled real data exclusively from the training set of CIFAR10, with batch size of 64 and used an equally size batch of noise. During our generator updates, we used a batch size of 128. We performed 50,000 total generator updates.

We used ADAM optimization (Kingma and Ba, 2014) with a learning rate of .0002, $\beta_1 = 0$ and $\beta_2 = .9$ for both the generator and discriminator. We linearly decayed the learning rate to 0 through the training process.

5.4 Benchmarking

We evaluated the models and our extensions using Inception Score of Salimans et al. (2016), which extracts features from our generated images via a neural network and compares these features to those extracted from the true dataset in a manner which has been seen to correspond well with human judgments.

We provide samples and interpolations in the generator latent space in Appendix C. Images were not cherry-picked. We note a distinct lack of mode-collapse among generated samples, although the small size of images in the CIFAR10 dataset (32x32) does not lend itself to particularly clear samples, especially in the unsupervised case.

Finally, we use the spectral plots in Appendix B to reason quantitatively about mode collapse in our GANs.

5.5 Results and Discussion

5.5.1 Toeplitz-normalized discriminator

Recall our provably 1-Lipschitz discriminator from the models section: we devised a modified version of Spectral Normalization which would allow

Results (Toeplitz-normalized Discriminator)	
k	Inception Score
1	2.62
2	7.34
4	5.77
8	3.05

FIGURE 5.5: Inception Scores for Toeplitz-normalized Discriminator as a function of reparametrized spectral norm k of convolutional layers

us to directly approximate the spectral norm of a convolution in an efficient computational way. Then, we showed a discriminator constrained in such a way would be $1-$Lipschitz.

Surprisingly, this constraint proved incredibly restrictive. The first row of figure 5.5 shows that naively enforcing a $1-$Lipschitz constraint caused the model to collapse. We hypothesize that this is due to the reduced model capacity of a $1-$Lipschitz discriminator. When the Lipschitz constant of the discriminator is constrained to be this low, the generator is not trained effectively as the discriminator simply performs too poorly.

To test this hypothesis, we perform a simple experiment to increase the model capacity. We add a hyperparameter k to each spectrally normalized convolutional layer, such that the output of each convolution is simply multiplied by k. This means the Lipschitz constant of the discriminator is exactly k^7, since our discriminator has 7 convolutional layers.

We see that as k increases, the Inception Score increases and then decreases. We posit a nonlinear trade-off between Lipschitz constant of the discriminator and GAN performance: when the constant is enforced to be too low, the discriminator has insufficient capacity to train the generator effectively. When the constant is too high, the Lipschitz assumption of Wasserstein distance is violated and training collapses. The sweet spot seems to be around $k = 2$. Due to the explosive nature of the Lipschitz constant as k increases, more hyperparameter investigation in the neighborhood of $k = 2$ is needed to determine if the Toeplitz-normalized discriminator can recover the original model performance.

Still, these results are surprising. They suggest that while truly approximating Wasserstein requires a $1-$Lipschitz discriminator, it is in the interest of one's model to relax this constraint to some K suitably large — but not too large — value.

5.5.2 Direct regularization of $r(W)$ in G

Recall our definition of the penalty $r(W)$ which approximately penalizes low-rank matrices with unequal distribution of spectra.

$$r(W) = \frac{\|W\|_F}{m\sigma(W)}$$

Results ($r(W)$ Penalty)	
λ	Inception Score
.1	7.95
1	1.18
10	.99

FIGURE 5.6: Inception Scores for different strengths of $r(W)$ penalty

$$R = -\lambda \sum_{i=1}^{l} r(W_l)$$

We experimented with adding the regularization R to the generator loss with various values of λ. It had no positive effect. Figure 5.6 gives the results — we see that with small enough λ, we recover the baseline, and anything larger collapses training.

To investigate the failure of this penalty, we turn to the spectral plots of D and G for our best model at $\lambda = .1$, figures B.1 and B.2 in Appendix B. If we consider these spectral plots, we see that rather than improving the spectral equality of the generator, adding the penalty $r(W)$ has collapsed it, though without affecting the inception score.

To see why this may have happened, we should investigate the penalty R further. By the quotient rule it has gradient

$$\nabla r(W) = -\frac{2W\sigma(W) - \|W\|_F \nabla\sigma(W)}{m^2\sigma^2(W)}$$

Ignoring the gradient of the spectral norm, we see that the penalty R corresponds to an inverse weight decay on the matrices W. In other words, the gradient pushes W in the direction of $\mathbb{1}^{n,m}$, a matrix with all ones. And this matrix has one singular value because it has rank 1. So actually, this penalty will push the largest singular value of the weight matrix to be equal to that of $\mathbb{1}^{n,m}$. Thus, a naive regularization of spectral inequality $r(W)$ will actually degrade $r(W)$, when coupled with spectral normalization.

5.5.3 Spectral Gradient Clipping

Recall our algorithnm for spectral gradient clipping: we clip the maximum singular value σ_0 of the gradients of a weight matrix W to a ratio r of the second largest singular value σ_1. We experimented with spectral gradient clipping in both the generator and the discriminator. Figures 5.6 and 5.7 give the results. This method did not improve over the baseline, but we gained some useful insights into the pathology of training G and D.

