electronics

MDPI

Review

Review on Generative Adversarial Networks: Focusing on Computer Vision and Its Applications

Sung-Wook Park [1], Jae-Sub Ko [2], Jun-Ho Huh [3,*] and Jong-Chan Kim [1,*]

[1] Department of Computer Engineering, Sunchon National University, Suncheon 57992, Korea; 411050@scnu.ac.kr
[2] Department of Electric Control Engineering, Sunchon National University, Suncheon 57992, Korea; kokos22@scnu.ac.kr
[3] Department of Data Science, (National) Korea Maritime and Ocean University, Busan 49112, Korea
* Correspondence: 72networks@kmou.ac.kr (J.-H.H.); seaghost@scnu.ac.kr (J.-C.K.); Tel.: +82-51-410-4371 (J.-H.H.); +82-61-750-3620 (J.-C.K.)

Abstract: The emergence of deep learning model GAN (Generative Adversarial Networks) is an important turning point in generative modeling. GAN is more powerful in feature and expression learning compared to machine learning-based generative model algorithms. Nowadays, it is also used to generate non-image data, such as voice and natural language. Typical technologies include BERT (Bidirectional Encoder Representations from Transformers), GPT-3 (Generative Pretrained Transformer-3), and MuseNet. GAN differs from the machine learning-based generative model and the objective function. Training is conducted by two networks: generator and discriminator. The generator converts random noise into a true-to-life image, whereas the discriminator distinguishes whether the input image is real or synthetic. As the training continues, the generator learns more sophisticated synthesis techniques, and the discriminator grows into a more accurate differentiator. GAN has problems, such as mode collapse, training instability, and lack of evaluation matrix, and many researchers have tried to solve these problems. For example, solutions such as one-sided label smoothing, instance normalization, and minibatch discrimination have been proposed. The field of application has also expanded. This paper provides an overview of GAN and application solutions for computer vision and artificial intelligence healthcare field researchers. The structure and principle of operation of GAN, the core models of GAN proposed to date, and the theory of GAN were analyzed. Application examples of GAN such as image classification and regression, image synthesis and inpainting, image-to-image translation, super-resolution and point registration were then presented. The discussion tackled GAN's problems and solutions, and the future research direction was finally proposed.

Keywords: artificial intelligence healthcare; computer vision; deep learning; generative adversarial networks

Citation: Park, S.-W.; Ko, J.-S.; Huh, J.-H.; Kim, J.-C. Review on Generative Adversarial Networks: Focusing on Computer Vision and Its Applications. *Electronics* **2021**, *10*, 1216. https://doi.org/10.3390/electronics10101216

Academic Editor: Giovanni Dimauro

Received: 23 April 2021
Accepted: 16 May 2021
Published: 20 May 2021

1. Introduction

Deep learning models directly learn the high-level features of unstructured data [1]. The real power of deep learning lies in its ability to handle unstructured data. Especially, generative modeling generates unstructured data such as new images or text; therefore, deep learning wields great influence on the field of generative models.

Generative modeling is the next frontier of machine learning. Deep learning has been applied to generative modeling for just a few years. At the 2014 NIPS (Neural Information Processing Systems) conference, Google Brain's Ian Goodfellow introduced GAN (Generative Adversarial Networks) [2]. GAN has given rise to a number of algorithms and has advanced this field further.

Since mid-2018, great progress has been made in the field of sequence modeling and image-based generative modeling. Sequence modeling was mainly driven by the

transformer [3], an attention-based module that eliminates circulatory or convolutional neural networks. Examples include Google's BERT (Bidirectional Encoder Representations from Transformers), GPT-3 (Generative Pretrained Transformer-3) for language modeling, Parallel WaveGAN for speech synthesis, and MuseNet for music composition [4–7]. GAN-based technologies such as PGGAN (Progressive Growing of Generative Adversarial Networks), SAGAN (Self-Attention Generative Adversarial Networks), BigGAN, and StyleGAN have been developed; thus improving the position of image generation [8–11].

Recently, media interest in generative modeling projects has increased. The StyleGAN introduced by NVIDIA generates an authentic face image. GPT-3 from open artificial intelligence generates a complete sentence by providing a short introduction syntax. As of 2021, GAN and attention-based methods have evolved significantly, generating video, text, speech, and music that even experts cannot distinguish.

GAN has two networks: generator and discriminator. The generator converts random noise into a real sample, whereas the discriminator distinguishes whether the input sample is real or synthesized by the generator. An example of Input-Output for both networks is shown in Figure 1.

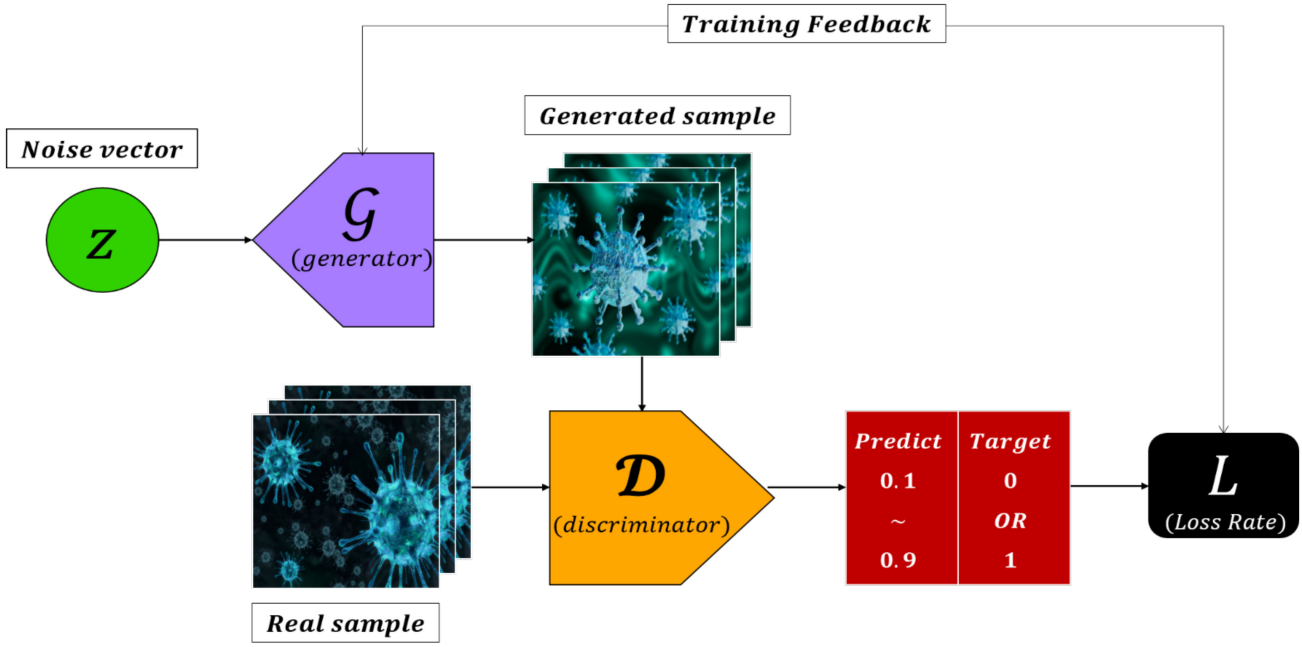


Figure 1. Training algorithm of GAN (Generative Adversarial Networks).

First, a real sample is selected from the training set for randomness. The output of the generator is then combined into a training set, and the discriminator is trained. The target of the real image is "1," and that of the generated image is "0." The real image outputs a value close to "1," and the synthetic image, a value close to "0."

It is difficult to train the generator because the real image is not mapped to any point in the latent space. When the output of the generator is inputted to the discriminator, the probability of being real is outputted. Such probability is the output of the GAN. The input is a randomly generated d-dimensional latent space vector, and the output is "1" to train the GAN by generating a training batch. The output should be set to "1" to generate a real sample.

Loss function is binary cross entropy between discriminator output and target "1". The target is a binary value, and it uses one output unit with the sigmoid activation function [12]. When training the GAN, the weight of the discriminator should be freezing so that only the weight of the generator is updated. Otherwise, it is adjusted to consider the generated image as real.

The criteria for selecting references appearing in this paper are as follows:

- A concept that first emerged in relation to a specific topic;
- It is not the first model to appear, but it shows a remarkable performance improvement compared to the existing model;
- Papers with higher citation index than existing models of similar concept.

