Date of publication SEP-30, 2023, date of current version JUN-25, 2023.
www.computingonline.net / computing@computingonline.net

Print ISSN 1727-6209
Online ISSN 2312-5381
DOI 10.47839/ijc.22.3.3223

Optimization Strategy for Generative Adversarial Networks Design

OLEKSANDR STRIUK [1], YURIY KONDRATENKO [2,1]

[1]Petro Mohyla Black Sea National University, 10 68th Desantnykiv st., Mykolaiv, 54003, Ukraine
oleksandr.striuk@gmail.com

[2] Institute of Artificial Intelligence Problems of Ministry of Education and Science and National Academy of Sciences of Ukraine, 11/5 Mala Zhytomyrska st., Kyiv, 01001, Ukraine
y_kondrat2002@yahoo.com

Corresponding author: Oleksandr Striuk (e-mail: oleksandr.striuk@gmail.com).

ABSTRACT Generative Adversarial Networks (GANs) are a powerful class of deep learning models that can generate realistic synthetic data. However, designing and optimizing GANs can be a difficult task due to various technical challenges. The article provides a comprehensive analysis of solution methods for GAN performance optimization. The research covers a range of GAN design components, including loss functions, activation functions, batch normalization, weight clipping, gradient penalty, stability problems, performance evaluation, mini-batch discrimination, and other aspects. The article reviews various techniques used to address these challenges and highlights the advancements in the field. The article offers an up-to-date overview of the state-of-the-art methods for structuring, designing, and optimizing GANs, which will be valuable for researchers and practitioners. The implementation of the optimization strategy for the design of standard and deep convolutional GANs (handwritten digits and fingerprints) developed by the authors is discussed in detail, the obtained results confirm the effectiveness of the proposed optimization approach.

KEYWORDS artificial intelligence; machine learning; deep learning; generative adversarial network; design; optimization; loss function.

I. INTRODUCTION

GENERATIVE Adversarial Networks (GANs) are a type of deep learning algorithm first proposed by Ian Goodfellow in 2014 [1]. The primary motivation behind GANs is to create high-quality synthetic data for various applications, such as image, photo, and video generation [2, 3]. Further advancements in GANs development eventually demonstrated huge progress in such significant scientific and applied domains as anomaly detection [4], cybersecurity [5], medicine and drug discovery, forensics, material science, and astronomy research [3, 6, 7].

A. BACKGROUND OF GANS

GANs operate by training two neural networks: a generator and a discriminator. The generator takes a random noise vector as input and produces a synthetic sample that resembles real data; Fig. 1.

The discriminator takes both real and synthetic samples as input and distinguishes between them. The two networks play a game-like adversarial training process, with the generator attempting to trick the discriminator into "believing" that its synthetic samples are "real," and the discriminator attempting to accurately identify real and fake samples [1].

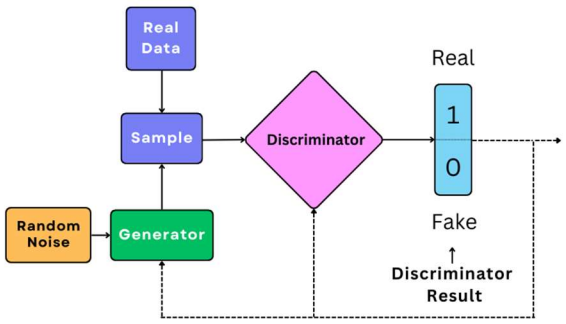

Figure 1. GAN Architecture.

The significant advantage of GAN models is that they do not require explicit modeling of the probability distribution of real data. This makes GANs suitable for generating high-dimensional, complex data such as images and photos.

Despite their general success, GANs can be difficult to train and require careful fine-tuning of hyperparameters. Furthermore, GANs can suffer from mode collapse, in which the generator produces only a limited set of samples that fail to capture the diversity of real data. Various methods have been

proposed to solve these problems, including the use of different loss functions, regularization methods, and architectures.

Overall, GANs have become a crucial tool in the deep learning toolkit, facilitating the creation of high-quality synthetic data for numerous applications [3].

B. TECHNICAL CHALLENGES IN GAN DESIGN AND OPTIMIZATION

GANs present several technical challenges in their design and optimization.

Vanishing gradients can occur when the discriminator becomes too good at distinguishing real from fake samples, leading to difficulty for the generator to learn [8].

Mode collapse can happen when the generator produces only a limited set of samples that do not fully capture the diversity of the real data [9]. Oscillations may occur when the generator and discriminator get stuck in a feedback loop and fail to converge to an optimal solution. Evaluation of the performance of GANs can be difficult, as there is no clear metric for measuring the quality of the generated samples [10].

Nevertheless, the primary benefit of GANs is that they eliminate the need for explicit modeling of the real data's probability distribution. This makes GANs well-suited for generating complex and high-dimensional data.

Addressing these technical challenges requires a combination of careful network design, tuning of hyperparameters, and novel optimization techniques.

The main goal of this article is to conduct thorough analytical research towards developing new approaches that will help to address these challenges and will help to improve the performance of GANs for a wide range of applications.

II. GAN DESIGN AND OPTIMIZATION TECHNIQUES

Loss functions play a crucial role in training the generator and discriminator networks. There are a variety of loss functions available for GANs, and selecting the appropriate loss for a specific GAN task is essential for achieving optimal results. It requires a deep understanding of the problem domain and expertise in the field.

Selecting the appropriate loss function for GAN models is a sophisticated art that combines experimental thinking, experience, and theoretical knowledge. It requires careful consideration and evaluation of various factors. Let's review some of the most prominent examples.

A. LOSS FUNCTIONS

One of the key components of every GAN model is the loss function used to train the discriminator and generator networks. The choice of loss function has a significant impact on the stability and quality of the generated output data. With the growing interest in GANs, there has been a surge of research in developing and improving different loss functions to enhance the performance of GANs.

Let's review some of the most commonly used GAN loss functions. By analyzing the characteristics of these loss functions, we can gain a better understanding of their strengths and limitations, and ultimately advance the state-of-the-art in GANs.

1. Adversarial Loss. Adversarial loss (or initial GAN loss) is the loss function used to update the generator weights in the GAN training process. The adversarial loss is defined as the negative log-likelihood of the discriminator output when the generator input is passed through the generator network [1].

The loss function of a GAN can be expressed mathematically using the following equation [1]:

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

where G represents the generator; D represents the discriminator; x is a vector – sample of real data; z is noise or a latent space vector extracted from a standard normal distribution; $p_z(z)$ is a prior on input noise variables; $p_{data}(x)$ is a prior on input real data variables; $\mathbb{E}$ is an expectation; $D(x)$ is the discriminator's output that represents the probability that x actually came from the data rather than from G; $V(D,G)$ is the value function of D and G in the two-player minimax game [1].