In particular, the values of r which collapsed training were vastly different depending on whether we applied clipping in the generator or discriminator. In early training, we found the discriminator to regularly update the gradients of its weights with extremely low rank updates — with σ_0/σ_1 on the

Results (Spectral gradient clamping with ratio r in Generator)	
r	Inception Score
1.05	3.782
1.5	7.82
2	7.94

FIGURE 5.7

Results (Spectral gradient clamping with ratio r in Discriminator)	
r	Inception Score
4	3.84
15	7.92
30	7.95

FIGURE 5.8

Results (Miscellaneous)	
Model	Inception Score
SN-GAN (baseline)	7.97
WGAN-GP	7.08
SN-4x	7.56
RESNET	3.28

FIGURE 5.9: Inception Scores for miscellaneous extensions

order of 30. Later, σ_0/σ_1 stabilized to around 3 for most updates. Thus, for small values of r, spectral clipping in the discriminator proved catastrophic.

By contrast, the generator is considerably more amenable to spectral clamping. Even fixing $r = 1.5$ essentially recovers the performance of the baseline model. Figures B.3 and B.4 in Appendix B give the spectral plots of a model with spectral clamping used in the discriminator, while figures B.5 and B.6 give the spectral plots of a model with spectral clamping used in the generator. We see that those models which did not collapse the Inception Score appeared to have little effect on the spectra of the generator or discriminator.

5.5.4 Miscellaneous Models

We experimented with several other extensions and modifications of the model of Miyato et al. (2018). We used a WGAN loss with gradient penalty in addition to the usual spectral normalization, denoted WGAN-GP. We tried quadrupling the batch size, denoted SN-4x. We tried removing spectral normalization altogether, denoted RESNET. The results are given in figure 5.9.

We compare the spectral plots of WGAN-GP with the baseline spectral normalization in figures in B.7, B.8 of Appendix B. We compare RESNET with the baseline as well, in figures B.9, B.10

Chapter 6

Conclusion

We began this thesis by attempting to understand why GAN training fails. After developing a mathematical model for GANs, we uncovered the Lipschitz condition as being key to the training of the Wasserstein GAN, which enjoys some favorable properties such as avoiding mode collapse and vanishing gradient issues.

We then developed several models for Lipschitz continuity enforcement, building off previous work in this space. We feel that the the Toeplitz normalized discriminator may warrant further investigation, as it allows us to reason directly about model performance under the Wasserstein objective as a function of Lipschitz constant K. We saw that if K was too low, the model failed, corresponding to a discriminator with insufficient capacity, but we also saw that if K was too high, the model also failed, corresponding to the fact that training such a discriminator is a poor approximation to Kantorovich-Rubinstein. Thus, determining the optimal K could prove fruitful for further theoretical and practical studies. In particular, investigating why $K = 1$ is not optimal could be worthwhile. We feel that our other models, namely spectral clamping and the $r(W)$ penalty, do not warrant any further investigation, though the former provided some interesting insights into the stability of GAN training; namely, the necessity of extremely low-rank updates in d for optimal training.

Appendix A

Background

A.1 Foundations, Kantorovich-Rubinstein

Proof (2.2.4): Consider the identity map from X equipped with the topology τ_1 to X equipped with the topology τ_2. Then for any sequence we have $d_1(x_n, x) \to 0$ implies $d_2(id(x_n), id(x)) \to 0$, so id is continuous. So the preimages of open sets in τ_2 are open, but since the map is the identity this just means $\tau_2 \subseteq \tau_1$

Proof (2.3.2): The ability to switch the inf and sup is a consequence of Sion's minimax theorem, and the derivation of the value of the Lipschitz term requires more probability theory, measure theory and topology than is within the scope of this thesis without a more considerable detour than we have already taken. The interested reader is encouraged to consult Villani (2009) which gives a complete derivation of Kantorovich-Rubinstein in chapter 5. For a middle ground, consult Herrmann (2017).

A.2 Lipschitz Constants of Neural Networks

Proof (2.4.3): $(1 \implies 2)$ Let $x, y \in \mathbb{R}$. Assume without loss of generality $x < y$. Take $s_1 ... s_{n-1}$ to be the elements of S between x and y, and say $x = s_0 <= s_1 < ... < s_n < s_{n-1} <= s_n = y$. Let $\epsilon > 0$ and take δ by uniform continuity of f in $[x, y]$. For sufficiently small δ we can write

$$|f(y) - f(x)| = \sum_{i=1}^{n} |f(s_{i+1} - \delta/2) - f(s_i + \delta/2)| + 2\epsilon$$

Then by the mean value theorem applied to the intervals (s_i, s_{i+1}) we get

$$|f(y) - f(x)| = \sum_{i=0}^{n} K|s_{i+1} - s_i + \delta| + 2\epsilon$$

$$|f(y) - f(x)| = \sum_{i=0}^{n} K|s_{i+1} - s_i| + 2\epsilon + \delta$$

$$|f(y) - f(x)| = K|y - x| + 2\epsilon + \delta$$

(2 $\implies$ 1) Suppose by way of contradiction that f is $K-$Lipschitz and there exists a point x at which f is differentiable, where $|f'(x)|K' > K$. By definition of the derivative we can take a sequence $x_i \to x$, and we must have

$$K' = |f'(x)| = \lim_{i \to \infty} \frac{|f(x_i) - f(x)|}{|x_i - x|}$$

By taking suitably large i we obtain a pair x_i, x which contradicts Lipschitzness of f.