The contributions of this paper are as follows: (1) You can check how the objective function, structure, and conditions of the GAN model affect the training results. At the same time, it is possible to acquire the knowledge necessary for model design and develop competencies. (2) Explain the theory of GAN. Based on this, observation of similar phenomena or future occurrences can be predicted, and tests can be conducted through experiments. In the absence of a theoretical perspective, we have no choice but to resort to empirical theory, which can lead to distortion. (3) By providing various application cases of GAN, you can learn application cases and make service improvements and performance in related fields. Furthermore, based on the theory and knowledge introduced in the thesis, it is possible to cultivate an eye that can be applied to other fields. (4) Major problems arising during GAN training were analyzed and countermeasures were suggested. (5) You can predict the future of GAN and prepare future response solutions based on this.

In Sections 1–4, the structure and operation principle of GAN, core models of GAN published so far, and theory of GAN was analyzed. In Section 5, application examples of GAN such as image classification and regression, image synthesis, image-to-image translation, super-resolution, and point registration were introduced. In Section 6, GAN's problems and solutions were described. In Section 7, the contents previously described were summarized and future research was forecasted.

2. Preliminaries

2.1. Notation

The generative model can synthesize images by grasping and learning the statistical distribution of training data. In any case, the network weight is learned through backpropagation [13]. The GAN literature deals with multidimensional vectors and italicizes vectors in the probability space. Latent vectors are usually denoted by z. In the signal processing field, vectors are represented by lowercase symbols to emphasize the multidimensional nature of variables. Therefore, $p_{data}(x)$ is the probability density function for random vector x of $R^{|x|}$. The probability that a continuous random variable will be included in a given interval is called probability density, and this is expressed as a probability density function. $p_G(x)$ represents the distribution of vectors generated in G. θ_D and θ_G are the weights learned from G and D, respectively.

As with all deep learning algorithms, training requires an objective function. At this time, loss function, objective function, and cost function are the same terms. To be precise, the cost function represents the sum of loss functions for all training data, and the objective function is the target function for optimization in more general terms. In general, however, these three terms are often used interchangeably. When the two objective functions are continuously updated, the objective functions of G and D are represented by $J_G(\theta_G;\theta_D)$ and $J_D(\theta_D;\theta_G)$ to remind that the parameter sets θ_G and θ_D are dependent on each other.

When the multidimensional gradient is updated, the gradient operator of the G weight is expressed as ∇_{θ_G}, and that of the D weight, as ∇_{θ_D}. ∇ is a differential operator wherein each component is represented as a formal vector with respect to Cartesian coordinates x, y, z. In the case of Expected gradient, it is denoted by $E\nabla$.

2.2. Data Distributions

The central issue in signal processing and statistics is density estimation, which acquires parametric or non-parametric data representations. The data generation distribution is used to represent the basic probability density or probability mass function of the observed data. GAN computes and learns the similarity between a candidate model distribution and a real data distribution.

Bayes's theorem can solve all inference problems in computer vision through conditional probability density functions [14]. It can be used as a model to learn the distribution of joints of interest and observation data. The problem is that it is difficult to construct a likelihood function for high-dimensional real images. GAN does not explicitly provide a method for evaluating density functions, and G implicitly captures the distribution of real data.

3. GAN Models

3.1. Objective Function

The objective function is very important because it is related to GAN's challenges. If you use an objective function that is not suitable for the task you are trying to solve, the GAN can go out of control during training. A typical example is when losses vibrate. The loss of the discriminator and the generator does not show a stable state for a long period, and it vibrates greatly.

As another example, although the image quality improves over time, the loss function of the generator may increase. This is due to the lack of association between the loss of the generator and the image quality. Lack of association makes it difficult to observe the GAN's training process. The objective functions introduced in this paper are considered to be suitable methods for training complex GANs.

3.1.1. WGAN (Wasserstein Generative Adversarial Networks)

The WGAN loss is significant in that it correlates the convergence of the generator and the quality of the sample. WGAN introduced Wasserstein Loss, which correlates the quality of samples with the convergence of generator [15]. Wasserstein loss improved stability during the optimization process. First, Arjovsky used $y_i = 1,\ y_i = -1$ instead of $y_i = 1,\ y_i = 0$ for binary cross-entropy loss. In addition, the sigmoid activation function was removed from the last layer of the discriminator. Therefore, prediction p_i is not limited to the [0,1] range but can be any number in the $[-\infty, \infty]$ range. For this reason, the discriminator of WGAN is called critic. The Wasserstein loss function is given by Equation (1).

$$-\frac{1}{n}\sum_{i=1}^{n}(y_i p_i) \tag{1}$$

WGAN compares prediction $p_i = D(x_i))$ and target $y_i = -1$ for the real image to train critic D. Prediction $p_i = D(G(z_i))$ and target $y_i = -1$ for the generated image are then compared to calculate the loss. The loss function of the WGAN critic can be minimized by Equation (2).

$$min_D - \left(E_{x\sim p_X}[D(x)] - E_{z\sim p_z}[D(G(z))]\right) \tag{2}$$

The WGAN critic maximizes the difference between the prediction of the real image and that of the generated image by increasing the score for the real image. Training the WGAN generator requires comparing the prediction and target for the generated image and calculating the loss. The loss function of the WGAN generator can be minimized by Equation (3).

$$min_G - \left(E_{z\sim p_Z}[D(G(z))]\right) \tag{3}$$

The Wasserstein loss function converges by training the discriminator so that the generator is updated correctly. This is different from the initial GAN, where it is important to ensure that the discriminator does not become too strong. Wasserstein loss can balance discriminator and generator training. WGAN trains the discriminator multiple times during the generator update to converge. In general, when updating a generator once, the discriminator is updated five times.

Because WGAN has clipped weights from critics, learning speed is greatly reduced. If the gradient is not correct, the generator cannot learn the weight update direction. For this reason, another method for the Lipchitz constraint, the WGAN-GP (Gradient Penalty), was presented [16].

3.1.2. WGAN-GP (Wasserstein GAN-Gradient Penalty)

WGAN-GP solves problems such as mode collapse and unstable training, and makes GAN training predictable and reliable. WGAN-GP included a gradient penalty term in the critic loss function [17]. Critics' weights are not clipped. Moreover, the batch normalization layer should not be used for critics. Batch normalization generates a correlation between images in the same batch, so gradient penalty loss has less effect [18]. WGAN-GP suggests another way to enforce the Lipchitz constraint on critics: adding a term to the loss function that penalizes when the gradient norm of critic deviates significantly from "1". As a result, the training process was greatly stabilized.

Gradient penalty loss is the squared difference between the gradient norm of the output and one. This model naturally finds weights that minimize the gradient penalty term. In other words, the model is made to follow the Lipchitz constraint. It is difficult to calculate the gradient everywhere during the training process. WGAN-GP only calculates the gradient at some point. In order not to be biased on one side, the real image-synthetic image pair are connected as shown in Figure 2, and the images interpolated using randomly selected points along a straight line are used.

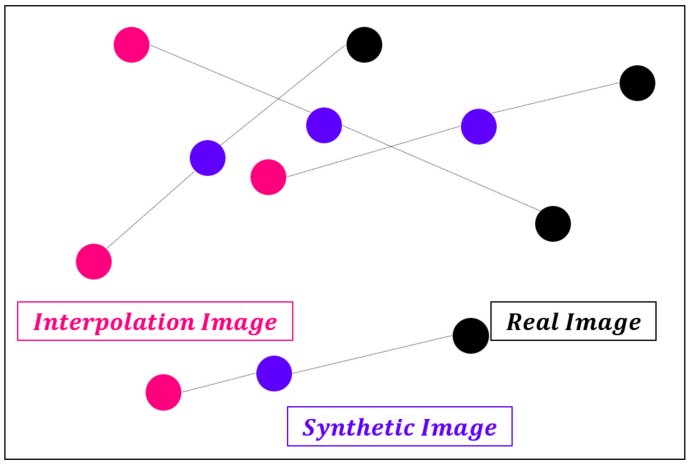


Figure 2. Interpolation between images.

