International Journal of Advanced Technology and Engineering Exploration, Vol 9(94)
ISSN (Print): 2394-5443 ISSN (Online): 2394-7454
http://dx.doi.org/10.19101/ IJATEE.2021.875738

Research Article

Mathematical analysis of loss function of GAN and its loss function variants

Rayeesa Mehmood, Rumaan Bashir* and Kaiser J. Giri
Department of Computer Science, Islamic University of Science and Technology, Kashmir, Jammu and Kashmir, 192122, India

Received: 04-April-2022; Revised: 19-September-2022; Accepted: 21-September-2022

Abstract
Generative adversarial networks (GANs) have turned up as the most widely used approaches for creating realistic samples. They're the effective latent variable models for learning complex real distributions. However, despite their enormous success and popularity, the process of training GANs remains challenging and suffers from a number of failures. These failures include mode collapse where the generator generates the same set of output for different inputs which finally leads to loss of diversity; non-convergence because of the diverging and oscillatory behaviors of both generator and discriminator while being trained; and vanishing or exploding gradients due to which either no learning or extremely slow learning takes place. In the past years, a variety of strategies for stabilizing GAN training have been explored which includes modified architectures, loss functions, and other methods. The choice of loss function has been found to be the most crucial part of the GAN model because it influences the vanishing gradient and model collapse directly. Viewing these loss functions as divergence minimization has provided a rich avenue of development. All of these factors make GAN training inherently unstable, and this instability is difficult to comprehend mathematically. This paper intends to provide a thorough mathematical explanation of these divergence minimization functions. It illustrates in great detail the two variants of the loss functions of the original GAN, their optimization to Kullback-Leibler (KL) divergence and Jensen-Shannon (JS) divergence along with their shortcomings. It also describes the loss functions of the different loss function GAN variants that have been proposed to mitigate these shortcomings as well as their minimization. The original GAN and its loss function variants have also been implemented using the standard MNIST, Fashion-MNIST, and CIFAR-10 datasets.

Keywords
Generative adversarial networks, Divergence minimization, Loss functions, Stable training, Mode collapse, Non-convergence.

1.Introduction
Machine learning has enjoyed a diversified history, having its origin in many interdisciplinary subjects like computational learning theory and pattern recognition, cognitive science, neuroscience, and other disciplines [1]. This field of computer science does not require machines to be programmed using static or rigid instructions in order to act. It focuses on developing algorithms that learn from the data and then make predictions based on that data set without human intervention. There are two important contemporary paradigms in machine learning. The first is generative or Bayesian learning and the other one is discriminative learning of classifiers [2, 3].

The generative model is a strong unsupervised learning method for learning any distribution of data and has experienced considerable success in a short amount of time. The idea behind such models is to generate new data instances or configurations. They can generate new photos of different types of objects that look like real ones. They enable users to provide information about the problem to the learning algorithm using prior distributions, structured models, independence graphs, probabilistic reasoning, Markov assumptions, and latent variables. It includes the data's distribution and indicates the likelihood of a certain example. For instance, the models used to determine the subsequent word in a series belong to the category of generative models because each word in the sequence is assigned a probability. These generative models include mixtures of multinomial, mixtures of experts, naïve

*Author for correspondence

Bayes, Bayesian networks, mixtures of Gaussians, hidden Markov models (HMM), Markov random fields, and sigmoidal belief networks [4]. Discriminative models, on the other hand, simply discriminate between different instances of data as being real or fake. These models instead of showing whether a given instance is likely, simply tell how likely it is to apply a label to the instance. The emphasis here is on categorization rather than generation, allowing for efficient allocation of computational resources to the task at hand. The most well-known models include support vector machines (SVMs), traditional neural networks, conditional random fields (CRF), logistic regression, and nearest neighbour [5]. Mathematically, data samples X and labels Y, generative algorithms measure the joint probability p(X, Y) if both instances, as well as labels, are present, or simply p(X) for the absence of labels, whereas discriminative models just capture the conditional probability of p(Y) given x as p(Y|X).

The conventional machine learning methods could not be applied directly to the raw data like .csv files, images, text, etc., and required some pre-processing prior to the application [6]. Also, they tend to succumb to different environmental changes and stop improving after reaching a certain saturation point. All this led to a new subset of machine learning called deep learning. It was introduced with the objective to bring traditional machine learning close to the artificial intelligence. Deep learning draws its roots from Kunihiko Fukushima's Neocognitron, which is an artificial neural network (ANN) [7] and uses multiple-layer neural networks that imitate the working of the neurons in the human brain. Unlike machine learning, they adapt to the changes by regular feedback and enhance the model. Deep learning models tend to get more accurate as the amount of training data grows [8]. To bring out a clear comparison between machine learning and deep learning, consider the following example: In a face detection task, if a machine learning algorithm learns about the characteristics of the face such as the eyes and nose, a deep learning algorithm will learn deeper aspect like the distance between the eyes and the shape of the nose. Furthermore, while previous machine learning generative models captured the real distribution of training data samples to produce new data points, understanding the actual distribution of the data was not always attainable, either implicitly or explicitly. Using the deep neural networks with these traditional generative models, it became possible to build a distribution that is very close to the true distribution of data. This combination represented a new class of models, named as deep generative models (DGMs). The idea behind DGMs is that generative neural network models have fewer parameters than the volume of data needed to train them on, thereby enabling them in identifying the essence of the data to produce it. When trained successfully, DGMs estimate the likelihood of each observation and thus generate images from that underlying distribution. DGMs have recently made headlines for their number of applications including producing false celebrity images from their real images. The emergence of what is referred to as "deep fakes", promises new beneficial technologies along with the number of scientific applications of DGMs in almost every field [9]. Deep learning generative algorithms include restricted Boltzmann machines (RBMs), deep Boltzmann machines (DBMs), deep belief networks (DBNs), generative stochastic networks (GSNs), denoising auto encoder (DAE), variational autoencoder and generative adversarial networks (GANs).

Compared to other DGMs, GANs [10], have sparked considerable interest [11, 12]. GANs have a variety of advantages, including the capacity to deal with sharp estimated density functions, effectively generate required output, eliminate deterministic bias, and, most importantly, are compatible with the neural network architecture. All these characteristics have contributed to the extraordinary success of GANs. However, GANs are not without problems. The problem with these GANs is that their training is unstable, making them hard to train. When it comes to being hard to train [13], it is challenging for both the neural networks of the GAN to attain the optimization goal i.e. Nash equilibrium [14]. They also suffer from the mode collapse since the generator is usually not able to learn the entire distribution of data, thereby producing limited samples from the real data [15]. Moreover, the selection of an optimizer for training the GANs is a serious issue [16]. To tackle these issues, a number of adversarial loss functions in order to train either generative or discriminative models have been proposed [17]. These loss functions are used to compute disparities between the samples produced by the models and the samples present in the real dataset, which finally is used by the model to learn towards the goal. For the minimization of the error, while training the network and to speed up their converging process, the optimization of loss functions is an extremely crucial issue to study.

The original paper on GAN introduces two loss functions viz, the saturated or min-max loss function and the non-saturating loss function. These loss functions have been further modified to yield different GAN loss variants. The mathematical intuition behind these loss functions is difficult to comprehend [18]. Also, none of the works have attempted to describe the complete mathematics and optimization of these loss functions to different divergences. The purpose of this study is to have a complete understanding of the underlying mathematics of the loss functions of original GAN and to derive the divergences to which these loss functions are finally optimized. Also, though a number of papers have surveyed the loss variants, none has touched the mathematical background in detail. The following sequence gives the structure of this paper. Section 2 discusses the related work in detail. Section 3 gives an in-depth study of GAN, and its objective functions. It shows that the loss function of GAN is actually a binary cross entropy (BCE). This section is subdivided into four subsections. The first subsection explains the preliminary of GAN and its objective function, the second subsection explains in great detail the min-max objective function and its minimization to KL-divergence firstly and then finally to JS-divergence, the third subsection describes the next loss function called as the non-saturating loss function and shows its minimization to reverse KL-divergence and JS-divergence, and lastly, fourth subsection describes the mathematical intuition behind different modified loss functions that have been presented in the literature to mitigate the shortcomings of the original loss functions. Section 4 gives details of the datasets, and the architectures of these loss function GAN variants used in experiments and presents the results obtained. Section 5 provides a summarization of the key findings, a summary of the weaknesses and strengths of different loss variants and the limitations of this study. Section 6 concludes this study with a conclusion.

2.Related work

GANs have several potential advantages as a result of which they have been gaining considerable attention. The research pertaining to GANs can be classified into two broad groups. The first group is concerned with the application of GANs to different real-world problems in different domains like computer vision [19], natural language processing [20], biology [21], astronomy [22], networking [23], and other areas. GANs have been successful in generating better-resolution samples from poor-resolution ones [24], generating images and videos from textual descriptions [25, 26], and Image-to-Image Translation [27]. A super-resolution generative adversarial network (SRGAN) was suggested by Ledig et al. [28] that employs a content loss in addition to the adversarial loss and pushes the output to the natural image manifold. To further increase the image resolution, Li et al. [29] proposed Beby-GAN for highly detailed image super-resolution. The authors introduced a region-aware-based adversarial learning technique that makes the model to adaptively produce details for textured areas. To generate images of complex backgrounds from textual inputs, Quan et al. [30] proposed attention regularization and region proposal networks-based GAN (ARRPNGAN). It leverages these techniques in order to obtain most of the semantics from a textual description. Isola et al. [31] used conditional-GAN (CGAN) and proposed pix2pix which is an image translation approach for converting image content from one domain to another. In this framework nothing is application specific, thereby making this setup considerably simpler. For unpaired image translation, Zhao et al. [32] presented a lightweight domain-attention generative adversarial network (LDA-GAN). This GAN uses an enhanced domain-attention module (DAM) to create a longer range dependency between two domains while using fewer parameters and less memory. Additionally, GANs are becoming more and more popular in the fields of medical image processing for tasks like segmentation and classification [33], cybersecurity [34], time series and sequence generation [35] as well as for speech processing [36]. GANs have been used in the field of medical imaging for analyzing images from radiography, computerized tomography (CT) scans, and magnetic resonance imaging (MRI). To convert two-dimensional (2D) CT image slices of the brain into 2D MR image slices of the brain, Jin et al. [37] presented the magnetic resonance-GAN (MR-GAN) architecture. Repecka et al. [38] developed protein GAN that generates functional protein sequences. It is based on the concept of self-attention and thus learns the diversity of natural protein sequences.

Despite widespread use and ongoing progress, training GANs is extremely unstable and hard to converge. To solve these issues, a wide number of possible solutions have been put forth by the research community. This has led to the second category of GAN research and it focuses on dealing with solving the training issues faced by GANs. Radford et al. [39] proposed a stable set of architectures referred to as deep convolutional GANs (DCGAN) for training GANs. Further proof was presented by the authors

that this adversarial network can learn effective image representations for both supervised learning and generative modeling. Arjovsky et al. [40] devised a new training model called Wasserstein GAN (WGAN) based on Wasserstein's distance that uses weight clipping to circumvent the mode collapse issue. Gulrajni et al. [41] suggested an alternative to clipping weights and proposed a gradient penalty in WGAN (WGAN-GP) to deal with the vanishing gradient problem. Different techniques for stabilizing these training issues and the underlying theory have also been studied in [42, 43]. The authors in [42] assessed different gradient penalty regularizers and found that they differ from each other from the theoretical point of view only. Kodali et al. [43] also suggested another new version of the gradient penalty to deal with the mode collapse issue.

