A Comparative Assessment of the Impact of Various Norms on Wasserstein Generative Adversarial Networks

Chandini Ramesh
cr4383@rit.edu

Follow this and additional works at: https://scholarworks.rit.edu/theses

Recommended Citation

This Thesis is brought to you for free and open access by RIT Scholar Works. It has been accepted for inclusion in Theses by an authorized administrator of RIT Scholar Works. For more information, please contact ritscholarworks@rit.edu.
A Comparative Assessment of the Impact of Various Norms on Wasserstein Generative Adversarial Networks

APPROVED BY

SUPERVISING COMMITTEE:

Dr. Reynold Bailey, Supervisor

Dr. Leon Reznik, Reader

Dr. Ernest Fokoue, Observer
A Comparative Assessment of the Impact of Various Norms on Wasserstein Generative Adversarial Networks

by

Chandini Ramesh

THESIS
Presented to the Faculty of the Department of Computer Science
Golisano College of Computer and Information Sciences
Rochester Institute of Technology

in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

Rochester Institute of Technology
May 2019
Acknowledgments

I want to thank my adviser Dr. Ernest Fukuoe for his guidance, support, and blessings throughout my thesis. I want to thank Dr. Reynold Bailey and Dr. Leon Reznik for serving as my committee members and for their time, valuable feedback and effort in assisting my research. I would like to thank everyone in the Computer Science Department for giving me the opportunity to pursue my thesis.
To my loving family for their support. Friends who motivated me not to take up thesis. Faiz Ur Rahman for his constant annoyance. To my seniors Preethi Vaidyanathan, Suhas Pillai and Anil Kumar Behera whose guidance proved crucial to embark this thesis...
Abstract

A Comparative Assessment of the Impact of Various Norms on Wasserstein Generative Adversarial Networks

Chandini Ramesh, M.S.
Rochester Institute of Technology, 2019

Supervisor: Dr. Reynold Bailey

Generative Adversarial Networks (GANs) provide a fascinating new paradigm in machine learning and artificial intelligence, especially in the context of unsupervised learning. GANs are quickly becoming a state of the art tool, used in various applications such as image generation, image super resolutions, text generation, text to image synthesis to name a few. However, GANs potential is restricted due to the various training difficulties. To overcome the training difficulties of GANs, the use of a more powerful measure of dissimilarity via the use of the Wasserstein distance was proposed. Thereby giving birth to the GAN extension known as Wasserstein Generative Adversarial Networks (WGANs).
Recognizing the crucial and central role played by both the cost function and the order of the Wasserstein distance used in WGAN, this thesis seeks to provide a comparative assessment of the effect of a various common used norms on WGANs. Inspired by the impact of norms like the $\ell_1$ norm in LASSO Regression, the $\ell_2$ norm Ridge Regression and the great success of the combination of the $\ell_1$ and $\ell_2$ norms in elastic net and its extensions in statistical machine learning, we consider exploring and investigating to a relatively large extent, the effect of these very same norms in the WGAN context. In this thesis, the primary goal of our research is to study the impact of these norms on WGANs from a pure computational and empirical standpoint, with an emphasis on how each norm impacts the space of the weights/parameters of the machines contributing to the WGAN. We also explore the effect of different clipping values which are used to enforce the $k$-Lipschitz constraint on the functions making up the specific WGAN under consideration. Another crucial component of the research carried out in this thesis focuses on the impact of the number of training iterations on the WGAN loss function (objective function) which somehow gives us an empirical rough estimate of the computational complexity of WGANs. Finally, and quite importantly, in keeping WGANs’ application to recovery of scenes and reconstruction of complex images, we dedicate a relative important part of our research to the comparison of the quality of recovery across various choices of the norms considered. Like previous researchers before us, we perform a substantial empirical exploration on both synthetic data and real life data. We specifically explore a simulated data
set made up of a mixture of 8 bivariate Gaussian random variables with large
gaps, the likes of which would be hard task for traditional GANs but can be
readily handled quite well be WGANs thanks to the inherent strength/power
of the underlying Wasserstein distance. We also explore various real data sets,
namely the ubiquitous MNIST datasets made up of handwritten digits and
the now very popular CIFAR-10 dataset used an de facto accepted benchmark
data set for GANs and WGANs. For all the datasets, synthetic and real, we
provide a thorough comparative assessment of the effect and impact of the
norms mentioned earlier, and it can be readily observed that there are indeed
qualitative and quantitative difference from one norm to another, with dif-
fences established via measures such as (a) shape, value and pattern of the
generator loss, (b) shape, value and pattern of the discriminator loss (c) shape,
value and pattern of the inception score, and (d) human inspection of quality
of recovery or reconstruction of images and scenes.
# Table of Contents

Acknowledgments iii

Abstract v

List of Tables xi

List of Figures xii

Chapter 1. Introduction 1
  1.1 Machine learning ................................. 1
  1.2 Supervised Learning vs. Unsupervised Learning .......... 3
  1.3 Reinforcement Learning ............................ 5
  1.4 Generative Modeling vs. Discriminative Modeling ......... 6
  1.5 Parametric Learning vs Nonparametric Learning .......... 8
  1.6 Generative Adversarial Networks (GANs) ................. 10
  1.7 Research ........................................ 15
    1.7.1 Effect of Cost functions on WGANs ............... 16
    1.7.2 Effect of the Clipping values in WGANs .......... 18
    1.7.3 Training of WGANs .............................. 19
    1.7.4 Experiments and Results ........................ 19
    1.7.5 Thesis Organization ............................ 20

Chapter 2. Background 21
  2.1 Introduction to Generative Adversarial Networks .......... 21
    2.1.1 Architecture of a Generative Adversarial Networks 22
  2.2 Training of a Generative Adversarial Networks .......... 23
  2.3 Applications of Generative Adversarial Networks .......... 24
  2.4 Challenges with GANs ................................ 26
  2.5 Motivation of Wasserstein Generative Adversarial Networks 27
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3.3 ℓ₁ Norm vs. ℓ_{enet}-Norm</td>
<td>68</td>
</tr>
<tr>
<td>4.3.3.1 Data set : MNIST</td>
<td>68</td>
</tr>
<tr>
<td>4.3.3.2 Data set : CIFAR-10</td>
<td>70</td>
</tr>
<tr>
<td>4.3.4 Clipping Values</td>
<td>73</td>
</tr>
<tr>
<td>4.3.4.1 MNIST - clipping values 0.1, 0.01, 0.001</td>
<td>73</td>
</tr>
<tr>
<td>4.3.4.2 CIFAR-10 - clipping values 0.1, 0.01, 0.001</td>
<td>76</td>
</tr>
<tr>
<td>4.3.4.3 Toy - 8 Gaussian - clipping values 0.01, 0.001</td>
<td>82</td>
</tr>
<tr>
<td>4.3.5 Number of iterations</td>
<td>84</td>
</tr>
</tbody>
</table>

Chapter 5.  Conclusion and Future Work  88

Bibliography  92
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Settings to obtain results for MNIST - $\ell_1$ norm and $\ell_2$ norm</td>
<td>54</td>
</tr>
<tr>
<td>4.2</td>
<td>Settings to obtain results for CIFAR-10 - $\ell_1$ and $\ell_2$ norm</td>
<td>57</td>
</tr>
<tr>
<td>4.3</td>
<td>Settings to obtain results for Toy - 8 Gaussian data using $\ell_1$ norm</td>
<td>60</td>
</tr>
<tr>
<td>4.4</td>
<td>Settings to obtain results for MNIST - $\ell_1$ norm and $\ell_\infty$ norm</td>
<td>62</td>
</tr>
<tr>
<td>4.5</td>
<td>Settings to obtain results for CIFAR-10 using $\ell_1$ and $\ell_\infty$ norm</td>
<td>64</td>
</tr>
<tr>
<td>4.6</td>
<td>Settings to obtain results for Toy - 8 Gaussian data using $\ell_1$ norm and $\ell_\infty$ norm</td>
<td>66</td>
</tr>
<tr>
<td>4.7</td>
<td>Settings to obtain results for MNIST using $\ell_1$ norm and $\ell_{enet}$ norm</td>
<td>68</td>
</tr>
<tr>
<td>4.8</td>
<td>Settings to obtain results for CIFAR-10 using $\ell_1$ and $\ell_{enet}$ norm</td>
<td>70</td>
</tr>
<tr>
<td>4.9</td>
<td>Settings to obtain results for different clipping values 0.1, 0.01 and 0.001 for MNIST data set</td>
<td>73</td>
</tr>
<tr>
<td>4.10</td>
<td>Settings to obtain results for different clipping values 0.1, 0.01 and 0.001 for CIFAR-10 data set</td>
<td>76</td>
</tr>
<tr>
<td>4.11</td>
<td>Settings to obtain results for Toy - 8 Gaussian data for clip value = 0.01</td>
<td>82</td>
</tr>
<tr>
<td>4.12</td>
<td>Settings to obtain results for MNIST, 160K iterations</td>
<td>84</td>
</tr>
</tbody>
</table>
# List of Figures

1.1 Tasks solved using Machine Learning ............................ 1  
1.2 Supervised Learning: Input data with class labels .......... 4  
1.3 Unsupervised Learning example input data without class labels 4  
1.4 Generative Model showing the two distributions of data points red and green. The model learns the distribution of the data shown in black dotted lines surrounding the data points. (Image Courtesy of Tu [57]) .................................................. 6  
1.5 Discriminative Model shows the data distribution in black and red. The model learns the class decision boundary separating the two classes in blue. (Image courtesy of Max [41]) .......... 7  
1.8 Text to Image GANs (Image Courtesy of Reed [46]) ........ 11  
1.6 Image generated by Progressive GANs from a Celeb-A data set (Image Courtesy of Karras [31]) ................................. 12  
1.7 Super Resolution GANs (Image Courtesy of Ledig [35]) ..... 13  
1.9 Due to the challenges in GANs, GANs face difficulty understand the global structure, perspective, and the count of the objects in an image ................................................................. 14  
1.10 Contours of LASSO vs Ridge on a 2-Dimensional parameter space where $\beta = (\beta_1, \beta_2)^T$ (Image Courtesy of Li [3]) ... 17  
1.11 Comparing the geometrical space of LASSO, ridge and elastic net Regression (Image Courtesy of Sosnovshchenko [53]) ... 18  
2.1 Architecture of a generative adversarial network. (Image courtesy [6]) ................................................................. 23  
3.1 Contours of LASSO vs Ridge on a 2-Dimensional parameter space where $\beta = (\beta_1, \beta_2)^T$ [3] ......................... 40  
3.2 $\ell_1$ norm geometric space (Image Courtesy [4]) ............ 42  
3.3 $\ell_2$ norm geometric space (Image Courtesy [4]) ............ 44  
3.4 $\ell_1$ norm + $\ell_2$ norm geometric space (Image Courtesy [7]) ... 45  
4.1 Sample from MNIST data set [32] .............................. 51
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2 Sample from CIFAR-10 data set, [2]</td>
<td>52</td>
</tr>
<tr>
<td>4.3 Sample from Toy - 8 Gaussian data set</td>
<td>53</td>
</tr>
<tr>
<td>4.4 $G$ and $D$ loss for MNIST - $\ell_1$ and $\ell_2$ norm</td>
<td>55</td>
</tr>
<tr>
<td>4.5 Recovery for MNIST - $\ell_1$ and $\ell_2$ norm</td>
<td>55</td>
</tr>
<tr>
<td>4.6 $D$ loss for CIFAR-10 - $\ell_1$ and $\ell_2$ norm</td>
<td>57</td>
</tr>
<tr>
<td>4.7 Inception score loss for CIFAR-10 - $\ell_1$ and $\ell_2$ norm</td>
<td>58</td>
</tr>
<tr>
<td>4.8 Recovery using CIFAR-10 - $\ell_1$ and $\ell_2$ norm</td>
<td>58</td>
</tr>
<tr>
<td>4.9 $D$ loss for Toy - 8 Gaussian - $\ell_1$ and $\ell_2$ norm</td>
<td>60</td>
</tr>
<tr>
<td>4.10 Recovery for Toy - 8 Gaussian - $\ell_1$ and $\ell_2$ norm</td>
<td>61</td>
</tr>
<tr>
<td>4.11 $G$ and $D$ loss for MNIST - $\ell_1$ and $\ell_\infty$ norm</td>
<td>62</td>
</tr>
<tr>
<td>4.12 Recovery for MNIST data set with $\ell_1$ and $\ell_\infty$ norm</td>
<td>63</td>
</tr>
<tr>
<td>4.13 $D$ loss for CIFAR-10 - $\ell_1$ and $\ell_\infty$ norm</td>
<td>64</td>
</tr>
<tr>
<td>4.14 Inception Score for CIFAR-10 - $\ell_1$ and $\ell_\infty$ norm</td>
<td>64</td>
</tr>
<tr>
<td>4.15 Recovery for CIFAR-10 - $\ell_1$ and $\ell_\infty$ norm</td>
<td>65</td>
</tr>
<tr>
<td>4.16 $D$ loss for Toy - 8 Gaussian - $\ell_1$ and $\ell_\infty$ norm</td>
<td>66</td>
</tr>
<tr>
<td>4.17 Recovery using $\ell_1$ and $\ell_\infty$ norm with Toy - 8 Gaussian data</td>
<td>67</td>
</tr>
<tr>
<td>4.18 $G$ and $D$ loss for MNIST - $\ell_1$ and $\ell_{enet}$ norm</td>
<td>69</td>
</tr>
<tr>
<td>4.19 Recovery for MNIST data set with $\ell_1$ and $\ell_{enet}$ norm</td>
<td>69</td>
</tr>
<tr>
<td>4.20 $D$ loss for CIFAR-10 - $\ell_1$ and $\ell_{enet}$ norm</td>
<td>70</td>
</tr>
<tr>
<td>4.21 Inception Score for CIFAR-10 - $\ell_{enet}$ norm</td>
<td>71</td>
</tr>
<tr>
<td>4.22 Recovery for CIFAR-10 - $\ell_1$ and $\ell_{enet}$ norm</td>
<td>71</td>
</tr>
<tr>
<td>4.23 The difference in the $D$ and $G$ loss corresponding to different clipping values for MNIST data set</td>
<td>74</td>
</tr>
<tr>
<td>4.24 The difference in the recovery corresponding to different clipping values for MNIST data set</td>
<td>75</td>
</tr>
<tr>
<td>4.25 The difference in the $D$ loss corresponding to different clipping values for CIFAR-10 data set</td>
<td>77</td>
</tr>
<tr>
<td>4.26 The difference in the inception Score corresponding to different clipping values for CIFAR-10 data set</td>
<td>78</td>
</tr>
<tr>
<td>4.27 The difference in the recovery corresponding to different clipping values for CIFAR-10 data set</td>
<td>79</td>
</tr>
<tr>
<td>4.28 The difference in the $D$ and $G$ loss corresponding to different clipping values for MNIST data set</td>
<td>82</td>
</tr>
</tbody>
</table>
4.29 The difference in the $D$ and $G$ loss corresponding to different clipping values for MNIST data set . . . . . . . . . . . . . . . . . . . . 83
4.30 The difference in the $D$ and $G$ loss corresponding to different number of training iterations for MNIST data set . . . . . . . 85
4.31 The difference in the recovery corresponding to different number of training iterations for MNIST data set . . . . . . . . . . . . . . . . . . . . 86
Chapter 1

