ISSN 2007-9737

Measuring Mode Collapse: Comparative Study between Architecture GAN

Miguel S. Soriano-Garcia[1,*], R. Sevilla-Escoboza[2], Angel Garcia-Pedrero[2]

[1] Centro de Investigación en Matemáticas,
Mexico

[2] Universidad de Guadalajara,
Centro Universitario de los Lagos,
Mexico

[3] Universidad Politécnica de Madrid,
Center for Biomedical Technology,
Spain

miguel.garcia@cimat.mx,
jesus.escoboza@academicos.udg.mx,
angel.garcia@ctb.upm.es

Abstract. Generative Adversarial Networks (GANs) are a type of generative model widely used in various applications, but they often suffer from a common problem called mode collapse. This phenomenon occurs when the generator learns to produce only a small group of images instead of diverse ones. Mode collapse can occur for two reasons: firstly, when the discriminator becomes so effective that the generator can no longer learn, and secondly, when the generator finds a way to deceive the discriminator with a small number of samples, causing it to lack the motivation to diversify its outputs. This study evaluates multiple GAN-based models with various metrics that measure mode collapse. The behavior of models with similar parameters is analyzed.

Keywords. Generative Models, Mode Collapse, Comparative, GAN, Metrics.

1 Introduction

Generative Adversarial Networks [5] (GANs) are a machine learning model used to generate new data from an existing dataset. GANs consist of two competing neural networks, the generator, and the discriminator, in an iterative training process.

The generator takes a random input, often referred to as ”noise,” and uses it to create a new image, sound, text, or another type of data that resembles the training dataset. The discriminator, on the other hand, receives both generated images from the generator and real images from the training dataset, and its job is to distinguish between the generated and real images.

During training, the generator tries to generate data that will fool the discriminator into mistaking it for real data. In contrast, the discriminator distinguishes between the generated and real data. This iterative process continues until the generator can generate indistinguishable data from real data for the discriminator.

Generative Adversarial Networks (GANs) have shown tremendous potential in various tasks such as generating images from text [15], super-resolution of images [11], video synthesis [19], realistic image rendering from a virtual world [7], image translation from one domain to another [21], natural protein sequence generation [16], and a myriad of medical applications including reconstruction, segmentation, and synthetic medical

image generation, as well as disease classification and detection [20].

However, despite their outstanding capabilities and usefulness, GAN architectures suffer from several problems during training. One of the most common problems is mode collapse, which hinders the generator from thoroughly learning the training data distribution. A mode can be defined as a value or category that occurs more frequently than others in a dataset and is used to describe the distribution of a variable, i.e., how the values of a variable are distributed across a dataset.

Mode collapse in GAN models is a phenomenon in which the generator learns to produce only a few or a single image similar to the training data instead of generating a diverse range of images. This can happen for two reasons: the discriminator becomes too efficient at distinguishing between real and fake images, which prevents the generator from learning, or the generator finds a small number of samples that deceive the discriminator, causing it to map all its samples to this small group. When mode collapse occurs, the generator can become insensitive to the diversity of its samples, leading it to lack incentives to diversify its outputs [4]. Various factors, such as poor choice of the loss function, lack of diverse data for training, poor network architecture, or imbalance between the generator and discriminator, can cause this phenomenon. Solving mode collapse may require adjusting the model's architecture, providing more diverse data for training, or modifying the loss function.

This study explores the behavior of different GAN models under various conditions. The research focuses on evaluating several models to measure the tendency of mode collapse using a variety of metrics. Multiple tests were conducted to analyze models trained under the same architecture and hyperparameters and examine their performance when trained with the initially presented architectures. The objective is to observe the limitations and capabilities of each model in terms of generalizing data from these architectures.

2 Materials & Methods

2.1 GAN Models

2.1.1 DCGAN

Introducing Deep Convolutional GAN (DCGAN) [14] marked a milestone in generative models. By combining the power of convolutional networks with GAN architecture, DCGAN proved its mettle in generating unsupervised data by training on various databases.

The essence of DCGAN lies in harnessing the benefits of convolutional networks to create a more stable architecture. The first change was to replace any deterministic spatial function layers like max-pooling with strided convolutions for the discriminator and fractional-strided convolutions for the generator. This eliminated the fully connected layers and allowed for deeper networks, enabling the model to learn intricate image features.

The generator and discriminator were equipped with batch normalization layers to further stabilize the network throughout the training. Additionally, the generator employed ReLU activation in all its layers except for the output layer, which used Tanh, while the discriminator had LeakyReLU layers.

The amalgamation of all these changes resulted in a more robust and less prone-to-mode-collapse model, thanks to the advantages offered by convolutional networks in adversarial models. DCGAN has taken a significant leap towards achieving more realistic and high-quality image generation.

2.1.2 WGAN

Wasserstein Generative Adversarial Network (WGAN) [1] introduces a novel approach to generative models, utilizing the Wasserstein distance function to measure the distance between the distributions of real and generated data. This unique approach provides a more stable model with fewer collapse issues during training.

Beyond its modification of the architecture by introducing a critic instead of a discriminator, with the same task but with an unbounded output ranging from negative to positive infinity instead of the conventional 0 to 1, WGAN employs a loss function known as Earth Mover's Distance (EMD).

The EMD measures the minimum cost required to transform the synthetic data distribution into the real data distribution, considering the necessary distance and the amount of data that needs to be moved. Thus, its objective function can be expressed as follows:

$$\min_G \max_D V(D,G) = E_{x\sim p_x} D(x) - E_{z\sim p_g} D(G(z)). \tag{1}$$

where p_x represents the distribution of the training data, while p_g corresponds to the synthetic distribution generated by the generator G, which takes a latent variable z as input. Additionally, D denotes the output of the discriminator, which in this case is referred to as the critic. Unlike a typical discriminator, the critic is not trained to discriminate between real and fake data but rather to provide a critical value of the difference between the distributions. The objective of the critic is to minimize the value of the function concerning the generator. Overall, the critic assesses the quality of the generated data and guides the generator towards producing more realistic outputs.