Table 1. Adversarial Loss Pros and Cons

Pros	Cons
High-quality samples. Adversarial loss drives the generator to create samples virtually indistinguishable from real data, fostering the creation of high-quality, authentic-like data.	***Training instability.*** Adversarial loss in GANs can cause training instability, especially if the discriminator overpowers or the generator can't closely mimic real data, leading to issues like mode collapse or fluctuating loss.
Flexibility. Adversarial loss is adaptable to diverse data and applications and can enhance generated sample quality when combined with other loss functions like reconstruction or cycle-consistency loss.	***Difficulty in evaluation.*** Assessing GANs' performance via adversarial loss is challenging since it doesn't directly show how well the generator mimics the data distribution. Often-used metrics like FID or IS may not always accurately represent the quality of the generated samples.
Conceptually simple. The adversarial loss function is straightforward and hence a preferred choice for GANs and similar generative models.	***Computational cost.*** Adversarial loss is computationally demanding, especially for large-scale GANs with high-dimensional data, which complicates GANs' scalability for more intricate datasets.

2. Binary Cross Entropy (BCE). BCE is a commonly used loss function in GANs that measures the difference between the discriminator's output and the target label. The BCE loss is used to update the discriminator weights during the training process and can be described as follows:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log h\left(x^{(i)},\theta\right) + \left(1-y^{(i)}\right)\log\left(1-h\left(x^{(i)},\theta\right)\right)\right], \quad (2)$$

where m is the number of examples in the entire batch, $-\frac{1}{m}\sum_{i=1}^{m}$ is the average loss of the whole batch, h represents predictions made by the model, y denotes labels for the different examples, x stands for features that are passed in through the prediction, θ represents parameters of the module that is computing the predictions.

Table 2. BCE Pros and Cons

Pros	Cons
Easy to compute. BCE loss is relatively simple to compute and is efficient in terms of both time and computation resources required for training.	***Can lead to mode collapse.*** Using BCE loss in GANs may cause mode collapse, where the generator creates limited outputs that deceive the discriminator, leading to less diverse generated samples.
Effective for binary classification tasks.	***Imbalanced class distribution.*** In some cases, the class distribution

BCE loss is well suited for binary classification tasks, where the output is either 0 or 1.	of the real and fake samples may be imbalanced. This can lead to bias in the discriminator's predictions, as it may focus on the more dominant class.
Encourages the discriminator to distinguish real from fake samples. BCE loss propels the discriminator in GANs to differentiate real from fake samples by penalizing wrong predictions, as it's trained to identify if an input is real (label 1) or fake (label 0).	***Sensitive to noise.*** BCE loss is sensitive to noise in the data. This can cause the discriminator to overfit to noisy samples, resulting in poor performance on unseen data.

3. Wasserstein Loss. Wasserstein loss (L), also known as Earth Mover's Distance (EMD), is a distance-based loss function that measures the difference between the distribution of real and synthetic data. The Wasserstein loss is used to update the discriminator weights during the training process and has been shown to be effective in stabilizing GAN training.

The Wasserstein GAN (WGAN) Loss is a modification of the standard GAN, where the discriminator outputs a number for each sample without classifying it as real or fake. Since the output number doesn't need to be between 0 and 1, a threshold of 0.5 can't be used to classify samples. Instead, the discriminator is trained to give higher scores to real instances than to fake ones [11].

The WGAN's discriminator is called a "critic" since it can't distinguish real from fake samples. While this has theoretical significance, practically, it acknowledges that the inputs to the loss functions don't necessarily have to be probabilities. Wasserstein Loss can be depicted as follows:

$$L_{critic} = D(x) - D\big(G(z)\big), \qquad (3)$$

where $D(x)$ stands for the critic's output for a real sample, $G(z)$ is the generator's output when given noise z, $D(G(z))$ is the critic's output for a fake sample.

The aim of the discriminator is to maximize this function, which involves maximizing the difference between its output on real samples and its output on fake samples [11].

Table 3. Wasserstein Loss Pros and Cons

Pros	Cons
Improved stability. Wasserstein loss aids in stabilizing GAN training as it provides a more continuous metric compared to the typical binary classification loss function used in GANs.	***Computational cost.*** Calculating Wasserstein distance is usually more computationally intensive than using traditional binary classification loss in GANs, potentially slowing down the training.
Improved sample quality. Wasserstein loss can improve the quality of generated samples in GANs as it offers a more significant and continuous metric to assess the difference between real and generated samples.	***Sensitivity to architecture.*** The performance of GANs using Wasserstein loss can be sensitive to the choice of architecture, hyper-parameters, and regularization techniques used.
Better gradient flow. Unlike traditional GANs, which suffer from the vanishing gradient problem, Wasserstein loss provides a more meaningful and stable gradient flow during the training process.	***Lack of diversity.*** The Wasserstein loss function is not specifically designed to encourage the generation of diverse samples, which can be a disadvantage if generating diverse samples is a goal of the GAN.

4. Least Squares Loss (LSGAN). LSGAN (V) is a modified version of the original GAN that uses a least-squares loss function instead of the binary cross-entropy loss. LSGAN has been shown to produce higher-quality samples and more stable training compared to the original GAN.

The objective functions for LSGANs can be formulated in the following manner [12]:

$$\begin{aligned} \min_D V_{\text{LSGAN}}(D) &= \frac{1}{2}\mathbb{E}_{x\sim p_{\text{data}}(x)}[(D(x)-b)^2] \\ &\quad + \frac{1}{2}\mathbb{E}_{z\sim p_x(z)}[(D(G(z))-a)^2] \\ \min_G V_{\text{LSGAN}}(G) &= \frac{1}{2}\mathbb{E}_{z\sim p_z(z)}[(D(G(z))-c)^2], \end{aligned} \qquad (4)$$

where V represents the value function, everything else is similar to *equation (1)* except for c, which denotes the value that G wants D to "believe" for fake data.

Table 4. LSGAN Pros and Cons

Pros	Cons
Robustness. The least squares loss is more robust to outliers compared to the binary cross-entropy loss. This means that it can handle situations where the generator produces samples that are far from the real data distribution.	***Sensitivity to outliers.*** While the least squares loss is more robust to outliers, it can also be more sensitive to them in some cases. This means that it can give too much importance to outliers and not enough to the bulk of the data.
Mode collapse avoidance. The least squares loss has been shown to be effective in avoiding mode collapse.	***Lack of diversity.*** The least squares loss can sometimes lead to a lack of diversity in the generated samples, as the generator may focus too much on matching the mean of the real data distribution and not enough on capturing its full range.
Stable training. The least squares loss can lead to more stable GAN training, as it avoids the problem of vanishing gradients that can occur with the binary cross-entropy loss.	***Requires careful tuning.*** The least squares loss requires careful tuning of its hyperparameters, such as the scaling factor, to achieve optimal performance. This can be time-consuming and challenging for practitioners.