Proof (2.4.5): For all $x_1, x_2 \in \mathbb{R}^n$, we can set $x := x_1 - x_2$, and the Lipschitz definition reduces by linearity of A.

$$\begin{aligned}
\sup_{x_1, x_2 \in \mathbb{R}^n} \frac{\|Ax_1 - Ax_2\|}{\|x_1 - x_2\|} &= \sup_{x_1, x_2 \in \mathbb{R}^n} \frac{\|A(x_1 - x_2)\|}{\|x_1 - x_2\|} \\
&= \sup_{x \in \mathbb{R}^n} \frac{\|Ax\|}{\|x\|} \\
&= \sigma(A)
\end{aligned}$$

Lemma A.2.1: A linearity with weight matrix $W \in [-c, c]^{n \cdot m}$ has Lipschitz constant bounded by c^2nm

Proof: For any z we have $z^T W^T W z = \|Wz\|_2^2 \geq 0$, so $W^T W$ is positive semidefinite. Therefore its eigenvalues are nonnegative. The trace of the matrix $W^T W$ is equal to the sum of its eigenvalues. Let λ_i be the eigenvalues of $W^T W$ ordered by size without loss of generality. Then

$$\sigma(W)^2 = \lambda_0 \leq \sum \lambda_i = \text{tr}(W^T W) = \|W\|_2^2 \leq c^2 nm$$

Thus

$$\sigma(W) \leq c\sqrt{nm}$$

Lemma A.2.2: Weight normalization ((Salimans and Kingma, 2016)) bounds the Lipschitz constant of a network.

Proof: To check that weight normalization bounds the Lipschitz constant, we just note it bounds the Frobenius norm by $\sqrt{n}$, where n is the number of rows. Since the Frobenius norm bounds $\sigma(W)$, we are done.

Appendix B

Spectral Plots

In all cases, the red baseline is the unsupervised spectral normalization model we introduced as the first model of chapter 4.

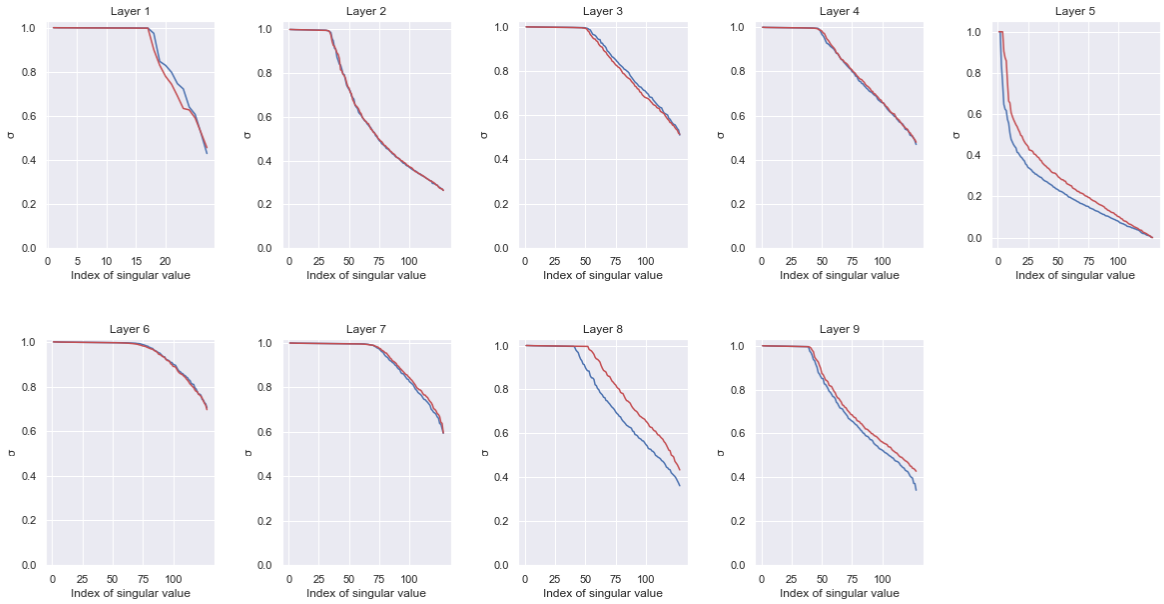

FIGURE B.1: Spectral plots of discriminator weights with $r(W)$ penalty in generator (red baseline)