2.1.3 WGAN-GP

WGAN-GP [6], which stands for Wasserstein Generative Adversarial Network with Gradient Penalty, is a variant of the WGAN model. This model also uses the Wasserstein distance to measure the difference between real and generated distributions. However, it includes a new technique called gradient penalty, which limits the weights' capacity to prevent them from becoming too large. This is crucial to stabilize the training process and avoid mode collapse in the generator.

The gradient penalty technique restricts the weights' capacity to be too large by adding a term to the model's cost function that penalizes large weights. This term is calculated by multiplying a penalty constant, denoted by ∇, with the L2 norm of the gradients of the generator's weights concerning a random point between the real and generated input. The L2 norm is the squared difference between a prediction and the actual value.

The cost function of the WGAN-GP model can be expressed mathematically as follows:

$$\min_G \max_D V(D,G) = E_{x\sim p_x} D(x) - E_{z\sim p_g} D(G(z)) + \lambda E_{\hat{x}\sim p_{\hat{x}}}[(\|\nabla_{\hat{x}} D(\hat{x})\| - 1)^2]. \tag{2}$$

where $\hat{x}$ is a random point between x and $G(z)$, λ is a penalty constant, and $p_{\hat{x}}$ is the distribution of points $\hat{x}$.

2.1.4 BEGAN

BEGAN [2] (*Boundary Equilibrium Generative Adversarial Networks*) is a generative image model based on adversarial networks. Unlike other GAN models, BEGAN uses a measure called the *auto-encoder loss* to automatically adjust the balance between image diversity and quality. This is achieved through a specific cost function that considers the similarity between the generated and real images and their distribution to achieve balance.

In practice, maintaining a balance between the generator's and the discriminator's losses is crucial. The model is considered in equilibrium when the discriminator can distinguish between real and generated images with a certain level of accuracy. At the same time, the generator can produce diverse and high-quality images. This balance is critical for the model's success in generating realistic images. It is considered to be in equilibrium when:

$$E[\mathcal{L}(x)] = E[\mathcal{L}(G(z))]. \tag{3}$$

The distribution of the loss function of the auto-encoder, with real images x, is denoted by $\mathcal{L}$, while $\mathcal{L}(G(z))$ is the distribution of the loss function of the auto-encoder with the synthetic images generated by G using a random vector input z.

If the generator can create images that fool the discriminator approximately half the time, the error distributions could be identical. This concept allows for balancing the losses of both agents so that neither one wins over the other. Therefore, it

is possible to achieve equilibrium by introducing a hyperparameter $\gamma \in [0, 1]$, defined as:

$$\gamma = \frac{E[\mathcal{L}(x)]}{E[\mathcal{L}(G(z))]}. \tag{4}$$

In the BEGAN model, the discriminator has two main objectives: encoding and decoding real images and discriminating between real and generated images. The hyperparameter γ helps to balance these objectives. Lower values of γ result in lower image diversity because the discriminator focuses more on encoding and decoding real images rather than discriminating between real and generated ones.

The objective function of the BEGAN model can be expressed as:

$$\begin{cases} \mathcal{L}_D = \mathcal{L}(x) - k_t\mathcal{L}(G(z)), & \text{For D,} \\ \mathcal{L}_G = \mathcal{L}(G(z)), & \text{For G,} \\ k_{t+1} = k_t + \lambda_k(\gamma(\mathcal{L}(x) - \mathcal{L}(G(z))), & \text{Each step t.} \end{cases} \tag{5}$$

This function implements a function k_t to control the emphasis on $\mathcal{L}(G(z))$ during training and having an initial value $k_0 = 0$. λ_k is proportional to the *learning rate* used during training.

2.1.5 BEGAN-CS

BEGAN-CS [3] (Boundary Equilibrium Generative Adversarial Networks with Conditional Structure) is a variation of the BEGAN model aimed at reducing mode collapse through a restricted space. By introducing a new term in the cost function that helps restrict the space produced by the discriminator's encoder, a more stable training process and improved performance can be achieved, especially when the training dataset is very small.

The cost function of the BEGAN-CS model is very similar to that of the BEGAN model, except that it adds a term $\mathcal{L}_c$ to restrict the encoder's space. Therefore, the objective function of this model is defined as:

$$\begin{cases} \mathcal{L}_D = \mathcal{L}(x) - k_t\mathcal{L}(G(z)) + \alpha\mathcal{L}_c, & \text{For D,} \\ \mathcal{L}_G = \mathcal{L}(G(z)), & \text{For G.} \end{cases} \tag{6}$$

Defining $\mathcal{L}_c$ and k_t as:

$$\begin{cases} \mathcal{L}_c = \|z - Enc(G(z))\|, & \text{Each step t,} \\ k_{t+1} = k_t + \lambda_k(\gamma(\mathcal{L}(x) - \mathcal{L}(G(z))), & \text{Each step t.} \end{cases} \tag{7}$$

where $\mathcal{L}_c$ is the magnitude in the difference between the generator input random vector z and the space generated by the encoder.

2.1.6 BEGANv3

BEGANv3 [13] is a variant of the BEGAN model that addresses the mode collapse issue by creating a handy and structured latent space. To achieve this, a variational autoencoder (VAE) is introduced, which adds statistical techniques to the autoencoder and the structured space created by the discriminator's encoder.

In the image compression process by the encoder, it is assumed that the input image is generated through statistical methods. This adds randomness to the encoding and decoding process of the space created by the discriminator. This model extracts a random image from the training dataset distribution, using the mean and variance, which is then encoded and restored as initially. By adding these statistical techniques, the resulting latent space becomes more structured and practical, which leads to a more stable training process and improved performance, mainly when the training dataset is small.