5. Hinge Loss. Hinge loss is another commonly used loss function in GANs that is based on the maximum margin principle. Hinge loss has been shown to be effective in stabilizing GAN training and producing high-quality samples [13].

The mathematical notation for hinge loss is as follows [14]:

$$L(y, f(x)) = max(0, 1 - yf(x)), \qquad (5)$$

where y is the true label of a sample, and $f(x)$ is the predicted score for that sample, max is the maximum function, which returns the larger of its two arguments.

In the context of GANs, the hinge loss is used to train the discriminator, where y is set to 1 for real samples and -1 for fake samples, and $f(x)$ is the discriminator's output for that sample.

The hinge loss penalizes the model when the predicted score for the true label is less than 1. If the score is greater than or equal to 1, the loss is 0.

Table 5. Hinge Loss Pros and Cons

Pros	Cons
Robust to outliers. Hinge loss handles outliers well, making it suitable for GANs where generated samples may not match real samples perfectly.	***Sensitivity to hyperparameters.*** Hinge loss's sensitivity to hyperparameters, like the margin value, can impact results if chosen incorrectly.
Encourages diversity.	***Difficulty in convergence.***

Hinge loss promotes diversity in generated samples by penalizing their similarity, preventing repetition in GAN output.	Converging hinge loss can be challenging, particularly with inadequate network design or complex datasets. This can lead to unstable training and subpar generated samples.
Works well with large datasets. Hinge loss works well with large datasets because it is computationally efficient and can handle a large number of samples without overfitting.	***Limited applicability.*** Hinge loss might not be appropriate for all GANs and datasets. It may struggle with datasets of limited samples or with GANs employing diverse architectures like conditional GANs.

6. Maximum Mean Discrepancy (MMD). MMD is a distance-based loss function that measures the difference between the distribution of real and synthetic data. MMD has been shown to be effective in stabilizing GAN training and producing high-quality samples [15].

In GANs, the generator is trained to minimize the MMD distance between the generated and real data distributions, which encourages the generated data to match the real data distribution.

Table 6. MMD Pros and Cons

Pros	Cons
Improved stability. MMD improves GAN training by promoting similarity between generated and real samples, preventing mode collapse and other issues.	***Limited effectiveness.*** Although MMD enhances GAN training stability, it may not capture the complete complexity of the data distribution, potentially limiting sample quality from the generator.
Flexibility. MMD is a versatile distance measure for distributions, adaptable to various data and applications through different kernel functions.	***Hyperparameter tuning.*** MMD requires careful hyperparameter selection (e.g., kernel function, bandwidth), which can be challenging and greatly affect GAN performance.
Efficient computation. MMD can be computed efficiently using a simple and fast algorithm. This makes it a practical choice for use in GANs and other machine learning applications.	***Computational cost.*** Although MMD is computationally efficient, it may struggle with large datasets or complex models, limiting its practicality for certain applications.

It is important to note that there is no universally optimal choice of loss function for a given GAN architecture. The selection of a suitable loss function for a specific task must take into consideration the unique characteristics of the problem at hand. As a result, loss functions must be chosen on a case-by-case basis.

The suitability of a loss function is dependent on the specific requirements of the task, the dataset, and the architecture. Therefore, an adaptive approach to the selection of a loss function is necessary to achieve optimal performance. This often requires the use of experimental techniques to explore the performance of different loss functions under different conditions.

B. ACTIVATION FUNCTIONS

Activation functions play a critical role in GANs. They are used to introduce non-linearity into the generator and discriminator networks, which allows the model to learn complex relationships between the input and output data. The choice of activation function can have a significant impact on the performance of the GAN model, its accuracy, and its stability.

1. Sigmoid. Sigmoid is an activation function that maps the input to the range [0,1]. It is defined as:

$$\text{Sigmoid } (x) = \sigma(x) = \frac{1}{1+\exp(-x)}, \quad (6)$$

where x is an input value, σ is a standard deviation, and exp is an exponential function; Fig. 2.

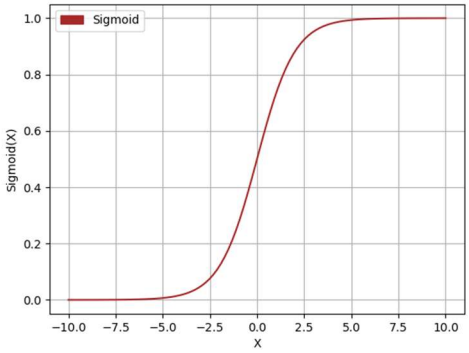

Figure 2. Logistic sigmoid activation function.

Sigmoid is commonly used as the activation function for the output layer of the generator in GANs to ensure that the generated samples are within the desired range [16, 17].

2. ReLU (Rectified Linear Unit). ReLU is one of the most widely used activation functions in deep learning, including GANs [16]. It is a non-linear activation function and can be expressed using the following notation:

$$f(x) = (x)^+ = max(0, x), \quad (7)$$

where x is the neuron input value, max is the output and denotes the maximum magnitude between zero and the input value; Fig. 3.

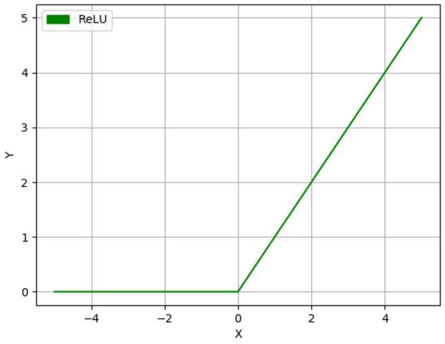

Figure 3. Rectified linear unit or ReLU – non-linear activation function.

ReLU is computationally efficient and allows for faster convergence during training.

3. LeakyReLU. LeakyReLU is a variant of ReLU that solves the problem of "dying ReLU," where the gradient of ReLU becomes zero for negative inputs, causing the corresponding neurons to stop learning [16, 17]. LeakyReLU is defined as:

$$\text{LReLU } (x) = max(0, x) + m * min(0, x), \quad (8)$$

where x is the neuron input value, max denotes the maximum magnitude between zero and the input value, m represents a negative slope, min represents the minimum value between zero and x; Fig. 4.