The popularity of GANs has also led to several studies and other survey papers as well that outline the concept and the applications of GANs. Sharma et al. [44] have focused on the application of GAN in the image and video domain. Jin et al. [45] examine the theoretical foundations of GANs, as well as some recently discovered GAN models. It also surveys the different applications of GAN in computer vision. Aggarwal et al. [46] review the theory of GAN along with its application in different applications for image segmentation. They have thrown light on the applications of GAN in medicine, pandemic, three-dimensional (3D) object generation, etc. Huang et al. [47] provide a taxonomy of the GAN models employed in the synthesis of images. The authors have reviewed different GANs for generating images from text and for translating images in different domains. Hitawala [48] presented a detailed and thorough comparison of the models of GAN on the basis of different parameters like methodology, architecture, and performance. Jabbar et al. [49] have reviewed different GAN variants, applications of GAN in image, audio, and video domains and have also described certain challenges faced by GAN during their training along with the methods to improve upon those challenges. Wali et al. [50] have presented a comprehensive review of GAN-frameworks for speech processing. They have also reviewed the datasets along with evaluation metrics and highlighted the issues faced by different speech GANs. Jozdani et al. [51] have presented a systematic review along with meta-analysis of GAN-based studies in remote sensing. The authors have also evaluated the GAN theories, applications, and difficulties and identified the research gaps that need to be addressed in the future by remote sensing researchers. Shahriar [52] has examined GANs for producing literary, musical, and artistic works. This survey paper has also done the performance analysis and has highlighted the challenges faced by these GAN in generating the visual art. Saxena and Cao [53] have surveyed the basic GANs framework, the key issues within them, the evolution of the better design of GANs, and their optimizations. Kurach et al. [54] described a few chosen loss functions, regularization normalization approaches along with the evaluation metrics used by GANs. Pan et al. [55] focused on the loss functions of the GANs as well as the loss function of the GAN variants used in different applications. Wiatrak et al. [56] provide a taxonomy of methods used for stabilizing the training procedure of GANs. Though all these papers mentioned above have described the basics of GAN, none has focused on the in-depth mathematical study of the original GAN loss functions or their optimization to different divergences. Also, none of them have entirely focused on the loss function variants describing their loss functions along with their optimization in detail.

3.Methods

3.1Preliminary of GAN and its objective function

GAN belongs to the class of DGM that is based on implicit deep learning [57]. More generally, GANs are the model architecture that is used to train generative models, and this architecture commonly uses deep learning models. The GAN architecture was first introduced in a paper in 2014, titled "Generative Adversarial Networks" by Ian Goodfellow. GAN is basically a network that combines two deep models: the generator G and the discriminator D as depicted in *Figure 1*. Generator, as the name suggests, creates the new sample instances which are discriminated by the discriminator from the real sample instances. The two models work in an adversary mode, competing with each other as a result of which both eventually get better and better in their own jobs. They play a two-player min-max game [58] that is a game wherein one seeks to maximize its winning probability whereas the other minimizes the probability of opponents winning. A noise vector z from Gaussian distribution is given as input to G which generates the G (z) as the output. Discriminator D classifies the input, it takes, as the true or generated data. It assigns a higher probability to real data that is if the input to the discriminator is x which is from real data distribution p_{data} and a lower probability to the generated data that is if x is extracted from the generators data distribution over x, p_g.

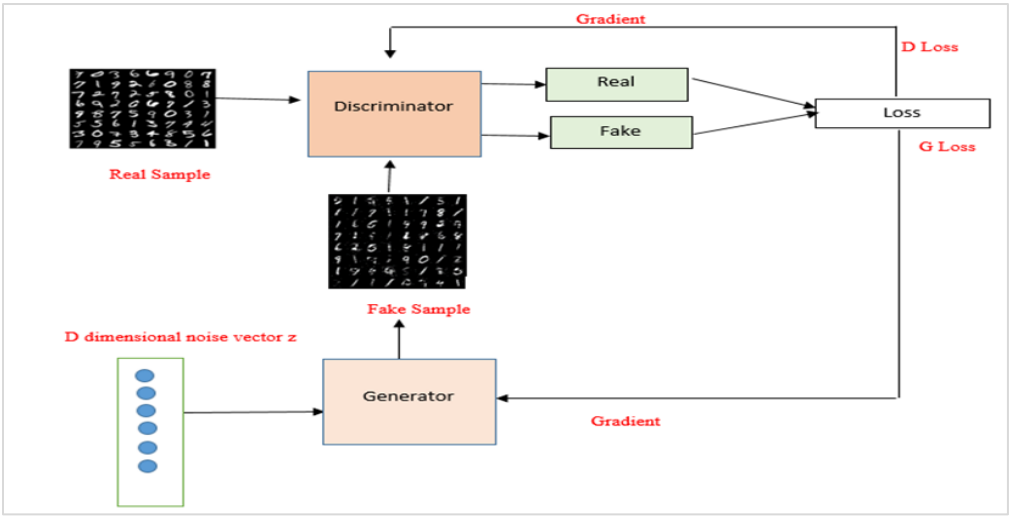

Figure 1 GAN Architecture

As mentioned above, the main objective of G is to synthesize the images that are comparable to the true images, whereas D seeks to discriminate between the real and produced instances of data and employs the back-propagation algorithm for optimizing the weights of the network. The discriminator wants to reduce its failure rate, but the generator wants to maximize it.

The discriminator maximizes the likelihood of correctly labelling instances from training data as well as samples produced by the generator. The image generated as an output by the generator must be given a minimum score for all possible values of z while the samples of x drawn from the real data distribution must receive a maximum score. It penalizes itself for wrongly classifying a true image as a fake image, or a fake image as a true image by maximizing the function given in Equation 1.

$$\max E_{x\sim p_{data}}[log(D(x))] + E_{z\sim p_g(z)}\left[log\left(1 - D(G(z))\right)\right] \quad (1)$$

In contrast, the generator wants the discriminator to give the images generated by it a higher score by producing images that are similar to those in the real data distribution. Mathematically, the objective function of G is to maximize the score $D(G(z))$, or to minimize $\log(1 - D(G(z))$ i.e. (Equation 2).

$$\max\left[log\left(D(G(z))\right)\right] \; or \; \min\left[log\left(1 - D(G(z))\right)\right] \quad (2)$$

So, the overall objective function of GAN is the combination of discriminators and generators loss functions, wherein D maximizes the loss so that the generator cannot fool it while discriminating between the true and fake samples and G minimizes the same loss or maximizes generated samples probability as being real [59]. Depending on the objective function we use for G, we get two objective function variants of the GAN: the min-max GAN (saturating GAN) and the non-saturating GAN [30]. Both of these objective functions were proposed in the original paper by Goodfellow.

3.2Original Min-Max objective function

Here, from the two choices of objective functions of G, the minimizing objective function is chosen which is mathematically given by Equation 3.

$$\min E_{z\sim p_g}\left[log\left(1 - D(G(z))\right)\right] \quad (3)$$

Combining (1) and (3), we get the overall minimax objective function which is applied for training G and D models. This overall min-max objective function is given by Equation 4.

$$min_G max_D\, V(G,D) = min_G max_D\; E_{x\sim p_{data}}[log(D(x))] + E_{z\sim p_g(z)}\left[log\left(1 - D(G(z))\right)\right] \quad (4)$$

This loss function is actually the BCE function [13]. BCE for a single input is given below by Equation 5.

$$L = y\, log\, \acute{y} + (1-y)\, log(1-\acute{y}) \quad (5)$$

where y = ground truth, $\acute{y}$= the value predicted by the mode

When we feed the model with real data that is when y=1 and $\acute{y}$= D(x), then Equation 5 becomes as Equation 6.

$$L = log\big(D(x)\big) \quad (6)$$

When we feed the model with the fake data that is when y=0 and $\acute{y}= D\big(G(z)\big)$ then (5) becomes as Equation 7.

$$L = log\big[1 - D\big(G(z)\big)\big] \quad (7)$$

Adding Equation 6 and Equation 7, we get

$$L = log[D(x)] + log\big[1 - D\big(G(z)\big)\big] \quad (8)$$

Equation 8 is the BCE function for one data point. For the entire dataset we use the expectation which is the average value and thus the BCE function changes as shown in Equation 9.

$$E(L) = \sum_{p_{data}(\mathrm{x})} log[D(x)] + \sum_{p_g(\mathrm{z})} log\big[1 - D\big(G(z)\big)\big] \quad (9)$$

Equation 9 is the BCE function for the discrete distribution of data. For continuous distributions of p_{data} and p_z , BCE is given by Equation 10 as following:

$$\int p_{data}\,(\mathrm{x}) log\ \mathrm{D}(\mathrm{x})\, dx + \int p_g(\mathrm{z})\, log\left(1 - D\big(G(z)\big)\right) dz \quad (10)$$

$$\mathrm{V(G,D)} = E_{\mathrm{x}\sim p_{data}}[log\, D(x)] + E_{\mathrm{z}\sim p_g(\mathrm{z})}\left[log\left(1 - D\big(G(z)\big)\right)\right] \quad (11)$$

So, comparing Equation 4 and Equation 11, it's clear that the GAN's loss function is actually the BCE function.

3.2.1Optimization of Min-Max objective function

While training the GANs, an alternate approach is used for training the generator and the discriminator. When one of them is being trained, the other is kept fixed. In theory, however, the training starts with the discriminator followed by the generator. The main objective is to achieve the Nash equilibrium of the min-max game [31]. Finding the (global) equilibrium, i.e., saddle point of min-max is referred to as optimization of the GANs. During the optimization of GANs, a solution that will maximize D(x) and minimize $1 - D\big(G(z)\big)$ is found. When and only when $p_{data} = p_g$, GAN is said to have reached the global optimal solution.

The objective function of GAN given by (4) can also be written as shown in Equation 12 below:

$$min_{\mathrm{G}} max_{\mathrm{D}}\, V(G,D) = min_{\mathrm{G}} max_{\mathrm{D}}\, \int_x p_{data}\,(\mathrm{x}) log\ \mathrm{D}(\mathrm{x}) dx + \int_z p_g(\mathrm{z})\, log\left(1 - D\big(G(z)\big)\right) dz \quad (12)$$

Since x is a function of z having the same domain, we can replace the second integral of Equation 12 and rewrite it as shown in Equation 13 below:

$$min_{\mathrm{G}} max_{\mathrm{D}}\, V(G,D) = min_{\mathrm{G}} max_{\mathrm{D}}\, \int_x p_{data}\,(\mathrm{x}) log\ \mathrm{D}(\mathrm{x}) dx + \int_x p_g(\mathrm{x})\, log\big(1 - D(x)\big) dx \quad (13)$$

$$min_{\mathrm{G}} max_{\mathrm{D}}\, V(G,D) = min_{\mathrm{G}} max_{\mathrm{D}}\, \int_x [p_{data}\,(\mathrm{x}) log\ \mathrm{D}(\mathrm{x}) + p_g(\mathrm{x})\, log\big(1 - D(x)\big)]\, dx \quad (14)$$

Now let's first see the condition for optimal D. The optimal condition of the D can be acquired by maximizing Equation 14 w.r.t every x when G is fixed. For that, we will take the derivate of the term within integral in Equation 14 w.r.t D(x) and set that to 0 as shown in Equation 15.

$$\frac{d}{d\big(D(x)\big)}\big(p_{data}(\mathrm{x})\, log\ \mathrm{D}(\mathrm{x}) + p_g(\mathrm{x})\, log\big(1 - D(x)\big)\big) = 0 \quad (15)$$

or $$\frac{p_{data}\,(\mathrm{x})}{\mathrm{D}(\mathrm{x})} + \frac{p_g(\mathrm{x})(-1)}{1 - \mathrm{D}(\mathrm{x})} = 0$$

or $$\frac{p_{data}\,(\mathrm{x})}{\mathrm{D}(\mathrm{x})} = \frac{p_g(\mathrm{x})}{1 - \mathrm{D}(\mathrm{x})}$$

or

$$p_{data}\,(\mathrm{x}) - p_{data}\,(\mathrm{x})\, D(x) = p_g(\mathrm{x})\ \mathrm{D}(\mathrm{x})$$

$$\mathrm{D}(\mathrm{x}) = \frac{p_{data}\,(\mathrm{x})}{p_{data}\,(\mathrm{x}) + p_g(\mathrm{x})} \quad (16)$$

Thus the optimal discriminator for any given generator is represented as shown in Equation 16 or Equation 17.

$$D_G^*\big(\mathrm{G}(\mathrm{x})\big) = \frac{p_{data}\,(\mathrm{x})}{p_{data}\,(\mathrm{x}) + p_g(\mathrm{x})} \quad (17)$$

For an optimal generator which produces the generated data as similar as possible to the real data, $p_g = p_{data}$. Hence, when $p_{data}\,(\mathrm{x}) = p_g(\mathrm{x})$, Equation 17 changes to Equation 18 which finally gives a constant value of $\frac{1}{2}$ as shown below:

$$D_G^* = \frac{p_{data}\,(\mathrm{x})}{p_{data}\,(\mathrm{x}) + p_{data}\,(\mathrm{x})} = \frac{1}{2} \quad (18)$$

This means that the discriminator will get confused and won't be able to make the distinction between actual and produced data. Substituting the value $\frac{1}{2}$ in the Equation 19 for training criteria C (G) where C (G) =$\max_D$ V (G, D), we have

$$C(G) = \int_x p_{data}\,(\mathrm{x}) \log \frac{1}{2} + p_g(\mathrm{x}) \log\left(1 - \frac{1}{2}\right) dx \quad (19)$$

$$= -\log 2 \int_x p_{data}\,(\mathrm{x}) dx - log\ 2 \int_x p_g(\mathrm{x})\, dx$$

$$= -2\ log\ 2 = -\ log\ 4 \quad (20)$$

Thus the discriminator's loss at optimal generator i.e., when $p_g = p_{data}$ is –log 4 as shown by Equation 20.