2.1.7 MGAN

MGAN [9] (*Mixture Generative Adversarial Nets*) is a novel model that combines the strengths of multiple generators to create images with improved quality and variety, aiming to cover diverse modes in the training dataset and overcome mode collapse.

In addition to multiple generators, MGAN adds a classifier to the base GAN architecture. The classifier's task is to identify which generator the sample comes from or if it comes from the training dataset.

To reduce the computational cost of training, the generators share the parameters of their layers, except for the input layer. Meanwhile, the discriminator and classifier also share their parameters, except for the last layer.

For this model, the following lost function was implemented:

$$\min_{G_{1:k},C}\max_{D} V(D,G_{1:k},C) = E_{x\sim p_x} logD(x) + E_{z\sim p_g} log(1-D(G(z)))- \beta\{\sum_{i=1}^{k}\pi_i E_{x\sim p_C} logC_i(x)\}. \quad (8)$$

The variable β represents a hyperparameter that controls the interaction between the generator and the classifier, while the multiple $\pi = [\pi_1, \pi_2, \ldots, \pi_k,]$ is an index containing the value of which generator the sample comes from, and k indicates the number of generators in the model.

2.1.8 MGO-GAN

MGO-GAN [12], or Multi-Generator Orthogonal GAN, is a variant of the MGAN model that also uses a mixture of generators. However, instead of using a classifier or shared parameters, this model employs a technique that uses orthogonal vectors to guide the multiple generators to learn complementary information to avoid overlap.

The multiple generators pass their samples to an encoder, which obtains a feature vector. These vectors' orthogonality is calculated between each possible pair, reflecting the correlation between the two vectors. This correlation value can be used to determine whether each of the generators is learning different information or, in some cases, is learning to generate similar images. The lower the correlation coefficient, the more it indicates that the generators are learning different information from each other.

For this model, the following loss function was implemented:

$$\min_{G_{1:k}}\max_{D} V(D,G_{1:k}) = E_{x\sim p_x} logD(x) + \frac{1}{k}\sum_{i=1}^{k}\{E_{z\sim p_g} log(1-D(G_i(z))) + \frac{1}{2}\sum_{i\neq j}\lambda O[E(G_i(z)), E(G_j(z))]\}. \quad (9)$$

In this loss function, k indicates the number of generators, and λ is a coefficient that gives more or less weight to the orthogonal value. E is the output of the encoder used to obtain the feature vectors for each sample, while O is the result of adding the orthogonal vectors, which is given by the following equation:

$$O(\alpha,\beta) = |\frac{(\alpha,\beta)}{|\alpha||\beta|}|, \quad (10)$$

where α and β are two different non-zero real number vectors.

2.2 Mode Collapse Metrics

To evaluate the collapse of mode and the correct distribution of generated data among the different modes of the training data, various metrics are used to evaluate the difference between the two data distributions. A generative model such as GAN aims to learn to generate images with a distribution similar to the data used to train the model. Additionally, the generated data should cover all modes and have a uniform distribution, meaning that the generative model should not have any preference for one or some data over others when generating images.

It's important to note that no metric is perfect, and they should be combined with visual inspections and other subjective criteria to evaluate the performance of a GAN model. By introducing several metrics to evaluate models, a more general idea of the performance of the models can be obtained, avoiding measuring high performance if any of the metrics suffer from overfitting. Therefore, to evaluate the performance of the models presented in section 2.1, the following metrics will be used to measure mode collapse.

2.2.1 Fréchet Inception Distance (FID)

The FID (Fréchet Inception Distance) metric is widely used to evaluate the quality of images generated by deep learning models, particularly in image generation tasks. It was introduced by Heusel et al. in 2017 [8] as an improvement over the Inception Score metric.

The FID metric combines Fréchet distance, a distance measure between two probability distributions, with features extracted from a pre-trained neural network called the Inception Network.

A set of images is first generated using the generative model, which is evaluated to calculate the FID. The features of these images are then extracted using the Inception network. The mean and covariance matrix of the extracted features from these images are computed.

Finally, this distribution of features is compared with the feature distribution of a reference set of images (typically, a set of real images from the same domain as the generated ones). The Fréchet distance between these two distributions is used to measure the quality of the images generated by the model.

Generally, the lower the FID value, the better the quality of the images generated by the evaluated model. The FID has become a commonly used measure in image generation research and is more robust than other quality measures, such as the Inception Score.

2.2.2 Samples Quality (SQ)

In their work, Srivastava et al. [18] proposed a novel approach for more accurately measuring mode collapse by introducing a new metric. The proposed method involves measuring collapse using synthetic data where both the distribution and modes are known, such as in a Gaussian mixture dataset.

First, some points are sampled from the generator to quantify mode collapse using this method. A sample is counted as high-quality if the generated sample is a certain distance away from the center of the nearest mode. Typically, a distance of 3 standard deviations is used for a two-dimensional dataset, while up to 5 standard deviations may be used for higher dimensions. After verifying which samples are of high quality according to this metric, a percentage is obtained to identify the number of high-quality samples. This metric ranges from 0 to 100, where 100 is the desired value.

2.2.3 Captured Modes (CM)

This metric measures how many modes from the training dataset were correctly learned by the generative model. To measure this, a mode is considered learned correctly when there are a certain number of high-quality generated samples within the standard deviation defined by the SQ metric. Typically, to determine whether a mode was learned, the following equation is used:

$$S_m = \frac{0.7 S_g}{M_t} : S_c \geq S_m, \tag{11}$$

here, S_m represents the minimum number of samples required for a mode to be counted as learned, S_g is the total number of generated samples, M_t is the total number of modes contained in the training dataset, and S_c is the number of high-quality samples in the mode.