3.1.3. SAGAN (Self-Attention Generative Adversarial Networks)

Attention is an algorithm used in sequence models, such as transformers [3]. SAGAN is a model that applies the attention algorithm to GAN [9]. The self-attention algorithm is shown in Figure 3.

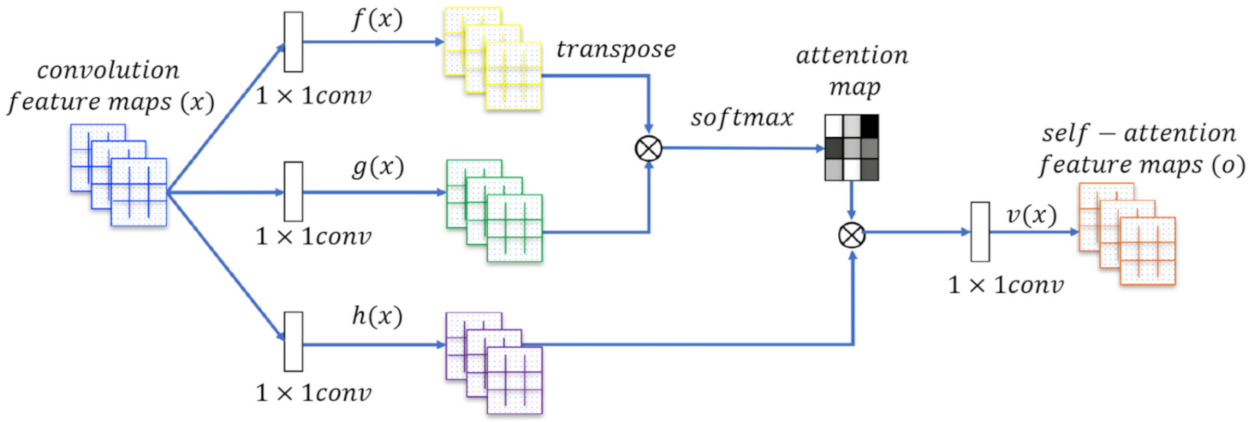


Figure 3. Self-attention algorithm of SAGAN (Self-Attention Generative Adversarial Networks) [9].

In a GAN without attention, the convolution feature map can process only local information. In order to connect pixel information on one side of the image to the other side, the channel must be increased to several convolutional layers, and the dimension of the image space must be reduced. This process, on the other hand, loses accurate location information instead of capturing high-level features. As such, it is inefficient for the model to learn the dependence between distant pixels. SAGAN solved the problem above by applying the attention algorithm to GAN. The outline of the proposed method is shown in Figure 4. As shown in Figure 4, attention focuses on different types of areas. SAGAN made a significant advancement using an attention mechanism that works similar to human perception.

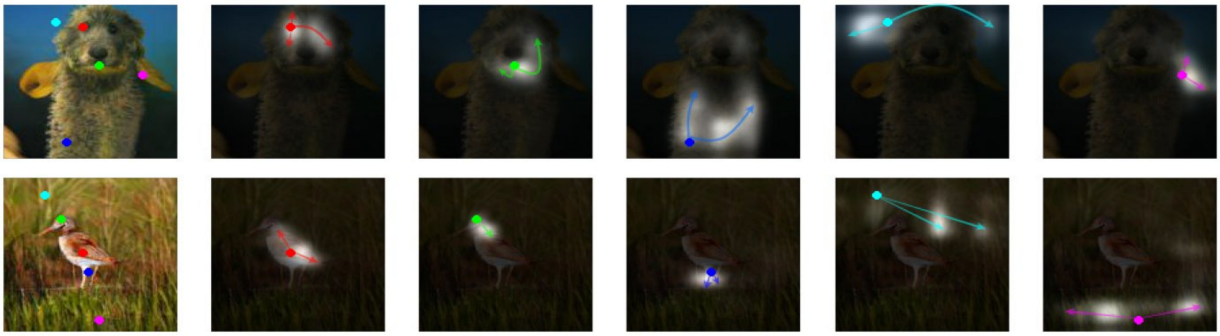


Figure 4. SAGAN generated image and its attention map [9].

3.2. Structure

3.2.1. DCGAN (Deep Convolutional Generative Adversarial Networks)

In recent years, supervised learning using CNN (Convolutional Neural Networks) has been widely applied in the field of computer vision [19]. Conversely, unsupervised learning using CNN has not received much attention.

Radford introduced DCGAN (Deep Convolutional Generative Adversarial Networks) in 2016 [20]. The interior of DCGAN is composed entirely of convolutional layers. The discriminator's pooling layer is replaced with stride convolution, and the generator's pooling layer with transpose convolution. After the convolutional layer, the fully connected classification layer is removed. Batch normalization is performed after each convolutional layer to promote the gradient flow.

The basic algorithm of DCGAN is the same as that of the traditional GAN. There is a generator that generates 100-dimensional noise, and the noise is mapped and transformed through the convolutional layer. Figure 5 presents DCGAN's generator structure.

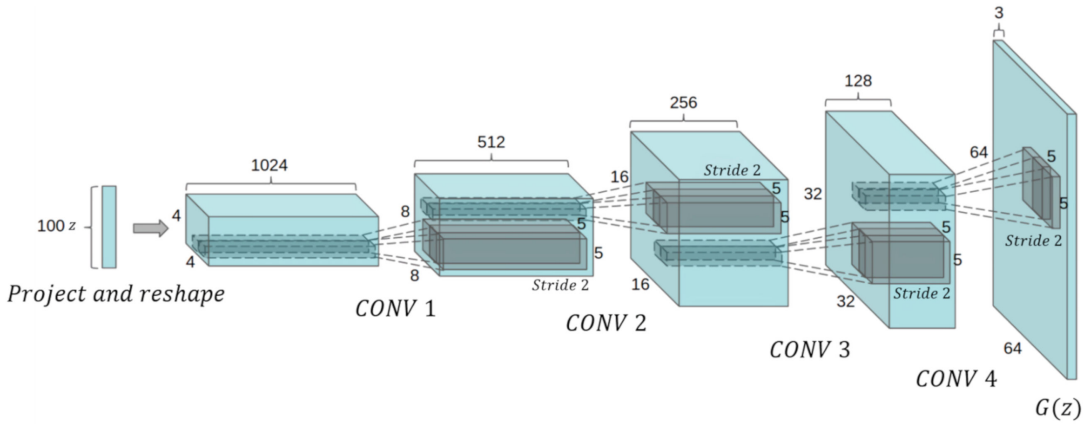


Figure 5. Generator structure of DCGAN (Deep Convolutional Generative Adversarial Networks) [19].

The generator serves the same purpose as the VAE (Variational Autoencoder) decoder [21]. The input of the generator is a vector extracted from the multivariate standard normal distribution. The generator freezes and trains the discriminator, and this process is repeated for thousands of epochs. The output image size is the same as the input image. DCGAN is optimal when using the Adam (Adaptive Moment estimation) optimizer and a learning rate of "0.002" [22].

DCGAN learns various representations from objects to scenes. In addition, the learned features can be used for other tasks to be applied as a general image representation. Raymond used DCGAN to fill in unwanted or missing parts of the image [23].

DCGAN is the first to use CNN as a generator and discriminator of GAN to improve performance. Currently, all GAN structures include a convolutional layer. Thus, GAN already implies the meaning of "DC".

3.2.2. BEGAN (Boundary Equilibrium Generative Adversarial Networks)

BEGAN (Boundary Equilibrium Generative Adversarial Networks) attracted attention for its discriminator being CAE (Convolutional Autoencoder) and the properties of convergence judgment that DCGAN does not have [24,25]. BEGAN learns the latent space of the image while maintaining and adjusting the balance between the generator and discriminator. BEGAN uses AE (Autoencoder) as discriminator, not as classifier. The discriminator learns that the real image has a small reconstruction error, and that the image generated by the generator has a large reconstruction error. The generator learns such that the reconstruction loss of the discriminator is small. Unlike DCGAN, BEGAN is capable of convergent adjudication.

The fundamental problem of GAN, mode collapse, also occurs in BEGAN. Accordingly, BEGAN-CS (BEGAN with Constrained Space), which has space limitations, was announced, but it did not solve the mode collapse [26].