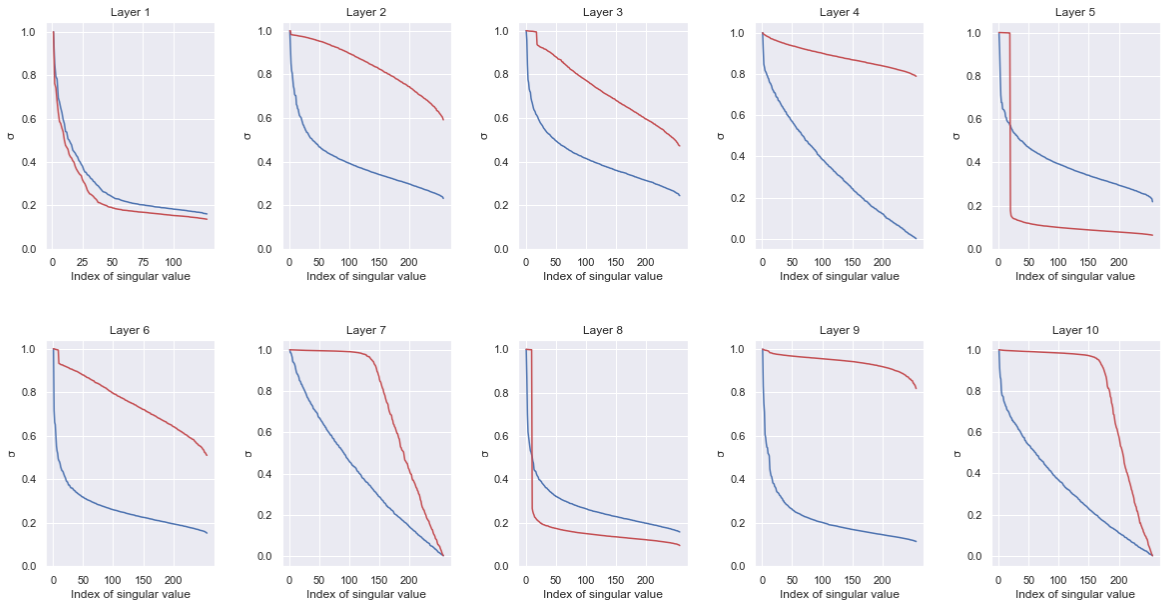

FIGURE B.2: Spectral plots of generator weights with $r(W)$ penalty in generator (red baseline)

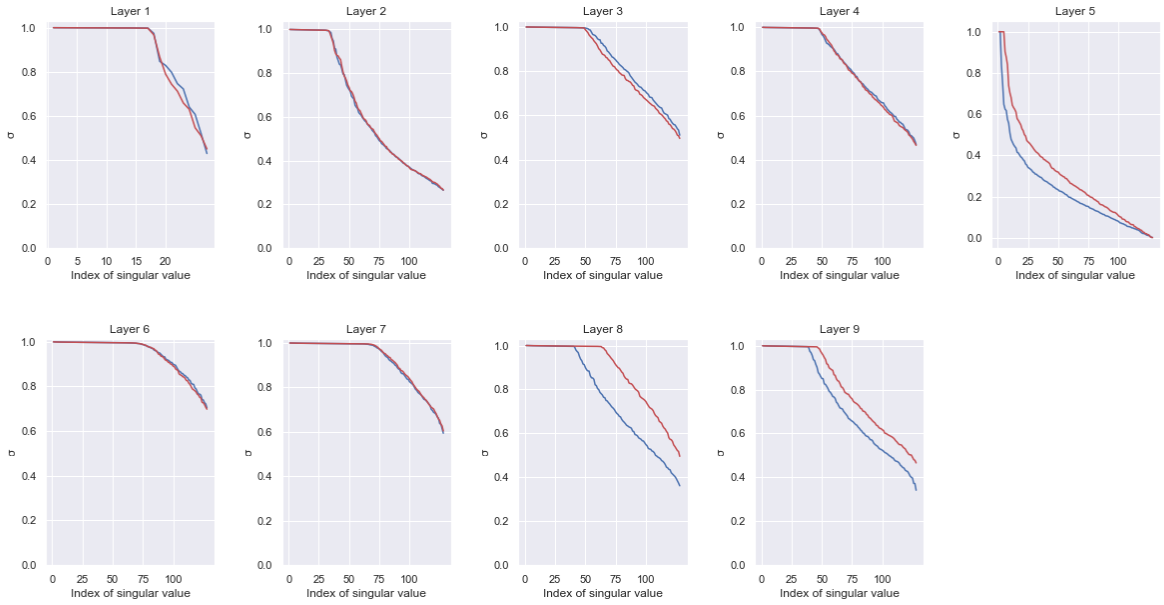

FIGURE B.3: Spectral plots of discriminator weights with spectral clamping applied in discriminator at $r = 15$ (red baseline)

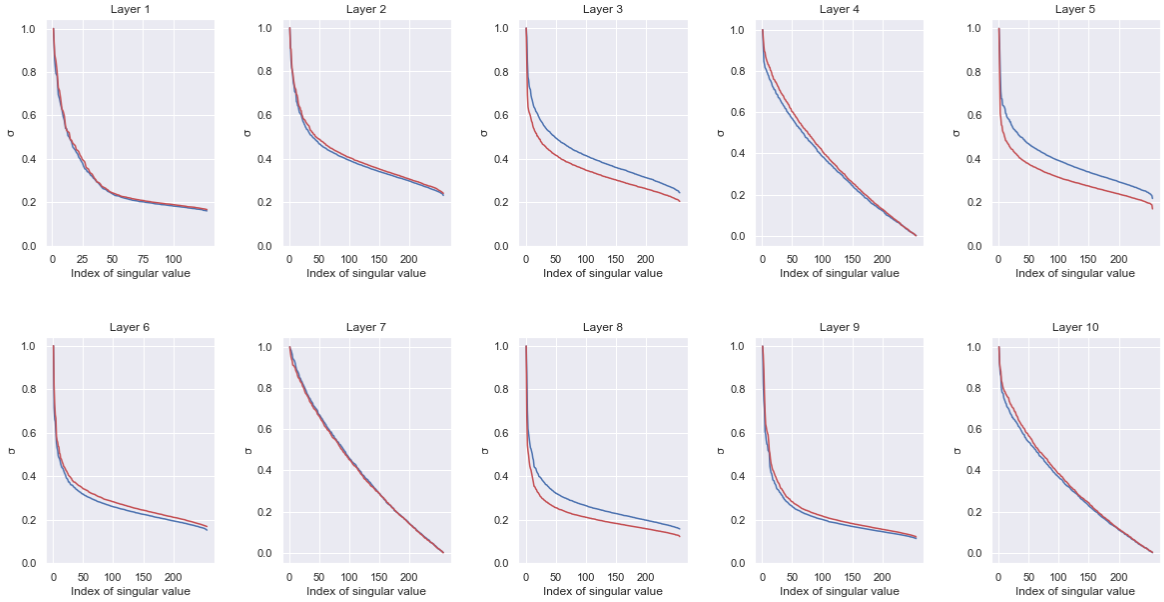

FIGURE B.4: Spectral plots of generator weights with spectral clamping applied in discriminator at $r = 15$ (red baseline)