2.2.4 Distribution of Samples by Class (SD)

In their study on mode collapse in natural data sets, Santurkar et. al. [17] proposed a method to identify if any of the modes in the data set are preferred or avoided by the generator during sample generation. By doing this, they aimed to uncover any biases or patterns in the generation process.

Their method involves training a generative model on a well-balanced data set (one that contains an equal number of samples per class or mode) and testing the generator to try and create a balanced data set for each class. The test involves the following three steps:

1. Train the unconditional generative model (without class labels) on the balanced data set.
2. Train a multiclass classifier on the same data set, this time with the class labels.

3. Generate samples with the generative model trained in step 1, then use the classifier trained in step 2 to determine which class each sample belongs to.

The metric evaluates whether all classes have the same number of generated samples. The goal is to have the same number of samples for each class among the generated samples. The range of this metric is from $[0, 200 - \frac{200}{C}]$, where C is the number of distinct classes, and the desired value is 0, which indicates that the generated samples are well balanced. To give a numerical value, the percentage of samples for each class is calculated using the following equation:

$$SD = \sum_{i=1}^{C} |\frac{100}{C} - p_i|. \tag{12}$$

where, C is the number of classes, and p_i is the percentage of samples that each class has.

2.3 Datasets

2.3.1 MNIST

The MNIST database is a collection of handwritten digit images widely used as a benchmark dataset for digit recognition. Created by Yann LeCun [10], it has become one of the most commonly used datasets in machine learning.

The MNIST dataset consists of a training set of 60,000 2828 pixel images and a test set of 10,000 images of the same dimension. Each image represents a digit between 0 and 9, handwritten and labeled accordingly.

The images in the MNIST database are grayscale, meaning each pixel is represented by a single intensity value ranging from 0 to 255, where 0 represents a black pixel and 255 represents a white pixel. Additionally, the images are normalized such that the average intensity of all pixels is zero, and the standard deviation of pixel intensities is one.

The MNIST dataset has been widely used as a benchmark for evaluating image classification algorithms. Furthermore, many researchers have used this database to train deep learning models, which has led to significant advances in digit recognition and the field of deep learning in general.

2.3.2 Sintetic Data

The synthetic training dataset consists of 500,000 two-dimensional points. These points are distributed in a circular pattern around 8 Gaussian. Each Gaussian has a standard deviation of 0.5 and is uniformly distributed along the range of -1 to 1 for both dimensions.

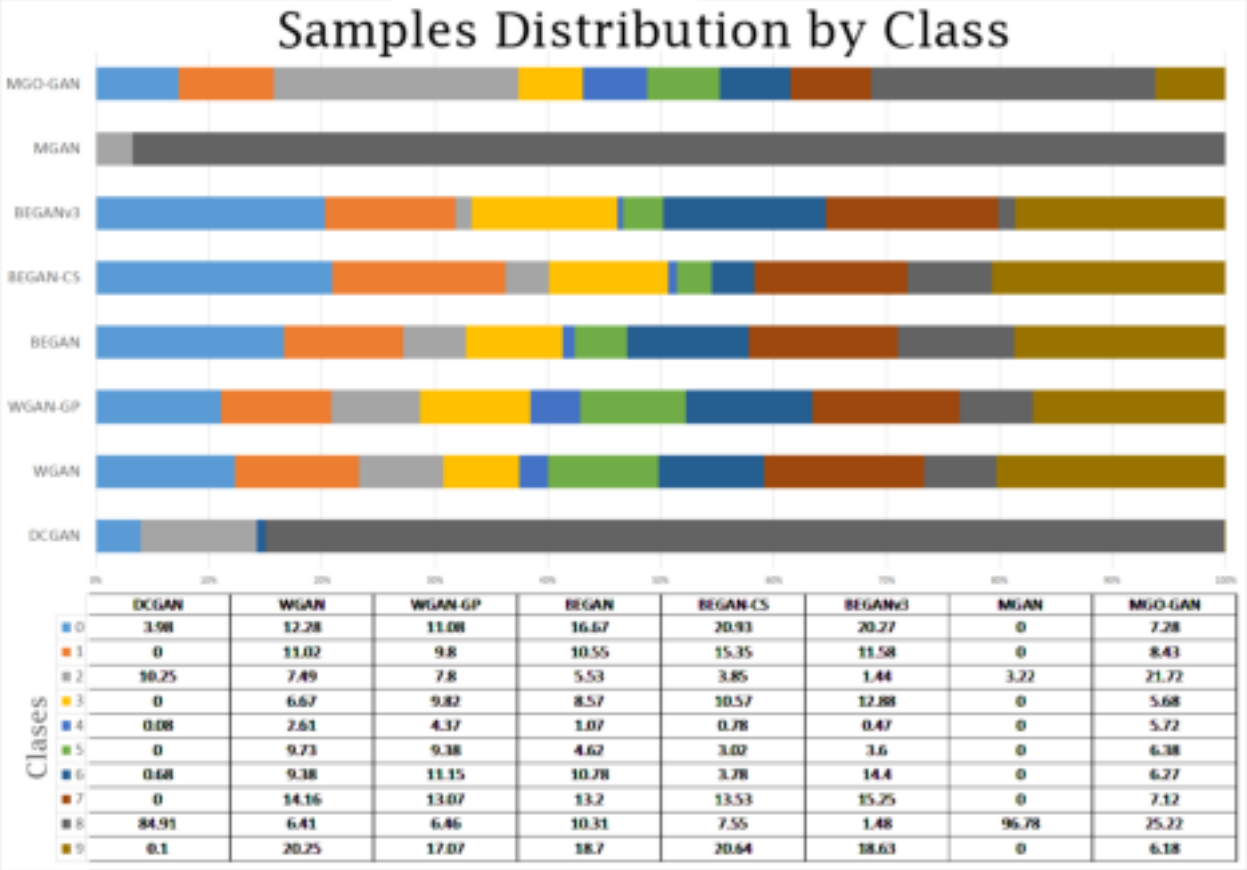
This dataset can be helpful for training models that need to learn to recognize patterns in data distributed in non-linear and complex ways. Being two-dimensional, this dataset can also be easily visualized and helpful in exploring data visualization techniques in machine learning. Additionally, the large number of points in the dataset makes it ideal for training models that require large amounts of data to achieve good performance.