Sung-Wook changed the structure of the BEGAN-CS discriminator from AE to VAE to solve the mode collapse and also changed the structure of the encoder and decoder [27]. The activation function changed from ELU (Exponential Linear Unit) to LReLU (Leaky Rectified Linear Unit) [28,29]. The KLD (Kullback–Leibler Divergence) term was added to the existing discriminator loss function [30]. The implementation model was able to solve the mode collapse, and the outline is shown in Figure 6.

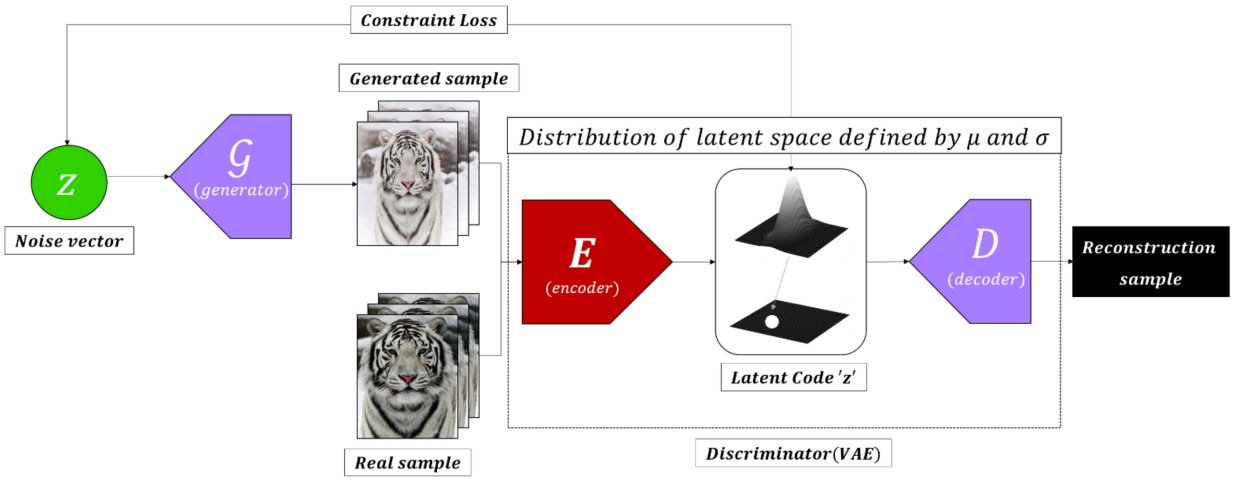


Figure 6. Structure and principle of BEGAN (Boundary Equilibrium Generative Adversarial Networks) v3.

3.2.3. PGGAN (Progressive Growing of Generative Adversarial Networks)

PGGAN is a model developed by NVIDIA to improve the speed and stability of GAN training [8]. PGGAN is a model that generates images of high quality and high resolution by adding a new layer during the training of the generator and discriminator. PGGAN

trains generators and discriminators from low-resolution images of 4× 4 pixels. Figure 7 shows the training process of PGGAN.

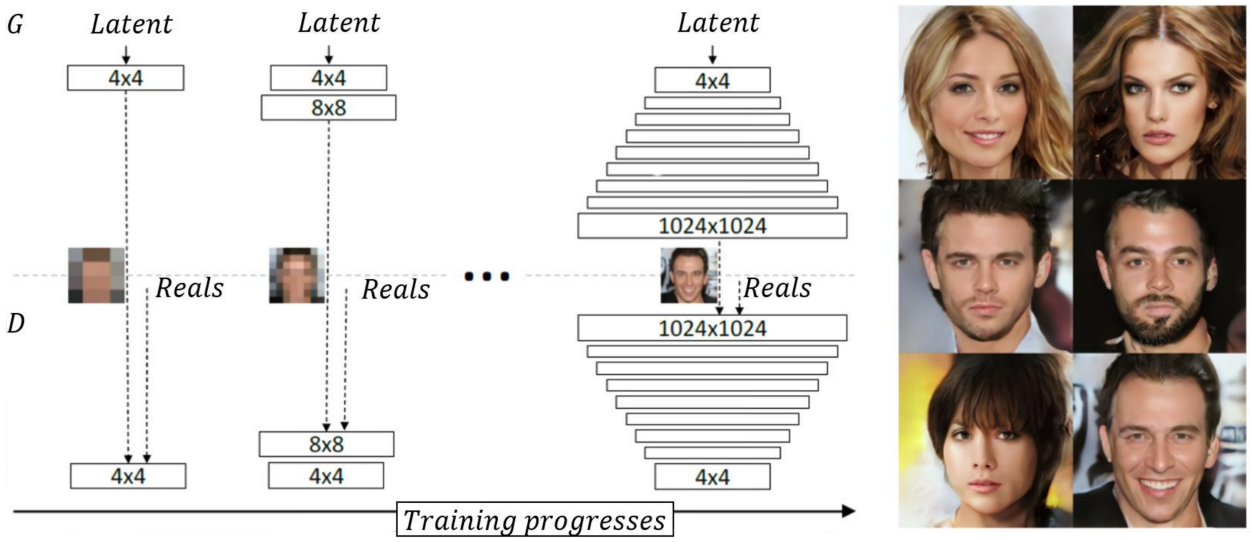


Figure 7. Training process of PGGAN (Progressive Growing of Generative Adversarial Networks) [7].

The added layer does not freeze but continues training. This algorithm was applied to the LSUN (large scale scene understanding) dataset image to obtain the results shown in Figure 8 [31].

Figure 8. 256 × 256 pixel images generated by PGGAN trained with LSUN (large scale scene understanding) datasets [8].

3.3. Condition

3.3.1. Info and Conditional GAN

The GAN model introduced so far has little or no control over the generated image. Info and Conditional GAN control the images to be generated [32,33]. InfoGAN (Information maximizing Generative Adversarial Networks) can control various properties of the generated image. It can apply the concept of information theory to predict the noise term for the output and convert it into a latent code in charge of control.

The generator of InfoGAN takes two inputs: latent space z and latent code c. The output of the generator is $G(Z, c)$. GAN is trained to maximize the mutual information between latent code c and generated image $G(Z, c)$. Figure 9 shows the structure of InfoGAN.

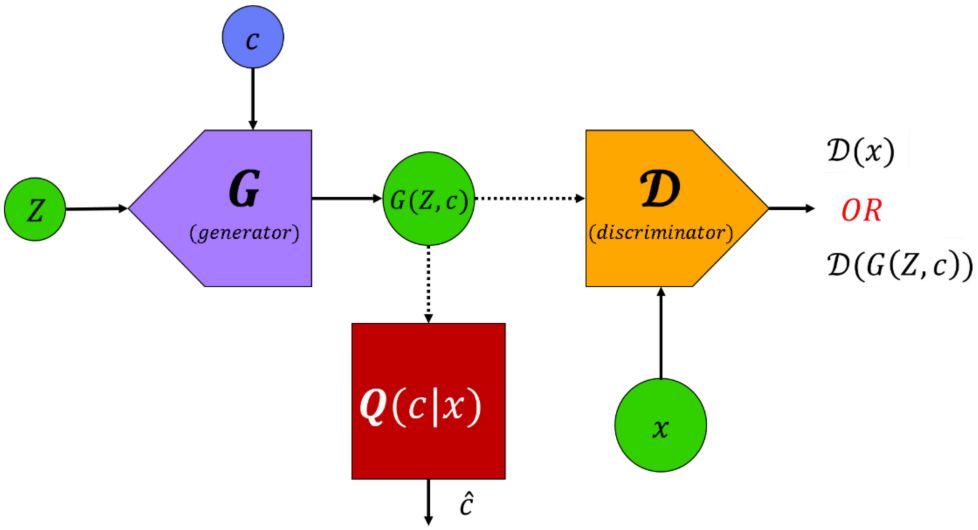


Figure 9. Structure and principle of InfoGAN (Information maximizing Generative Adversarial Networks).

Concatenated vector $G(Z, c)$ is entered as a generator. $Q(c|x)$ is also a neural network. Combined with the generator, it forms a mapping of random noise Z and latent code $\hat{c}$. For InfoGAN, the training goal is to estimate c for a given X. This is done by adding a normalization term to the objective function of GAN.

$$min_D max_G V_1(D, G) = V_G(D, G) - \lambda(c; G(Z, c)) \quad (4)$$

The term $V_G(D, G)$ in Equation (4) is the loss function of GAN, and the second term is the normalization term. λ is a constant, and the value is "1". $I(c; G(Z, c))$ is the mutual information between the latent code c and the generator's image $G(Z, c)$.