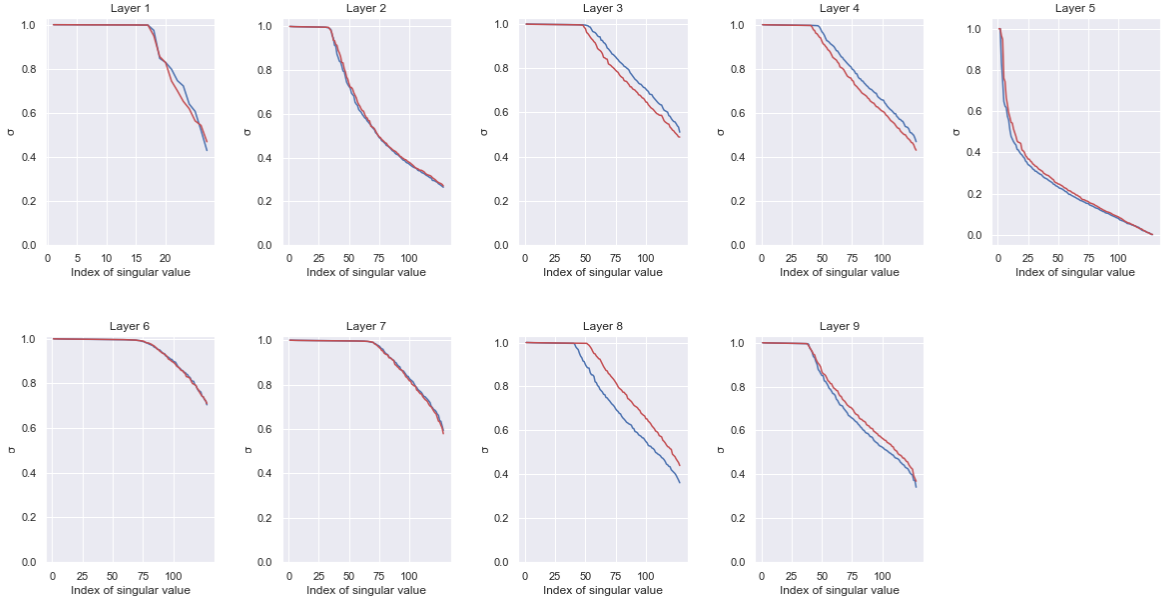

FIGURE B.5: Spectral plots of discriminator weights with spectral clamping applied in generator at $r = 1.5$ (red baseline)

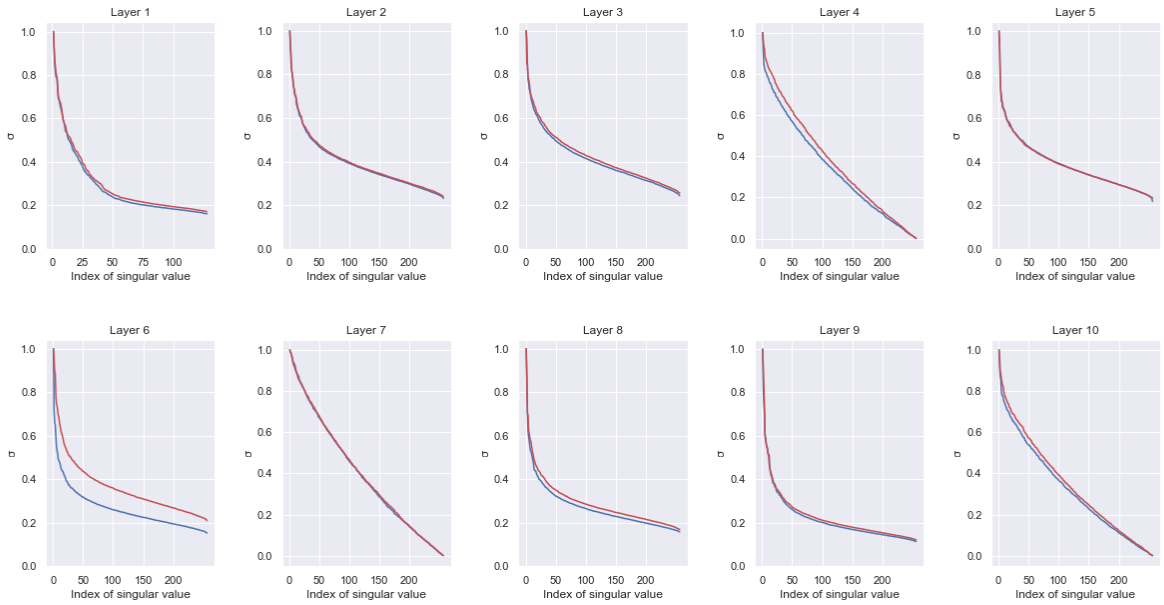

FIGURE B.6: Spectral plots of generator weights with spectral clamping applied in generator at $r = 1.5$ (red baseline)