3 Results

Three comparative studies were conducted on various GAN models, which were evaluated using different metrics to observe their behavior in the face of mode collapse under varying conditions.

The models were trained on the MNIST dataset in the first experiment and evaluated using the FID and SD metrics. They shared a similar architecture, with the generator consisting of three blocks, each beginning with a transposed convolutional layer with a square kernel of size 4, followed by a Batch-normalization layer and finally a ReLU layer. The blocks were then followed by a last transposed convolutional layer with a kernel similar to the previous ones but followed by a Tanh layer. The discriminator also had three blocks similar to the generator. Still, it uses a convolutional layer instead of the transposed convolutional one and a LeakyReLU layer in place of the ReLU. A convolutional layer followed these blocks and, finally, a Sigmoid layer. The WGAN and WGAN-GP models were trained using RMSProp optimizers, while the others were trained using Adam. All models were trained for 200 epochs, with a learning rate of 0.001

ISSN 2007-9737



Clases	DCGAN	WGAN	WGAN-GP	BEGAN	BEGAN-CS	BEGANv3	MGAN	MGO-GAN
0	3.98	12.28	11.08	16.67	20.93	20.27	0	7.28
1	0	11.02	9.8	10.55	15.35	11.58	0	8.43
2	10.25	7.49	7.8	5.53	3.85	1.44	3.22	21.72
3	0	6.67	9.82	8.57	10.57	12.88	0	5.68
4	0.08	2.61	4.37	1.07	0.78	0.47	0	5.72
5	0	9.73	9.38	4.62	3.02	3.6	0	6.38
6	0.68	9.38	11.15	10.78	3.78	14.4	0	6.27
7	0	14.16	13.07	13.2	13.53	15.25	0	7.12
8	84.91	6.41	6.46	10.31	7.55	1.48	96.78	25.22
9	0.1	20.25	17.07	18.7	20.64	18.63	0	6.18

Fig. 1. Distribution of the samples of the different GAN models, trained with a similar architecture and MNIST database

for the discriminator and 0.005 for the generator. The BEGAN, BEGAN-CS, and BEGANv3 models used the same structure as the discriminator for their encoders, while the decoder had the same structure as the generator. The MGAN and MGO-GAN models had 10 generators.

As observed in Table 1, the MGAN model obtained the highest FID score with 4.7102, indicating that the model suffered from collapse during training. In contrast, the WGAN-GP model performed best in this metric, scoring 0.1788. Considering the SD metric, models with high FID scores also had high SD scores. Once again, the MGAN model was the worst performer in this metric, with a score of 173.56, confirming its collapse during training. On the other hand, the WGAN-GP model had the best evaluation in the SD metric, with a score of 24.74.

In Fig 1, we can observe the amount of data generated for each class by each evaluated model. The model with the worst performance (MGAN) in FID and SD metrics generated only class 2 and 8 images. Another collapsed model was DCGAN, with FID and SD scores of 2.91 and 150.32, respectively, showing that 84.91% of its output data belonged to class 8. An interesting fact was the MGO-GAN model, with FID and SD scores of 2.2249 and 53.88, respectively. Together with the DCGAN and MGAN models, it was one of the only models with an FID score higher than 0.4, indicating that this model also suffered from collapse. However, it performed well in the SD metric, ranking as the fourth-best performer.