Let's see if this discriminator loss can be ever lower than –log4. For that put $p_g \neq p_{data}$ in the equation for training criteria C (G).Thus we get Equation 21 as under:

$$C(G) = \int_x p_{data}\,(\mathrm{x}) \log \tfrac{1}{2} + p_g(\mathrm{x}) \log\left(1 - \tfrac{1}{2}\right) dx \quad (21)$$

Substituting optimal value of D(x) in Equation 21, we get

$$C(G) = \int_x \left[[p_{data}(x)\ log\left(\frac{p_{data}\,(\mathrm{x})}{p_{data}\,(\mathrm{x}) + p_g\,(\mathrm{x})}\right) + p_g(\mathrm{x})\ log\left(1 - \frac{p_{data}\,(\mathrm{x})}{p_{data}\,(\mathrm{x}) + p_g(\mathrm{x})}\right)\right] dx$$

$$= \int_x \left[p_{data}\,(\mathrm{x}) log\ \left(\frac{p_{data}\,(\mathrm{x})}{p_{data}\,(\mathrm{x}) + p_g\,(\mathrm{x})}\right) + p_g(\mathrm{x})\ log\left(\frac{p_g(\mathrm{x})}{p_{data}\,(\mathrm{x}) + p_g(\mathrm{x})}\right) + (log2 - log2)(p_{data}\,(\mathrm{x}) + p_g\,(\mathrm{x}))\right] dx$$

$$= \int_x \left[p_{data}\,(\mathrm{x}) \left(log\ 2 + log\ \left(\frac{p_{data}\,(\mathrm{x})}{p_{data}\,(\mathrm{x}) + p_g\,(\mathrm{x})}\right)\right) + p_g\,(\mathrm{x}) \left(log\ 2 + log\ \left(\frac{p_g(\mathrm{x})}{p_{data}\,(\mathrm{x}) + p_g(\mathrm{x})}\right)\right)\right] dx + \int_x -log2 \left(p_{data}\,(\mathrm{x}) + p_g\,(\mathrm{x})\right)\, dx$$

$$= \int_x \left[p_{data}\,(\mathrm{x})\ log \left(\frac{\frac{p_{data}\,(\mathrm{x})}{p_{data}\,(\mathrm{x}) + p_g\,(\mathrm{x})}}{2}\right) + p_g\,(\mathrm{x}) log \left(\frac{\frac{p_g\,(\mathrm{x})}{p_{data}\,(\mathrm{x}) + p_g\,(\mathrm{x})}}{2}\right)\right] dx - log2(1+1) \quad (22)$$

The term within the integral in Equation 22 is in the form of KL divergence [60]. KL divergence is the asymmetric measure of how much two distributions differ from one another. It's either equal to zero or greater than that and is defined by Equation 23.

$$KL(\mathrm{P}\|Q) = \int p(x) log \frac{p(x)}{q(x)} \quad (23)$$

Using this KL form, Equation 22 can be rewritten as Equation 24.

$$C(G) = -log4 + KL\left(p_{data}\,\|\,\frac{p_{data} + p_g}{2}\right) + KL\left(p_g\,\|\,\frac{p_{data} + p_g}{2}\right) \quad (24)$$

Since the KL divergence can take values from 0 onwards [61], so the minimum value C(G) can take is –log4. Thus, we can conclude that the global minimum of the training criteria C(G) where C(G) = $\max_D$ V (G, D) is –log4 and can be reached only when $p_g = p_{data}$.

The KL-divergence forms the base for a more promising and more tractable objective function known as the JS-divergence [62]. It's also a measure for determining the difference (or similarity) between two distributions. It has a finite value and is symmetrical. It's also a smoothed and normalized version of KL-divergence with the scores limited by a number between 0 and 1. Its definition is given by Equation 25 as under:

$$JS(\mathrm{P}\|Q) = \tfrac{1}{2} KL(\mathrm{P}\|M) + \tfrac{1}{2} KL(\mathrm{Q}\|M) \quad (25)$$

where $M = \frac{P+Q}{2}$ [63]. In terms of JS-divergence, (24) can be written as Equation 26 below:

$$C(G) = -log4 + 2JS(p_{data}\,\|p_g) \quad (26)$$

This shows that optimizing the generator G simply means to minimize the JS-divergence between p_{data} and p_g, thus making them similar to each other. Early researchers were of the opinion that optimizing JS-divergence rather than KL-divergence, though produced less diverse images but were of better quality [64]. Later on, Arjovsky and Bottou [65] put forth the weaknesses of JS-divergence used in GANs. It was found that distributions of ground truth and fake data are disjoint and there is a high probability that these two distributions have negligible overlap as the parameter dimensions of the generator's input data and the real data lie on two extreme ends. The JS-divergence between such distributions is constant which is equal to $2log2$ in case of GANs. As a result of this, the gradient of discriminator and generator becomes 0 and is referred to as the vanishing gradient. This gradient disappearance makes it difficult to train the model.

3.3Non-saturating objective function

During the initial learning phase, the generator creates samples that are completely dissimilar from the samples of actual data. As a result, it becomes easier for the discriminator to identify the fake samples. The

gradient provided by (4) drops down and becomes increasingly smaller as a result of which the generator stops learning and the data distributions it generates overlap. At this time $\log\left(1 - D\left(G(z)\right)\right)$ saturates and eventually leads to the vanishing gradient problem. To address this problem of vanishing gradient, Gui et al. (2021) recommended maximization of $log\left(D\left(G(z)\right)\right)$ rather than minimization $\log\left(1 - D\left(G(z)\right)\right)$ [66]. This alternate loss function of G is called as non-saturating loss function, due to non-saturating behavior of the gradient [67]. It is also termed as – log D trick. So, instead of Equation 27 and Equation 28 was used.

$$E_{z\sim p_z}(z)[log\left(1 - D\left(G(z)\right)\right)] \quad (27)$$

$$E_{z\sim p_x}(z)[-\log\left(D\left(G(z)\right)\right)]$$

or

$$E_{x\sim p_g}[-log\left(D(x)\right)] \quad (28)$$

Where x=G(z).

Unlike (4), it provides larger gradients during the initial phase of training which is sufficient enough for generator to learn the real distribution of data.

3.3.1Optimization of non-saturating objective function

We know that the optimal discriminator is as follows

$$D^*(x) = \frac{p_{data}\ (x)}{p_{data}\ (x) + p_g(x)}$$

Also KL-divergence is given by Equation 29 as under:

$$KL\left(p_g \| p_{data}\right) = E_{x\sim p_g} log\ \frac{p_g}{p_{data}} \quad (29)$$

$$= E_{x\sim p_g} log\ \frac{\frac{p_g(x)}{p_{data}\ (x)+p_g(x)}}{\frac{p_{data}(x)}{p_{data}\ (x)+p_g(x)}}$$

$$= E_{x\sim p_g} log\ \frac{1- D^*(x)}{D^*(x)}$$

or

$$KL\left(p_g \| p_{data}\right) = E_{x\sim p_g}\left[log\left(1 - D^*(x)\right)\right] - E_{x\sim p_g}\left[log\left(D^*(x)\right)\right]$$

or

$$-E_{x\sim p_g}\left[log\left(D^*(x)\right)\right] = KL\left(p_g \| p_{data}\right) - E_{x\sim p_g}\left[log\left(1 - D^*(x)\right)\right] \quad (30)$$

Also, from Equation 21 and Equation 26, we get

$$E_{x\sim p_{data}}\left[log\left(D^*(x)\right)\right] + E_{x\sim p_g}\left[log\left(1 - D^*(x)\right)\right] = 2JS\left(p_{data} \| p_g\right) - log4 \quad (31)$$

$$E_{x\sim p_g}\left[log\left(1 - D^*(x)\right)\right] = 2JS\left(p_{data} \| p_g\right) - log4 - E_{x\sim p_{data}}\left[log\left(D^*(x)\right)\right] \quad (32)$$

By substituting (32) in (30), we get

$$-E_{x\sim p_g}\left[log\left(D^*(x)\right)\right] = KL\left(p_g \| p_{data}\right) - \left[2JS\left(p_{data} \| p_g\right) - log4 - E_{x\sim p_{data}}\left[log\left(D^*(x)\right)\right]\right] \quad (33)$$

or

$$E_{x\sim p_g}\left[-log\left(D^*(x)\right)\right] = KL\left(p_g \| p_{data}\right) - 2JS\left(p_{data} \| p_g\right) + log4 + E_{x\sim p_{data}}\left[log\left(D^*(x)\right)\right] \quad (34)$$

Since the final two terms of Equation 34 do not depend on G, so they won't affect the loss for G. From Equation 34, it is clear that optimization of non-saturating loss of G is contradictory where the KL-measure seeks to minimize the difference between the true and generated data distributions as much as possible whereas the JS-measure attempts to increase the same. As a result of this contradiction, the gradient for training generator is unstable. Also, the first term here is reverse KL-divergence. Since KL-divergence is asymmetrical measure, we get a value of KL-divergence that is different from the value of reverse KL-divergence as shown below:

$$If\ \ p_{data}(x) \to 1\ and\ p_g(x) \to 0\ , then\ KL\left(p_g \| p_{data}\right) \to 0$$

$$If\ p_{data}(x) \to 0\ and\ p_g(x) \to 1\ , then\ KL\left(p_g \| p_{data}\right) \to +\infty$$

The first case results in a small range of samples and hence lacks diversity whereas the second case produces implausible and inaccurate samples. Thus, there is a tradeoff between generating diverse images and generating accurate images. In the first case, though samples of less diversity are generated, they are safe samples while in the second case competing to get diverse images leads to unsafe samples. In such a case, the generator cannot find a balance between real and generated samples and hence causes a mode collapse problem. Moreover in Equation 34, since there is reverse KL-divergence, so optimization on it will result in a mode collapse problem during training GANs.

In summary, the original min-max objective function result in the vanishing gradient problem whereas using the alternative loss i.e., non-saturating objective function though removes vanishing gradient problem but incurs mode collapse problem. To handle the problems with GANs, a lot of work has been done. The two main directions include changing the network architectures and designing new loss functions [68]. It

has been found that GANs problem ultimately arises from the design of the loss function since different loss function GANs variants have been successful in improving the training of GANs compared to the architectural GANs. For the stability of training GANs, different loss variants of GANs have been proposed. The mathematical intuition behind them is discussed in the following subsection.