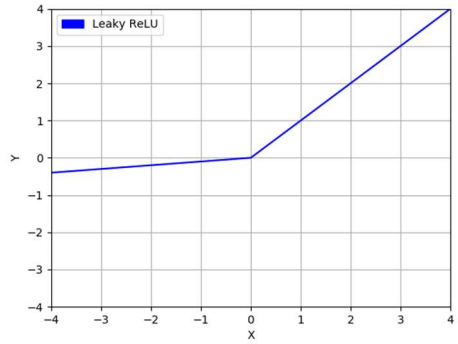

Figure 4. Leaky rectified linear unit or LReLU – linear variant of ReLU.

LeakyReLU allows for some negative values to pass through the activation function, providing a more robust gradient flow during training [16].

4. Tanh (Hyperbolic Tangent). Tanh is an activation function that maps the input to the range [-1,1]. It is defined as:

$$\text{Tanh}\,(x) = \tanh\,(x) = \frac{\exp\,(x) - \exp\,(-x)}{\exp\,(x) + \exp\,(-x)}, \qquad (9)$$

where tanh is a hyperbolic tangent function, x is the output of the generator, and exp is an exponential function.

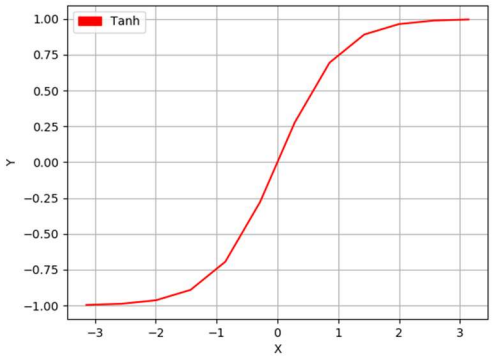

Figure 5. Hyperbolic tangent or tanh – hyperbolic activation function.

The hyperbolic tangent function (tanh) is helpful in GANs because it generates values between -1 and 1, compatible with the same range as normalized real data. With its smooth gradient, it assists in stabilizing the training process and mitigates the issue of exploding gradients, though it can still suffer from the vanishing gradient problem. Additionally, tanh can help the generator network produce a wider range of output values, making it more diverse and realistic [16, 17].

5. Softmax. Softmax is an activation function used for multi-class classification problems in GANs [16]. It maps the output of the model to a probability distribution over multiple classes. Softmax is defined as:

$$\text{Softmax}\,(x_i) = \frac{\exp(x_i)}{\Sigma_j \exp\,(x_j)}, \qquad (10)$$

where x_i is the output of the i-th neuron in the output layer and the sum is taken over all neurons in the output layer.

Activation functions reviewed above are a critical component of GANs that help to introduce non-linearity into the models. There is no single "best" activation function, as each function has its own set of strengths and weaknesses. Sigmoid and Tanh functions are commonly used in GANs for their ability to normalize output values between -1 and 1, but they suffer from the vanishing gradient problem. ReLU and LeakyReLU are popular due to their simplicity and ability to address the vanishing gradient problem, but they are not suitable for all applications. Softmax is used in multi-class classification tasks to generate probability distributions, but it is not appropriate for regression tasks. The choice of activation function depends on the specific requirements of the task and an understanding of the characteristics and trade-offs of each function.

C. BATCH NORMALIZATION

The process of normalizing inputs to every layer is referred to as *batch normalization* and is widely utilized in deep neural networks. This technique enhances the network's robustness to varying input distributions [18]. In the domain of GANs, batch normalization serves as a valuable tool to ensure the stability of both generator and discriminator network training [19].

In a GAN, the generator network leverages a random noise vector to generate synthetic images or samples, while the discriminator network takes an image as input and endeavors to differentiate between genuine and counterfeit images. The training of these networks occurs iteratively, wherein the generator strives to fabricate images that can deceive the discriminator, and the discriminator aims to accurately classify both genuine and fake images.

Batch normalization involves normalizing the inputs to each layer of the network based on the statistics of the mini-batch of data being processed. Specifically, batch normalization involves subtracting the mean and dividing by the standard deviation of the input values for each layer. This has the effect of centering and scaling the data, making it easier for the network to learn [18, 19, 20].

In GANs, batch normalization is typically applied to the inputs of the discriminator network, as this can help to reduce the problem of mode collapse. By normalizing the inputs to the discriminator, the network is less likely to overfit specific image features and can better distinguish between real and fake images [21].

Batch normalization can also be applied to the generator network, but this is less common as it can make the network more sensitive to small changes in the input noise vector, which can lead to instability in the training process. However, some variations of GANs, such as conditional GANs, may benefit from batch normalization in the generator network [12].

D. WEIGHT CLIPPING AND GRADIENT PENALTY

Weight clipping and *gradient penalty* are two additional techniques commonly used in the training of GANs to improve stability and encourage the generation of higher-quality images [22].

Weight clipping refers to a strategy employed to prevent the weights of the discriminator network from growing excessively large during the training process. This technique involves establishing a predetermined threshold value and, after each training iteration, capping the discriminator's weights to ensure they remain under this threshold. Through restricting the weights to a predetermined range, weight clipping can mitigate the issue of mode collapse, and curtail the discriminator from becoming excessively "confident" in its predictions [22].

However, weight clipping has some drawbacks. First, it can result in the loss of important discriminative information,

particularly when the weights are clipped too aggressively. Additionally, weight clipping can lead to instability in the training process, particularly when used in combination with other techniques like batch normalization [22].

While weight clipping can be an effective technique for preventing discriminator weights from becoming too large during training, it has a few potential drawbacks [22, 23].

First, weight clipping can lead to gradient vanishing or exploding, particularly when the threshold is set too low or too high, respectively. This can hinder the learning process and negatively impact the performance of the model.

Second, weight clipping can result in a non-smooth optimization objective, which can make it more difficult to find the global optimum during training.

Third, weight clipping can be difficult to set optimally, and the optimal value may vary depending on the specific dataset and model architecture.

Lastly, weight clipping can reduce the expressive power of the model and limit its ability to capture complex patterns in the data.

By penalizing the discriminator for producing gradients that deviate from the desired values, *gradient penalty* can help to reduce the problem of mode collapse and improve the overall quality of generated images. Additionally, gradient penalty has been shown to be more effective than weight clipping in improving the stability and convergence of the GAN training process [23].

However, gradient penalty has some limitations as well. First, it can be computationally expensive to calculate, particularly for large GAN models with many layers. Additionally, the effectiveness of gradient penalty depends on the specific choice of penalty function and hyperparameters used and may require careful tuning to achieve optimal results. Gradient penalty can lead to vanishing or exploding gradients, particularly when the penalty term is set too high. This can hinder the learning process and negatively impact the performance of the model [23].

One-Lipschitz (1-L) continuity is a property of functions that measures how much their output can change when the input changes by a small amount. In the context of GANs, 1-L continuity is often used to ensure the stability and convergence of the training process by preventing the discriminator function from becoming too sensitive to small changes in the input data. By enforcing this constraint, GANs can produce more stable and higher-quality generated samples [11, 23].