Introduction

1.1 Machine learning

In Figure 1.1 problems such as recognizing the handwritten digits, object detection from a scene, recognizing whether a given image is of a cat and face recognition are shown. These are a few challenging problems which are
difficult to solve by writing a specific computer program. These problems are challenging because there is no set of finite rules that would guide the computer program leading to a solution which would solve the problem. The number of rules and exceptions would grow exponentially making it a cumbersome task. Problems like these are easier to solve using machine learning [24], [61], [39], [25].

Machine Learning uses mathematical models to learn from the data and improve the model’s performance over time.

Mitchell describes machine learning as “A computer program is said to learn from experience ‘E’, with respect to some class of tasks ‘T’ and performance measure ‘P’ if its performance at tasks in ‘T’ as measured by ‘P’ improves with experience ‘E’ [42].” Consider a task of recognizing handwritten digits. Here the task ‘T’ is to recognize the handwritten digits and the experience ‘E’ is the process by which the model learns from looking at many samples of these digits. The performance ‘P’ of the model is the success of the model’s performance on recognizing the given handwritten digits.

More formally, let’s consider data set $\mathcal{D} = \{(X_i, Y_i) \sim \mathbb{P}_{xy}(x, y), i = 1, \cdots, n\}$, where $X_i \in \mathcal{X}, i = 1, \cdots, n$. $\mathcal{X}$ is the input space and $Y_i \in \mathcal{Y}, i = 1, \cdots, n$. $\mathcal{Y}$ is the output space. The samples are drawn from identically independent distribution $(iid), \mathbb{P}_{xy}(x, y)$. $\mathbb{P}_{xy}(x, y)$ is the probability distribution of $(X, Y)$. 
In machine learning given a task, we try to learn function mapping $f$.

$$f : \mathcal{X} \rightarrow \mathcal{Y}$$

$$x \mapsto f(x) \in \mathcal{Y}$$

The model learns a function $f$, that maps the input from the input space $\mathcal{X}$ to output space $\mathcal{Y}$. For a new observation $x$, $f(x)$ is used to find the mapping $y$ such that it belongs to $\mathcal{Y}$. This $f$ is called the generic mapping. The loss is calculated by $\ell(y, f(x))$ and is measured as the loss incurred between the model prediction $f(x)$ and the actual true label $y$ in $\mathcal{Y}$. The expected value of the loss, is called the risk functional of $f$ and is given by Eq. 1.1

$$\mathbb{R}(f) = \mathbb{E}[\ell(y, f(X))] = \int_{\mathcal{X} \times \mathcal{Y}} \ell(y, f(x))d\mathbb{P}(x, y) \quad (1.1)$$

In machine learning for any task, the goal is to find $f^*$ of all the functions in $\mathcal{Y}^\mathcal{X}$, which is the function that minimizes the risk. The function $f^*$ over the space $\mathcal{X}^\mathcal{Y}$ of all the measurable functions from $\mathcal{X}$ to $\mathcal{Y}$ is given by Eq. 1.2:

$$f^* = \arg \inf_f \mathbb{R}(f) \quad (1.2)$$

$$\mathbb{R}(f^*) = \mathbb{R}^* = \inf_f \mathbb{R}(f)$$

### 1.2 Supervised Learning vs. Unsupervised Learning

In machine learning, there are two types of learning algorithms. They have supervised learning and unsupervised learning algorithms. In supervised
learning, the mathematical model is given the data with both the input and the output variables i.e, $\{x_i, y_i\}_{i=1}^{n}$.

![Figure 1.2: Supervised Learning : Input data with class labels](image)

The model tries to learn the relationship between $x$ and $y$. In supervised learning, we perform classification or regression. In classification, the target variables is usually a finite set consisting of finite variables. Usually, they are assigned a categorical variable. Whereas, in regression, the output predicted is often a real number and belongs to continuous variables. As shown in the Figure 1.2 an input with the handwritten digit seven, is given a class label *seven*, and the image of a cat is given the class label *cat* during the training. Algorithms such as K-nearest neighbors, Decision Trees, Random Forest, Support Vector Machines are supervised learning algorithms.

![Figure 1.3: Unsupervised Learning example input data without class labels](image)

In contrast to supervised learning, unsupervised learning methods are
more challenging. When the model is trained, with the input variables \( \{x_i\}_{i=1}^n \) there is no response variable \( y \) given. Therefore, to make some sense out in this setting we need to find the relationship between the observations. One of the ways is cluster analysis. Here we try to group the input variables into different groups. Along with these in unsupervised learning, we have density estimation methods where the model tries to determine the distribution of the input data. Unsupervised learning can also be used when we would want to project the data from the higher dimensional space to the lower dimensional space for visualization. As shown in Figure 1.3 an input with the handwritten digit seven, is not given the label, similarly, with an image of a cat, only the image is provided during the learning without the class label. Algorithms such as principal component analysis and K-means clustering are examples of unsupervised learning machines.

### 1.3 Reinforcement Learning

In reinforcement learning method usually an agent is considered, which learns in an interactive environment. They are guided with set of states, and actions. With every correct action the agent is rewarded and with incorrect action it is penalized. In reinforcement learning we usually have a feedback loop. This feedback loop provides a set of correct actions that should have been performed when the agent commits a mistake. The goal is to collect the maximum reward for the correct set of actions. It finds its applications in robotics, game theory and self driving cars.
1.4 Generative Modeling vs. Discriminative Modeling

In machine learning, we have generative and discriminative modeling algorithms. The difference lies in the main focus and ingredients used to build the learning machines.

![Figure 1.4: Generative Model showing the two distributions of data points red and green. The model learns the distribution of the data shown in black dotted lines surrounding the data points. (Image Courtesy of Tu [57])](image_url)

\[
f(x^{\text{new}}) = \arg \max_{g=1,\ldots,G} \{ p(Y = (g|x^{\text{new}})) \} \tag{1.3}
= \arg \max_{g=1,\ldots,G} \left\{ \frac{p(Y = g)p(x^{\text{new}}|y = g)}{p(x^{\text{new}})} \right\}
\]

In some cases, machine learning models learn by attempting to reconstruct or infer the distribution from which the data was generated. With generative learning machines as shown in Figure 1.4, the focus is explicitly on learning the distribution of the data and not directly the surface that separates
the classes/groups, even though that is implicit in the distribution. In other words, generative machines seek to learn how the members of each class were generated as depicted in the Eq. 1.3. From a theoretical and methodological statistics perspective, generative learning typically involves extensive density estimation. If the learning machine of interest is predicated on learning the distribution of the data, it is Generative. In Bayesian learning, the $f(x^{new})$ is the label that has the largest posterior probability of class membership. In machine learning such generative learning works by estimating the class conditional densities $p(x|y = g)$ where $g = 1, \ldots, G$ which are generators of the data for each class. Such machines are said to be Generative. Examples include Naive Bayes, Gaussian Mixture models, Latent Dirichlet allocation.

![Figure 1.5: Discriminative Model](image.png)

Discriminative modeling does not explicitly/directly focus on the distribution of the data. Instead, discriminative models directly and explicitly aim
at learning the decision boundaries that best separates the data observations into the underlying classes as shown in Figure 1.5. Every class \( j \) will have a discriminant function \( \delta_j(x) \). Given the input \( x \) the reasonable definition of \( f \) can be formulated as

\[
f(x) = \arg \max_{j \in \{1, \ldots, k\}} \{\delta_j(x)\}
\]

The decision boundary between classes \( j \) and \( l \) is the set

\[
\{x \in \mathbb{R}^q : \delta_j(x) = \delta_l(x), \forall j \neq l\}
\]

(1.5)

For the new observation \( x \), function \( f \) can be estimated using \( \hat{f} \)

\[
\hat{f}(x) = \arg \max_{j \in \{1, \ldots, k\}} \{\hat{\delta}_j(x)\}
\]

(1.6)

\[
f(x^{\text{new}}) = \text{sign} \left( \sum_{i=1}^{n} \alpha_i y_i K(x_j x_i) + b \right)
\]

(1.7)

The SVM binary classifier as in Eq. (1.7) provides an example of a discriminative learning machine since the explicit focus is on learning the decision boundary without directly attempting to learn the distribution of the data.

1.5 Parametric Learning vs Nonparametric Learning

Typically the function \( f \) is \( f : X \to Y \). When this function is driven by set of parameters \( \Theta \), we say \( f(x) := f(x; \Theta) \) is a parametric function. They learn a set of parameters while modeling. In parametric learning, the number of parameters learned is finite, and there is almost always an assumption
made about the shape of underlying $f$. Logistic Regression, Linear Discriminant Analysis, Perceptron, Naive Bayes and Artificial Neural Networks are all examples of parametric learning machines.

Consider a neural network, the $q$ vectors $\beta_1, \cdots, \beta_q$ are the $p$-dimensional projection vectors. The $q$ scalars $\gamma_1, \cdots, \gamma_q$ scalars are the coefficients. The $q$ scalars $\beta_{01}, \cdots, \beta_{0q}$ scalars are the shifts. $\gamma_0$ is known as intercept. These are the parameters of a neural network. Generally known as weights of the neural network, represented by $\mathcal{W} = \{\beta_j, \gamma_j, \beta_{0j}, \gamma_0\}$

$$\text{net}(x, \cdot) = \mu(x) = \phi(\eta(x))$$

(1.8)

Eq. 1.8 represents a neural network [20]. The linear combination of the nodes is $\mu(x)$ and the output is given to the non linear function $\phi(\cdot)$

Nonparametric learning is more powerful because it can model complex decision boundaries. While dealing with a non-parametric method, the model does not make any assumptions about the shape of $f$, and typically makes substantially fewer assumptions, the underlying spirit of the paradigm being to allow for more adaptable and adaptive functions as representations of the data. Essentially, nonparametric methods are designed to perform estimation with an inherent potential to cover a more extensive range of different shapes of $f$. In nonparametric modeling, the number of quantities to estimate (sometimes also referred to as parameters) grows with the sample size. For this reason, nonparametric models are often called infinite-dimensional models. Examples
include K-nearest Neighbors learning machines, Classification and Decision Trees learners, Support Vector Machines and Gaussian processes, to name a few. Although not having to make any assumptions about the underlying function may appear as an absolute advantage of nonparametric learners, it must be said that the price to pay comes in the form of a large (potentially infinite) number of parameters to be estimated. It is worth mentioning that nonparametric learning typically requires a vast number of observations to estimate \( f \), and typically suffers from the so-called curse of dimensionality.