3.4Mathematical intuition behind loss function GAN variants

3.4.1Vanilla GAN

The original GAN discussed above is referred to as the vanilla GAN. Its generator and the discriminator consist of a simple multi-layer perceptron [69]. Vanilla-GAN is a simple algorithm that uses stochastic gradient descent (SGD) to optimize mathematical equations [70]. It can be of two kinds depending on the loss function it uses for the generator: minimax GAN and non-saturating GAN. This Vanilla GAN suffers the following shortcomings which need to be addressed: (a) non-convergence (b) mode-collapse (c) vanishing gradient (d) overfitting (e) high sensitiveness [71]. A substantial quantity of research has been done, modifying the vanilla GAN to impart stability, drawing on a growing body of empirical and theoretical insights. Radford et al. [39], introduced architectural changes to vanilla GAN and set other parameters of learning rate which helped stabilize training significantly. This is referred to as deep convolutional GANs (DCGANs). Though it is an architectural variant of GAN, the reason for its inclusion in this paper is that the DCGAN architecture is used in the majority of GANs nowadays and all the loss variants discussed and implemented in this paper use the DCGAN architecture. So for that reason, this variant has been discussed as well as implemented here.

3.4.2Deep convolutional generative adversarial networks (DCGANs)

DCGAN stands for deep convolution GAN and primarily uses the convolutional layers. It is the first to use a deconvolutional neural networks architecture for the generator G. The architectural changes in DCGAN are as under:

1. The structure of DCGAN is borrowed from a convolutional network. Strided convolutions in G and fractional-strided convolutions in D replace pooling layers.
2. Except the generator's final layer and discriminator's first layer, it uses batch normalisation in the rest of the layers of the generator and discriminator.
3. Except for last layer of the generator that uses the Hyperbolic Tangent Function (Tanh), all other layers use Rectified Linear Unit (ReLU) as the activation function. In all layers of the discriminator, leaky ReLU is used.
4. It uses the Adam optimizer with momentum replacing SGD.

3.4.3Wasserstein GAN (WGAN)

To deal with the issues of vanishing gradient, mode collapse and non-convergence in the training of GAN, [40] presented the WGAN. Rather than employing JS-divergence for measuring distance between true data distribution $p_{data}(x)$ and synthesized data distribution $p_g(x)$ in vanilla GAN, WGAN uses Wasserstein distance. This modification is one of the most important developments in the topic since the inception of GAN [71]. Wasserstein Distance is a metric for comparing the distance between two different probability distributions. Another name for this distance is Earth Mover's distance or EM distance for short. It is so named because it can be read informally as the cost of shifting and altering a pile of dirt from one probability distribution's shape to the shape of another probability distribution. This distance has been found to effectively mitigate the common failure modes in the training of GANs. For the continuous probability domain, the distance formula introduced in [72] is given by Equation 35 as under:

$$W(p_{data}, p_g) = inf_{\gamma \sim \prod(p_{data}, p_g)} E_{(x,y)\sim\gamma}[\|x - y\|] \quad (35)$$

$\prod(p_{data}, p_g)$ a collection of all feasible joint probability distributions between p_{data} and p_g. Infimum, which is also known as the greatest lower bound, implies that our aim is to find out the lowest cost. Unlike KL-divergence and JS-divergence which give infinity when p_{data} and p_g do not overlap, EM provides a meaningful gradient when the two distributions are disjoint. It can possibly alleviate the problem of disappearing gradients because of its higher smoothing abilities compared to KL-divergence and JS-divergence.

Equation 35 cannot be evaluated using only samples from the distributions because there is no explicit expression for the target distribution. It is not possible to exhaust all the joint distributions that can exist in $\prod(p_{data}, p_g)$ to compute $inf_{\gamma \sim \prod(p_{data}, p_g)}$. [40] suggested a sophisticated Kantorovich-Rubinstein duality to transform to a dual problem which is tractable and is given as under by Equation 36:

$$W(p_{data}, p_g) = \frac{1}{K} \sup_{\|f\|_L \leq K} E_{x\sim p_{data}} f(x) - E_{x\sim p_g} f(x) \quad (36)$$

Here, sup is used for supremum. It is the opposite of inf that represents infimum. Here, the aim is to measure maximum value or to put it in another way, the least upper bound. The function f in the modified form of the Wasserstein metric is required to satisfy$\|f\|_L \leq K$, which means that it should be K-Lipschitz continuous i.e. Equation 37 should be satisfied [73].

$$\frac{\|f(x_1)-f(x_2)\|}{\|x_1-x_2\|} \leq K \quad (37)$$

where K is called the Lipschitz constant. Assume that the function f belongs to the K-Lipschitz family of continuous functions, $\{f_w\}_{w\in W}$ parameterized by w where w is the parameters in "discriminator" D. The model D learns w in order to find a good w and makes the optimization distance equivalent to Wasserstein distance as under.

$$W(p_{data}, p_g) = \max_{w\in W} E_{x\sim p_{data}}[f_w(x)] - E_{z\sim p_z} p(z)[f_w(G(z))] \quad (38)$$

Then G seeks to minimise Equation 38 in order to have the two distribution that is true data and synthesized data distribution as much similar to each other as possible. As a result, WGAN's overall objective function [72] is as shown in Equation 39 and 40.

$$\min_G \max_{w\in W} E_{x\sim p_{data}}[f_w(x)] - E_{z\sim p_z} p(z)[f_w(G(z))] \quad (39)$$

or

$$= \min_G \max_D E_{x\sim p_{data}}[D(x)] - E_{z\sim p_z} p(z)[D(G(z))] \quad (40)$$

Thus the "discriminator" of Vanilla GAN and the WGAN are different from each other. The vanilla GAN discriminator is a binary classifier that simply distinguishes between false and real samples, whereas the discriminator of WGAN, has to calculate the Wasserstein distance which it does by learning the K-Lipschitz continuous function during training. The WGAN removes the sigmoid in the final layer of Vanilla D. The training of the WGAN is also unstable when using a momentum-based optimizer like the Adam optimizer, so the Adam optimizer is replaced with the RMSProp optimizer in WGAN. To preserve the Lipschitz continuity introduced to deal with the intractability of the infimum term, it uses a technique called as weight clipping wherein the weights "w" are clipped so that they fall in a small fixed window [-0.01, 0.01]. This forms a compact parameter space w. Besides, dealing with the instability issues of training, WGAN also presents a meaningful indicator of the training process in relation to the quality of the samples being generated. However, the Wasserstein loss is inferior to the original loss in quality of fake data in practice and also the gradient regularization depends on the model that varies during the training [74]. It is also heavily dependent on the hyper-parameter setting [75]. Moreover maintaining the constraint of K-Lipschitz continuity during the training is one of the most difficult tasks.

3.4.4Wasserstein GAN-gradient penalty (WGAN-GP)

It has been established that Wasserstein distance enhances training stability and convergence, especially when working with distributions that support low-dimensional manifolds. However, weight clipping is not an ideal method for enforcing a Lipschitz requirement. A significant drawback of weight clipping is that it incurs pathological behaviour that leads to sluggish convergence and also to incomplete stable training. To circumvent these problems of WGAN, [72] have replaced the weight clipping in the loss function with a gradient penalty. In other words, WGAN uses gradient penalty for constraining$\|f\|_L \leq K$, for the discriminator to increase the network's convergence speed and stability. The modified loss for the discriminator now becomes Equation 41.

$$L = -E_{x\sim p_{data}}[D(x)] + E_{\tilde{x}\sim p_g}[D(\tilde{x})] + \lambda E_{\hat{x}\sim p_{\hat{x}}}[(\|\nabla_{\hat{x}} D(\hat{x}\|_2 - 1)^2] \quad (41)$$

The initial two terms in Equation 41 represent the objective function of WGAN. In the penultimate term, $\hat{x}$ is sampled from $p_{\hat{x}}$ which samples evenly in a straight direction between pairs of points that are taken from the true distribution p_{data} and the generated distribution of data p_g. WGAN-GP creates generally good images and avoids mode collapse to a large extent, and it is simple to extend this training framework to various other GAN models. It detects overfitting in discriminator instead of generator and measures overfitting against same loss that the network minimizes. In many GAN frameworks, the WGAN-GP performs better than the original WGAN with essentially little hyper-parameter adjustment.

3.4.5Least squares GAN (LSGAN)

Least squares GANs (LSGANs) were presented to deal with the issue of vanishing gradient in original GANs [76]. The discriminator D in the original GAN uses sigmoid cross entropy loss to classify the real or true samples and the synthesized samples and if D misclassifies the synthesized sample as a real sample

then the generator G will stop improving and will continue to generate similar samples which in a real sense might not be close to the true data. While updating the generator, it causes no loss for the samples which though, present on the right side of the boundary of decision, are still different than true data points. Contrary to that, LSGAN will penalize even those samples despite being correctly classified. To create more gradients to update G, LSGAN penalizes the samples based on their distances from the decision boundary. This penalty will push generated samples closer to the decision border, bringing them closer to the real data, even if they are correctly classified This is beneficial for obtaining data samples of better quality. Also, penalizing samples according to their distances from the boundary decision, helps it to produce additional gradients and addresses the vanishing gradient issue, thus boosting the stability of the learning process. Assume that the LSGANs discriminator D employs an "a-b" coding scheme [77], with label 'a' for the generated sample and 'b' for the real sample. The discriminator loss and generators loss of LSGAN [77] are then calculated as shown under by Equation 42.

$$\min_D V_{LSGAN}(D) = E_{\mathrm{x}\sim p_{data}}[(D(x)-b)^2] + E_{\mathrm{z}\sim p_z}\left[\left(D\left(G(z)\right)-a\right)^2\right]$$

$$\min_G V_{LSGAN}(G) = E_{\mathrm{z}\sim p_z}\left[\left(D\left(G(z)\right)-c\right)^2\right] \quad (42)$$

Here, c is the hyper parameter that G would like D to trust for created samples. Optimization of LSGAN has been proved to be comparable to minimizing the Pearson $\chi 2$ divergence. Let us have a deeper look into that. Let us take an extended version of Equation 43 as follows:

$$\min_D V_{LSGAN}(D) = \frac{1}{2}E_{\mathrm{x}\sim p_{data}}(x)[(D(x)-b)^2] + \frac{1}{2}E_{\mathrm{z}\sim p_z}\left[\left(D\left(G(z)\right)-a\right)^2\right]$$

$$\min_G V_{LSGAN}(G) = \frac{1}{2}E_{\mathrm{x}\sim p_{data}}(x)[(D(x)-c)^2] + \frac{1}{2}E_{\mathrm{z}\sim p_z}\left[\left(D\left(G(z)\right)-c\right)^2\right] \quad (43)$$

Here, in Equation 43, we added term $\frac{1}{2}E_{\mathrm{x}\sim p_{data}}(x)[(D(x)-c)^2]$ to the objective function of LSGANs generator. Since it does not contain the parameters of G, it won't affect the Gs' objective function. For a fixed G, the optimal D [77] is given in Equation 44 below:

$$D^*(\mathrm{x}) = \frac{bp_{data}(\mathrm{x}) + ap_g(\mathrm{x})}{p_{data}(\mathrm{x}) + p_g(\mathrm{x})} \quad (44)$$

Substituting the value in the objective function of G, we get the Equation 4 and 46.