If the GAN discriminator is too sensitive to small changes in the input data, it may assign very different scores to two similar images, even if they are both real. This can lead to instability and poor performance in GAN.

To address this issue, some GANs use gradient penalty, which enforces 1-L continuity by adding a regularization term to the discriminator loss function. This term penalizes the discriminator if its gradients with respect to the input data exceed a certain threshold, which helps to ensure that the discriminator is not too sensitive to small changes in the input [11, 23].

Overall, enforcing 1-L continuity is an important consideration when designing and training GANs, as it can help to prevent instability and improve the quality of generated data.

In summary, weight clipping and gradient penalty are two additional techniques commonly used in the training of GANs to improve stability and encourage the generation of higher quality images. While weight clipping can help prevent the discriminator from becoming too "confident" in its predictions, gradient penalty can encourage the discriminator to produce more realistic gradients and improve the overall quality of generated images. However, both techniques have limitations and must be carefully tuned to achieve optimal results [22, 23].

E. STABILITY PROBLEM

Stability is a major challenge in training GANs and refers to the difficulty of maintaining a balance between the generator and discriminator networks during training. Instability can manifest in a number of ways, such as oscillating or exploding loss functions, vanishing gradients, or mode collapse [1, 19, 22].

There are several techniques that can be used to address stability issues in GANs, including the ones already mentioned such as batch normalization, weight clipping, and gradient penalty [24]. In addition to these, some other commonly used techniques to improve stability in GANs are:

1. One-sided Label Smoothing. This technique involves smoothing the labels for the real images to be less than 1 and the labels for the fake images to be greater than 0. By doing so, the discriminator is encouraged to be less certain in its predictions, which can help to prevent it from becoming too dominant during training [19, 25].

2. Feature Matching. Feature matching involves modifying the generator's loss function to encourage the generated images to match the intermediate features extracted by the discriminator, rather than just the final output. By doing so, the generator is encouraged to produce more diverse images, and the discriminator is less likely to overfit to specific image features [19].

3. Spectral Normalization. Spectral normalization involves normalizing the weights of the discriminator based on the spectral norm of each weight matrix. This can help to reduce the Lipschitz constant of the discriminator and improve the stability of the training process [26].

3. Diversity Regularization. Diversity regularization involves modifying the generator's loss function to encourage the production of diverse images, by penalizing the generator for generating similar images. This can help to reduce the problem of mode collapse and improve the overall quality of generated images [27].

4. Dropout. Dropout is a regularization technique used in the generator network of GANs to prevent overfitting and improve generalization performance. It works by randomly deactivating a proportion of the neurons during each training iteration, forcing the generator to rely on different subsets of neurons to generate samples. Dropout has been shown to be effective in preventing overfitting and improving the generalization performance of GANs, but its application requires careful tuning of the dropout rate and selection of the layers to apply dropout to [42].

These techniques are not mutually exclusive and can be combined depending on the specific needs of a GAN model. Ultimately, achieving stability in GANs requires careful tuning of the model architecture, hyperparameters, and training procedure, as well as the use of appropriate techniques to address stability issues.

F. TRANSPOSED CONVOLUTION

Transposed convolution, also known as deconvolution or fractionally-strided convolution, is a key operation in many

GANs. It is used in the generator network to upsample the low-resolution feature maps produced by the initial layers of the network [28].

The transposed convolution operation is essentially the reverse of a regular convolution operation. In a regular convolution operation, a kernel is applied to the input image to produce a set of output feature maps. In a transposed convolution operation, a kernel is applied to the output feature maps to produce a set of upsampled feature maps [28].

Transposed convolution is typically used in conjunction with a stride parameter to perform upsampling. The stride parameter determines the spacing between the output pixels, and by setting it to a value greater than 1, the operation effectively performs upsampling. For instance, a stride of 2 will double the resolution of the feature maps [28].

One important consideration when using transposed convolution is the choice of padding. In regular convolution, the output size is smaller than the input size due to the convolution operation "removing" some pixels around the edges. In transposed convolution, the output size can be larger than the input size if appropriate padding is applied. The choice of padding can affect the quality of the generated images and is an important hyperparameter to consider when designing a GAN-based model [28].

Transposed convolution allows the generator to produce high-resolution images from low-resolution inputs. By using a series of transposed convolution layers to gradually increase the resolution of the feature maps, the generator can produce images that are much larger and more complex than the initial input noise vector [29].

Although transposed convolution is a powerful technique, it is not without its challenges. One issue is that it can generate artifacts and checkerboard patterns in the resulting images. Another potential challenge is that training can be difficult, especially when using large stride values or deep neural networks. This is due to the operation's ability to amplify small errors in the input, which can result in unstable training. However, these challenges can be addressed by carefully tuning the transposed convolution parameters, such as adjusting the kernel sizes and strides, implementing skip connections, or using alternative upsampling techniques [29].

To conclude, transposed convolution is a valuable asset in the GAN toolkit as it enables the creation of high-quality, high-resolution images from low-dimensional input vectors.

G. CURSE OF DIMENSIONALITY

The term *curse of dimensionality* refers to the difficulties that arise when handling high-dimensional data. In the context of GANs, this can present significant challenges for both the generator and discriminator networks [1, 30].

One of the main challenges posed by the curse of dimensionality is the sparsity of high-dimensional data. As the dimensionality of the input data increases, the amount of data required to fully cover the input space increases exponentially. This can make it difficult to train GANs on high-dimensional data, as the amount of training data required to achieve good performance can quickly become prohibitively large [1].

Another challenge posed by the curse of dimensionality is the increased complexity of the generator and discriminator networks. As the dimensionality of the input data increases, the number of parameters in the generator and discriminator networks also increases. This can make it more difficult to train the networks, as the optimization problem becomes more complex, and the risk of overfitting increases [1].

Several approaches have been proposed to tackle the curse of dimensionality in GANs, such as dimensionality reduction, regularization, and transfer learning. Dimensionality reduction can be leveraged to decrease the input data's dimensionality, thus facilitating GAN training. Regularization methods can be utilized to prevent overfitting and improve the networks' generalization capacity. Transfer learning can be applied to benefit from pre-trained models on related tasks, which can reduce the amount of training data required for optimal performance [16, 30].

III. EVALUATION OF GAN PERFORMANCE

Due to the absence of intrinsic evaluation metrics, assessing the performance of GANs can be challenging. Their performance is often evaluated using metrics such as Fréchet inception distance (FID) and Inception score, which measure the quality and diversity of the generated images. In addition to FID and Inception score, precision and recall, as well as Perceptual Path Length (PPL), can be employed to comprehensively evaluate the performance of GANs. Let's review these metrics in greater detail.