In the subsequent sections, we will be focusing on the emerging class of learning machines known as Generative Adversarial Networks, featuring a combination of the generative and discriminative ideas mentioned earlier, along with many aspects of parametric modeling of functions of interest.

### 1.6 Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs) [22] provide a fascinating new paradigm in machine learning and artificial intelligence, especially in the context of unsupervised learning. GANs are quickly becoming a state of the art tool, used in various applications such as image generation, image synthesis, super image resolutions, text generation, text to image synthesis. The generic goal is to generate the samples which are as similar as possible to the real data. Hence the appellation of generative networks. The architecture of a GAN consists of two neural networks namely, the Generator(Imitator) \( G \) and the Discriminator(Critic) \( D \) featuring in the adversarial framework. In the
original formulation of GANs, the idea is that two neural networks\textsuperscript{1} would compete with each other, such that the Generator(Imitator) would generate the samples trying to imitate the real data distribution by fooling the Discriminator(critic) $D$, which would try to identify if the samples generated were real or fake.

As stated by Ian Goodfellow in [22], the training of a GAN corresponds to a minimax game between the $D$ and $G$ formalized [22] in the following objective function:

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

$$= \mathbb{E}_{x \sim p_{\text{data}}} \left[ \log D_D^*(x) \right] + \mathbb{E}_{z \sim p_z} \left[ \log (1 - D_D^*(G(z))) \right]$$

Here the real data is denoted by $x$ and the generator learns this starting with the input noise $z$, and the distance between the resulting samples and the original data is measured using Jensen-Shannon Divergence (JSD) [19] between the samples from $G$ and $D$ to evaluate the performance of $D$ and $G$.

---

\textsuperscript{1}The learning machines involved in a GAN need not be neural networks. Any other learning machines can be used/considered.
GANs are said to be a universal simulator. One of the recent advances include progressive GANs [31]. In this work, the generator and the discriminator grow progressively and generates the high-quality images as shown in figure 1.6. Super-resolution generative adversarial networks (SR-GAN) [35] captures the finer texture details of an image and recover photo-realistic textures as shown in one of the samples in Figure 1.7. Another interesting application includes the GANs capability to convert the texts to images [46], with this GAN given a phrase describing an image the GAN is capable of generating the image from the textual description an example of which is shown in Figure 1.8.

These were just a few examples of the applications of GANs. Though
GANs have numerous applications, but the full inherent potential has not been harnessed due to various challenges. Training of GANs are extremely difficult. In the field of research for GANs, and the entire study is dedicated to making the training efficient. GANs suffer from training difficulties which includes the problem of mode collapse and vanishing gradients. In the early development stages of GANs, they were poor at understanding the global structure, perspective in the images. They also had difficulties generating the image with multiple objects an example is shown in the Figure 1.9.

To overcome these training difficulties, Martín Arjovsky proposed the use of a more powerful measure of dissimilarity via the use of the Wasserstein distance, thereby giving birth to the GAN extension known as Wasserstein Generative Adversarial Networks (WGANs) [11]. The motivation for using WGANs was to introduce a new more robust and more powerful way to train the original GAN, while also providing a theoretical background on how dif-
Figure 1.9: Due to the challenges in GANs, GANs face difficulty understand the global structure, perspective, and the count of the objects in an image. Different distance measures impact the convergence in the training of the model. They compared various distances such as total variation, Jensen-Shannon (JS) divergence, Kullback-Leibler (KL) divergence and the Wasserstein Distance (Earth movers distance), and found the Wasserstein Distance to be a advantageous cost function than the Jensen-Shannon (JS) divergence which was used in GANs until then.

\[
W(P_r, P_g) = \frac{1}{k} \sup_{\|f\|_L \leq k} \mathbb{E}_{x \sim P_r}[f(x)] - \mathbb{E}_{x \sim P_g}[f(x)],
\]  

(1.10)

\[
W(P_r, P_g) = \max_{w \in W} \mathbb{E}_{x \sim P_r}[f_w(x)] - \mathbb{E}_{z \sim P_g(z)}[f_w(G(z))]
\]  

(1.11)
The $k$-Lipschitz constraint is enforced by weight clipping applied to the function $f$, which essentially limits the weights learned by the discriminator to a compact space. It is important to note that this compactification resulted in greater stability in the training of GAN with the yielding of more meaningful learning curves.

1.7 Research

The goal of our research is to investigate the different cost functions of the WGANs, which for our purposes will be induced norms. We specifically consider different norms such as (a) $\ell_1$ norm, (b) $\ell_2$ norm (c) $\ell_\infty$ norm, (d) and inspired by the great success of elastic net in statistical regression and classification, we also consider the linear combination of $\ell_1$ and $\ell_2$. Primary goal of our research is to study the impact of these norms, along with this we also explore the effect of training the WGANs for relatively larger number of iterations. In this case 1 Million. Furthermore we also explore the impact of different clipping values which are used to enforce the $k$-Lipschitz constraint on the cost function of WGANs. Indeed, it is also quite interesting to consider exploring various traditional constraints on the weight space $W$, such (i) standard (ii) cubitization (projection in the unit hypercube) (iii) unitization (iv) and certainly typically normalization via the above norms, since the key idea of compact weight space lends itself to these.

We describe each of them in the following sections - 1.7.1, 1.7.2, 1.7.3. We discuss the implementation and results in 1.7.4.
1.7.1 Effect of Cost functions on WGANs

It is interesting to note that the compact space referred to depends heavily on the cost function $d(X,Y)$. Earlier implementations of WGANs predominantly used the $\ell_1$ norm, namely $d_1(X,Y) = \|X - Y\|_1 = \sum_{j=1}^{p} |X_j - Y_j|$. However, it makes natural sense to consider exploring a wide variety of norms when it comes to clipping, namely natural counterpart to $\ell_1$ like the $\ell_2$ norm $d_2(X,Y) = \|X - Y\|_2 = \sqrt{\sum_{j=1}^{p} |X_j - Y_j|^2}$, or the supremum norm $\ell_\infty$ with $d_\infty(X,Y) = \|X - Y\|_\infty = \max_{i=1,\ldots,n} |X_i - Y_i|$. In fact many authors have done just that namely [23]. Further emphasizing the importance of this cost function on the performance of WGANs, distances based on Banach spaces have been considered by [9] with tremendous improvement in performance as measured by inception score. It is not surprising that changes in the cost function would have impacts on the performance. Such drastic qualitative and quantitative differences in the effect of distances are prevalent in machine learning as one can see in the difference between LASSO (Least Absolute Shrinkage and Selection Operator) which is $\ell_1$ regularization and Ridge which is $\ell_2$ regularization i.e., . As Figure 1.10 shows, LASSO gives a sparse solution while ridge does not.

We also study the generator and the discriminator loss inspired by elastic net regression [62] which essentially combines the properties of both LASSO and ridge regression i.e, $d(X,Y) = \|X - Y\|_1 + \|X - Y\|_2 = \sum_{j=1}^{p} |X_j - Y_j|$.
Figure 1.10: Contours of LASSO vs Ridge on a 2-dimensional parameter space where $\beta = (\beta_1, \beta_2)^T$ (Image Courtesy of Li [3])

$$Y_j + \sqrt{\sum_{j=1}^{p} (X_j - Y_j)^2}.$$ Here the $\ell_1$ and the $\ell_2$ norm is added linearly and the geometric representation is as show in figure 1.11. In regression and classification such differences are paramount.

Although, various authors have investigated the effects of the cost function on the performance of WGANs we have noticed that the contributions are scattered and no research exists that comprehensively and cohesively provides a taxonomy of those effects. The motivation of this thesis is to systematically consider a variety of cost functions both already explored as indicated previously, along with new extensions, and provide a comprehensive empirical assessment of the effect on measures of performance of GANs like inception score. For completeness, we intend to use a wide variety of synthetic and real
1.7.2 Effect of the Clipping values in WGANs

In WGANs the parameters learned by the generator are confined to the compact space and the K-Lipschitz is enforced by clamping [11]. The clamping is usually done by clipping the values of the parameters over an interval. In [11] these clipping values were \([c, -c]\) where \(c = 0.01\). In this thesis research, we would like to explore the effect on the loss of the generator and discriminator with different clipping values. While the authors of WGAN have used hard clipping [11], we refrain from using doing the same. Instead, we consider parameters learned by the WGAN and product of the average of the weights with the clipping values for the compactification. The results for the different clipping values \(c = [0.1, 0.01, 0.001]\) are collected. As we use
norms we do not confine the compact space with the negative clipping values rather use zero for the lower bound.

1.7.3 Training of WGANs

In our research, we would like to explore whether the number of iteration used for training matters. We have seen in [22], [11], [23] the number of iterations for the training varies. While there is no fixed method to depict the ideal number of training iterations, our interest mainly lies in exploring their impact of the training on the samples generated. To answer questions like how did the GAN perform on a lesser number of iterations than the higher number of iterations. Did we get meaningful samples generated when trained for larger numbers (say, 1 Million)?

1.7.4 Experiments and Results

To verify our hypothesis on the effect of the cost function of WGANs with different distance measurements, we use both real data sets like MNIST, CIFAR - 10 and synthetic dataset like Toy - 8 Gaussian. Quantitatively we use the WGAN loss for Generator and the Discriminator. The Inception score evaluates the quality and the diversity of the images generated. Along with this to measure the recovery we also include human inspection as a way to judge the samples.
1.7.5 Thesis Organization

- Chapter 1. Introduction: This chapter gives the brief introduction to the various concepts of machine learning which is required to understand this thesis. Along with the introduction to GANs and WGANs, the research carried out in the interest of this thesis is also explained.

- Chapter 2. Background: This chapter explains the training of GANs. It discusses the various challenges faced during the training of GANs and the ways to circumvent this using Wasserstein Distance and WGANs. It also explains WGANs in detail.

- Chapter 3. Methodology: This chapter starts with explaining the importance of norms in statistical machine learning. It explains the various norms and their impact. This is followed by explaining the impact of norms and their significance in WGANs and the methodology carried out to test the hypothesis.

- Chapter 4. Results and Experiments: In this chapter the various metrics used for the evaluation of GANs in this thesis is explained. This is followed by discussing the data sets. We include the results of our research.

- Chapter 5. Conclusion and Future Work: This chapter gives the conclusion of this thesis along with the potential future work.
Chapter 2

Background

2.1 Introduction to Generative Adversarial Networks

Until recently, discriminative deep neural networks models have been found to perform well on the tasks in deep learning, e.g: convolutional neural networks [34], rectifier neural networks [21] performs well on classification with supervised learning. The success of discriminative models is evident with applications in [18], [47]. However, with generative models, early success was limited to Boltzmann machines [49] and auto-encoders [48].

In 2014, Ian Goodfellow introduced Generative Adversarial Networks (GANs) [22]. It is a fascinating new paradigm in machine learning and artificial intelligence, especially in the context of unsupervised learning. The advent of GANs has given new promises. It can generate any type of data and are quickly becoming a state of the art tool used in various applications such as image generation [22], image synthesis [45], image super resolutions [35], text generation [26], text to image synthesis [13], image editing [28], molecule development [30] to name a few.

Given the real data distribution $p_{data}(x)$, the goal is to imitate the samples from real data distribution and generate the samples which are as
similar as possible to the real data. To accomplish this, GANs have a Gener-
erator (imitator) $G$ and a Discriminator (critic) $D$, $D$ is given the real data
distribution, and $G$ is fed with an input noise $p(z)$. $G$ learns to map this
noise to the real data via an adversarial process.

2.1.1 Architecture of a Generative Adversarial Networks

The Generator $G$ and the Discriminator $D$ in GANs can be any neural
network [54], [43]. The training is via an adversarial process. We can think of it
as two neural networks of any architecture called the Generator neural network
$G$ and the Discriminator neural network $D$ competing with each other such
that, the $G$ captures the data distribution and $D$ estimates the probability
that the sample came from the training data rather $G$. In the training $G$ tries
to maximize the probability that $D$ makes a mistake.

To give an intuition about how the GANs function in adversarial setup,
let’s consider a two-player game with player 1, and player 2. In this game, one
of the players will be strong. Consider if Player 1 is a stronger player, Player
2 tries to learn the techniques from Player 1 and eventually tries to win over
Player 1. Simultaneously Player 1 will try its best to win the game. The idea
is over the time both the players become strong, and they try to win over each
other. This is derived from two player minimax game from game theory [52].

In GANs, $G$ (imitator) and the $D$ (critic) play the minimax game trying
to fool each other. As shown in the Figure 2.1, the Discriminator is given the
sample $x$ from the probability distribution of the input data $p_{data(x)}$, and the
Figure 2.1: Architecture of a generative adversarial network. (Image courtesy [6])

Generator is given a random sample $z$ from the noise distribution $p(z)$. The Generator generates an image from the noise sample and the discriminator is trained on the data from the real distribution $x$. The Discriminator after the training of Generator has to predict if the generated sample was real or fake.