$$2\mathrm{C}(G) = E_{\mathrm{x}\sim p_{data}}[(D^*(\mathrm{x})-c)^2] + E_{\mathrm{x}\sim p_g}[(D^*(\mathrm{x})-c)^2] \quad (45)$$

$$= E_{\mathrm{x}\sim p_{data}}\left[\left(\frac{bp_{data}(\mathrm{x})+ap_g(\mathrm{x})}{p_{data}(\mathrm{x})+p_g(\mathrm{x})}-c\right)^2\right] + E_{\mathrm{x}\sim p_g}\left[\left(\frac{bp_{data}(\mathrm{x})+ap_g(\mathrm{x})}{p_{data}(\mathrm{x})+p_g(\mathrm{x})}-c\right)^2\right] \quad (46)$$

$$= \int_x p_{data}(x)\left(\frac{(b-c)p_{data}(\mathrm{x})+(a-c)p_g(\mathrm{x})}{p_{data}(\mathrm{x})+p_g(\mathrm{x})}\right)^2 dx + \int_x p_g(x)\left(\frac{(b-c)p_{data}(\mathrm{x})+(a-c)p_g(\mathrm{x})}{p_{data}(\mathrm{x})+p_g(\mathrm{x})}\right)^2 dx$$

$$= \int_x \frac{\left(p_{data}(\mathrm{x})+p_g(\mathrm{x})\right)\left(\begin{matrix}(b-c)p_{data}(\mathrm{x})\\+(a-c)p_g(\mathrm{x})\end{matrix}\right)^2}{\left(p_{data}(\mathrm{x})+p_g(\mathrm{x})\right)^2} dx$$

$$= \int_x \frac{\left((b-c)p_{data}(\mathrm{x})+(a-c)p_g(\mathrm{x})\right)^2}{p_{data}(\mathrm{x})+p_g(\mathrm{x})} dx$$

$$= \int_x \frac{\left(\begin{matrix}(b-c)\left(p_{data}(\mathrm{x})+p_g(\mathrm{x})\right)\\-(b-a)p_g(\mathrm{x})\end{matrix}\right)^2}{p_{data}(\mathrm{x})+p_g(\mathrm{x})} dx \quad (47)$$

Setting $(b-c)=1$ and $(b-a)=2$ in Equation 47, we get the Equation 48.

$$2\mathrm{C}(G) = \int_x \frac{\left(2p_g(\mathrm{x})-\left(p_{data}(\mathrm{x})+p_g(\mathrm{x})\right)\right)^2}{p_{data}(\mathrm{x})+p_g(\mathrm{x})} dx \quad (48)$$

$$= \chi^2_{Pearson}\left(p_{data}+p_g \| 2p_g\right)$$

where $\chi^2_{Pearson}$ is the Pearson divergence between $p_{data}+p_g$ and $2p_g$. Thus, Equation 49 shows that optimizing the generator G simply means to minimize the Pearson divergence between $p_{data}+p_g$ and $2p_g$ when (b – c) = 1 and (b – a) = 2. As a result, LSGANs outperform GANs in terms of training stability and resultant image quality [78].With real datasets, however, LSGANs are unable to achieve good results in producing diverse images since they do not expand the mode range or the semantic diversity of generated data samples. *Table 1* summarizes the loss functions of the discriminators and generators of each variant of the loss variants of GAN.

Table 1 Summary of loss functions of loss variants of GAN

Types of GAN	Loss function of discriminator	Loss Function of Generator
Min-max vanilla GAN	$L_{\mathrm{D}} = E_{\mathrm{x}\sim p_{data}}[\log \mathrm{D}(\mathrm{x})] + E_{\mathrm{z}\sim p_g(\mathrm{z})}\left[\log\left(1 - D\left(G(z)\right)\right)\right]$	$L_{\mathrm{G}} = E_{\mathrm{z}\sim p_g}\left[log\left(1 - D\left(G(z)\right)\right)\right]$
Non-saturating vanilla GAN	$L_{\mathrm{D}} = E_{\mathrm{x}\sim p_{data}}[\log \mathrm{D}(\mathrm{x})] + E_{\mathrm{z}\sim p_g(\mathrm{z})}\left[\log\left(1 - D\left(G(z)\right)\right)\right]$	$L_{\mathrm{G}} = E_{\mathrm{z}\sim p_x}(z)\left[-log\left(D\left(G(z)\right)\right)\right]$
WGAN	$L_{\mathrm{D}} = E_{\mathrm{x}\sim p_{data}}[\mathrm{D}(\mathrm{x})] + E_{\mathrm{z}\sim p_g(\mathrm{z})}\left[\left(1 - D\left(G(z)\right)\right)\right]$	$L_{\mathrm{G}} = E_{\mathrm{z}\sim p_g}\left[\left(-D\left(G(z)\right)\right)\right]$
GP-WGAN	$L_{\mathrm{D}} = L_D^{WGAN} + \lambda\ E_{\mathrm{x}\sim p_{data}}\left[\left(\left\|\nabla D\left(\alpha x + (1-\alpha G(z))\right)\right\|_2 - 1\right)^2\right]$	$L_{\mathrm{G}} = E_{\mathrm{z}\sim p_g}\left[\left(-D\left(G(z)\right)\right)\right]$
LSGAN	$L_{\mathrm{D}} = E_{\mathrm{x}\sim p_{data}}[(D(x)-b)^2] + E_{\mathrm{z}\sim p_z}\left[\left(D\left(G(z)\right)-a\right)^2\right]$	$L_{\mathrm{G}} = E_{\mathrm{z}\sim p_z}\left[\left(D\left(G(z)\right)-c\right)^2\right]$

4.Results

4.1Datasets used

We run experiments on the MNIST dataset, the Fashion-MNIST dataset, and the CIFAR-10 dataset to assess the effectiveness of the loss functions used in the aforementioned GANs versions. Brief description of these datasets is provided below, and examples of training samples from these datasets are displayed in *Figure 2*.

MNIST

MNIST dataset is made up of handwritten digits and the tags that go with them. The dataset is separated into two parts: the training part that consists of 60,000 samples, and the test part that has 10,000 cases. It can be found at http://yann.lecun.com/exdb/mnist/. This dataset was chosen because of its simplicity and size, which allows researchers to test and develop deep learning methods quickly. In addition, most machine learning libraries and deep learning frameworks have functions for loading and preparing the dataset.

CIFAR-10

CIFAR-10 dataset comprises of 60,000 coloured images from 10 classes. Each class contains 6000 images. The images are of 32×32 pixels and belong to classes of frogs, ships, airplanes, etc. The dataset is separated into two parts: the training part that consists of 50,000 samples, and the test part that has 10,000 cases. It can be found at https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz

Fashion-MNIST

In terms of image size and the design of the training and testing portions, the Fashion-MNIST dataset is comparable to the MNIST dataset. There are 10,000 test samples in the test component and 60,000 training samples in the training set. The size of the images in this dataset is also 28×28 and are grayscale images. It can be found at http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-images-idx3-ubyte.gz.

Figure 2 Training samples of different datasets

4.2Model architectures

The details about the architecture of the generator and the discriminator models used by the variants are given below. The optimizers, activation functions, and loss functions that are specific to each variant are also mentioned.

Vanilla GAN: Optimizer applied was Adam optimizer. Both the discriminator and generator networks had four layers. All of the layers in discriminator use Leaky ReLU as activation function excluding the final layer that uses sigmoid. In all layers of generator, Leaky ReLU activation function is employed excluding the final layer, which employs the Tangent Hyperbolic activation function. A dropout was also used. Finally, the BCE is used as a loss function.

DCGAN: Optimizer applied was Adam optimizer. Both discriminator and generator networks had five layers. In the discriminator, Leaky ReLU is used as an activation function in all layers excluding the final layer that uses the sigmoid function. In the generator, excluding the last layer that uses the Tangent Hyperbolic activation function, all other layers use ReLU as the activation function. It employs the BCE function as its loss function.

WGAN: RMSProp optimizer was used as an optimizer. Five layers constitute the discriminator and generator. In every layer in the discriminator excluding the output layer, the activation function used is Leaky ReLU, whereas in every layer in the generator ReLU is used as an activation function except the final layer, which uses a Tangent Hyperbolic. As a loss function, a Wasserstein distance was used.

WGAN-GP: Optimizer applied was RMSProp optimizer. The discriminator and the generator consist of five layers. A leaky ReLU activation function is used in every layer in discriminator except the output layer which does not use any activation function, whereas the ReLU activation function is used in every layer in generator except the output layer that makes use of the Tangent Hyperbolic activation function. Moreover, it replaces weight clipping by the gradient penalty.

LSGAN: It uses an Adam optimizer. The discriminator and the generator both have five layers each. Activation function used by every layer in the discriminator is the Leaky ReLU, with the exception of the last layer, which has no activation function. In addition, except for the last layer, which employs Tangent Hyperbolic activation, the generator uses the ReLU activation function for all layers. As a loss function, Mean Square Error is adopted.

4.3Parameters and experimental setup

The images of all three datasets are resized to 64×64. On all three datasets, all parameters for each variant are set the same. In every experiment, the batch size is fixed to 128 for all datasets. Latent vector's dimension is set at to100. The variants were trained for 100 epochs on all three datasets. The learning rate is chosen to be 0.0002 and momentum is fixed to 0.5. All the variants were implemented using Python language, and the PyTorch framework. The models were trained online in the Google Colab environment with the GPU enabled.

4.4Evaluation

4.4.1Qualitative evaluation

The synthetic images of the MNIST dataset generated by the above-mentioned GAN variants are displayed in *Figure 3*. First column shows images synthesized by different GANs after the first epoch. Vanilla GAN and LSGAN generate nothing but the noise in the first epoch whereas the rest of the GANs generate some patterns after the first epoch. From the last column, which shows the results generated after 60 epochs, it is clear that the digits generated by LSGAN are more clear and more vivid than the rest of the GANs. The output of the Vanilla GAN even after 60 epochs is not realistic. On the other hand, DCGAN produces better samples than WGAN and WGAN-GP whereas WGAN-GP produces better samples than WGAN. Moreover, the vanilla GAN suffered from vanishing gradient wherein the generator prematurely converges to an unrealistic output. In comparison to Vanilla GAN, DCGAN has been found to be more stable, easy to train, and, most importantly, provides high-quality outputs. However, it still faced the problems of mode collapse and non-convergence.

Figure 4 and *Figure 5* show examples of mode collapse and non-convergence of DCGAN when trained on the MNIST dataset. *Figure 4* shows that DCGAN has generated different kinds of digits in epoch 63. In other words, it has generated all the 10 modes of the MNIST dataset in this epoch. In comparison to this, after epochs 70, 74, and 75, DCGAN has generated just one mode in each epoch. This exhibits that there was a modal collapse of DCGAN after epoch 63. Also, *Figure 5* shows that after epoch 63, better results have been obtained. In other words, it means that the generator has learned the distribution and it can be thought of as the model is converging. But the results produced after the 63rd epoch are of extremely bad quality which means that the model has failed to converge.