1. Density estimation. Density estimation is a fundamental aspect of GANs, as the generator network learns to model the underlying distribution of the input data. The quality of the generated samples is directly related to the accuracy of the underlying distribution learned by the generator network. As a result, density estimation plays a crucial role in both training and evaluating GANs and is closely related to the performance evaluation metrics used for GAN-generated images [1, 8].

2. Fréchet inception distance (FID). FID is a commonly used metric for evaluating the quality of GAN-generated images. It measures the distance between the feature representations of real and generated images, as computed by a pre-trained Inception network. Lower FID scores indicate better quality generated images [19, 31].

3. Inception score. The Inception score is another commonly used metric for evaluating the quality of GAN-generated images. It measures the diversity and quality of generated images by computing the entropy of the predicted class labels for each image, and then taking the exponent of the mean entropy [19, 32].

4. Precision and recall. Precision and recall are metrics commonly used in classification tasks, but they can also be used to evaluate the performance of GANs. Precision measures the proportion of generated images that are classified as real, while recall measures the proportion of real images that are classified as such [33].

5. Perceptual Path Length (PPL). PPL is a metric for evaluating the diversity of GAN-generated images. It measures the average distance in feature space between pairs of images that are near each other in latent space. Lower PPL scores indicate greater diversity and higher quality generated images [34, 35].

Performance evaluation is an essential aspect of any machine learning model, and GANs are no exception. In the context of GANs, the evaluation is particularly challenging due to the absence of a clear objective function and the subjective nature of the generated samples' quality. The reviewed evaluation metrics are designed to provide a quantitative measure of GANs' performance and guide the development and improvement of GAN architectures. As this type of neural network continues to advance and finds new applications, the

development of more effective and robust performance evaluation techniques will remain an active area of research.

IV. ADDITIONAL GAN TECHNIQUES

There are additional methods that can improve the performance of GAN-based models. Let's review some of them.

A. ONE-SIDED LABEL SMOOTHING

One-sided label smoothing is a technique used to improve the generalization performance of GANs. It involves modifying the labels used for the discriminator network during training, by replacing the binary labels of 0 and 1 with smoothed labels of 0 and α, where α is a value between 0 and 1.

The purpose of label smoothing is to prevent the discriminator from becoming too "confident" in its predictions, which can lead to overfitting and poor generalization performance. By smoothing the labels, the discriminator is encouraged to be less certain in its predictions, which can help to improve the overall performance of the GAN model.

One-sided label smoothing is called "one-sided" because only the positive label (i.e., the label for real data) is smoothed, while the negative label (i.e., the label for generated data) is left unchanged. This is because smoothing the negative label can lead to unstable training and worse performance [19, 26].

B. MINI-BATCH DISCRIMINATION

Mini-batch discrimination is a GAN technique that enhances the diversity and quality of generated images. It integrates an extra layer to the discriminator network, which calculates a distance measure between the features of generated images and those of other images within the same mini-batch [19].

The purpose of mini-batch discrimination is to encourage the generator network to produce more diverse images by penalizing it for generating images that are too similar to each other. This can help to prevent mode collapse.

The distance metric used in mini-batch discrimination can take many forms, but a commonly used approach is to compute the L1 or L2 distance between the features of the generated images and the features of other images in the same mini-batch. The resulting distance values are then concatenated with the features of the generated images and passed through the discriminator network.

L1 and L2 distances are two types of distance metrics used in machine learning to measure the difference or similarity between two vectors. The L1 distance, also known as the Manhattan distance, is the sum of the absolute differences between the corresponding elements of two vectors. The L2 distance, also known as Euclidean distance, is the square root of the sum of the squared differences between the corresponding elements of two vectors.

Overall, mini-batch discrimination is a simple and effective technique for improving the diversity and quality of GAN-generated images, by encouraging the generator network to produce more diverse images and preventing mode collapse. It is particularly useful for generating high-resolution images or for tasks where the target distribution has many modes [19].

C. SAMPLING AND TRUNCATION TRICK

The sampling and truncation trick is a technique used to improve the diversity and quality of GAN-generated images. It involves randomly sampling the latent space of the generator network and truncating the samples to a certain percentile of their distribution. This can help to encourage the generation of diverse and high-quality images [36].

V. ALGORITHMS FOR IMPLEMENTING OPTIMIZATION STRATEGY

In order to enhance the performance of GAN models, optimization algorithms based on gradient descent have been explored, such as SGD, RMSProp, AdaGrad, Momentum, Adadelta, Adagrad, and ADAM [37].

1. Stochastic Gradient Descent (SGD) is a simple and widely-used optimization algorithm for neural networks, which updates the model parameters by computing the gradient of the loss function with respect to the parameters and adjusting them in the opposite direction of the gradient.

2. RMSProp is a gradient-based optimization algorithm that adapts the learning rate for each parameter based on the moving average of squared gradients, which reduces the impact of noisy gradients.

3. AdaGrad adapts the learning rate for each parameter based on the sum of the squares of past gradients, which gives larger updates for infrequent parameters and smaller updates for frequent ones.

4. Momentum uses a moving average of past gradients to accelerate SGD, which helps to avoid local minima and converge faster.

5. Adadelta is an extension of the AdaGrad algorithm that adapts the learning rate based on the moving average of gradients and updates, which further reduces the impact of noisy gradients.

6. ADAM adapts the learning rate for each parameter based on the first and second moments of the gradients, which results in more robust convergence and improved performance compared to other optimization algorithms.

Among these algorithms, ADAM has been shown to achieve superior performance compared to RMSProp, regardless of hyper-parameter settings, as demonstrated by experimental evaluation with logistic regression, multi-layer neural networks, and convolutional neural networks [37].

The optimization of GANs still remains an active research area, with ongoing efforts directed towards developing mathematical foundations and improving software and hardware implementations.

VI. OPTIMIZATION STRATEGY IN GAN DESIGN: EXPERIMENTAL RESULTS

In order to evaluate the effectiveness of some of the reviewed methods and approaches for optimizing GAN performance, we conducted experimental tests on two similar GAN models: the standard GAN and DCGAN with a BCE loss function. We trained the GAN model on the MNIST dataset, and the DCGAN model on the Sokoto Coventry Fingerprint Dataset with similar training and testing setups. We performed the training in a web-based Kaggle environment using a GPU as the main processing unit. The models were trained for a fixed number of epochs [6, 7].

We used Keras and PyTorch as the two main libraries, with Keras for GAN and PyTorch for DCGAN.

We used ADAM as an optimization algorithm and compared its performance to different benchmarks and the performance

indicators of other optimizers such as RMSProp and SGD. We analyzed their ability to optimize the standard GAN and DCGAN models. Our experimental results showed that Adam outperformed the other optimization algorithms in terms of convergence speed and final performance for both models.