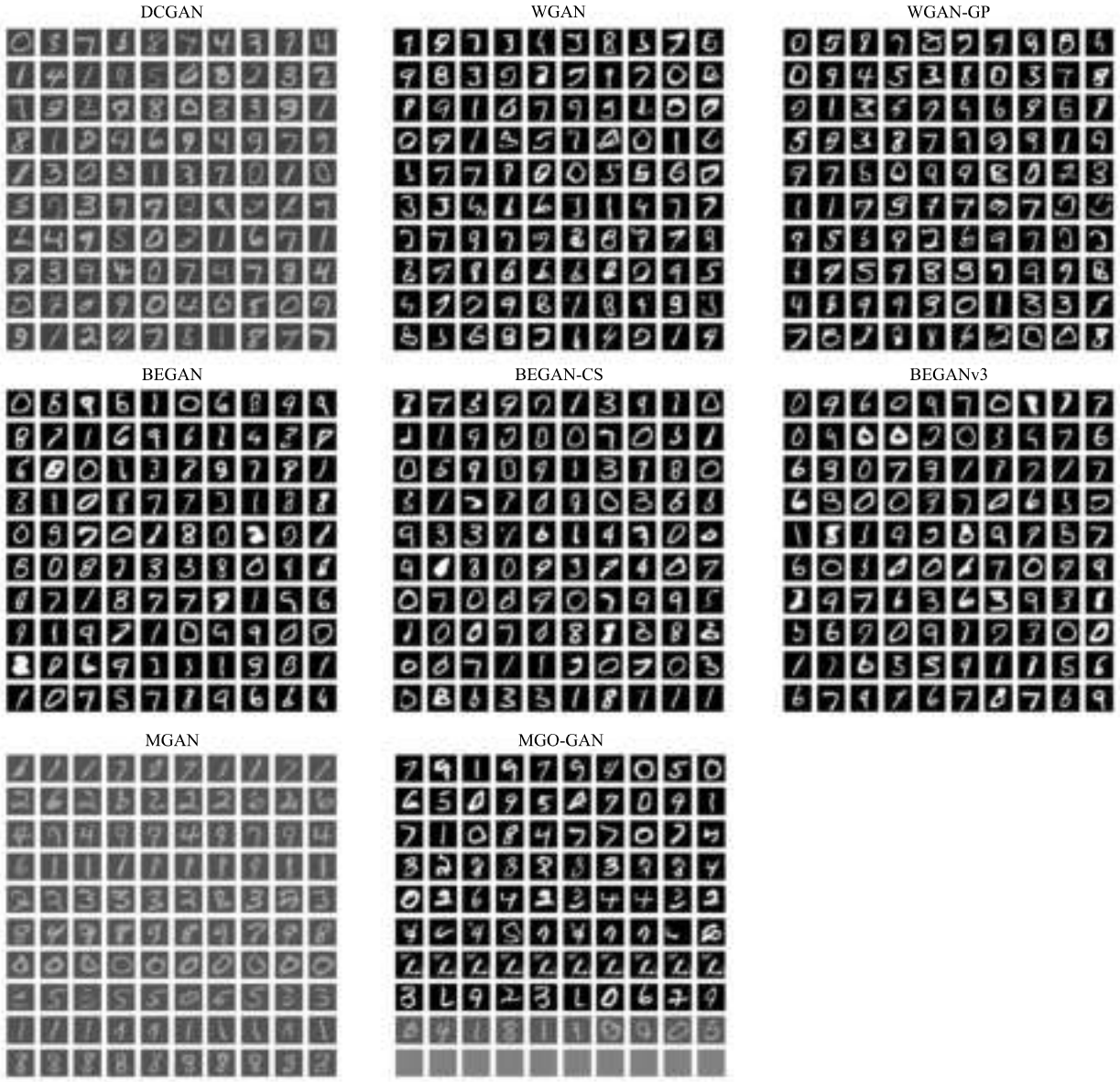


Fig. 2. Samples generated by the different models with the MNIST database

This was confirmed in Fig 1, where it could generate samples from all classes, even in a somewhat balanced way.

This may be due to the model's architecture, where having several generators allowed each one to learn to generate different information, aided by their orthogonal vectors cost function.

In Fig 2, several samples created by each of the compared generators in the experiment can be observed. It is evident that when the models are

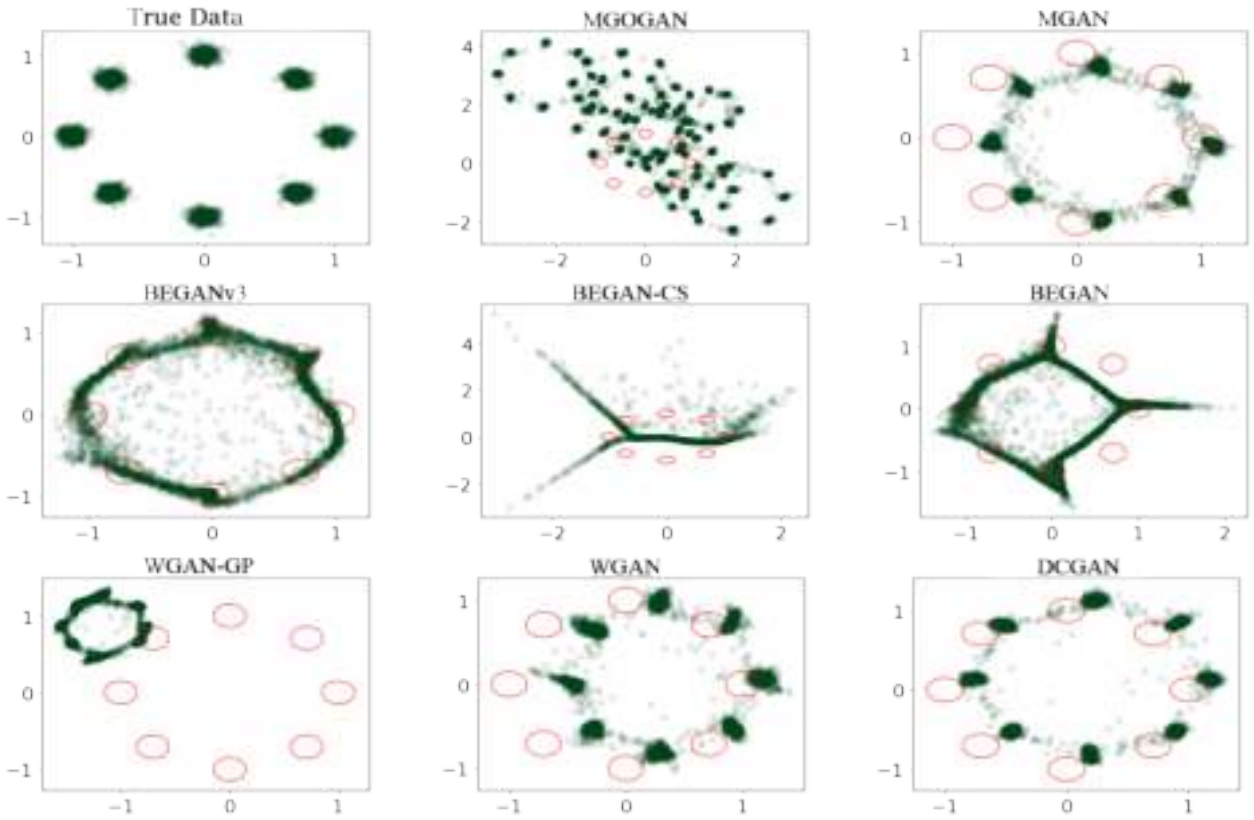


Fig. 3. Samples created by the generator of each model, training with the synthetic database of 8-ring Gaussians. The red circle has a diameter of 3 standard deviations of the modes and is in the center of the modes in the training database. Since the generated output is a vector of size two, the x axis is the value of the first value, while the y axis is the value of the second value

Table 1. Evaluation of the models with FID and SD metrics with the MNIST database as training. Taking into account that the models have the same architecture and the same hyperparameters

Model	FID	SD
DCGAN	2.9100	150.32
WGAN	0.3987	35.42
WGAN-GP	0.1788	24.74
BEGAN	0.1829	40.42
BEGAN-CS	0.2655	62.04
BEGANv3	0.2933	66.02
MGAN	4.7102	173.56
MGO-GAN	2.2249	53.88

collapsed, they create gray images or form figures that do not belong to the MNIST dataset.