### 2.2 Training of a Generative Adversarial Networks

As we mentioned earlier, GANs are trained via an adversarial process. In the simplest form let us consider the Generator $G$ and the Discriminator $D$ to be a multi-layer perceptron in GANs. As stated by Ian Goodfellow in [22], the training is via the minimax game is realized such that, the value function $V(G, D)$, $D$ tries to maximize the probability of assigning the correct label
to both training examples and samples from $G$, and $G$ tries to minimize the $\log(1 - D(G(z)))$. The objective function [22] is formalized by Eq. 2.1:

$$
\min_G \max_D V(G, D) = \mathbb{E}_{x \sim p_{data}}[\log D(x)] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))] 
$$  \hspace{1cm} (2.1)

$G$ learns the mapping to transform the input noise $z$ to the real data $x$. The distance between the resulting samples from $G$ and the original data is measured using the measure of between the two probability distributions given by Jensen Shannon Divergence (JSD).

Algorithm 1 is used to train the GAN. As stated by the authors [22], GAN is trained using minibatch stochastic gradient descent. The number of the number of training iterations differ for $G$ and $D$, and this is controlled by hyper-parameter $k$, where $k = 1$.

In the GAN set up; first the Discriminator is trained against a Generator. For a fixed $k$ steps, samples from $p_g(z)$ noise distribution along with the samples from $p_g(x)$ is given to $D$ and the weight is updated using the ascending stochastic gradient. After training $D$, we train $G$ with the input samples from $p_g(z)$ and using descending its stochastic gradient to update $G$.

### 2.3 Applications of Generative Adversarial Networks

One of the exciting applications of GANs is the conversion of text to image. Though in deep learning there had been advances in with a different recurrent neural network to learn the features of the text representations, and
Algorithm 1 Training GANs

1: for number of iterations do
2:     for k steps do
3:         Sample minibatch of $m$ noise samples $\{z^{(1)},...,z^{(m)}\}$ from prior available noise distribution $p_g(z)$
4:         Sample minibatch of $m$ samples $\{x^{(1)},...,x^{(m)}\}$ from real data distribution $p_g(x)$
5:         By ascending its stochastic gradient, update discriminator.
6:             \[ \nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[ \log (x)^i + \log (1 - D(G(z^{(i)}))) \right] \] (2.2)
7:     end for
8:     Sample minibatch of $m$ noise samples $\{z^{(1)},...,z^{(m)}\}$ from prior available noise distribution $p_g(z)$
9:     By descending its stochastic gradient, update the generator.
10: \[ \nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \log (1 - D(G(z^{(i)}))) \] (2.3)
11: end for

25
GANs to generate the image. The authors of [46] were able to combine these two frameworks to create the images from textual description.

The quality of the images that are generated by GANs has improved over time. In super resolution GANs, authors [35], were able to generate the images that could capture the more delicate (or finer) texture of the images and generate the photo-realistic pictures. They use deep residual networks for this framework of GAN and with the changes in loss function, they enhance the resolution of an image.

2.4 Challenges with GANs

Unfortunately, despite the exciting applications of GANs and their vast potential, there are many challenges [50] that make it difficult to fully harness GANs inherent modelling power.

- The biggest problem is to train the GANs itself, and due to the complexities it has a whole new area of research to find the optimal way to train GANs.

- The stability of the Generator training and Discriminator training is essential for GANs to perform well. The training process of GANs is such that $G$ and $D$ is trained individually. If the balance between the training of $G$ and $D$ fails, one of them might become more powerful than the other. This leads to variation in the stability of learned parameters and it causes failure in convergence.
• Vanishing gradients caused by the constant loss (due to $G$ or $D$) impacts the GAN training resulting in model collapse problem.

• The training of the GANs like any other deep learning models is highly sensitive to the hyperparameter selection.

2.5 Motivation of Wasserstein Generative Adversarial Networks

The objective function used for training the GANs had a few setbacks, which essentially made the GANs training difficult and thereby limiting their potential. This was carefully analyzed by the authors of [10] and it gave the birth to Wasserstein Generative Adversarial Networks (WGANs) [11].

2.5.1 Cost Function and Training of Generative Adversarial Networks

An optimal training of GAN is achieved when $D$ fails to distinguish between the samples generated by $G$ and the samples from $p_{data}(x)$. The cost function for training the GANs to attain optimal $D$ is given by Eq. 2.4 and the optimal $D$ at this point is $D^*$ given by Eq. 2.5. The cost function for this optimally is further proved by the authors in Eq. 2.6.

\[
C(G) = \min_{G} \max_{D} V(G, D) \\
= \mathbb{E}_{x \sim p_{data}} [\log D^*_G(x)] + \mathbb{E}_{z \sim p_{z}} [\log (1 - D^*_G(G(z)))]
\]
\[ D^*_G(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \] (2.5)

\[
V(G, D) = \int_x p_{data}(x) \log(D(x)) dx + \int_z p_z(z) \log(1 - D(g(z))) dz \tag{2.6}
\]

\[
= \int_x p_{data}(x) \log(D(x)) + p_g(x) \log(1 - D(x)) dx
\]

As stated in [22], any function \( y \rightarrow \log(y) + b \log(1 - y) \) where \((a, b) \in \mathbb{R}^2 \setminus \{0, 0\}\), reaches its maximum in \([0, 1]\) at \( a \left/ \left(a + b\right) \). For given \( p_{data} \) and \( p_g \), we need not define the discriminator outside \( \text{Supp}(p_{data}) \cup \text{Supp}(p_g) \).

The authors reformulate the cost function by considering the maximum log-likelihood. Consider the conditional probability \( P(Y = y|x) \), here \( Y \) acts as a class label whether the sample came from the real data distribution when \( y = 1 \) or the noise distribution when \( y = 0 \). The Discriminator tries to maximize this log-likelihood by estimating \( Y \). Thus reformulating \( V(G, D) \) in Eq. 2.4 to Eq. 2.7.

\[
C(G) = \max_D V(G, D) \tag{2.7}
\]

\[
= \mathbb{E}_{x \sim p_{data}} \left[ \log D^*_G(x) \right] + \mathbb{E}_{z \sim p_z} \left[ \log(1 - D^*_G(G(z))) \right]
\]

\[
= \mathbb{E}_{x \sim p_{data}} \left[ \log D^*_G(x) \right] + \mathbb{E}_{z \sim p_g} \left[ \log(1 - D^*_G(x)) \right]
\]

\[
= \mathbb{E}_{x \sim p_{data}} \left[ \log \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \right] + \mathbb{E}_{z \sim p_g} \left[ \log \frac{p_g(x)}{p_{data}(x) + p_g(x)} \right]
\]
Note that, in Eq. 2.7 $D$ should maximize the cost $G$ should minimize the cost. The value obtained at $p_g = p_{data}$ for the global minimum is $C(G) = -\log 4$. When $p_g = p_{data}$, Eq. 2.7 is given by Eq. 2.8,

$$\mathbb{E}_{x \sim p_{data}}[-\log 2] + \mathbb{E}_{z \sim p_z}[-\log 2] = -\log 4 \quad (2.8)$$

These formulations to get the cost function for the training of GANs has few limitations. Due to these limitations, the training sometimes proved difficult and a daunting task. The measures due to which these difficulties were induced is discussed in the next section.

### 2.5.2 Divergence measure in Generative Adversarial Networks

The cost function of GANs applies Kullback–Leibler divergence (KL) [16] and Jensen–Shannon divergence (JSD)[19] in the original formulation [22].

Given any two probability distributions $P_1$ and $P_2$, Kullback–Leibler divergence is given by Eq. 2.9. The Jensen–Shannon divergence for the same is given by Eq. 2.10.

$$KL(P_1||P_2) = \mathbb{E}_{x \sim P_1} \log \frac{P_1}{P_2} \quad (2.9)$$

$$JS(P_1||P_2) = \frac{1}{2} KL(P_1||\frac{P_1 + P_2}{2}) + \frac{1}{2} KL(P_2||\frac{P_1 + P_2}{2}) \quad (2.10)$$
Now, in the training of GANs if Eq. 2.8 is subtracted from Eq. 2.7
\[ C(G) = V(D^{*}_G, G), \]
we get Eq. 2.11 for a given optimal discriminator.

\[ C(G) = -\log(4) + KL\left(p_{data}||\frac{p_{data} + p_g}{2}\right) + KL\left(p_g||\frac{p_{data} + p_g}{2}\right) \]  
\[ (2.11) \]

Using Jensen–Shannon divergence in Eq. 2.11 we get Eq. 2.12

\[ C(G) = -\log(4) + 2 \cdot JSD\left(p_{data}||p_g\right) \]  
\[ (2.12) \]

Notice in 2.11, we are interested to get the value of \( C(G) \) as \(-\log 4\), which corresponds to the global minima \( C^*(G) \). The JSD should become 0 and should be non-negative, for us to get this desired value. This is possible, only when the two distributions here \( p_{data} \) and \( p_g \) are equal i.e, \( p_{data} = p_g \).

### 2.5.3 Jensen–Shannon divergence

JSD is used to measure the divergence between the generated data distribution and the real data distribution and the training goal is to minimize this divergence. JSD particularly comes with the drawback. It works well if there is an intersection between two distributions and helps the cost function of the GAN to attain a meaningful loss. But when there is no intersection, JSD does not let the \( C^*(G) \) to attain minima as the value is no longer \(-\log 4\), rather it is fixed to a constant. Since there is no overlap, this constant does not change, and thus the gradient is the same throughout leading to the problem
of vanishing gradients. One of the ways to get around this problem was to use methods such as log trick [50], which later caused additional issues of instability in training of GANs.

These theoretical properties were analyzed and an alternative to JSD was proposed by the authors of [10]. A new metric called the Wasserstein Distance was used as a replacement of JSD and thereby giving birth to WGANs.

2.6 Wasserestein Distance

Optimal transport [15] cost between the two measures is cost incurred to transport the masses between the two probability measures. The cost is given by 2.13

\[ C(\mu, \nu) = \inf_{\pi \in \Pi(\mu, \nu)} \int c(x, y) d\pi(x, y) \]  (2.13)

In this equation the cost incurred to transport a unit mass from \(x\) to \(y\) is given by \(c(x, y)\). We use this cost function and define Wasserstein distances [59] in its primal form given by Eq. 2.14

\[ \mathcal{W}_p(\mu, \nu) = \left( \inf_{\pi \in \Pi(\mu, \nu)} \int_X d(x, y)^p d\pi(x, y) \right)^{\frac{1}{p}} \]  (2.14)

\[ = \inf \left\{ \left[ \mathbb{E}d(X, Y)^p \right]^{\frac{1}{p}} : \text{law}(X) = \mu, \text{law}(Y) = \nu \right\} \]

In Eq. 2.14, \(p\) is the order of Wasserstein distance, between any two probability measure \(\mu\) and \(\nu\) on \(X\), and \(p \in [1, \infty]\) and consider \((X, d)\) to be a polish metric space.
The distance in Eq. 2.14 becomes extremely difficult to compute because of the inf. To circumvent that difficulty, the so-called Kantorovich-Rubinstein duality \cite{59} is used to modify the primal form to the dual form, which for the Wasserstein distance of order 1 takes the form below:

\[
\mathbb{W}_1(\mu, \nu) = \sup_{\|\psi\|_{\text{Lip}} \leq 1} \left\{ \int_X \psi d\mu - \int_X \psi d\nu \right\}
\]  

(2.15)

Wasserstein distance metric has many advantages \cite{59}:

- Wasserstein distances take care of large distances in polish metric space in \( X \), which becomes difficult to handle with other weak distances.

- When we have problems which involve optimal transport and others which come from partial differential equations because of continuity of Wasserstein distances it works well.

- One can leverage the benefits of original to the dual definition and switch accordingly.

- Since they are bounded by infimum any product space will give a bound on the distance between any two distributions.

- A variety of geometry of spaces can be incorporated with Wasserstein distance.

In WGANs the measure \( \mu \) is attained through the probability density functions \( p_{\text{data}}(x) \) and \( \nu \) is \( p(z) \).
In the coming sections we discuss how the use of Wasserstein's distance enabled WGANs to overcome the training difficulties of GANs.