Mirza made generators and discriminators class-conditional and extended the GAN structure to conditional settings.

Figure 10 shows the structure of cGAN (Conditional Generative Adversarial Networks). cGAN performs conditional discrimination of real and synthetic images from discriminators, providing better representation than DCGAN in generating various data.

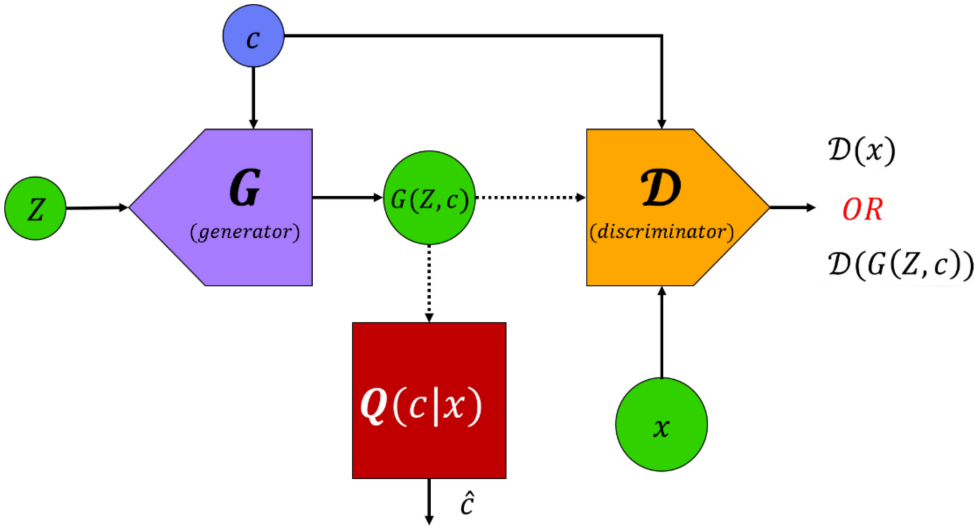


Figure 10. Structure and principle of cGAN (Conditional Generative Adversarial Networks).

3.3.2. Inference Model Based on GAN

Before the models introduced in this section appeared, GAN lacked inference capacity to map a given observation x to a latent space vector. Therefore, several techniques have been proposed to cultivate reasoning capability by reversing the G of the pretrained GAN [34,35].

ALI (Adversarially Learned Inference) and BiGAN (Bidirectional Generative Adversarial Networks) are inference models wherein D tests $data - latent$ pairs [36,37]. G consists of an inference network, an encoder, and a decoder. The input of D is (z, x) pair to identify whether it is a real image and its encoding or a synthetic image and its latent vector.

The quality of the synthetic image using ALI/BiGAN is not good, but it can be improved with additional cost for the sample and reconstruction distribution [38]. Figure 11 shows the structure of a GAN with an added inference network.

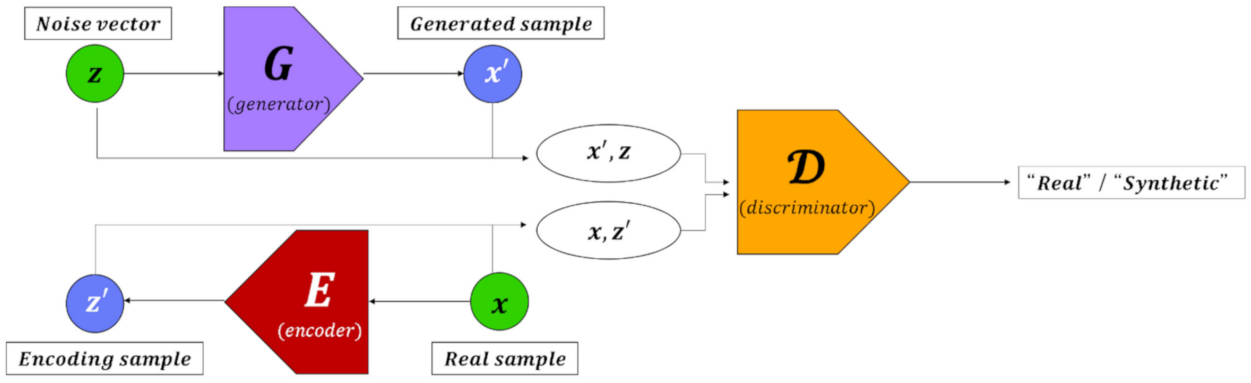


Figure 11. Structure of GAN with the addition of the inference network.

3.3.3. AAE (Adversarial Autoencoder)

As a network composed of encoders and decoders, AE learns how to map data to internal latent representations and re-output. In other words, it learns how to map from data space to latent space and vice versa. Reconstruction is outputted by these two mapping operations and is trained to be as close to the source as possible.

AE is reminiscent of a reconstructed filter bank widely used for image and signal processing. Note, however, that AE generally learns bidirectional nonlinear mapping. The filter bank is a repository of filters from which features can be extracted. The parameters of

the encoder and decoder are learned using backpropagation between the reconstructed image and the original.

Ancestral sampling can also be performed in AE [39]. Adversarial training may be applied between the latent space and the ideal distribution. The latent space GAN is similar to the VAE, which serves as the KLD term of the loss function [40]. KLD is a measure of how the two probability distributions differ. There have also been attempts to apply Variational Bayesian Methods (VAEs) to GAN. Mescheder integrates VAE through adversarial training in the form of AVB structures [41]. A similar algorithm was proposed in Ian Goodfellow's NIPS 2016 tutorial [42]. AVB tries to optimize targets such as VAE but uses adversarial training targets rather than KLD.

3.3.4. StarGAN

Image-to-image translation should ultimately have excellent quality of the generated image, and translation between various domains should be possible. In addition, images of various styles must be generated in each domain. The previously announced models for image-to-image translation were difficult to satisfy the aforementioned conditions as a single model. StarGAN is possible.

StarGAN has a limitation, i.e., the style translation is limited to the local area [43]. StarGANv2 has added multi-task discriminator AdaIN (Adaptive Instance Normalization), latent code z, mapping network F, R_1, and diversity regularization in StarGAN [44]. As a single generative model that generates images of various styles in multiple domains, StarGANv2 consists of four neural networks: Generator G, mapping network F, style encoder E, and discriminator D. The generator has style and domain as inputs. It uses multi-tasking mapping network, style encoder, and discriminator. A method for projecting an image into a style space was presented. The residual blocks in the network are all pre-activation structures [45]. The structures of G, F, E, and D are shown in Figure 12.

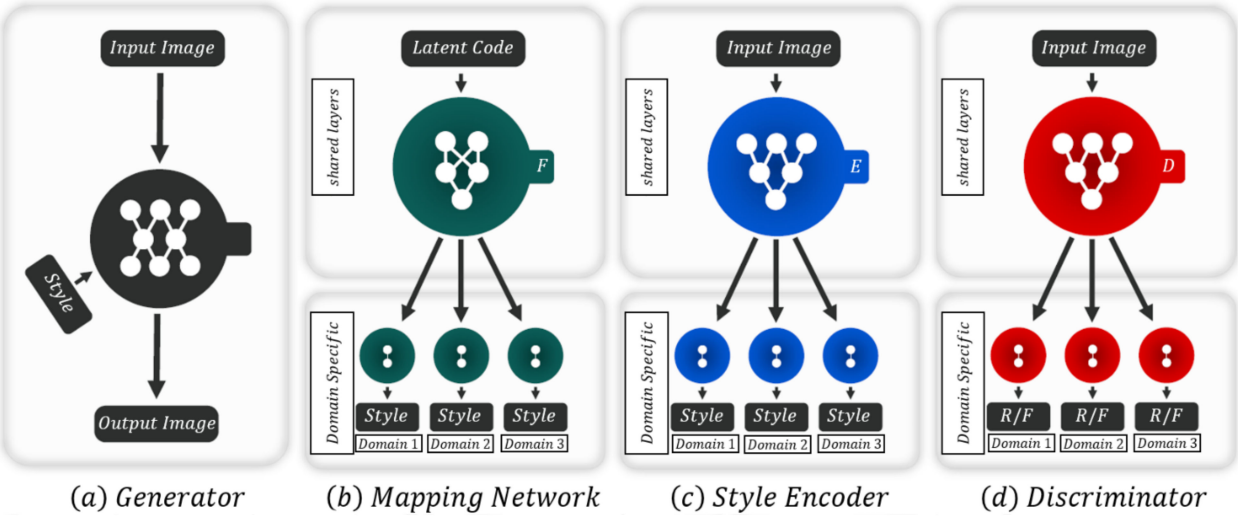


Figure 12. StarGANv2 with four neural networks [44].