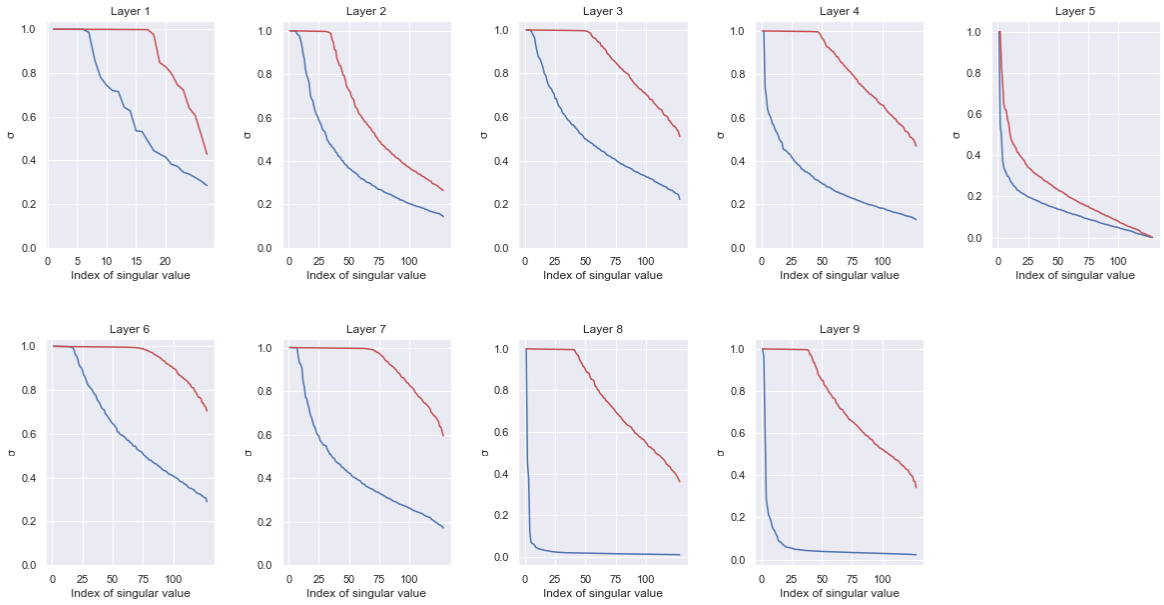

FIGURE B.7: Spectral plots of discriminator weights with gradient penalty applied in generator at $\lambda = 10$ (red baseline)

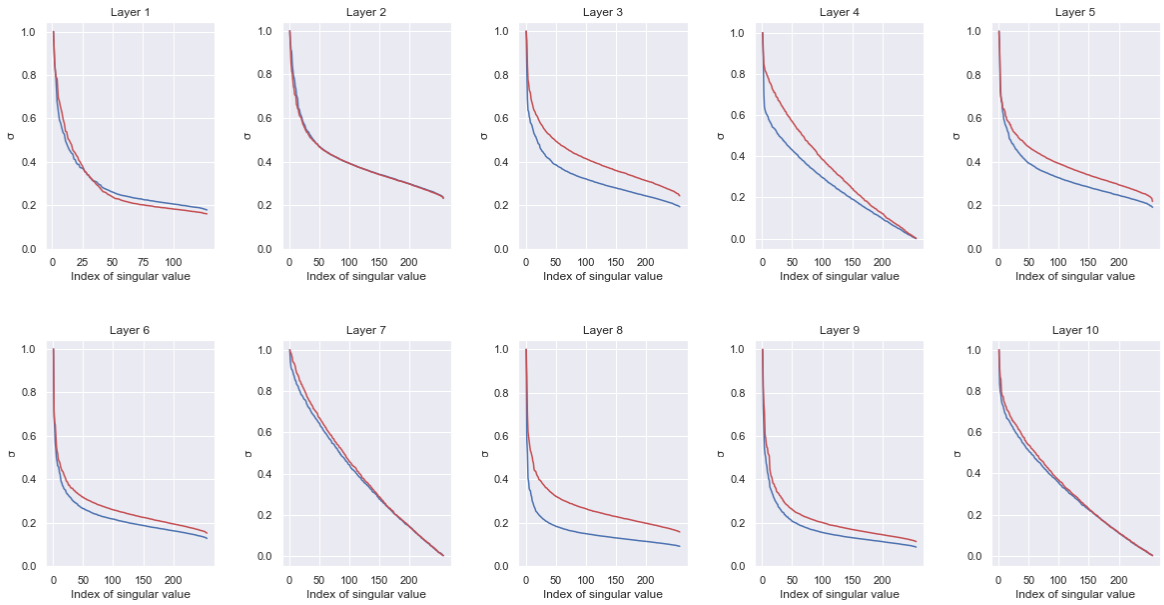

FIGURE B.8: Spectral plots of generator weights with gradient penalty applied to discriminator at $\lambda = 10$ (red baseline)