2.7 Wasserestein Generative Adversarial Networks

In this section, the theory behind WGANs and the advantages of WGANs are presented.

\[ W(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} E_{(x,y) \sim \gamma} [||x - y||] \] (2.16)

The Wasserstein Distance in WGANs is formulated as Eq. 2.16, where \( \Pi(P_r, P_g) \) is the set of all all joint distributions \( \gamma(x, y) \) whose marginals are given by \( P_r \) and \( P_g \). Here, \( \gamma(x, y) \) represents the mass that has to be transported from \( x \) to \( y \) to transform the distributions \( P_r \) into \( P_g \). The Wasserstein distance is the cost of the optimal transport plan.

\[ W(P_r, P_\theta) = \sup_{||f||_L \leq 1} \mathbb{E}_{x \sim P_r}[f(x)] - \mathbb{E}_{x \sim P_\theta}[f(x)] \] (2.17)

Due to the complexity involved in infimum with Eq.2.16, the Kantorovich-Rubinstein dual formulation [59] of Wasserstein distance used in WGANs is given by Eq. 2.17. The supremum sup in Eq. 2.17 is all over the 1-Lipschitz functions, where \( f : \mathcal{X} \mapsto \mathbb{R} \). Instead of considering \( ||f||_L \leq 1 \), if 1 is replaced by a constant \( K \) we get \( ||f||_L \leq K \) this will be K-Lipschitz for some constant \( K \). Then, in Eq. 2.17 you end up with \( K \cdot W(P_r, P_\theta) \).
In WGANs, therefore for any parameterized family of functions driven
by the set of weights $w$ (In this case, a neural network) $\{f_w\}_{w \in W}$ that are all
$K$-Lipschitz for some $K$, we could consider solving the problem in Eq.2.20.

$$\max_{w \in W} \mathbb{E}_{x \sim \mathbb{P}_r} \left[ f_w(x) \right] - \mathbb{E}_{z \sim p(z)} \left[ f_w(g_{\theta}(z)) \right] \quad (2.18)$$

$$\mathbb{E}_{z \sim p(z)} \left[ \nabla_{\theta} f_w(g_{\theta}(z)) \right] \quad (2.19)$$

In WGANs the authors assume that if in Eq. 2.17 the supremum is
attained for some weights $w$ in $W$, this would lead to the calculation of the
Wasserstein distance between the probability distribution of $W(\mathbb{P}_r, \mathbb{P}_{\theta})$ up to
a multiplicative constant. $W(\mathbb{P}_r, \mathbb{P}_{\theta})$ can also be considered differentiating by
backpropagating through Eq. 2.20 and via estimating Eq. 2.21.

$$\max_{\|f\|_{L^1} \leq 1} \mathbb{E}_{x \sim \mathbb{P}_r} \left[ f(x) \right] - \mathbb{E}_{x \sim \mathbb{P}_{\theta}} \left[ f(x) \right] \quad (2.20)$$

$$\nabla_{\theta} W(\mathbb{P}_r, \mathbb{P}_{\theta}) = -\mathbb{E}_{z \sim p(z)} \left[ \nabla_{\theta} f(g_{\theta}(z)) \right] \quad (2.21)$$

Consider the maximization of Eq. 2.17, this is reformulated as shown
in Eq. 2.20. This Eq. 2.20 means a function $f$ should be found such that it
solves the maximization problem. The authors [11] suggests to accomplish this one can train a neural network which is driven by the set of parameters (or weights) lying in a compact space $W$, this is followed by updating the weights using backpropagation through Eq. 2.21.

The most important thing to note here is the weight in the compact space. In WGANs, the $W$ lies within the compact space; this implies that all the functions $f_w$ will be $K$-Lipschitz for some $K$, that depends on $W$ and not the individual weights. Therefore, in WGANs they approximate the weight upto an irrelevant scaling factor and the capacity of the critic $f_w$.

To enforce the compact space, on the set of weights $w$ the authors use clamping method. The weights are clipped to a fixed box $W = [-c, c]^l$ where $c$ can take different clipping values and $l$ is the dimension of $W$.

### 2.8 Training Wasserstein Generative Adversarial Networks

Arjovsky et. al [11] propose the following algorithm to train WGANs:

To train they use $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $\eta_{\text{critic}} = 5$. The hyper-parameters required for the training are $\alpha$ (learning rate), $c$ is the clipping parameter, $m$ is the batch size and they set $\eta_{\text{critic}}$ such that is the number of iterations of the critic per generator iteration. Initial critic parameter $w_0$ and initial generator parameter $\theta_0$ is required.

WGANs training differs with the use of different loss function compared
Algorithm 2  Training Wasserstein Generative Adversarial Networks

\begin{algorithm}
\begin{algorithmic}[1]
\While {$\theta$ has not converged}
\For {$t = 0, \ldots, n_{\text{critic}}$}
\State Sample $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$ a batch from real data.
\State Sample $\{z^{(i)}\}_{i=1}^m \sim p_z$ a batch of priors.
\State $g_w \leftarrow \nabla_w \left[ \frac{1}{m} \sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m} \sum_{i=1}^m f_w(g_\theta(z^{(i)})) \right]$ \Comment{Eq. 9}
\State $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$ \Comment{Eq. 10}
\State $w \leftarrow \text{clip}(w, -c, c)$ \Comment{Eq. 11}
\EndFor
\State Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
\State $g_\theta \leftarrow -\nabla_\theta \frac{1}{m} \sum_{i=1}^m f_w(g_\theta(z^i))$ \Comment{Eq. 12}
\State $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$ \Comment{Eq. 13}
\EndWhile
\end{algorithmic}
\end{algorithm}
to training in GANs Algorithm 1, and the optimizer used here is RMS prop.
The advantages of using the WGANs is discussed in the next section.

2.9 Advantages of WGANs

- The problem of vanishing gradient which occurred in the original GAN was solved using the WGAN. This is due to the fact that the Wasserstein Distance is continuous and differentiable almost everywhere.

- The critic, $D$ does not saturate and converge to a linear function, this is because the weights are constrained and it limits the possible growth of the function.

- Since Wasserstein Distance is continuous and differentiable almost everywhere gives the optimal critic and thus helps to avoid the mode collapse problem.

It turns out that the practical implementation and use of WGANs end up revolving around elements and properties of the set $W$, which is essentially the weight space of the learning machine/function $f_w$.

Our subsequent sections explore various aspects of $W$ with the finality of achieving the compactness crucial to the realization of the $k$-Lipschitz condition mentioned earlier. We specifically touch on various norms as indicated in previous sections and provide what we believe to be a fair detail account of the impact of weights on the performance of WGANs. It is important to note
that our goal as stated earlier is not to achieve any particular improvement in performance but rather to compare performances whatever they may be but crucially to provide evidence that different choice of $W$ and different norms therein affect WGANs qualitatively and quantitatively.
3.1 Revisiting WGANs

In the dual formulation of Wasserstein distance of WGANs Eq. 2.16, the underlying learning machine is now appropriately and conveniently represented by a parametric function $f_w$, whose weight $w = (w_1, \cdots, w_l)^T$ are taken from weight space $W$. It turns out that the Lipschitz condition $|w_j| < 0.001$, $j = 1, \cdots, l$, can be achieved in the weight space $W$. In WGANs this can be achieved by constructing a function whose parameters (or weights) come from a compact set, the initial work of WGAN enforces compactness by clamping the weights into the fixed box. In other words, the crucial aspect of building WGANs comes down to controlling or enforcing the compactness of $W$.

Looking at these clamping, there are few obvious questions.

- What is the origin of limiting the compact space to $[-0.01, +0.01]$?
- Does the absolute value of $W$ have to be symmetric around zero?
- Is it possible to have a sparse version of the weights, where many of the weights are dropped down to zero’s and with a few numbers of weights that have non zero values?
• And when these parameters are kept inside the compact space, how the norm effect the compactness?

• We may ask if one considers that this compact space is in two dimensional then what does the shape of \( W \) look like.

### 3.2 Impact of Norms

In traditional machine learning, norms have had a significant impact. Figure 3.1: Contours of LASSO vs Ridge on a 2 - Dimensional parameter space where \( \beta = (\beta_1, \beta_2)^T \) [3]

It is known in the research [51] that although neural networks may be large, we need only a few weights that are nonzero, this is accomplished through regularization. The traditional regression methods like LASSO [37], Ridge[44] and elastic nets that is well established, act on the parameter (weight) space constraining the set of parameters is done via the norms.
Typically $\ell_1$, $\ell_2$ norms imposes an apriori shape on the space of parameters. In figure 3.1, we can see the difference in the contours of LASSO vs Ridge. Obviously, the $\ell_1$ norm is shown to have a different contour and as a result different impact on the norms.

### 3.3 Significance of $\ell_1$ norm in LASSO Regression

For our purpose let us consider a set of weights $W = (w_1, \ldots, w_l)$, $\ell_1$ norm is given by Eq. 3.1

$$
||w||_1 = \sum_{j=1}^{l} |w_j| 
$$

$$
||w||_1 = |w_1| + |w_2| + \cdots + |w_l|
$$

$\ell_1$ norm is used in LASSO regularization. LASSO is known as Least Absolute Shrinkage and Selection Operator. It is used in machine learning for feature selection and regularization. The method of shrinking or regularization is nothing but penalizing the coefficients to zero. This allows feature selection, where only the non-zero values of the coefficients are allowed to be the part of the model.

$$
\mathcal{E}(w) = \sum_{i=1}^{n} \left( Y_i - \sum_{j=1}^{p} X_{ij}w_j \right)^2 + \lambda \sum_{j=1}^{p} |w_j| 
$$

$$
= ||Y - Xw||_2^2 + ||w||_1
$$
The LASSO minimizes the sum of square errors, with an upper bound on sum of the absolute values of the model parameters as shown in Eq. 3.3. \( \lambda \) controls the strength of the penalty, bigger the lambda more is the penalty forced. Notice, the norm 1 acting on the parameters to ensure the compactness and introduces sparseness in the data. In this case, if the first point is in the proximity of the corner, then the parameter is equal to zero, and thus the model always seeks a sparse solution. This sparsity can be seen by the geometric representation of the \( \ell_1 \) norm shown in Figure 3.2 Using LASSO increases the models performance by decreasing the variance and without substantially increasing the bias. As shown in Figure 3.2 the constraint region for LASSO is
a turned square (or a diamond). It is usually used when we have data with a smaller number of observations but a larger number of features. It also helps in model interpretability. During the shrinkage, it reduces the features that are not related to the target variable and thus reducing the model overfitting.

3.4 Significance of $\ell_2$ norm in Ridge Regression

Interestingly, the counterpart of $\ell_1$, $\ell_2$ is also used in regularization. $\ell_2$ norm is given by (3.3)

$$||w||_2 = \sqrt{\sum_{j=1}^{l} w_j^2} \quad (3.3)$$

$$||w||_2^2 = \sum_{j=1}^{l} w_j^2 = w_1^2 + w_2^2 + \cdots + w_l^2$$

This is called Ridge regularization. Similar to $\ell_1$ norm regularization $\ell_2$ also induces penalties on the coefficients learned. Ridge regularization is used when the data has high collinearity. When the data is co-linear small changes in the data can amplify the noise.

$$E(w) = \sum_{i=1}^{n} \left(Y_i - \sum_{j=1}^{l} X_{ij}w_j \right)^2 + \lambda \sum_{j=1}^{l} w_j^2 \quad (3.4)$$

$$= ||Y - Xw||_2^2 + ||w||_2^2$$

Ridge regularization is the least square regression combined with the $\ell_2$ norm regularization. The coefficients are penalized to avoid overfitting. It
Figure 3.3: $\ell_2$ norm geometric space (Image Courtesy [4])

has a square penalty on the weights which is controlled by $\lambda$ Eq. 3.4. We can think of this as an, optimizing the error while keeping the $\ell_2$ norm (the square values of the $w$) of the weights bounded and is the tuning parameter usually obtained by the cross-validation. If the value of lambda increases, a larger weight is added leading to underfitting. For irrelevant features, a small weight is assigned which non zero. The goal, however, remains to avoid overfitting, and reduce the model complexity. The constraint region for ridge regression is a circle as shown in Figure 3.3. In Ridge, the weights are constrained using the $\ell_2$ norm. In neural networks, the ridge regression is called the weight decay.
3.5 Significance of $\ell_1$ norm, $\ell_2$ norm in Elastic Net

In machine learning, we have another regularization technique which uses the convex combination of $\ell_1$ and $\ell_2$ norm, called the elastic net, given by Eq. 3.5.

$$||w||_{enet} = \alpha||w||_1 + (1 - \alpha)||w||_2 \quad (3.5)$$

![Figure 3.4: $\ell_1$ norm + $\ell_2$ norm geometric space (Image Courtesy [7])]()
variables before it saturates. In the case of grouped variables, LASSO tends to choose only one variable ignoring others. The elastic net was introduced to overcome the difficulties of LASSO. Elastic net uses the both LASSO and Ridge regularization properties Eq. 3.6. It uses the $\ell_1$ norm and the $\ell_2$ with the coefficients as shown in Figure 3.4. It eliminates the limitation on the variable selection and stabilizes the selection of grouped variables. The geometric space would be as shown in Figure 3.4

### 3.6 Norms in WGANs

It is important to note that, all these techniques in traditional machine learning have used norms to limit the compactness of the coefficients learned.