E projects the image into the style space. The input is the image of the specific domain, and the output is the style code of the domain. F projects latent code z into the style space of a specific domain. The input is latent code z as the specific domain, and the output is the style code of the domain. G generates an image likely to be in the target domain. The input is the image corresponding to the content, the style code of the specific domain; the output is the image of the corresponding domain style. D distinguishes whether the input image is a real image or a synthetic one generated by the generator. The inputs are real or synthetic image, and the outputs are truly false in each domain. This algorithm was

applied to CelebA (large-scale Celeb faces' Attributes High Quality) and AFHQ (Animal Faces High Quality) dataset images to obtain the results shown in Figure 13 [46,47].

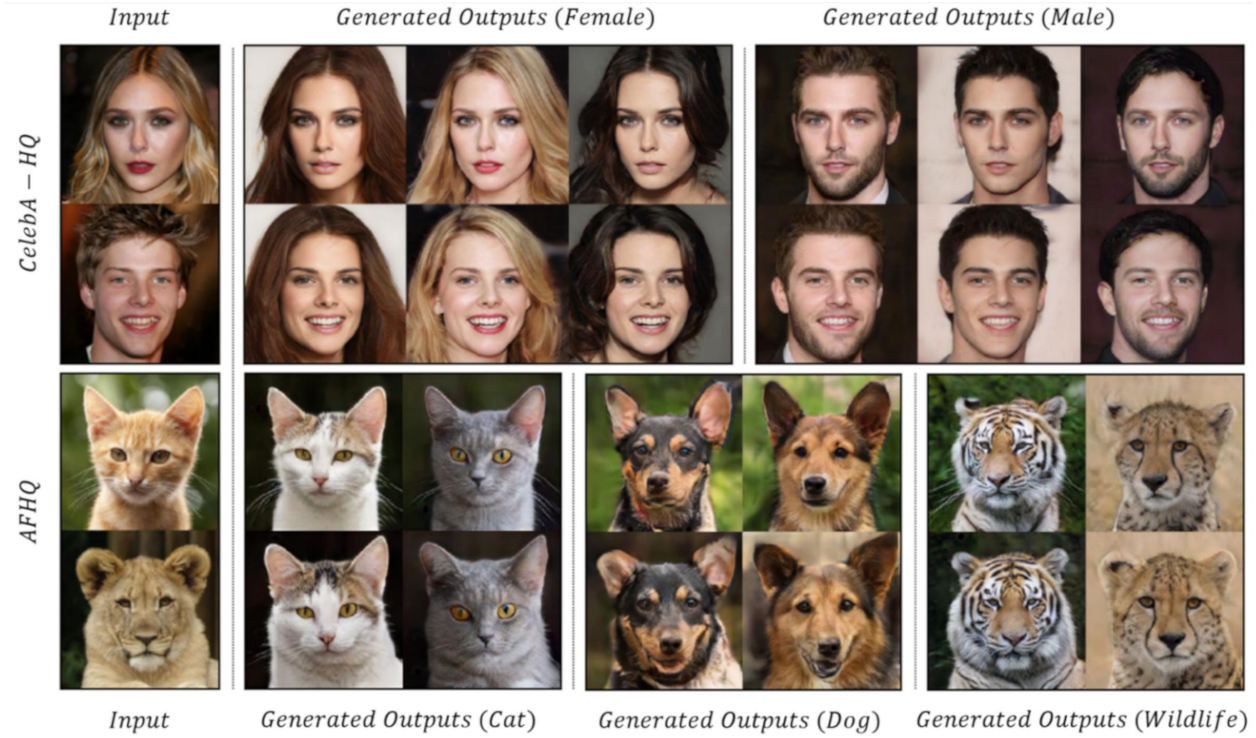


Figure 13. Image generated by StarGANv2 trained with CelebA (large-scale Celeb faces' Attributes High Quality), AFHQ (Animal Faces High Quality) datasets [44].

3.4. Mixing

3.4.1. BigGAN

BigGAN is an extended version of SAGAN, developed by DeepMind, and currently achieves the best performance in image generation trained with ImageNet datasets [10]. Figure 14 is an image generated by BigGAN trained with an ImageNet dataset [48].

Figure 14. Image generated by BigGAN trained with ImageNet dataset [10].

BigGAN currently has the best performance in generating images trained with ImageNet datasets, using $z \sim N(0,1)$ as the distribution of latent vectors when training. Truncated normal distribution is used for sampling. In other words, z smaller than a certain threshold is sampled. The smaller the cut threshold is, the less the diversity, but the greater the reliability of the resulting sample. Figure 15 shows this well.

Figure 15. Truncation technique of BigGAN [10].

BigGAN is as big as its name. Its deployment size is 2048, which is eight times larger than SAGAN. The channel size of each layer also increased by 50%. Shared embedding and orthogonal regularization have been added, and latent vector z is used for each layer of the generator as well as the first layer.

3.4.2. StyleGAN

StyleGAN uses a mixture of PGGAN and neural-style transfer technologies [8,11,49]. StyleGAN has been in the spotlight by creating full high definition-level results with several steps of control from the details of the image to the whole. Figure 16 shows the generator structure of StyleGAN.

A in Figure 16 is a fully connected layer. StyleGAN solved the problem of latent space entanglement by proposing a method called AdaIN, which uses reference style bias $y_{b,i}$ and scale $y_{s,i}$. $y_{b,i}$ and $y_{s,i}$ are used to adjust the mean and variance of feature map x_i outputted from the layers within the synthesis network. AdaIN is as shown in Equation (5).

$$\mathrm{AdaIN}(x_i, \mathrm{y}) = y_{s,i}\frac{x_i - \mu(x_i)}{\delta(x_i)} + y_{b,i} \tag{5}$$

To calculate the style parameters, latent vector z is passed through mapping network f, with intermediate vector ω generated. It then passes through the fully connected layer to generate $y_{b,i}$ and $y_{s,i}$ vectors of length n. This is to separate the style selection process of the image. The AdaIN layer prevents style information from leaking between layers. The style vector injected into each layer makes it affect only the features of that layer. This latent vector ω is better than the original z vector.

The synthesis network is based on the PGGAN structure, and the style vector of the front layer of the synthesis network affects features larger than the style vector of the back layer. StyleGAN had full control over the image generated using latent vector ω and changed the style to various levels by changing the position of the ω vector in the synthesis network.

A and B in Figure 17 were generated with different ω vectors. To merge the two images, A's ω vector is passed through the synthesis network and is converted at a certain point into B's ω vector. When deformation occurs early, styles such as posture, appearance, and glasses are transferred to A. When deformation occurs later, styles such as color and fine shape of the face are transferred to A. Both features of the A image are maintained.