For the standard GAN, we used Adam for both generator and discriminator models with a learning rate of 0.0002 and a beta_1 parameter of 0.5.

For DCGAN, we also applied Adam for both networks, with the same beta1 parameter but set two different learning rates for the generator and discriminator – 0.0001 and 0.0002, respectively.

Next, we tested the effectiveness of dropout as a regularization technique for improving GAN model performance and batch normalization for DCGAN.

Both techniques appeared to be effective and simple approaches for improving the stability and convergence speed of the standard GAN and DCGAN models.

We trained the generator and discriminator alternately, with the discriminator trained first and then the generator [6, 7].

Below are two samples of synthetic images generated by our GAN and DCGAN models (Fig. 6 and Fig. 7).

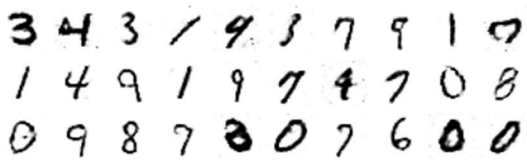

Figure 6. GAN-generated samples (trained on MNIST).

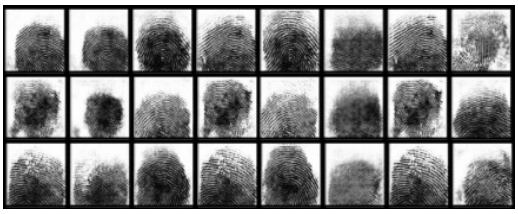

Figure 7. DCGAN-generated samples (trained on SOCOFing).

The challenges we faced included:

- High computational demands of the models.
- Overall low-speed performance.
- High rates of mode collapse.
- Low quality of the generated data in terms of accuracy.

Moving forward, we plan to practically implement all the design methods listed in the article into our newly developed GAN-based anomaly detection system, which also incorporates a fuzzy logic component. With the optimal configuration of hyperparameters, the use of appropriate optimization techniques, and the correct choice of evaluation metrics, authors expect to achieve high-performance results.

Overall, our experiments demonstrate the effectiveness of these techniques in improving GAN performance and highlight the importance of selecting appropriate optimization approaches for specific GAN models.

VII. CONCLUSION AND FUTURE RESEARCH DIRECTIONS

The article provides a comprehensive analysis of the technical peculiarities and challenges involved in designing and optimizing Generative Adversarial Networks to generate realistic, qualitative synthetic data of various range of types.

Through a range of topics, including loss functions, activation functions, batch normalization, weight clipping, gradient penalty, stability problems, performance evaluation, and others, various techniques used to address these challenges are reviewed, highlighting recent advancements in the field.

Moving forward, there is a need for further research into the development of more efficient and effective GAN optimization techniques, as well as their practical applications in various fields such as medicine, cybersecurity, computer vision, robotics, graphics, data augmentation, and intelligent systems [38-41].

The conducted analysis aims to provide a better understanding of how GAN models can be improved in terms of performance and computational efficiency. Despite the huge spike in popularity of large language models like GPT-4 and diffusion models like Stable Diffusion, GANs still possess the potential to significantly contribute to scientific research, technology, and business, and thus should be thoroughly researched. Also, the provided analysis gives a better intuition on that GANs should be designed and fine-tuned carefully taking into account a particular area of application. Hence, the analyzed materials have practical value for researchers in the field.

The continued improvement of GAN architectural optimization research is an essential area of focus for researchers and domain experts. By addressing the technical challenges outlined in this article and building upon recent advancements in the field, we can gain a more comprehensive understanding of GANs and harness their potential to create new, innovative solutions to complex problems.

References

[1] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, J. Bengio, "Generative adversarial networks," *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*, 2014, pp. 2672–2680.

[2] N. Aldausari, A. Sowmya, N. Marcus, and G. Mohammadi, *Video Generative Adversarial Networks: A Review*, 2022. https://doi.org/10.1145/3487891.

[3] O. S. Striuk, Y. P. Kondratenko, "Generative adversarial neural networks and deep learning: Successful cases and advanced approaches," *International Journal of Computing*, vol. 20, issue 3, pp. 339-349, 2021. https://doi.org/10.47839/ijc.20.3.2278.

[4] F. Di Mattia et al., *A Survey on GANs for Anomaly Detection*, 2021, [Online]. Available at: https://arxiv.org/abs/1906.11632

[5] O. S. Striuk, Y. P. Kondratenko, "Generative adversarial networks in cybersecurity: Analysis and response," in: Y. Kondratenko, V. Kreinovich, W. Pedrycz, A. Chilrii, A. M. Gil-Lafuente (Eds.), *Artificial Intelligence in Control and Decision-making Systems: Dedicated to Prof. Janusz Kacprzyk. Studies in Computational Intelligence*, vol. 1087, Springer, Cham, 2023, pp. 373-388. https://doi.org/10.1007/978-3-031-25759-9_18.

[6] O. Striuk and Y. Kondratenko, "Adaptive deep convolutional GAN for fingerprint sample synthesis," *Proceedings of the 2021 IEEE 4th International Conference on Advanced Information and Communication Technologies (AICT)*, Lviv, Ukraine, September 21-25, 2021, pp. 193-196. https://doi.org/10.1109/AICT52120.2021.9628978.

[7] O. Striuk, Y. Kondratenko, I. Sidenko, A. Vorobyova, "Generative adversarial neural network for creating photorealistic images," *Proceedings of 2020 IEEE 2nd International Conference on Advanced Trends in Information Theory*, Kyiv, Ukraine, November 27, 2020, pp. 368-371. https://doi.org/10.1109/ATIT50783.2020.9349326.

[8] M. Arjovsky, L. Bottou, *Towards Principled Methods for Training Generative Adversarial Networks*, 2017, [Online]. Available at: https://arxiv.org/abs/1701.04862.

[9] R. Ayari, *Generative Adversarial Networks*, 2020, [Online]. Available at: https://bit.ly/3Uk4GBw.

[10] A. Borji, *Pros and Cons of GAN Evaluation Measures*, 2018, [Online]. Available at: https://arxiv.org/abs/1802.03446.

[11] M. Arjovsky, S. Chintala, L. Bottou, *Wasserstein GAN*, 2017, [Online]. Available at: https://arxiv.org/abs/1701.07875.

[12] X. Mao et al., *Least Squares Generative Adversarial Networks*, 2016, [Online]. Available at: https://arxiv.org/abs/1611.04076.

[13] J. H. Lim, J. C. Ye, *Geometric GAN*, 2017, [Online]. Available at: https://arxiv.org/abs/1705.02894.