Compactness of $w$ can undoubtedly be achieved by the arbitrary clipping of the values around the zero symmetric space. But, it is our view a more principled approach should involve aspects of the norms in the spirit of LASSO, Ridge and elastic nets. This approach used here putting the weights into a closed interval and setting the parameters into compact space to the processing activities like standardization guaranteeing that weight have 0 mean and one standard deviation, cubitization putting all the weights to 0’s and 1’s and unitization ensuring that the weights are unit length.

In effect achieving the lipschtiz condition through the control of the weights is far more complex than clipping them into some arbitrary interval. A more complete treatment should definitely consider a cross-validation approach to finding the threshold on the norms of the weights i.e., $||w|| \leq \tau$. For instance
in the original sup norm clipping used to achieve compactness as indicated earlier one could consider $||W||_\infty \leq \tau$ with $\tau$ adaptively found by $c$. We conjugate that such adaptive scheme will be more optimal.

We do not try these methods in this thesis because of the limitation to the deeper access to tensor flow than the time of the thesis allowed. But, this should be ideal.

The importance of weights has been well documented in statistical machine learning. It is refreshing to know in the field of WGAN this is also extensively done. Banach GAN provides a quite principled way to do this by namely, what we are advocating by regularizing the weights [9]. They have an inception score of 8. Another variant of WGAN called the WGAN with gradient penalty [23] also uses the Lagrange form of regularizer and gets the results whose images have the highest inception score and considered to be the state of the art WGAN producing better results.

Given all these, in this thesis, we explore a variety of norms and threshold. In the spirit of the interplay between $\ell_1$ and $\ell_2$ as seen with ridge and LASSO, we first use an alternative to $\ell_1$, $\ell_2$. But later, we also consider other norms like $\ell_\infty$ norm, and in the spirit of the elastic net and we explore it empirically $\ell_1 + \ell_2$.

In the next chapter, we discuss the results and experiments of our research. We use a variety of real and synthetic data sets which is discussed and get the results for WGANs.
Chapter 4

Results and Discussion

In this chapter, we discuss the results of our hypothesis. Methods to evaluate GANs is presented in section 4.1. Followed by the introduction to the data sets 4.2 used to test and assess the hypothesis. Finally, the results of the experiments are shown in section 4.3

4.1 Evaluating GANs

In the original GAN [22] the performance was measured using the generator and the discriminator loss. Nonetheless, since the presentation of GAN there has been researched regarding which metric would best depict its performance. Various metrics and their usage has been discussed in [12], [14], [40]. With many choices available we use the $D$ loss and the $G$ loss to measure the loss incurred during the training. To evaluate the recovery quantitatively, we use inception score. In addition to this, we resort to a human inspection to evaluate the quality of the image generated by the WGAN, as this was one of the ways to evaluate the quality of recovery in early GANs [50].
4.1.1 Inception score

As mentioned earlier, in this thesis we use a metric called inception score as one of the measures to evaluate the performance of GANs. Ian Goodfellow [50] first used inception score and he introduced this as one of the metric to quantify the performance of GAN. When the inception score was proposed these scores had a good correlation with the human annotators (Amazon Mechanical Turk (MTurk) [1]). Since then, it has become one of the popular choices among researchers, though it is not a standard. To get inception score, we use the inception model trained on inception v3 network [56]. When the Generator produces an image, this image is given to the classifier to see if it can classify the image generated by the WGAN. If the generated images have meaningful objects, the conditional label distribution \( p(y|x) \) obtained will have low entropy. Also, the generator is expected to generate the variety of images, and for highly variable images the marginals \( \int p(y|x = G(z))dz \) should have high entropy. Combination of these two scores gives the inception score and is given by \( \exp(\mathbb{E}_x KL(p(y|x)||p(y))) \). Here \( KL \) is the kullback-liebler divergence, and for easier comparison, the exponent of the result is taken. In our research, we use the Inception score to evaluate the recovery for CIFAR-10 [2] images. Inception score is used primarily with the RGB images, and we refrain from using this metric on grayscale images.
## 4.2 Data sets

For empirical evaluation of the impact of various norms on WGANs we use two real data sets. Modified National Institute of Standards and Technology database which is popularly known as MNIST, Canadian Institute For Advanced Research - 10 which is known as a CIFAR-10 data set. Along with these two we use the synthetic dataset called the Toy-8 Gaussian dataset.

### 4.2.1 MNIST

MNIST data set [17] is one of the most popular benchmark data set used in the various field of research in artificial intelligence. This database consists of 60,000 training images and 10,000 testing images. The images in the MNIST database is collected from the National Institute of Standards and Technology (NIST) [29] specialized databases 1, and database 3. These images are handwritten digits by the American high school students and the employees of the American Census Bureau. These grayscale images are normalized and scale to give a 28x28 image. The sample from the database is as shown in Figure 4.1

MNIST is used by many researchers for applications like recognition tasks and is one of the most sought out preliminary databases for the works involving the neural networks.
4.2.2 CIFAR-10

The CIFAR-10 data set [2] consists of 60000 images, which are divided into 50000 training images and 10000 test images. CIFAR - 10 was first used in [33] to learn multiple layers of features from tiny images. Since then, along with MNIST, CIFAR-10 is one of the widely used data set amongst the researchers for RGB images.

These images are RGB (color) images of size 32x32, and they belong to 10 classes. In each of these class, we have 6000 pictures, and the classes are namely, airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks as shown in Figure 4.3. Note these classes are mutually exclusive. As stated in [2], the images in the automobile class contain only sedans, SUVs which separates from a truck class which has the images of only big trucks. So mostly the researchers are interested in finding the object in these images.
4.2.3 Toy Data set

\[ P(x) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \]  

(4.1)

In this experiment, we use a mixture of 8 Gaussians with different mean and the variance is given by Eq. 4.1 (each of the Gaussian), where \( \mu \) is mean and \( \sigma \) is the standard deviation. Authors [11] use a variety of mixture of Gaussian because it is easy to visualize the mode collapse problem. These data sets are popular to show the effect of Wasserstein Distance on the distributions with no overlap.

In the next section, we present the results of our hypothesis.
4.3 Results and Experiments

In this section, we present the results for WGAN loss namely, $G$ and $D$ loss. The $\ell_1$ norm is compared with the other norms - $\ell_2$, $\ell_\infty$ and $\ell_{\text{treneto}}$. We discuss the results for MNIST, CIFAR - 10 and toy - 8 Gaussian data set. All the experiments are conducted on NVIDIA TITAN V 12GB hardware configuration [5]. We use python - Tensorflow [8] and develop our research on the existing implementation of [27]. The results are shown via the graphical representation of the W-GAN loss - $G$ and $D$ loss, along with the Inception score and the recovery. The results are arranged such that we first present the results obtained by changing the norms, followed by the plots showing the WGAN loss when the clipping values of WGAN are changed and finally the impact of the training on the results are presented in the end.
4.3.1  $\ell_1$ Norm vs $\ell_2$ Norm

In this section, we present the comparison between the WGAN loss and the recovery for $\ell_1$ norm and $\ell_2$ norm.

4.3.1.1 Data set: MNIST

This section discusses the results for MNIST data set for $\ell_1$ in comparison with $\ell_2$ norm. The WGAN settings such as norm used, data set, clipping values and the number of iterations to obtain the result are shown in Table 4.1. The $D$ and the $G$ loss for $\ell_1$ norm is shown in Figure 4.4a and Figure 4.4b shows the loss for $\ell_2$ norm. For better comparison both the results are put together in Figure 4.4. The recovery of the samples generated by WGANs is compared in Figure 4.5, where Figure 4.5a shows the recovery from $\ell_1$ norm and Figure 4.5b for $\ell_2$ norm.

We shall note that in this thesis, we do not claim that the changes in the norm will give us better results. The hypothesis as stated earlier is to show that the changes in the norm have an impact on the learning of WGANs which translates to the effect on the $D$ and the $G$ loss and thus the recovery.

<table>
<thead>
<tr>
<th>Norm</th>
<th>$\ell_1$</th>
<th>$\ell_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data set</td>
<td>MNIST</td>
<td>MNIST</td>
</tr>
<tr>
<td>Clipping values</td>
<td>[-0.1, +0.1]</td>
<td>[0, 0.01 * avg($\ell_2$)]</td>
</tr>
<tr>
<td>Number of Iterations</td>
<td>1M</td>
<td>1M</td>
</tr>
</tbody>
</table>

Table 4.1: Settings to obtain results for MNIST - $\ell_1$ norm and $\ell_2$ norm
Figure 4.4: $G$ and $D$ loss for MNIST - $\ell_1$ and $\ell_2$ norm

(a) Recovery using $\ell_1$ norm with MNIST
(b) Recovery using $\ell_2$ norm with MNIST

Figure 4.5: Recovery for MNIST - $\ell_1$ and $\ell_2$ norm

The $D$ loss and the $G$ loss for $\ell_1$ norm converge and the WGAN starts learning after training for few thousands of iterations. The recovery for $\ell_1$ norm has a fair quality of the image generated, and the digits are quite recognizable as shown in Figure 4.5. It is interesting to notice that the $D$ loss and the $G$ loss for $\ell_2$ norm shows a significant difference when compared to $\ell_1$. Initially using $\ell_2$ the WGAN has a higher bias, and it does not learn meaningful weights. After training WGAN for few thousands of iterations, we see WGAN starts
learning which continues in the convergence of the loss as shown in Figure 4.5b. In the case of MNIST, we rely on human inspection to evaluate the recovery.

Comparing the training of WGAN with $\ell_1$, we can infer that the WGAN when trained with $\ell_2$ norm learns slowly, but the recovery using both the norms are similar.
4.3.1.2 Data set : CIFAR-10

This section discusses the results for CIFAR-10 data set for $\ell_1$ in comparison with $\ell_2$ norm. The WGAN settings such as norm used, data set, clipping values and the number of iterations to obtain the result are shown in Table 4.2. The results of the comparison are displayed using the $D$ loss and the inception score. We also compare the recovery for both the norms.

<table>
<thead>
<tr>
<th>Norm</th>
<th>$\ell_1$</th>
<th>$\ell_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data set</td>
<td>CIFAR-10</td>
<td>CIFAR-10</td>
</tr>
<tr>
<td>Clipping values</td>
<td>[-0.1, +0.1]</td>
<td>[0, 0.01 * avg $\ell_2$ norm]</td>
</tr>
<tr>
<td>Number of Iterations</td>
<td>1M</td>
<td>1M</td>
</tr>
</tbody>
</table>

Table 4.2: Settings to obtain results for CIFAR-10 - $\ell_1$ and $\ell_2$ norm norm

![Figure 4.6](image)

(a) $D$ loss for CIFAR-10 - $\ell_1$ norm  
(b) $D$ loss for CIFAR-10 - $\ell_2$ norm

Figure 4.6: $D$ loss for CIFAR-10 - $\ell_1$ and $\ell_2$ norm

The comparison of $D$ loss is shown in Figure 4.6. The $D$ loss for $\ell_1$ norm is shown in Figure 4.6a and Figure 4.6b shows the $D$ loss for $\ell_2$ norm.
With $\ell_2$ norm, we see the WGAN learning rate is slower when compared to $\ell_1$ norm. The $D$ loss with $\ell_2$ norm shows high variance as the learning continues.

The inception score is compared in Figure 4.7. The $\ell_1$ norm inception score is shown in Figure 4.7a, and for $\ell_2$ norm we see the plot on Figure 4.7b. Inception score gradually increases and becomes stable as and when the training is stable. Towards the end when $D$ loss increases, this is reflected in
the inception score. Inception score drastically decreases towards the end.

Finally the recovery is shown in Figure 4.8, where 4.8a shows the recovery for $\ell_1$ norm and 4.8b shown the recovery for $\ell_2$ norm. Using $\ell_2$ norm, we have lesser number of samples recovered with finer features of CIFAR-10 in comparison to $\ell_1$ norm.
4.3.1.3 Data set: Toy-8 Gaussian

In this section, the results obtained for Toy data set with 8 Gaussian is presented. The WGAN settings are shown in table 4.3.