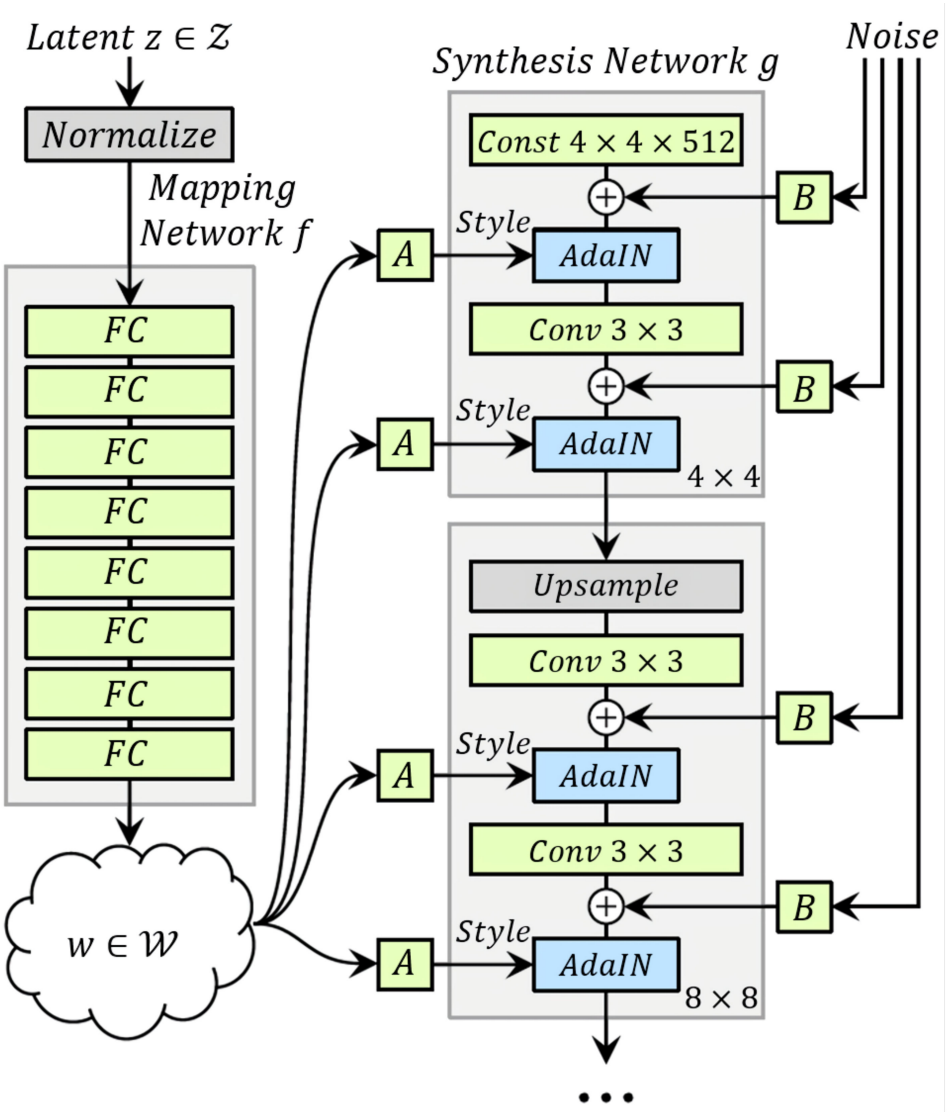


Figure 16. Generator structure of StyleGAN [11].

Finally, the StyleGAN structure adds noise behind each convolutional layer to capture areas such as hair position or face background. The noise injection location determines the fineness and roughness in the image.

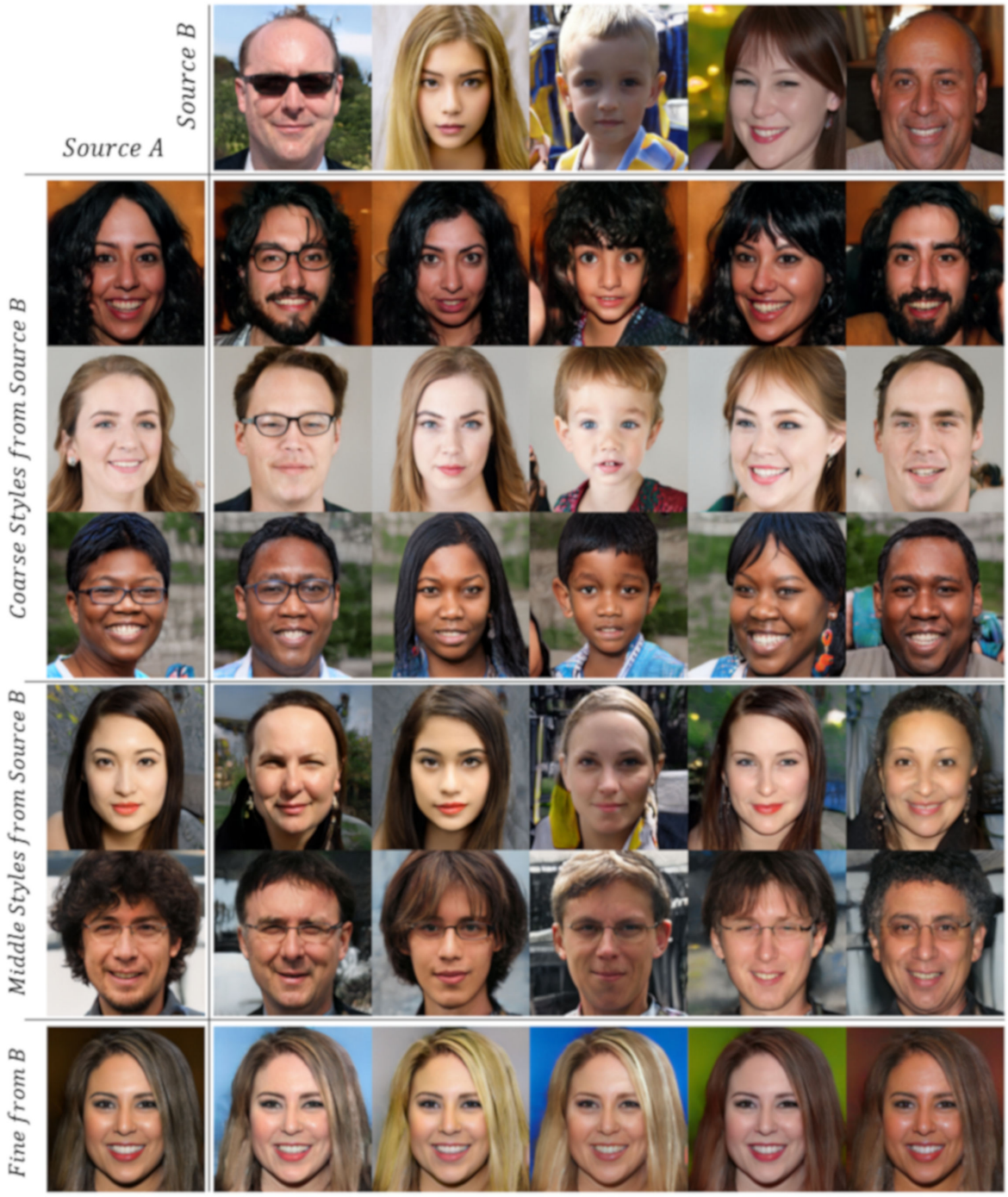


Figure 17. Result of style synthesis in two generated images [11].

4. Theory Analysis of GAN

In GAN's training, *D* explores parameters that maximize classification accuracy, and *G* explores those that can confuse *D* as much as possible. Figure 18 shows the process of exploring parameters for generators and discriminators.