[14] C. Cortes, V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, issue 3, pp. 273-297, 1995. https://doi.org/10.1007/BF00994018.

[15] C.-L. Li et al., *MMD GAN: Towards Deeper Understanding of Moment Matching Network*, 2017, [Online]. Available at: https://arxiv.org/abs/1705.08584.

[16] I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning*, MIT Press, Cambridge, Massachusetts, 2016, 66 p., 178 p., 187 p., 189p.

[17] M. P. Deisenroth, A. A. Faisal, C. S. Ong, *Mathematics for Machine Learning*, 1st ed., Cambridge University Press, Cambridge, 2020, 160 p., 213 p., 315 p. https://doi.org/10.1017/9781108679930.

[18] S. Ioffe, C. Szegedy, *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*, 2015, [Online]. Available at: https://arxiv.org/abs/1502.03167.

[19] T. Salimans et al., *Improved Techniques for Training GANs*, 2016, [Online]. Available at: https://bit.ly/3L8qjBM.

[20] A. Radford, L. Metz, S. Chintala, *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*, 2016, [Online]. Available at: https://arxiv.org/abs/1511.06434,

[21] S. Xiang, H. Li, *On the Effects of Batch and Weight Normalization in Generative Adversarial Networks*, 2017, [Online]. Available at: https://arxiv.org/abs/1704.03971.

[22] M. Arjovsky, S. Chintala, L. Bottou, *Wasserstein GAN*, 2017, [Online]. Available at: https://arxiv.org/abs/1701.07875.

[23] I. Gulrajani et al., *Improved Training of Wasserstein GANs*, 2017, [Online]. Available at: https://arxiv.org/abs/1704.00028.

[24] K. Roth et al., *Stabilizing Training of Generative Adversarial Networks through Regularization*, 2017, [Online]. Available at: https://arxiv.org/abs/1705.09367.

[25] J. Hui, *GAN – Ways to improve GAN performance*, 2018, [Online]. Available at: https://bit.ly/3A8d11Z.

[26] T. Miyato et al., *Spectral Normalization for Generative Adversarial Networks*, 2018, [Online]. Available at: https://arxiv.org/abs/1802.05957.

[27] B. O. Ayinde, K. Nishihama, J. M. Zurada, *Diversity Regularized Adversarial Learning*, 2019, [Online]. Available at: https://arxiv.org/abs/1901.10824. https://doi.org/10.1007/978-3-030-19823-7_24.

[28] V. Dumoulin, F. Visin, *A Guide to Convolution Arithmetic for Deep Learning*, 2018, [Online]. Available at: https://arxiv.org/abs/1603.07285.

[29] J. Brownlee, *How to Use the UpSampling2D and Conv2DTranspose Layers in Keras*, 2019, [Online]. Available at: https://bit.ly/3oq94TA.

[30] J. Brownlee, *A Gentle Introduction to Transfer Learning for Deep Learning*, 2017, [Online]. Available at: https://bit.ly/3GTmdeC.

[31] M. Heusel et al., *GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium*, 2018, [Online]. Available at: https://arxiv.org/abs/1706.08500.

[32] S. Barratt, R. Sharma, *A Note on the Inception Score*, 2018, [Online]. Available at: https://arxiv.org/abs/1801.01973.

[33] M. S. M. Sajjadi, *Assessing Generative Models via Precision and Recall*, 2018, [Online]. Available at: https://arxiv.org/abs/1806.00035.

[34] T. Karras, S. Laine and T. Aila, "A style-based generator architecture for generative adversarial networks," *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Long Beach, CA, USA, June 15-20, 2019, pp. 4396-4405. https://doi.org/10.1109/CVPR.2019.00453.

[35] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen and T. Aila, "Analyzing and improving the image quality of StyleGAN," *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Seattle, WA, USA, June 13-19, 2020, pp. 8107-8116. https://doi.org/10.1109/CVPR42600.2020.00813.

[36] J. Brownlee, *A Gentle Introduction to BigGAN the Big Generative Adversarial Network*, 2019, [Online]. Available at: https://bit.ly/3om09CM.

[37] D. P. Kingma, J. Ba, *Adam: A Method for Stochastic Optimization*, 2014, [Online]. Available at: https://arxiv.org/abs/1412.6980.

[38] Y. Kondratenko, I. Atamanyuk, I. Sidenko, G. Kondratenko, S. Sichevskyi, "Machine learning techniques for increasing efficiency of the robot's sensor and control information processing," *Sensors*, vol. 22, issue 3, 1062, 2022. https://doi.org/10.3390/s22031062.

[39] M. Derkach, I. Skarga-Bandurova, D. Matiuk and N. Zagorodna, "Autonomous quadrotor flight stabilisation based on a complementary filter and a PID controller," *Proceedings of the 2022 12th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Athens, Greece, December 09-11, 2022, pp. 1-7. https://doi.org/10.1109/DESSERT58054.2022.10018623.

[40] A. Shevchenko, M. Vakulenko, and M. Klymenko, "The Ukrainian AI strategy: Premises and outlooks," *Proceedings of the 12th International Conference on Advanced Computer Information Technologies (ACIT)*, Ruzomberok, Slovakia, September 26-28, 2022, pp. 511-515. https://doi.org/10.1109/ACIT54803.2022.9913094.

[41] V. N. Opanasenko, S. L. Kryvyi, "Synthesis of neural-like networks on the basis of conversion of cyclic Hamming codes," *Cybernetics and Systems Analysis*, vol. 53, issue 4, pp. 627–635, 2017. https://doi.org/10.1007/s10559-017-9965-z.

[42] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

***OLEKSANDR STRIUK,** Ph.D. student and researcher at Petro Mohyla Black Sea National University (PMBSNU). Master of Science in System Analysis. Research interests include AI, AGI, ML, DL, GANs.*

***YURIY KONDRATENKO,** Doctor of Science, Professor, Honour Inventor of Ukraine (2008), Corr. Academician of Royal Academy of Doctors (Barcelona, Spain), Leading Researcher at the Institute of Artificial Intelligence Problems (IAIP) of MES and NAS of Ukraine, Head of the Department of Intelligent Information Systems at Petro Mohyla Black Sea National University (PMBSNU), Ukraine. He has received (a) the Ph.D. (1983) and Dr.Sc. (1994) in Elements and Devices of Computer and Control Systems from Odessa National Polytechnic University, (b) several international grants and scholarships for conducting research at Institute of Automation of Chongqing University, P.R.China (1988-1989), Ruhr-University Bochum, Germany (2000, 2010), Nazareth College and Cleveland State University, USA (2003), (c) Cleveland State University, USA (2015/2016, Fulbright Program). Research interests include robotics, automation, sensors and control systems, intelligent decision support systems, fuzzy logic.*