<table>
<thead>
<tr>
<th>Norm</th>
<th>$\ell_1$</th>
<th>$\ell_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data set</td>
<td>TOY-8 Gaussian</td>
<td>TOY-8 Gaussian</td>
</tr>
<tr>
<td>Clipping values</td>
<td>$[-0.1, +0.1]$</td>
<td>$[0, 0.01 \times \text{avg(norm-2)}]$</td>
</tr>
<tr>
<td>Number of Iterations</td>
<td>1M</td>
<td>1M</td>
</tr>
</tbody>
</table>

Table 4.3: Settings to obtain results for Toy - 8 Gaussian data using $\ell_1$ norm

![Figure 4.9: D loss for Toy - 8 Gaussian - $\ell_1$ and $\ell_2$ norm](image)

Figure 4.9: $D$ loss for Toy - 8 Gaussian - $\ell_1$ and $\ell_2$ norm

The comparison of $D$ loss is shown in Figure 4.9. The $D$ loss for $\ell_1$ norm is shown in Figure 4.9a and Figure 4.9b shows the $D$ loss for $\ell_2$ norm. With $\ell_1$ norm, WGAN’s learning is stable. As a result of WGAN training with $\ell_2$ norm we notice loss is fanning out towards the end of the training. The reason for this may be because of the outlying nature of $\ell_2$ norm. The $\ell_2$ norm is highly sensitive to the outliers and less robust than $\ell_1$ norm.
Figure 4.10: Recovery for Toy - 8 Gaussian - $\ell_1$ and $\ell_2$ norm

This effect is similar to the least square error analysis in regression when the data has many outliers. The recovery for $\ell_1$ norm shows it has stable contours. In the case of $\ell_2$ norm it is qualitatively different with larger variance.
4.3.2 $\ell_1$ Norm vs $\ell_\infty$ Norm

In this section, we present the comparison between the WGAN loss and the recovery for $\ell_1$ norm and $\ell_\infty$ norm.

4.3.2.1 Data set: MNIST

This section discusses the results for MNIST data set for $\ell_1$ in comparison with $\ell_\infty$ norm. The WGAN settings such as norm used, data set, clipping values and the number of iterations to obtain the result are shown in Table 4.4.

<table>
<thead>
<tr>
<th>Norm</th>
<th>$\ell_1$</th>
<th>$\ell_\infty$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data set</td>
<td>MNIST</td>
<td>MNIST</td>
</tr>
<tr>
<td>Clipping values</td>
<td>[-0.1, +0.1]</td>
<td>[0, 0.01 * avg($\ell_\infty$) norm]</td>
</tr>
<tr>
<td>Number of Iterations</td>
<td>1M</td>
<td>1M</td>
</tr>
</tbody>
</table>

Table 4.4: Settings to obtain results for MNIST - $\ell_1$ norm and $\ell_\infty$ norm

![Loss vs Iterations](image1)

(a) $G$ and $D$ loss for MNIST - $\ell_1$ norm

![Loss vs Iterations](image2)

(b) $G$ and $D$ loss for MNIST - $\ell_\infty$ norm

Figure 4.11: $G$ and $D$ loss for MNIST - $\ell_1$ and $\ell_\infty$ norm
The $D$ and the $G$ loss for $\ell_1$ norm is shown in Figure 4.11a and Figure 4.11b shows the loss for $\ell_\infty$ norm. The results are put together in Figure 4.11. In the case of $\ell_\infty$ norm, the $D$ loss and the $G$ loss do not converge which was observed when $\ell_2$ norm was used. The variance remains high. The recovery of the samples generated by WGANs is compared in Figure 4.12, where Figure 4.12a shows the recovery from $\ell_1$ norm and Figure 4.12b for $\ell_\infty$ norm.

### 4.3.2.2 Data set: CIFAR-10

This section discusses the results for CIFAR-10 data set for $\ell_1$ in comparison with $\ell_\infty$ norm. The WGAN settings such as norm used, data set, clipping values and the number of iterations to obtain the result are shown in Table 4.5. The results of the comparison are displayed using the $D$ loss and the Inception score. We also compare the recovery for both the norms.
<table>
<thead>
<tr>
<th>Norm</th>
<th>$\ell_1$</th>
<th>$\ell_\infty$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data set</td>
<td>CIFAR-10</td>
<td>CIFAR-10</td>
</tr>
<tr>
<td>Clipping values</td>
<td>[-0.1, +0.1]</td>
<td>$[0, 0.001 \times \text{avg}(\ell_\infty\text{-norm})]$</td>
</tr>
<tr>
<td>Number of Iterations</td>
<td>1M</td>
<td>1M</td>
</tr>
</tbody>
</table>

Table 4.5: Settings to obtain results for CIFAR-10 using $\ell_1$ and $\ell_\infty$ norm

Figure 4.13: $D$ loss for CIFAR-10 - $\ell_1$ and $\ell_\infty$ norm

Figure 4.14: Inception Score for CIFAR-10 - $\ell_1$ and $\ell_\infty$ norm
The comparison of $D$ loss is shown in Figure 4.13. The $D$ loss for $\ell_1$ norm is shown in Figure 4.13a and Figure 4.13b shows the $D$ loss for $\ell_\infty$ norm. With inf norm, we see the WGAN learning is highly unstable compared to $\ell_1$ norm. During the training for a brief period, the WGAN fails to learn anything and the loss is extremely high. Towards the end of the training, the loss reduces, and WGAN starts learning meaningful weights.

The inception score is compared in Figure 4.14. The $\ell_1$ norm inception score is shown in Figure 4.7a, and for $\ell_\infty$ norm we see the plot on Figure 4.7b. We notice that the inception score is not steady and it highly varies. It corresponds to the training of WGAN with $\ell_\infty$ norm. Towards the end, after 500000 iterations there is an increase in the inception score.

Finally the recovery is shown in Figure 4.15, where 4.15a shows the recovery for $\ell_1$ norm and 4.15b shown the recovery for $\ell_\infty$ norm. Using $\ell_2$ norm, the number of samples recovered with finer features of CIFAR-10 is
lesser when compared to $\ell_1$ norm. When the inception score is more than 3.0 we notice the recovery generated has samples with finer textures.

### 4.3.2.3 Data set: Toy - 8 Gaussian

In this section, the results obtained for Toy data set with 8 Gaussian is presented. The WGAN settings are shown in table 4.6.

<table>
<thead>
<tr>
<th>Norm</th>
<th>$\ell_1$</th>
<th>$\ell_\infty$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data set</td>
<td>TOY-8 Gaussian</td>
<td>TOY-8 Gaussian</td>
</tr>
<tr>
<td>Clipping values</td>
<td>[-0.1, +0.1]</td>
<td>$[0, 0.01 \times \text{avg}(\ell_\infty\text{-norm})]$</td>
</tr>
<tr>
<td>Number of Iterations</td>
<td>1M</td>
<td>1M</td>
</tr>
</tbody>
</table>

Table 4.6: Settings to obtain results for Toy - 8 Gaussian data using $\ell_1$ norm and $\ell_\infty$ norm.

![Figure 4.16](image)

(a) $D$ loss for Toy - 8 Gaussian data - $\ell_1$ norm  
(b) $D$ loss for Toy - 8 Gaussian data - $\ell_2$ norm

Figure 4.16: $D$ loss for Toy - 8 Gaussian - $\ell_1$ and $\ell_\infty$ norm

The comparison of $D$ loss is shown in Figure 4.16. The $D$ loss for $\ell_1$ norm is shown in Figure 4.16a and Figure 4.16b shows the $D$ loss for $\ell_\infty$ norm.
The recovery using the $\ell_1$ norm is shown in Figure 4.17a and Figure 4.17b recovery for $\ell_\infty$ norm.

With Toy-8 Gaussian data we observe that the results obtained for $\ell_\infty$ norm is similar to $\ell_2$ norm. The training becomes difficult, and the WGAN loss starts fanning out in the end. In the recovery, the variance remains high using $\ell_\infty$ norm.
4.3.3 $\ell_1$ Norm vs. $\ell_{enet}$ Norm

In this section, we present the comparison between the WGAN loss and the recovery for $\ell_1$ norm and $\ell_{enet}$ norm. Note that $\ell_{enet}$ is the linear combination of $\ell_1$ norm and $\ell_2$ norm.

4.3.3.1 Data set: MNIST

This section discusses the results for MNIST data set for $\ell_1$ in comparison with $\ell_{enet}$ norm. The WGAN settings such as norm used, data set, clipping values and the number of iterations to obtain the result are shown in Table 4.7.

<table>
<thead>
<tr>
<th>Norm</th>
<th>$\ell_1$</th>
<th>$\ell_{enet}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data set</td>
<td>MNIST</td>
<td>MNIST</td>
</tr>
<tr>
<td>Clipping values</td>
<td>[-0.1, +0.1]</td>
<td>[0, 0.01 * (avg($\ell_{enet}$)) norm]</td>
</tr>
<tr>
<td>Number of Iterations</td>
<td>1M</td>
<td>1M</td>
</tr>
</tbody>
</table>

Table 4.7: Settings to obtain results for MNIST using $\ell_1$ norm and $\ell_{enet}$ norm

The $D$ and the $G$ loss for $\ell_1$ norm is shown in Figure 4.18a and Figure 4.18b shows the loss for $\ell_{enet}$ norm. The results are put together in Figure 4.18.

The $D$ and the $G$ loss obtained for $\ell_{enet}$ norm is similar to $\ell_{\infty}$ norm. It does not fully converge when compared to $\ell_2$ norm. Even in this case, the variance remains high after the loss starts slightly converging after 600K iterations.
Figure 4.18: $G$ and $D$ loss for MNIST - $\ell_1$ and $\ell_{\text{enet}}$ norm

Figure 4.19: Recovery for MNIST data set with $\ell_1$ and $\ell_{\text{enet}}$ norm

The recovery of the samples generated by WGANs is compared in Figure 4.19, where Figure 4.19a shows the recovery from $\ell_1$ norm and Figure 4.19b for $\ell_{\text{enet}}$ norm. The quality of recovery is similar to $\ell_1$ norm in this case.
4.3.3.2 Data set: CIFAR-10

This section discusses the results for CIFAR-10 data set for $\ell_1$ in comparison with $\ell_{enet}$ norm. The WGAN settings such as norm used, data set, clipping values and the number of iterations to obtain the result are shown in Table 4.8. The results of the comparison are displayed using the $D$ loss and the Inception score. We also compare the recovery for both the norms.

<table>
<thead>
<tr>
<th>Norm</th>
<th>$\ell_1$</th>
<th>$\ell_{enet}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data set</td>
<td>CIFAR-10</td>
<td>CIFAR-10</td>
</tr>
<tr>
<td>Clipping values</td>
<td>[-0.1, +0.1]</td>
<td>[0, +0.001 * (avg ($\ell_{enet}$ norm))]</td>
</tr>
<tr>
<td>Number of Iterations</td>
<td>1M</td>
<td>1M</td>
</tr>
</tbody>
</table>

Table 4.8: Settings to obtain results for CIFAR-10 using $\ell_1$ and $\ell_{enet}$ norm

Figure 4.20: $D$ loss for CIFAR-10 - $\ell_1$ and $\ell_{enet}$ norm

70
The comparison of $D$ loss is shown in Figure 4.20. The $D$ loss for $\ell_1$ norm is shown in Figure 4.20a and Figure 4.20b shows the $D$ loss for $\ell_{enet}$ norm. CIFAR-10 data set with $\ell_{enet}$ norm is unsuccessful in learning any parameters. We notice from the $D$ loss that WGAN loss remains high when $\ell_{enet}$ is used. This loss indicates that WGAN’s performance also depends on the data set used and this is verified empirically because with MNIST data set, and the
same WGAN setting we could see WGAN performed fairly well. The Inception score is compared in Figure 4.21. The $\ell_1$ norm inception score is shown in Figure 4.21a, and for $\ell_{enet}$ norm we see the plot in the Figure 4.21b. Inception score corresponds to the training. It decreases and reaches as low as 2.0. Finally the recovery is shown in Figure 4.22, where 4.22a shows the recovery for $\ell_1$ norm and 4.22b shown the recovery for $\ell_{enet}$ norm. We see a blurry image and the recovery does not correspond to any meaningful image.
4.3.4 Clipping Values

The performance of WGAN was highly sensitive to the clipping values. In this section, we present the results of WGAN loss for $\ell_2$ norms with different clipping values. These results were obtained during the initial stages of our exploration.

4.3.4.1 MNIST - clipping values 0.1, 0.01, 0.001

This result corresponds to $\ell_2$ norm with different clipping constants used are shown in Table 4.9. In this case, we compare the results obtained for $c = 0.1, 0.01$ and 0.001.

<table>
<thead>
<tr>
<th>Norm</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data set</td>
<td>MNIST</td>
</tr>
<tr>
<td>Clipping values - (i)</td>
<td>$[0, 0.1 \times \text{avg}(\ell_2 \text{ norm})]$</td>
</tr>
<tr>
<td>Clipping values - (ii)</td>
<td>$[0, 0.01 \times \text{avg}(\ell_2 \text{ norm})]$</td>
</tr>
<tr>
<td>Clipping values - (iii)</td>
<td>$[0, 0.001 \times \text{avg}(\ell_2 \text{ norm})]$</td>
</tr>
<tr>
<td>Number of Iterations</td>
<td>1M</td>
</tr>
</tbody>
</table>

Table 4.9: Settings to obtain results for different clipping values 0.1, 0.01 and 0.001 for MNIST data set
Figure 4.23: The difference in the $D$ and $G$ loss corresponding to different clipping values for MNIST data set
Figure 4.24: The difference in the recovery corresponding to different clipping values for MNIST data set

The results are shown in Figure 4.23 corresponds to the $G$ and $D$ loss obtained for WGAN using the MNIST data set. The results presented here uses $\ell_2$ norm. $D$ and $G$ loss for different clipping constants are presented.