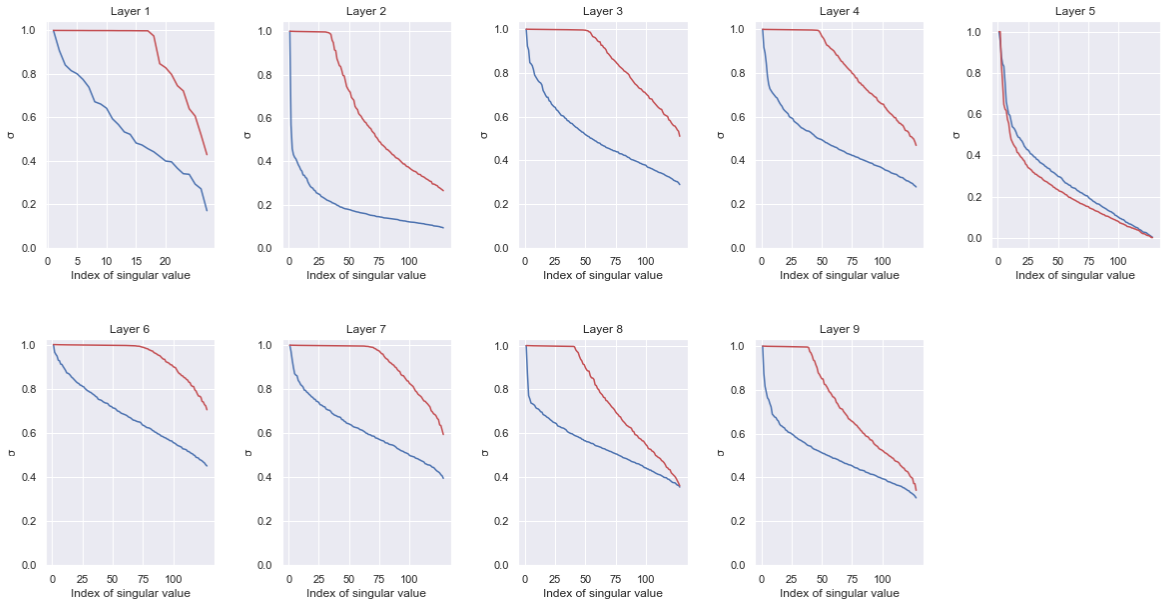

FIGURE B.9: Spectral plots of discriminator weights of a model without spectral normalization (red baseline)

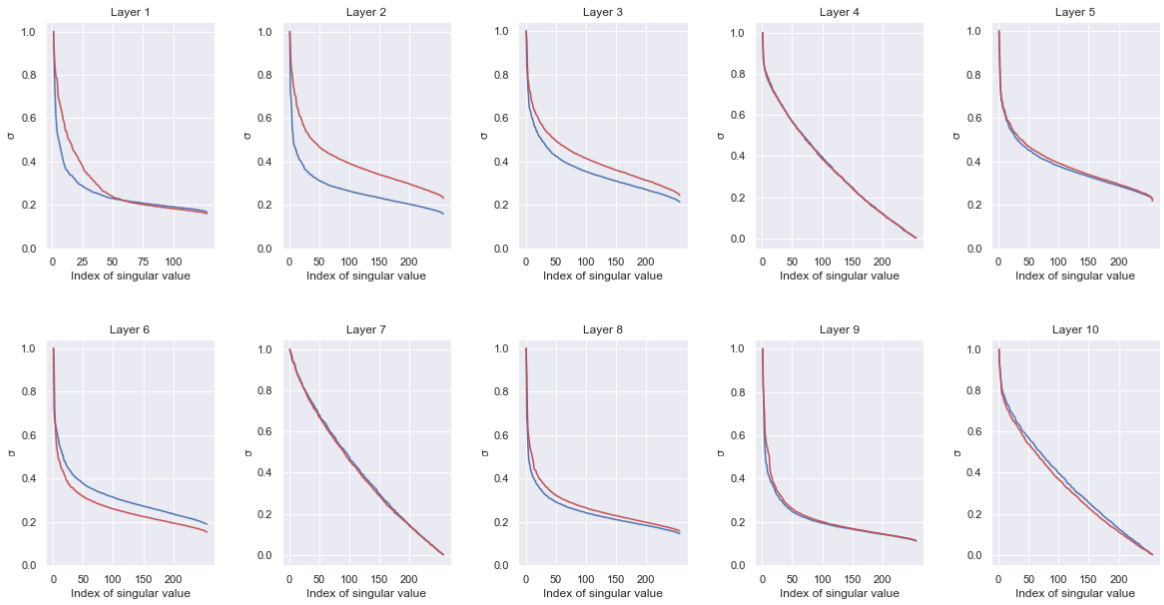

FIGURE B.10: Spectral plots of generator weights of a model without spectral normalization (red baseline)