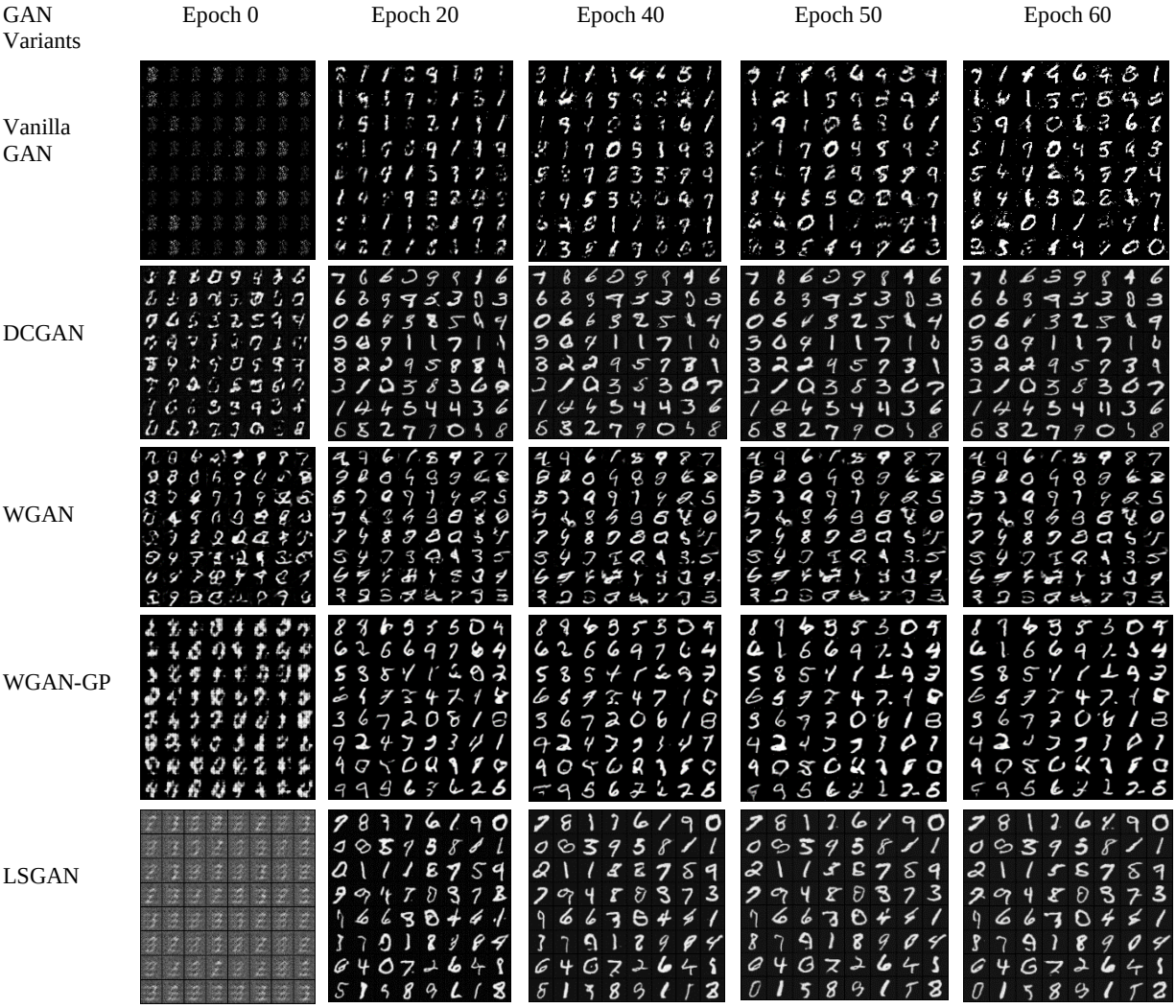

Figure 3 On the MNIST dataset, digits generated by different loss variants of GAN at different epochs

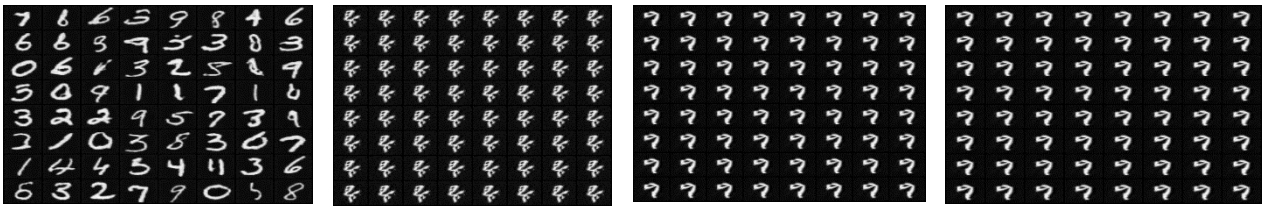

Figure 4 Example of the mode collapse problem in DCGANs on the MNIST dataset

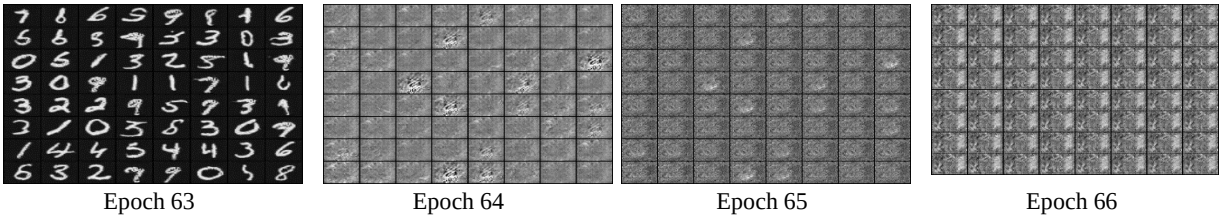

Figure 5 Example of the non-convergence problem in DCGANs on the MNIST dataset

The synthetic images of CIFAR-10 and Fashion-MNIST generated by different loss function GAN variants while experimentation is presented in *Figure 6* and *Figure 7* respectively. It was found that in the case of CIFAR-10 dataset, DCGAN produces better results than LSGAN and other variants. However, DCGAN still suffered from mode collapse and non-convergence problems. Also, though LSGAN was found to generate better results than Vanilla GAN, WGAN, and WGAN-GP, it also suffered from mode collapse. On Fashion-MNIST, better images were produced by LSGAN followed by DCGAN. The samples produced by Vanilla GAN after the 20th epoch are still of very poor quality compared to the other GAN variants.

Figure 6 Images generated at the end of the 50-th epoch by Vanilla GAN, DCGAN, WGAN, WGAN-GP, and LSGAN on the CIFAR-10 dataset

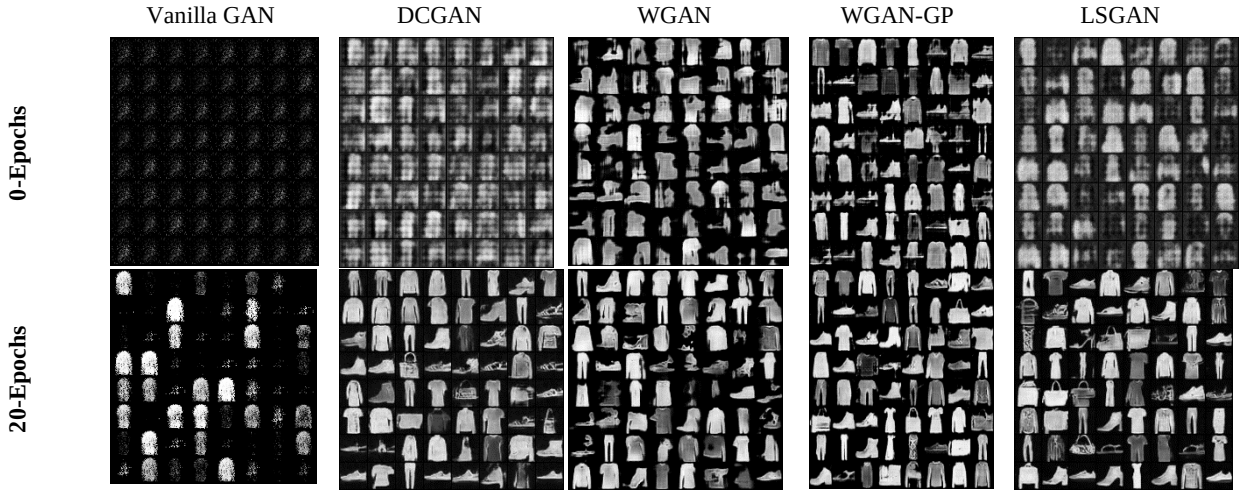

Figure 7 On the Fashion-MNIST dataset, images generated by different loss variants of GAN at 0-th and 20-th epochs

The training losses of generator and discriminator for each variant implemented are shown in *Figure 8*. These can help to visualize the learning process. In the initial epochs, the discriminator's efficiency is low i.e. the error is high since it does not know how to classify the images correctly as actual or generated. As the discriminator starts to learn and becomes better, its error starts decreasing. Since the discriminator and generator are in a competitive relationship, an improvement on one means a higher loss on the other. Thus, with the decrease in discriminators error, the generators error increases. As the training continues and time passes, the generator error decreases meaning that the images it generates are better and better. With the generator improving and generating more and more realistic images, the discriminator's error increases and it misclassifies actual data instances and synthesized data instances. In the plots given below, discriminator loss is discriminator's loss value, generator loss is generator's loss value, and x-coordinate gives the number of epochs used for training. In vanilla GAN, the generator's loss initially increases to a higher peak and then drops significantly followed by an increase again and then a continual decrease in the loss. Discriminator's loss also decreases in initial phase and then begins to rise, implying that the model is progressing to convergence. The loss curve of DCGAN shows that there is no sign of convergence. The generator's loss initially drops and then keeps on oscillating and does not converge. Compared to the DCGAN loss curve, the WGAN loss curve shows that it converges and that too faster than Vanilla GAN. In case of LSGAN, In the case of LSGAN, training is largely steady throughout the run, with some initially noted very high peaks.

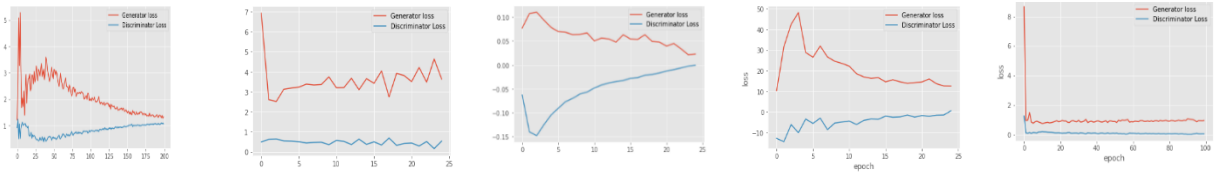

Figure 8 On the MNIST dataset, discriminator and generator losses of different loss variants of GAN

4.4.2Quantitative evaluation

For each of the three datasets considered, we measured visual quality of the images produced by the aforementioned GAN variations using two quantitative criteria. Inception score (IS) and a Frechet inception distance (FID) are two of these measurements. IS is frequently used to assess the produced samples since it has a strong correlation with human annotator quality judgments. Better results are indicated by higher values of IS. FID approximates the inception's feature space for the calculation of separation between the actual and fake samples. Better-quality samples are indicated by lower FIDs, and worse-quality images are indicated by higher FIDs. *Table 2* presents the IS and FID scores.

Table 2 IS and FID achieved by different GAN variants on MNIST, Fashion-MNIST and CIFAR-10 Datasets

GAN-Variants	Inception score			Frechet inception distance		
	MNIST	Fashion-MNIST	CIFAR-10	MNIST	Fashion-MNIST	CIFAR-10
Vanilla GAN	2.98± 0.02	2.9± 0.14	4.61 ± 0.02	20.33	49.54	87.2
DCGAN	8. 73± 0.09	5.83± 0.10	6. 802± 0.11	6.93	26.81	38.1
WGAN	5.62± 0.02	5.37± 0.07	5.751± 1.4	12.94	24.05	52.8
WGAN-GP	8.10 ± 0.12	6.11± 0.10	6.631± 0.09	7.81	24.62	37.4
LSGAN	9.22± 0.06	6.35± 0.12	6.582± 0.10	6.56	25.32	43.6

4.4.3Training time

The training time taken by the models is tabulated below in *Table 3*. WGAN was found to be extremely slow compared to the other GANS. GP-WGAN, an upgraded WGAN, was found to be quicker than WGAN. However, when compared to both WGAN and WGAN-GP, DCGAN and LSGAN took lesser time.