In the second experiment, the behavior of GAN models was observed using a much simpler synthetic two-dimensional database. The models' architectures were similar, where the generator consisted of two blocks with a fully connected layer, a batch-normalization layer, and a ReLU layer. The first block takes a vector of size 100 as input and outputs a vector of size 64, while the second block takes the 64-sized vector as input and outputs a vector of size 32. After these two blocks, the last fully connected layer takes the 32-sized vector and outputs a vector of size 2. The discriminator also consists of two blocks, with the first block taking a vector of size 2 as input and outputting

Table 2. Evaluation of the models with FID, Sample Quality, and Captured Modes with the synthetic database (8 ring Gaussians) as training

Model	FID	SQ (%)	CM
DCGAN	0.0668	1.92	0
WGAN	0.1376	6.46	0
WGAN-GP	2.4715	8.56	1
BEGAN	0.0060	27.84	2
BEGAN-CS	0.0803	1.04	0
BEGANv3	0.0032	46.24	3
MGAN	0.0552	18.10	2
MGO-GAN	2.4075	0.62	0

a vector of size 32, while the second block takes that vector and outputs a vector of size 64. Lastly, a fully connected layer takes the 64-sized vector and outputs a single-feature output. The training parameters for this experiment were the same as the previous one, with the only difference being that the models were trained for only 100 epochs.

The comparison between the models can be seen in Table 2, where we can observe that the BEGANv3 model obtained the best evaluation in all three metrics when trained on the synthetic dataset, with an FID metric of 0.0032, a CM of 46.24, and an MC of 3. These results are consistent with expectations, as BEGANv3 is the only model to capture 3 out of the 8 total modes, giving it a higher probability of generating higher quality samples than the other models. Additionally, as the model with the highest number of captured modes, its samples are more diverse, which would also result in a better FID metric evaluation. On the other hand, when a higher FID metric is obtained, models are usually poorly evaluated in the CM and MC metrics. In this case, the MGO-GAN and WGAN-GP models are the worst evaluated, with 2.4075 and 2.4715, respectively, in the FID metric. However, while MGO-GAN has the worst evaluation in the CM metric with 0.62, the WGAN-GP model is the fourth best evaluated with 8.56. This last case may be because, at its collapse, the generator attempted to focus on only one of the modes and, in some way, learned to generate high-quality samples from that single mode.

When we use the generators of each trained model to create new data, we can observe that each model has its difficulties, as shown in Fig 3. The True Data figure displays the synthetic database used to train the models, while the other figures show the data generated by each model. We can see that the MGO-GAN model is trained to have each of its generators learn different information. Still, each generator attempted to replicate the shape of the training data with different values. On the other hand, models with an auto-encoder had difficulty generating a well-structured space, resulting in the generated data being disorganized and dissimilar to the training data. As mentioned before, during the training of the WGAN-GP model, a collapse occurred where the generator focused on generating information from a single mode. Although it attempted to create a structure similar to the training data, it only generated data from one mode.

The third experiment involved training the top-performing models according to their original papers while also observing and comparing them with their original architectures and optimal hyperparameters found in their respective works. In this case, models were selected based on their overall performance and having the same loss function or architecture, which led to the selection of WGAN-GP, BEGANv3, and MGO-GAN models. This comparison of models under their original specifications provides valuable insights into the robustness and reliability of these models in real-world applications.

The results of the third experiment are shown in Table 3. All models performed better when trained with their optimal architectures and hyperparameters than when all models were trained with the same architecture. Interestingly, the WGAN-GP model had the best performance, with an FID of 0.0445 and an SD of 9.26. On the other hand, models with an auto-encoder structure struggled to form a structured space, resulting in the worst SD score of 24.33. The MGO-GAN model, with its multiple generators, achieved a higher diversity in its samples and obtained an SD of 11.5. However, it struggled to create high-quality samples compared to BEGANv3, which achieved

Table 3. Evaluation of the models with FID and SD metrics with the MNIST database as training. Trained with original architecture and hyperparameters for each model

Model	FID	SD
WGAN-GP	0.0445	9.26
BEGANv3	0.0496	24.33
MGO-GAN	0.0868	11.5

FID scores of 0.0868 and 0.0496, respectively. In conclusion, the WGAN-GP model performed the best overall with these metrics and when trained on the MNIST dataset.

4 Conclusions

This study presents a comparative analysis of various GAN models to observe their performance when measured with different metrics that evaluate mode collapse. The models were subjected to three different tests, two of them involving training with the MNIST dataset. One test involved models with similar architecture, while the other evaluated them with their original architecture. The third test trained the models with a synthetic dataset with known modes and low complexity to visually observe when the models collapsed.

The study reveals the different limitations of the models, as varying the architecture, hyperparameters, and loss function resulted in poorer performance. With the metrics used, it was observed that most of the models experienced mode collapse, generating poorly distributed or low-quality samples. However, when evaluated with their original architectures, the results improved. This study underscores that GAN models are highly prone to mode collapse and emphasizes the need for ongoing research to address this issue

Acknowledgments

R.S.E. acknowledges support from Consejo Nacional de Ciencia y Tecnología call SEP-CONACYT/CB-2016-01, grant number 285909 and Universidad de Guadalajara ”Programa Fortalecimiento de Institutos, Centros y Laboratorios de Investigación 2021”. M.S.G. acknowledges financial support from ”Programa de Becas de CONACYT” with number of CVU: 924212. A.G.P. acknowledges financial support from UPM project RP200060107.