Appendix C

Samples and Interpolations

FIGURE C.1: Samples from the baseline model

FIGURE C.2: More samples from the baseline model

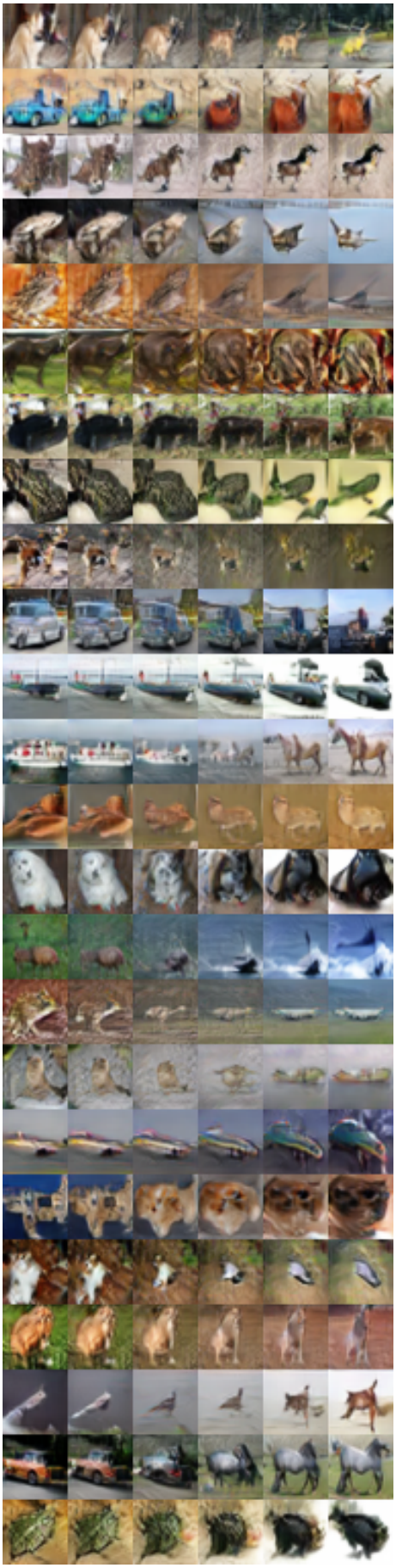

FIGURE C.3: Some interpolations from the baseline model

FIGURE C.4: A close-up of some interpolations from the baseline model

Bibliography

Arjovsky, Martin and Leon Bottou (2017). "Towards Principled Methods for Training Generative Adversarial Networks". In: *CoRR* abs/1704.00028. arXiv: 1704.00028. URL: http://arxiv.org/abs/1704.00028.

Arjovsky, Martin et al. (2017). "Wasserstein GAN". In: URL: https://arxiv.org/pdf/1701.07875.pdf.

Axler, Sheldon (1997). "Linear Algebra Done Right". In:

Brock, Andrew et al. (2018). "Large Scale GAN Training for High Fidelity Natural Image Synthesis". In: *CoRR* abs/1809.11096. arXiv: 1809.11096. URL: http://arxiv.org/abs/1809.11096.

Dumoulin, Vincent and Francesco Visin (2016). "A guide to convolution arithmetic for deep learning". In:

Goodfellow, Ian et al. (2014). "Generative Adversarial Nets". In: ed. by Z. Ghahramani et al., pp. 2672–2680. URL: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

Goodfellow, Ian J. (2017). "NIPS 2016 Tutorial: Generative Adversarial Networks". In: *CoRR* abs/1701.00160. arXiv: 1701.00160. URL: http://arxiv.org/abs/1701.00160.

Gulrajani, Ishaan et al. (2017). "Improved Training of Wasserstein GANs". In: *CoRR* abs/1704.00028. arXiv: 1704.00028. URL: http://arxiv.org/abs/1704.00028.

He, Kaiming et al. (2015). "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385. arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

Herrmann, Vincent (2017). "Wasserstein GAN and the Kantorovich-Rubinstein Duality". In: URL: https://vincentherrmann.github.io/blog/wasserstein/.

Kingma, Diederik P. and Jimmy Ba (2014). "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980. arXiv: 1412.6980. URL: http://arxiv.org/abs/1412.6980.

Kingma, Diederik P and Max Welling (2014). "Auto-encoding Variational Bayes". In: *ICLR*. URL: https://arxiv.org/pdf/1606.05908.pdf.

Krizhevsky, Alex (2009). "Learning Multiple Layers of Features from Tiny Images". In: URL: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

Larochelle, Hugo and Iain Murray (2011). "Neural Autoregressive Density Estimators". In: *AISTATS* 15. arXiv: 1701.00160. URL: https://arxiv.org/pdf/1701.00160.pdf.

Mescheder, Lars M. (2018). "On the convergence properties of GAN training". In: *CoRR* abs/1801.04406. arXiv: 1801.04406. URL: http://arxiv.org/abs/1801.04406.

Mirza, Mehdi and Simon Osindero (2014). "Conditional Generative Adversarial Nets". In: *CoRR* abs/1411.1784. arXiv: 1411.1784. URL: http://arxiv.org/abs/1411.1784.

Miyato, Takeru et al. (2018). "Spectral Normalization for Generative Adversarial Networks". In: *CoRR* abs/1802.05957. arXiv: 1802.05957. URL: http://arxiv.org/abs/1802.05957.

Nowozin, Sebastian et al. (2016). "f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization". In: URL: https://arxiv.org/abs/1606.00709.

Pascanu, Razvan et al. (2012). "Understanding the exploding gradient problem". In: *CoRR* abs/1211.5063. arXiv: 1211.5063. URL: http://arxiv.org/abs/1211.5063.

Radford, Alec et al. (2015). "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". In: *CoRR* abs/1511.06434. arXiv: 1511.06434. URL: http://arxiv.org/abs/1511.06434.

Salimans, Tim and Diederik P. Kingma (2016). "Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks". In: *CoRR* abs/1602.07868. arXiv: 1602.07868. URL: http://arxiv.org/abs/1602.07868.

Salimans, Tim et al. (2016). "Improved Techniques for Training GANs". In: *CoRR* abs/1606.03498. arXiv: 1606.03498. URL: http://arxiv.org/abs/1606.03498.

Tsuzuku, Yusuke et al. (2018). "Lipschitz-Margin Training: Scalable Certification of Perturbation Invariance for Deep Neural Networks". In: *CoRR* abs/1802.04034. arXiv: 1802.04034. URL: http://arxiv.org/abs/1802.04034.

Villani, Cedric (2009). "Optimal Transport: Old and New". In:

Yuichi Yoshida, Takeru Miyato (2017). "Spectral Norm Regularization for Improving the Generalizability of Deep Learning". In: URL: https://arxiv.org/abs/1705.10941.

Zhang, Han et al. (2018). "Self-Attention Generative Adversarial Networks". In: URL: https://arxiv.org/pdf/1805.08318.pdf.