Table 3 Time taken per epoch and Total time taken by loss variants of GAN when trained on MNIST, Fashion-MNIST and CIFAR-10 datasets

GAN-Variants	Time taken/Epoch			Total Time Taken/100 epochs		
	MNIST	Fashion-MNIST	CIFAR-10	MNIST	Fashion-MNIST	CIFAR-10
Vanilla GAN	0.08s	0.11s	0.11s	8.60s	11.07s	12.02s
DCGAN	1.39s	1.44s	1.27s	138.52s	144.08s	127.00s
WGAN	1.35s	4.50s	4.04s	135.00s	450.19s	403.54s
WGAN-GP	2.24s	2.27s	2.08s	223.89s	227.09s	208.32s
LSGAN	1.45s	1.40s	1.36s	145.55s	136.02s	136.46s

5.Discussion

5.1Summary of the key findings

In this paper, we implemented 5 different loss function versions of GAN for generating images from noise input. These models have been tested on three alternative datasets including MNIST, Fashion-MNIST, and CIFAR-10 datasets. Below is the summary of the findings from the results:

a. From *Figure 3*, it is clear that the samples generated by LSGAN are more clear and realistic than other variants on the MNIST dataset. Among DCGAN, WGAN, and WGAN-GP, DCGAN produces better results. On the CIFAR-10 dataset, DCGAN produces better visual results than the rest of the variants. On Fashion-MNIST, better results were obtained by LSGAN followed by DCGAN.

b. Though DCGAN produced better results on CIFAR-10 dataset than rest of the GANs and produced better results than vanilla GAN, WGAN as well as its modified version WGAN-GP on MNIST and Fashion-MNIST datasets, it suffered from mode collapse and non-convergence on all the three datasets. The examples of these issues on MNIST dataset are given in *Figure 4* and *Figure 5*. Also, LSGAN suffered from the mode collapse issue on the CIFAR-10 dataset. On the other hand, there was no sign of mode collapse and other issues when implementing WGAN and WGAN-GP. Moreover, *Figure 8* shows that WGAN-GP converges faster than WGAN and both of them converge faster than vanilla GAN and DCGAN. LSGAN also gets stable early and continues to remain so.

c. In terms of quantitative metrics, on the MNIST dataset, LSGAN gives the higher IS of 9.22± 0.06 followed by DCGAN whose inception score is 8. 73± 0.09. WGAN-GP has a higher IS than WGAN and the lowest IS is shown by vanilla GAN. On the CIFAR-10 dataset, the higher IS is achieved by DCGAN. Moreover, WGAN-GP shows better IS on this dataset compared to LSGAN. On the Fashion-MNIST dataset, LSGAN has the highest IS. Also, WGAN-GP shows better IS than DCGAN on this dataset. On the Fashion-MNIST, better scores of FID are shown by WGAN followed by WGAN-GP. On the CIFAR-10 dataset, better FID scores are shown by WGAN-GP followed by DCGAN. On the MNIST dataset, LSGAN shows lower FID scores than the rest of the variants.

d. In terms of training time, almost all the variants take more time when trained on Fashion-MNIST and CIFAR-10 datasets compared to when trained on the MNIST dataset. Among the GAN variants, WGAN is extremely slow to be trained and takes

more time per epoch than rest of the variants. WGAN-GP is faster than WGAN but slower than the rest of the variants. Vanilla GAN takes lesser time per epoch on all the datasets.

Furthermore, the summary and strengths of above-mentioned GAN variants are tabulated below in *Table 4*.

Table 4 Summary of strengths and weaknesses of loss variants of GAN

Loss function GAN variants	Strengths	Weaknesses
Vanilla GAN	• Faster generation of the samples. • Easy to implement	• Vanishing gradient problem • Results generated are of lower quality
DCGAN	• Create samples with a high likelihood of being real.	• Incurs mode collapse problem. • Incurs non-convergence problem
WGAN	• Solves the issue of vanishing gradient. • Solves mode collapse problem • Converges faster than DCGAN	• It does not clearly perform significantly better than the DCGAN in terms of results generated. • It uses weight clipping which limits the ability of the models to learn complex distributions. • Selecting the weight clipping is very sensitive because the small one might cause vanishing gradient and the large one may cause the slow convergence.
WGAN-GP	• Faster convergence rate than WGAN. • During training, it is more stable	• Batch normalisation is not feasible since each sample in the batch is subjected to gradient penalization. • Results generated though are better than WGAN but not better than DCGAN
LSGAN	• Solves the vanishing gradient problem and improves GAN training stability. • Increases the model's mode diversity. • It's simple and easy to implement.	• Instead of real data, produced samples are pushed to the decision boundary, which may impair the quality of the output image. • Shows some sign of mode collapse on complex datasets like CIFAR-10

From the experimental evaluation, it is clear that loss function selection in GANs significantly impacts the performance. DCGAN, on one hand, produces better quality samples but suffers from mode collapse and vanishing gradient. WGAN and WGAN-GP, though, solve the problems of instability and mode collapse but are not successful in generating quality samples. Moreover, setting the parameters in WGAN was very cumbersome since it requires setting up of lot of parameters. The performance of LSGAN was found to be better compared to the other algorithms as it undergoes minimal mode collapse and deals with vanishing gradient and non-convergence issues of GANs.

5.2Limitation of the study

a. In this study, we focussed more on analysing and studying the effect of loss functions in dealing with the challenges faced by traditional GAN. We used the DCGAN-based architecture in all the variants. So, in the future different architectural variants can be also studied that mitigate the issues faced by GAN.
b. We considered classic datasets in our study which have images of low resolution. In the future, these variants can be implemented with complex datasets having higher resolution images.
c. Moreover, in the future, the effect of various parameters and other attributes on the generation of results can be also studied.
d. Also, here we have studied only the basic loss function GAN variants, so in the future, different application-oriented loss variants of GAN can be studied in detail along with their mathematical intuition, other underlying theory, and their implementation.

A complete list of abbreviations is shown in *Appendix I.*

6.Conclusion and future work

GAN, a class of DGM has attracted the interest of the research community for its extraordinary ability to simulate complex real-world distributions. GANs have demonstrated outstanding performance in synthesizing a variety of datasets, particularly natural images. Despite the fact that GANs have produced excellent outcomes in different domains, training them is difficult, and suffer from serious instability difficulties. The original GAN referred to as vanilla GAN used the min-max objective function, also called as saturating objective function. Goodfellow et al. (2014) showed training procedure of conventional GAN can be viewed as approximately minimizing Jensen-Shannon divergence. The problem with this

function is that it results in the vanishing gradient. To mitigate this issue, the authors recommended the use of an alternate loss function termed as non-saturating loss function. Although it removed the challenging issue of vanishing gradient but incurred the problem called as mode collapse. To attain training stability and alleviate these challenges, a variety of GAN derivative models that have improved loss functions, have emerged endlessly. In this paper, we have first of all explored the complete mathematics behind the two versions of the loss functions used by the original GAN and summarized their strengths and weaknesses. It has been found that GANs problem ultimately arises and depends upon the design of loss function used and the new loss function variants have been found to improve the process of training in GANs compared to architectural GANs as a consequence of which a plethora of objective function GANs variants have been proposed. An in-depth study of the various loss function variants of GAN has been provided in this paper giving a mathematical insight into them. It also provides a summary of their advantages and disadvantages. Finally, we implemented the vanilla GAN, DCGAN, and the other GAN variants mentioned in this paper using MNIST, Fashion-MNIST, and CIFAR-10 datasets, and performed a detailed qualitative and quantitative analysis of the results obtained.

Acknowledgment
None.

Conflicts of interest
The authors have no conflicts of interest to declare.

Author's contribution statement
Rayeesa Mehmood: Conceptualization, writing original draft and editing, analysis and interpretation of results. **Rumaan Bashir:** Conceptualization, writing, review and supervision. **Kaiser J. Giri:** Conceptualization, review and writing.

References
[1] Simon A, Singh M, S. Venkatesan S, Babu DRR. An overview of M learning and its application. International Journal of Electrical Sciences Electrical Sciences & Engineering. 2015; 1(1): 22-4.
[2] Jebara T. Machine learning: discriminative and generative. Springer Science & Business Media; 2012.
[3] Harshvardhan GM, Gourisaria MK, Pandey M, Rautaray SS. A comprehensive survey and analysis of generative models in machine learning. Computer Science Review. 2020.
[4] Jebara T. Discriminative, generative and imitative learning (Doctoral Dissertation, PhD thesis, Media laboratory, MIT). 2001.
[5] https://iq.opengenus.org/discriminative-model/. Accessed 4 March 2022.
[6] https://towardsdatascience.com/what-is-deep-learning-and-how-does-it-work-2ce44bb692ac. Accessed 4 March 2022.
[7] Bishop CM, Nasrabadi NM. Pattern recognition and machine learning. New York: Springer; 2006.
[8] Salakhutdinov R. Learning deep generative models. Annual Review of Statistics and Its Application. 2015; 2:361-85.
[9] Ruthotto L, Haber E. An introduction to deep generative modeling. GAMM-Mitteilungen. 2021; 44(2).
[10] Goodfellow I. Nips 2016 tutorial: generative adversarial networks. arXiv preprint arXiv:1701.00160. 2016.
[11] Hong Y, Hwang U, Yoo J, Yoon S. How generative adversarial networks and their variants work: an overview. ACM Computing Surveys. 2019; 52(1):1-43.
[12] Singh U. Generative adversarial networks: a survey. 2021:1-28.
[13] Mescheder L, Geiger A, Nowozin S. Which training methods for GANs do actually converge? In international conference on machine learning 2018 (pp. 3481-90). PMLR.
[14] Ratliff LJ, Burden SA, Sastry SS. Characterization and computation of local Nash equilibria in continuous games. In 2013 51st annual Allerton conference on communication, control, and computing 2013 (pp. 917-24). IEEE.
[15] Barnett SA. Convergence problems with generative adversarial networks (GANS). arXiv preprint arXiv:1806.11382. 2018.
[16] Dutt RK, Premchand P. Generative adversarial networks (GAN) review. CVR Journal of Science and Technology. 2017; 13:1-5.
[17] Dong HW, Yang YH. Towards a deeper understanding of adversarial losses under a discriminative adversarial network setting. arXiv preprint arXiv:1901.08753. 2019.
[18] Chu C, Minami K, Fukumizu K. Smoothness and stability in GANS. arXiv preprint arXiv:2002.04185. 2020.
[19] Park SW, Ko JS, Huh JH, Kim JC. Review on generative adversarial networks: focusing on computer vision and its applications. Electronics. 2021; 10(10):1-40.
[20] De RGH, Papa JP. A survey on text generation using generative adversarial networks. Pattern Recognition. 2021.
[21] Park J, Kim H, Kim J, Cheon M. A practical application of generative adversarial networks for RNA-seq analysis to predict the molecular progress of Alzheimer's disease. PLoS Computational Biology. 2020; 16(7):1-20.
[22] Dia M, Savary E, Melchior M, Courbin F. Galaxy image simulation using progressive GANs. arXiv preprint arXiv:1909.12160. 2019.