Firstly when clipping constant $c = 0.1$ is used, the corresponding $D$ and $G$ loss is shown in Figure 4.23a. Using this constant the loss does not converge. Whereas, when we use clipping constant $c = 0.01$ the WGAN loss starts converging 4.23b with larger variance. The results for clipping constant $c = 0.001$ is similar to $c = 0.1$ as shown in Figure 4.23c.
The recovery for various clipping values are shown in Figure 4.24. Though the recovery may look similar, we notice that in the case of clipping value \( C = 0.01 \) higher quality images are generated throughout after the loss converges, one of the samples shown in Figure 4.24b. The convergence is not similar to the clipping value of 0.01 when clipping values 0.1 and 0.001 is used on WGAN. The samples may not have a good recovery during certain stages of training.

With these plots and recovery samples, we can infer that clipping values have a significant impact on the training and the recovery.

4.3.4.2 CIFAR-10 - clipping values 0.1, 0.01, 0.001

This result corresponds to \( \ell_2 \) norm with different clipping constants used are shown in Table 4.10 using CIFAR-10 data set. In this case, we compare the results obtained for \( c = 0.1, 0.01 \) and 0.001.

| Clipping values - (i) | \([0, 0.1 * (\text{avg}(\ell_2 \text{ norm}))]\) |
| Clipping values - (ii) | \([0, 0.01 * (\text{avg}(\ell_2 \text{ norm}))]\) |
| Clipping values - (iii) | \([0, 0.001 * (\text{avg}(\ell_2 \text{ norm}))]\) |

| Table 4.10: Settings to obtain results for different clipping values 0.1, 0.01 and 0.001 for CIFAR-10 data set |

The results are shown in Figure 4.25 corresponds to the \( D \) loss obtained for WGAN using the CIFAR-10 data set. The results presented here uses \( \ell_2 \)
(a) $D$ loss for CIFAR-10 for clip value: 0.1  
(b) $D$ loss for CIFAR-10 for clip value: 0.01  
(c) $D$ loss for CIFAR-10 for clip value: 0.001

Figure 4.25: The difference in the $D$ loss corresponding to different clipping values for CIFAR-10 data set
(a) Inception Score for CIFAR-10 for clip value : 0.1

(b) Inception Score for CIFAR-10 for clip value : 0.01

(c) Inception Score for CIFAR-10 for clip value : 0.001

Figure 4.26: The difference in the inception Score corresponding to different clipping values for CIFAR-10 data set
Figure 4.27: The difference in the recovery corresponding to different clipping values for CIFAR-10 data set
Finally when clipping constant $c = 0.1$ is used, the corresponding $D$ loss is shown in Figure 4.25a. Using this constant, the WGAN learning rate is slower in the starting of the training and the loss is high. When the training continues, the loss decreases and becomes stable.

Whereas, when we use clipping constant $c = 0.01$ the WGAN training is stable in the beginning 4.25b and the loss reduces. Towards the end, the training abruptly fails and the loss increases as shown in 4.25b. The results for clipping constant $c = 0.001$ shows that it learns slower than $c = 0.1$ and the loss remains high in the beginning. After training for a few thousands of iterations, the loss reduces as shown in Figure 4.25c. In this thesis, we use this clipping values for our exploration with the CIFAR-10 data set.

The inception scores is shown in Figure 4.26 and the recovery is shown in Figure 4.27. Both the inception score and the recovery corresponds to the $D$ loss obtained for $c = 0.1, 0.01, 0.001$. With clipping value $c = 0.1$ and $c = 0.01$ we see that the inception score do not show a steady growth. Rather in the case of $c = 0.1$ it decreases at certain iterations around 600K and 800K iterations as shown in Figure 4.26a, and at $c = 0.01$ the inception score decreases drastically toward the end of the training as shown in Figure 4.26b. The quality of the recovery for $c = 0.1$ is better than $c = 0.01$. A comparison can be seen in Figure 4.27a and 4.27c.

In the case of CIFAR-10 data set clipping value $c = 0.001$ outperforms
the others. It has a steady growth in the inception score which continues to increase around 3.5 as we can see in Figure 4.26c. The impact of this score can be seen in recovery where more number of samples have a finer texture as shown in Figure 4.27c.
4.3.4.3 Toy - 8 Gaussian - clipping values 0.01, 0.001

These results are obtained for Toy-8 Gaussian data with $\ell_2$ norm. We have the results for clip values $c = 0.01$, and $c = 0.001$ as shown in Table 4.11.

<table>
<thead>
<tr>
<th>Norm</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data set</td>
<td>Toy - 8 Gaussian</td>
</tr>
<tr>
<td>Clipping values - (i)</td>
<td>$[0, 0.01 \times \text{avg}(\ell_2 \text{ norm})]$</td>
</tr>
<tr>
<td>Clipping values - (ii)</td>
<td>$[0, 0.001 \times \text{avg}(\ell_2 \text{ norm})]$</td>
</tr>
<tr>
<td>Number of Iterations</td>
<td>1M</td>
</tr>
</tbody>
</table>

Table 4.11: Settings to obtain results for Toy - 8 Gaussian data for clip value $= 0.01$

![Graphs](image)

(a) $D$ loss for Toy - 8 Gaussian data for clip value : 0.01  
(b) $D$ loss for Toy - 8 Gaussian data for clip value : 0.001

Figure 4.28: The difference in the $D$ and $G$ loss corresponding to different clipping values for MNIST data set

The results are shown in Figure 4.28 corresponds to the $D$ loss obtained for WGAN using the Toy - 8 Gaussian data set. The results presented here uses $\ell_2$ norm. $D$ loss for different clipping constants is presented.
Figure 4.29: The difference in the $D$ and $G$ loss corresponding to different clipping values for MNIST data set

The $D$ loss for both the clipping values fans out at the end. We can see that in the Figure 4.28a for clip value $c = 0.01$ and Figure 4.28b for clip value $c = 0.001$. The corresponding recovery we can observe when $c = 0.01$ is used we have a lower variance as shown in Figure 4.29a when compared to $c = 0.01$ as shown in Figure 4.29b.
4.3.5 Number of iterations

The number of training iterations matters in WGANs. To demonstrate this, we show the below results. We can see as the number of training iterations increases the loss decreases, and we get better recovery. We test this on MNIST data set and present the results for $\ell_2$ norm.

<table>
<thead>
<tr>
<th>Norm</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data set</td>
<td>MNIST</td>
</tr>
<tr>
<td>Clipping values</td>
<td>$[0, 0.01 \times (\text{avg}(\ell_2 \text{ norm}))]$</td>
</tr>
<tr>
<td>Number of Iterations - (i)</td>
<td>160K</td>
</tr>
<tr>
<td>Number of Iterations - (ii)</td>
<td>400K</td>
</tr>
<tr>
<td>Number of Iterations - (iii)</td>
<td>800K</td>
</tr>
<tr>
<td>Number of Iterations - (iv)</td>
<td>1M</td>
</tr>
</tbody>
</table>

Table 4.12: Settings to obtain results for MNIST, 160K iterations
Figure 4.30: The difference in the \( D \) and \( G \) loss corresponding to different number of training iterations for MNIST data set
Figure 4.31: The difference in the recovery corresponding to different number of training iterations for MNIST data set

The settings used to demonstrate the impact of the number of training iterations is shown in Table 4.3.5. Comparing the different $G$ and the $D$ loss for MNIST as shown in the Figure 4.30 it is evident that the learning rate for WGAN is slower. When they are given enough time and compute capacity it the loss decreases and the samples recovered have better quality. The loss
for 160 K number of training iteration is shown in Figure 4.30a, following by 400K in Figure 4.30b, 800K in Figure 4.30c, and 1M iterations in Figure 4.30d.

Similarly recovery for 160 K number of training iteration is shown in Figure 4.31a, following by 400K in Figure 4.31b, 800K in Figure 4.31c, and 1M iterations in Figure 4.31d.

In the next chapter, we discuss the conclusion and future work of this thesis research.
Chapter 5

Conclusion and Future Work

In this thesis, we give a comparative study of the impact of norms on WGANs. Inspired by the success of the $\ell_1$ norm in LASSO Regression, $\ell_2$ norm in Ridge Regression and the success of elastic net which uses the linear combination of $\ell_1$ and $\ell_2$ norm, we explore these norms in WGANs. We have empirically evaluated the impact of $\ell_1$, $\ell_2$, $\ell_\infty$ and $\ell_{enet}$ on real data sets like MNIST, CIFAR-10 along with the synthetic data set like eight bivariate Gaussian random variable. We established the qualitative and quantitative difference between these norms by measuring the (a) shape, value and pattern of the generator loss, (b) shape, value and pattern of the discriminator loss (c) shape, value and pattern of the inception score, and (d) human inspection of quality of recovery or reconstruction of images and scenes.

Firstly, on MNIST data set when WGANs weights are clamped with $\ell_1$ norm the $D$ and $G$ loss has a lower variance. Hence the recovery of the digits are of decent quality images during the initial training and as the training continues the quality of the image decreases. This decrease in the quality of the images is mainly because the $D$ and $G$ loss starts diverging after the initial convergence. Whereas, when the other norms such as $\ell_2$, $\ell_\infty$ and $\ell_{enet}$ norms
are compared with $\ell_1$ norm, the $D$ and $G$ loss converges when $\ell_2$ norm is used. Using $\ell_2$ norm, $D$ and $G$ loss have high variance, and WGAN produces the best quality of the images when recovery is analyzed. The $\ell_\infty$ and $\ell_{enet}$ norm have a similar $G$ and $D$ loss which showing the slight convergence. The recovery of these two norms is similar. With our results and analysis, when data sets similar to MNIST with simpler data distributions are used, $\ell_2$ norm may give us a better recovery compared to other norms.

When CIFAR-10 data set is empirically evaluated with different norms, $\ell_1$ norm had smaller $D$ loss and with an inception score of 5.0 and they had the best recovery amongst the other norms like $\ell_2$, $\ell_\infty$ and $\ell_{enet}$ used for comparison. With $\ell_2$ norm we observe that the $D$ loss has very high variance and the highest inception score achieved is 3.5. Thus it gives a lower quality of the images in the recovery. With $\ell_\infty$ norm we get an inception score of decreases further with a value of $xyz$. With $\ell_{enet}$ norm, WGANs performance further deteriorates. This deterioration of the performance of $\ell_{enet}$ with CIFAR-10 data set may be due to the dominance of the $\ell_2$ norm for that particular type of scenario. Which could be remedied by giving greater weight to $\ell_1$ than $\ell_2$. For instance $\alpha = 3/4$ in $\ell_{enet}$. It is also desirable like with other norms to directly control the optimization. From our analysis, we can recommend the use of $\ell_1$ norm for the compactness of the weights in WGANs when the data sets similar to CIFAR-10 is used. The $\ell_1$ norm’s performance may be better than $\ell_2$ because $\ell_2$ norm may not perform well with various data distributions observed in data sets like CIFAR-10.
Finally, with Toy - 8 Gaussian data set the only norm which has the best performance is $\ell_1$ norms. With $\ell_1$ norm the WGAN has a smaller loss and better recovery compared to the other norms. With $\ell_2$ and $\ell_\infty$ norm, the $D$ loss is much larger with higher variance, and it fans out during the training. The larger $D$ loss may be because the $\ell_2$ norm is sensitive to the outliers when compared to $\ell_1$ norm. The $\ell_1$ norm is more robust to the outliers, and this may be attributed to the least square analysis used in the regression.

In this thesis, we also demonstrate the impact of the clipping values on WGANs performance. With the change in the clipping values, the results obtained for different data sets also changes. We see with the different data sets the clipping value giving the best result changes. We empirically evaluate with constants $c = [0.1, 0.01, 0.001]$. With MNIST data set clipping constant of 0.01 gave better results whereas with CIFAR-10 0.001 worked better. In the case of Toy data, the results did not vary with any of the clipping constant used. We believe in selecting the clipping values a more principled approach like cross-validation would be ideal. However, this would be computationally expensive. We also see the impact of the number of training iterations on the $D$ loss, $G$ loss and the recovery. The results obtained by changing the norm from $\ell_1$ to any other norm significantly improves when the WGAN is given enough training time and compute capacity.

As an extension of this thesis, in future work, we would be interested in giving a taxonomy of the impact of various norms on WGANs and giving a thorough analysis to check which norms would provide the best results for
the applications. We would also like to see the impact on WGAN loss and the recovery when other methods of the projection of weights such as standardization, cubitization, unitization of the weights are used in order to enforce the compactness.
Bibliography


[30] Artur Kadurin, Sergey I. Nikolenko, Kuzma Khrabrov, Alex Aliper, and Alex Zhavoronkov. drugan: An advanced generative adversarial autoen-