References

1. **Arjovsky, M., Chintala, S., Bottou, L. (2017).** Wasserstein Generative Adversarial Networks. In Proceedings of the 34th International Conference on Machine Learning, Journal of Machine Learning Research, Vol. 70, pp. 214–223.
2. **Berthelot, D., Schumm, T., Metz, L. (2017).** BEGAN: Boundary Equilibrium Generative Adversarial Networks. arXiv, pp. 1–10. DOI: 10.48550/arXiv.1703.10717.
3. **Chang, C.-C., Lin, C. H., Lee, C.-R., Juan, D.-C., Wei, W., et al. (2018).** Escaping from Collapsing Modes in a Constrained Space. In Proceedings of the European Conference on Computer Vision, Springer International Publishing, pp. 204–219. DOI: 10.48550/arXiv.1808.07258.
4. **Foster, D. (2019).** Generative Deep Learning: Teaching Machines to Paint, Write, Compose, and Play. O'Reilly Media.
5. **Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., et al. (2014).** Generative Adversarial Nets. Advances in Neural Information Processing Systems, Curran Associates, Vol. 27, pp. 1–9. DOI: 10.48550/arXiv.1406.2661.
6. **Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A. C. (2017).** Improved Training of Wasserstein GANs. Advances in Neural Information Processing Systems, Vol. 30, pp. 1–20. DOI: 10.48550/arXiv.1704.00028.
7. **Hao, Z., Mallya, A., Belongie, S., Liu, M.-Y. (2021).** GANcraft: Unsupervised 3D Neural Rendering of Minecraft Worlds. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 14072–14082. DOI: 10.48550/arXiv.2104.07659.
8. **Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S. (2017).** GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Curran Associates, Vol. 30, pp. 6629–6640.

9. **Hoang, Q., Nguyen, T. D., Le, T., Phung, D. (2017).** Multi-Generator Generative Adversarial Nets. arXiv, pp. 1–23. DOI: 10.48550/arXiv.1708.02556.
10. **LeCun, Y., Bottou, L., Bengio, Y., Haffner, P. (1998).** Gradient-Based Learning Applied to Document Recognition. In Proceedings of the IEEE, IEEE, Vol. 86, No. 11, pp. 2278–2324. DOI: 10.1109/5.726791.
11. **Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., et al. (2017).** Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. In IEEE Conference on Computer Vision and Pattern Recognition, IEEE Computer Society, pp. 4681–4690. DOI: 10.1109/CVPR.2017.19.
12. **Li, W., Fan, L., Wang, Z., Ma, C., Cui, X. (2021).** Tackling Mode Collapse in Multi-Generator GANs with Orthogonal Vectors. Pattern Recognition, Elsevier, Vol. 110, pp. 1–12. DOI: 10.1016/j.patcog.2020.107646.
13. **Park, S.-W., Huh, J.-H., Kim, J.-C. (2020).** BEGAN v3: Avoiding Mode Collapse in GANs Using Variational Inference. Electronics, Multidisciplinary Digital Publishing Institute, Vol. 9, No. 4, pp. 1–31. DOI: 10.3390/electronics9040688.
14. **Radford, A., Metz, L., Chintala, S. (2015).** Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. arXiv, pp. 1–16. DOI: 10.48550/arXiv.1511.06434.
15. **Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., et al. (2016).** Generative Adversarial Text to Image Synthesis. In Proceedings of the 33rd International Conference on Machine Learning, Proceedings of Machine Learning Research, Vol. 48, pp. 1060–1069. DOI: 10.48550/arXiv.1605.05396.
16. **Repecka, D., Jauniskis, V., Karpus, L., Rembeza, E., Rokaitis, I., et al. (2021).** Expanding Functional Protein Sequence Spaces Using Generative Adversarial Networks. Nature Machine Intelligence, Nature Publishing Group, Vol. 3, No. 4, pp. 324–333. DOI: 10.1038/s42256-021-00310-5.
17. **Santurkar, S., Schmidt, L., Madry, A. (2018).** A Classification-Based Study of Covariate Shift in GAN Distributions. In Proceedings of the 35th International Conference on Machine Learning, Proceedings of Machine Learning Research, Vol. 8, pp. 4480–4489. DOI: 10.48550/arXiv.1711.00970.
18. **Srivastava, A., Valkov, L., Russell, C., Gutmann, M. U., Sutton, C. (2017).** VEEGAN: Reducing Mode Collapse in GANs Using Implicit Variational Learning. Advances in Neural Information Processing Systems, Vol. 30, pp. 3310–3320.
19. **Wang, T.-C., Liu, M.-Y., Zhu, J.-Y., Liu, G., Tao, A., et al. (2018).** Video-to-Video Synthesis. arXiv, pp. 1–14. DOI: 10.48550/arXiv.1808.06601.
20. **Yi, X., Walia, E., Babyn, P. (2019).** Generative Adversarial Network in Medical Imaging: A Review. Medical Image Analysis, Elsevier, Vol. 58, pp. 1–20. DOI: 10.1016/j.media.2019.101552.
21. **Zhu, J.-Y., Park, T., Isola, P., Efros, A. A. (2017).** Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. In Proceedings of the IEEE International Conference on Computer Vision, arXiv, pp. 2223–2232. DOI: 10.48550/arXiv.1703.10593.

Article received on 22/04/2023; accepted on 13/01/2025.
**Corresponding author is Miguel S. Soriano-Garcia.*