[23] Navidan H, Moshiri PF, Nabati M, Shahbazian R, Ghorashi SA, Shah-mansouri V, et al. Generative adversarial networks (GANs) in networking: a comprehensive survey & evaluation. Computer Networks. 2021.
[24] Karras T, Aila T, Laine S, Lehtinen J. Progressive growing of GANS for improved quality, stability, and variation. arXiv preprint arXiv:1710.10196. 2017.
[25] Liao W, Hu K, Yang MY, Rosenhahn B. Text to image generation with semantic-spatial aware GAN. In proceedings of the IEEE/CVF conference on computer vision and pattern recognition 2022 (pp. 18187-96).
[26] Zhu M, Pan P, Chen W, Yang Y. DM-GAN: dynamic memory generative adversarial networks for text-to-image synthesis. In proceedings of the IEEE/CVF conference on computer vision and pattern recognition 2019 (pp. 5802-10).
[27] Zhu JY, Park T, Isola P, Efros AA. Unpaired image-to-image translation using cycle-consistent adversarial networks. In proceedings of the IEEE international conference on computer vision 2017 (pp. 2223-32).
[28] Ledig C, Theis L, Huszár F, Caballero J, Cunningham A, Acosta A, et al. Photo-realistic single image super-resolution using a generative adversarial network. In proceedings of the IEEE conference on computer vision and pattern recognition 2017 (pp. 4681-90). IEEE.
[29] Li W, Zhou K, Qi L, Lu L, Lu J. Best-buddy GANS for highly detailed image super-resolution. In proceedings of the AAAI conference on artificial intelligence 2022 (pp. 1412-20).
[30] Quan F, Lang B, Liu Y. ARRPNGAN: text-to-image GAN with attention regularization and region proposal networks. Signal Processing: Image Communication. 2022.
[31] Isola P, Zhu JY, Zhou T, Efros AA. Image-to-image translation with conditional adversarial networks. In proceedings of the IEEE conference on computer vision and pattern recognition 2017 (pp. 1125-34).
[32] Zhao J, Lee F, Hu C, Yu H, Chen Q. LDA-GAN: lightweight domain-attention GAN for unpaired image-to-image translation. Neurocomputing. 2022; 506:355-68.
[33] Jeong JJ, Tariq A, Adejumo T, Trivedi H, Gichoya JW, Banerjee I. Systematic review of generative adversarial networks (GANS) for medical image classification and segmentation. Journal of Digital Imaging. 2022; 35:137-52.
[34] Arora A, Shantanu. A review on application of GANs in cybersecurity domain. IETE Technical Review. 2022; 39(2):433-41.
[35] Brophy E, Wang Z, She Q, Ward T. Generative adversarial networks in time series: a survey and taxonomy. arXiv preprint arXiv:2107.11098. 2021.
[36] Kong J, Kim J, Bae J. Hifi-gan: generative adversarial networks for efficient and high fidelity speech synthesis. Advances in Neural Information Processing Systems. 2020; 33:17022-33.
[37] Jin CB, Kim H, Liu M, Jung W, Joo S, Park E, et al. Deep CT to MR synthesis using paired and unpaired data. Sensors. 2019; 19(10):1-19.
[38] Repecka D, Jauniskis V, Karpus L, Rembeza E, Rokaitis I, Zrimec J, et al. Expanding functional protein sequence spaces using generative adversarial networks. Nature Machine Intelligence. 2021; 3(4):324-33.
[39] Radford A, Metz L, Chintala S. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434. 2015.
[40] Arjovsky M, Chintala S, Bottou L. Wasserstein generative adversarial networks. In international conference on machine learning 2017 (pp. 214-23). PMLR.
[41] Gulrajani I, Ahmed F, Arjovsky M, Dumoulin V, Courville AC. Improved training of Wasserstein GANs. Advances in Neural Information Processing Systems. 2017.
[42] Fedus W, Rosca M, Lakshminarayanan B, Dai AM, Mohamed S, Goodfellow I. Many paths to equilibrium: GANs do not need to decrease a divergence at every step. arXiv preprint arXiv:1710.08446. 2017.
[43] Kodali N, Abernethy J, Hays J, Kira Z. On convergence and stability of GANs. arXiv preprint arXiv:1705.07215. 2017.
[44] Sharma A, Jindal N, Rana PS. Potential of generative adversarial net algorithms in image and video processing applications–a survey. Multimedia Tools and Applications. 2020; 79(37):27407-37.
[45] Jin L, Tan F, Jiang S. Generative adversarial network technologies and applications in computer vision. Computational Intelligence and Neuroscience. 2020.
[46] Aggarwal A, Mittal M, Battineni G. Generative adversarial network: an overview of theory and applications. International Journal of Information Management Data Insights. 2021; 1(1):1-9.
[47] Huang H, Yu PS, Wang C. An introduction to image synthesis with generative adversarial nets. arXiv preprint arXiv:1803.04469. 2018.
[48] Hitawala S. Comparative study on generative adversarial networks. arXiv preprint arXiv:1801.04271. 2018.
[49] Jabbar A, Li X, Omar B. A survey on generative adversarial networks: variants, applications, and training. ACM Computing Surveys. 2021; 54(8):1-49.
[50] Wali A, Alamgir Z, Karim S, Fawaz A, Ali MB, Adan M, et al. Generative adversarial networks for speech processing: a review. Computer Speech & Language. 2022.
[51] Jozdani S, Chen D, Pouliot D, Johnson BA. A review and meta-analysis of generative adversarial networks and their applications in remote sensing. International Journal of Applied Earth Observation and Geoinformation. 2022.
[52] Shahriar S. GAN computers generate arts? a survey on visual arts, music, and literary text generation using generative adversarial network. Displays. 2022.

[53] Saxena D, Cao J. Generative adversarial networks (GANs) challenges, solutions, and future directions. ACM Computing Surveys. 2021; 54(3):1-42.
[54] Kurach K, Lucic M, Zhai X, Michalski M, Gelly S. The GAN landscape: losses, architectures, regularization, and normalization. 2018.
[55] Pan Z, Yu W, Wang B, Xie H, Sheng VS, Lei J, et al. Loss functions of generative adversarial networks (GANs): opportunities and challenges. IEEE Transactions on Emerging Topics in Computational Intelligence. 2020; 4(4):500-22.
[56] Wiatrak M, Albrecht SV, Nystrom A. Stabilizing generative adversarial networks: a survey. arXiv preprint arXiv:1910.00927. 2019.
[57] https://www.cs.toronto.edu/~rgrosse/courses/csc321_2018/readings/L19%20GANs.pdf. Accessed 4 March 2022.
[58] Berard H, Gidel G, Almahairi A, Vincent P, Lacoste-julien S. A closer look at the optimization landscapes of generative adversarial networks. arXiv preprint arXiv:1906.04848. 2019.
[59] Wang Z, She Q, Ward TE. Generative adversarial networks: a survey and taxonomy. arXiv preprint arXiv:1906.01529. 2019.
[60] http://www.moreisdifferent.com/assets/science_notes/notes_on_GAN_objective_functions.pdf. Accessed 4 March 2022.
[61] Uddin SM. Intuitive approach to understand the mathematics behind GAN. Intuitive Approach Math. 2019.
[62] Huszár F. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? arXiv preprint arXiv:1511.05101. 2015.
[63] Theis L, Oord AV, Bethge M. A note on the evaluation of generative models. arXiv preprint arXiv:1511.01844. 2015.
[64] Manisha P, Gujar S. Generative adversarial networks (GANs): the progress so far in image generation. arXiv. 2019.
[65] Arjovsky M, Bottou L. Towards principled methods for training generative adversarial networks. arXiv preprint arXiv:1701.04862. 2017.
[66] Gui J, Sun Z, Wen Y, Tao D, Ye J. A review on generative adversarial networks: algorithms, theory, and applications. IEEE Transactions on Knowledge and Data Engineering. 2021.
[67] Shannon M, Poole B, Marioryad S, Bagby T, Battenberg E, Kao D, et al. Non-saturating GAN training as divergence minimization. arXiv preprint arXiv:2010.08029. 2020.
[68] Brock A, Donahue J, Simonyan K. Large scale GAN training for high fidelity natural image synthesis. arXiv preprint arXiv:1809.11096. 2018.
[69] Mallick PK, Meher P, Majumder A, Das SK. Electronic systems and intelligent computing: proceedings of ESIC 2020. Springer; 2020.
[70] Carneiro G. Why are generative adversarial networks so fascinating and annoying? In 33rd SIBGRAPI conference on graphics, patterns and images 2020 (pp. 1-8). IEEE.
[71] Wang Y. A mathematical introduction to generative adversarial nets (GAN). arXiv preprint arXiv:2009.00169. 2020.
[72] Weng L. From GAN to WGAN. arXiv preprint arXiv:1904.08994. 2019.
[73] Qin Y, Mitra N, Wonka P. How does Lipschitz regularization influence GAN training? In European conference on computer vision 2020 (pp. 310-26). Springer, Cham.
[74] Nakamura K, Korman S, Hong BW. Stabilization of generative adversarial networks via noisy scale-space. arXiv preprint arXiv:2105.00220. 2021.
[75] Pinetz T, Soukup D, Pock T. On the estimation of the Wasserstein distance in generative models. In German conference on pattern recognition 2019 (pp. 156-70). Springer, Cham.
[76] Sampath V, Maurtua I, Aguilar MJJ, Gutierrez A. A survey on generative adversarial networks for imbalance problems in computer vision tasks. Journal of Big Data. 2021; 8(1):1-59.
[77] Mao X, Li Q, Xie H, Lau RY, Wang Z, Paul SS. Least squares generative adversarial networks. In proceedings of the IEEE international conference on computer vision 2017 (pp. 2794-802).
[78] Bhatia H. Generalized loss functions for generative adversarial networks (Doctoral Dissertation, Queen's University (Canada)).

Rayeesa Mehmood received B.Tech in Computer Science and Engineering from the University of Kashmir in 2016 and received her M.Tech from Central University of Punjab, Bathinda in 2018. Currently, she is pursuing Ph.D in Computer Science at Islamic University of Science and Technology (IUST), Awantipora, J&K, India. Her research interests include Image Processing, Machine Learning, Deep Learning, Computer Vision, Natural Language Processing and related applications.
Email: rayeesa.mehmood@iust.ac.in

Rumaan Bashir received a Master's Degree in Computer Applications with Gold Medal from University of Kashmir, India in the year 2005. She also received M.Phil. & Ph.D. in Computer Science from University of Kashmir, India in 2011 & 2016 respectively. Currently , she is working as Associate Professor in the Department of Computer Science, Islamic University of Science & Technology, and Awantipora, J & K, India since 2006. She has participated in more than 50 national & international conferences, seminars & workshops. She has more than 25 papers in various journals & conferences of international repute to her credit. Her research interests include Image Processing, Signal Processing, Pattern Recognition, natural Language Processing, Information Security, Multimedia Security and

recently Machine Learning, Artificial Intelligence and related applications.
Email: rumaan.bashir@islamicuniversity.edu.in

Kaiser J. Giri received Master's Degree, M.Phil. & Ph.D. in Computer Applications from University of Kashmir, India in the year 2004, 2011 & 2016 respectively. He is working as an Associate Professor in the Department of Computer Science, Islamic University of Science & Technology, and Awantipora, J & K, India since 2006. He has participated in more than 50 national & international conference, seminars & workshops. He has more than 30 papers in various journals & conferences of international repute to his credit. His research interests include Image Processing, Signal Processing, Natural Language Processing, Information Security, Multimedia Security, Digital Image Watermarking, and Big-Data Analytics.
Email: kaiser.giri@islamicuniversity.edu.in

Appendix I

S. No.	Abbreviation	Description
1	ANN	Artificial Neural Network
2	ARRPNGAN	Attention Regularization and Region Proposal Networks Based GAN
3	BCE	Binary Cross Entropy
4	CGANs	Conditional Generative Adversarial Network
5	CRF	Conditional Random Fields
6	CT	Computed Tomography
7	DAE	Denoising Auto Encoder
8	DAM	Domain-Attention Module
9	DBMs	Deep Boltzmann Machines
10	DBNs	Deep Belief Networks
11	DCGAN	Deep Convolutional GANs
12	DGM	Deep Generative Models
13	FID	Frechet Inception Distance
14	GANs	Generative Adversarial Networks
15	GSN	Generative Stochastic Network
16	HMM	Hidden Markov Models
17	IS	Inception Score
18	JS	Jensen-Shannon
19	KL	Kullback-Leibler
20	LDA-GAN	Lightweight Domain-Attention Generative Adversarial Network
21	LSGAN	Least Squares Generative Adversarial Network
22	MRI	Magnetic Resonance Imaging
23	MR-GAN	Magnetic Resonance Generative Adversarial Network
24	RBMs	Restricted Boltzmann Machines
25	ReLU	Rectified Linear Unit
26	SRGAN	Super-Resolution Generative Adversarial Network
27	SGD	Stochastic Gradient Descent
28	SVMs	Support Vector Machines
29	Tanh	Hyperbolic Tangent Function
30	WGAN	Wasserstein Generative Adversarial Network
31	WGAN-GP	Wasserstein GAN-Gradient
32	2D	Two Dimensional
33	3D	Three Dimensional