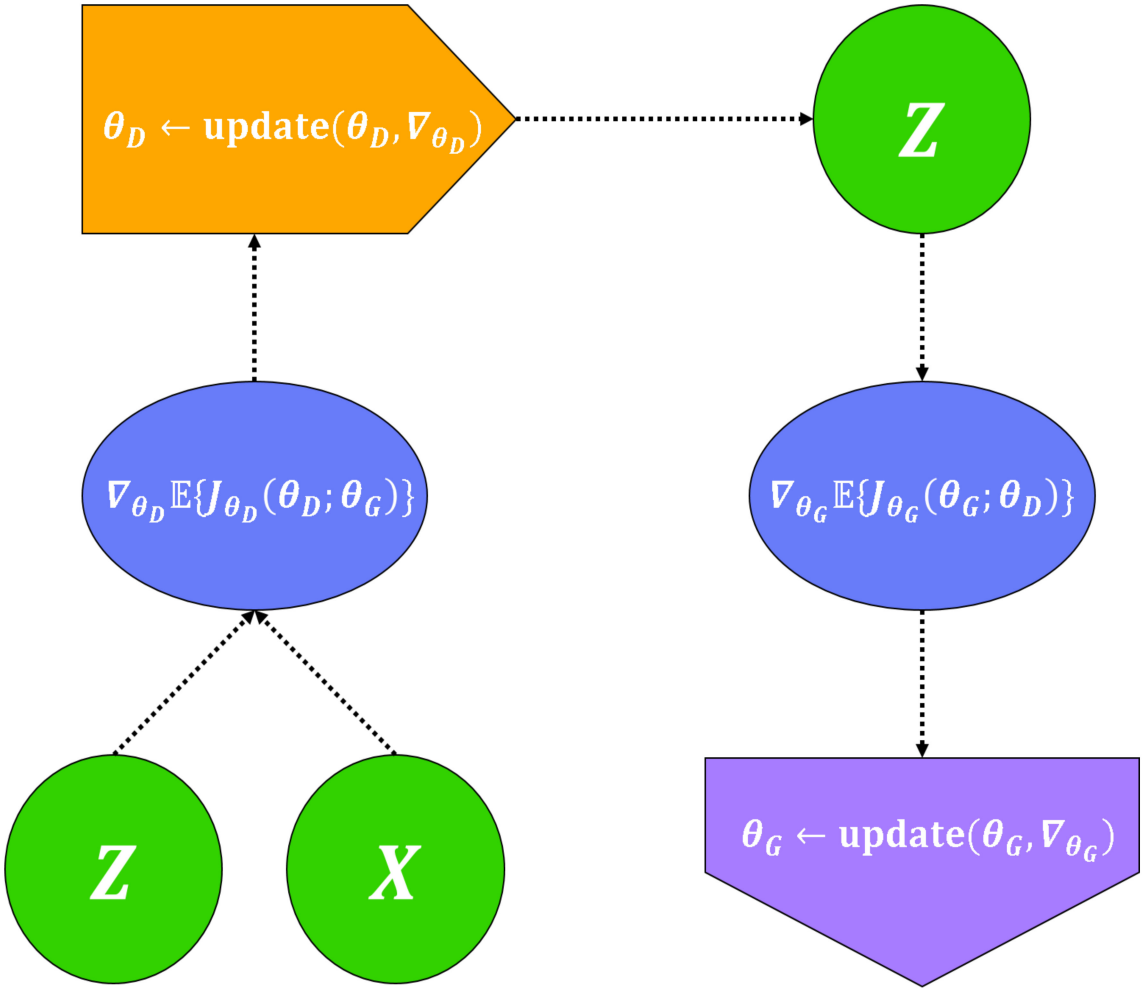


Figure 18. Process of navigating parameters for generators and discriminators.

The training cost is evaluated using the $V(G, D)$ function as shown in Equation (6).

$$V(G, D) = E_{p_{data}(x)} log D(x) + E_{p_G(x)} \log(1 - D(x)) \tag{6}$$

When the parameters of one network are updated, those of the other network are fixed. Goodfellow presented the optimal D as Equation (7) when G is fixed.

$$D^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_G(x)} \tag{7}$$

Optimal G is when $p_G(x) = p_{data}(x) \rightarrow D$ predicts 0.5 for all samples x. In other words, G is optimal when D cannot distinguish between a real image and a synthetic one. The ideal situation is when D is trained until G is optimal, and then G is updated again. Note that it may not be trained until D is optimal, and the parameters of G and D are updated at the same time.

In general, training criteria other than saturation for G using $\max_G log(1 - D(G(z)))$ instead of $\min_G log(1 - D(G(z)))$ are used. Despite these methods, GAN training is unstable due to the difficulty in converging a pair of models, and collapsing occurs wherein only a few data are generated for different input data. In addition, D's loss converges rapidly to "0" and often does not give good feedback to [20,50,51].

Goodfellow and Salimans explained that GAN training is unstable because GD (Gradient Descent) is an inadequate solution to form GAN's saddle point [2,50]. This does not mean that GD is not good. As a low-dimensional space inherent in a high-dimensional

space, Manifold exists in a high-dimensional space that can represent real data. When the color image sample is $N \times N \times 3$ size of $[0, R^+]^3$, the space that can be called X is a $3N^2$ dimension. Each dimension is "0" and the maximum measurable pixel. Note that p_{data} constitutes a manifold of real data related to a particular problem and occupies a small portion of total space X. Likewise, the sample generated in G should occupy only a small portion of X.

Arjovsky showed that the dimensional space of $P_G(x)$ and $p_{data}(x)$ was lower than that of X [51]. $P_G(x)$ and $p_{data}(x)$ may not overlap, so D can classify real image $x \sim p_{data}(x)$ and synthetic image $x \sim p_G(x)$ with 100% accuracy. In this case, the error of D converges to zero quickly. On one side, the parameter of G is updated only through D, and the gradient used when updating the parameter of G also converges to "0". Therefore, it may no longer be useful for updating G.

Goodfellow proved that, when D is optimal, G is equivalent to minimizing JSD (Jensen–Shannon Divergence) between $P_G(x)$ and $p_{data}(x)$ [2]. JSD is a measure of the similarity between two probability distributions devised by Jensen and Shannon. If A is not optimal, the update may not be meaningful or accurate. Such insight led to the study of alternative distance-based cost functions.

4.1. Training Methods

Radford's DCGAN designed G and D similar to the existing CNN structure [20]. At the heart of DCGAN are stride convolution and transpose convolution [52]. This greatly contributed to improving the quality of the image synthesis work. To stabilize deep model training, both G and D proposed batch normalization. Furthermore, the fully connected layer was minimized. It was also shown that it is better to use LReLU instead of ReLU (Rectified Linear Unit) as an activation function [53].

Salimans proposed a heuristic solution to stabilize GAN's training [50]. First is to change G's training goals to increase the amount of information available. G is trained to match features that are intermediate predictions of the real image and the synthetic image outputted from D. Second is to add a minibatch discrimination input function to D. This function encodes the distance between an input sample and another sample in minibatch units. In addition, it is easy to know whether D and G have the same output, which is effective for preventing mode collapse. Third is the heuristic average, which is to assign a penalty to the parameter when the moving average of the parameters in the network exceeds the previous average. This helps G and D maintain equilibrium. Fourth is virtual batch normalization, which calculates batch statistics using samples of minibatch size at the start of training and continues to refer to them. This method reduces the dependence of specific samples. The final one-sided label smoothing is to smooth D's classification criteria. In this method, the intensity of the gradient transmitted to G is adjusted to generate a better image.

Sonderby added noise before entering data into D [54]. One-sided label smoothing is done because D can be biased in a specific direction. Instance noise narrows the manifold of real and synthetic images. At the same time, it prevents A from easily finding the boundary separating the real image from the synthetic one. This method is implemented by adding Gaussian noise to both real image and synthetic image and annealing standard deviation with time. As such, studies have been actively conducted to add noise to the sample for the stabilization of GAN training.

4.2. Alternative Formulations

This chapter describes the interpretation and overview of GAN's information theory, including the cost function to improve the vanishing gradient. For the information theory interpretation and overview, Nowozin showed that it can be generalized to minimize f-divergence other than JSD [55]. f-divergence includes a divergence scale similar to KLD, and it is approximated by applying fenchel conjugates to the generated sample. The fenchel conjugate is a pair function devised by fenchel, which is not a new algorithm but

is meaningful since it is applied to GAN training. f means convex. Depending on the choice of f-divergence, it provides a list of activation functions available in the fenchel conjugate and the last layer of G. When the object of G was maximized in the experiment, the gradient became faint. At the beginning of the training, a cost function was proposed to update G, which is less likely to saturate. In G, only the prediction of f-divergence is minimized. Uehara expanded f-GAN, and Goodfellow also offered an alternative to JSD [42,56].

Weight clipping reduces the capacity of D to force simpler feature learning [15]. Gulrajani proposed an improved training method in WGAN by penalizing the D gradient [17].

4.3. Disentangled Representation

Interpretable disentangled representation has been studied for a long time [57]. The generative model aims to capture the generative factors of the training data. Disentangled representation is associated with symmetry transformations, wherein any property is changed but other properties are preserved. Symmetry transformation transforms certain properties but preserves others. In order to achieve symmetry transformation in neural networks, neurons must have no connection to other neurons. In other words, each neuron is isolated. Symmetry is often used in quantum mechanics and is more comprehensive than the symmetry mentioned in geometry.

Disentangled representation is the process of learning symmetry, disentangling through training even if it starts from the fully connected layer. This is because latent units are sensitive to changes in generative factors. In terms of information theory, disentangled representation is highly useful. Because it compresses the information, it is more efficient than other algorithms, and it can increase the number of things to a lot. Disentangled representation is only effective for latent vectors.

InfoGAN achieved better disentanglement by maximizing the same index code mutual information. AAE achieved better disentanglement by minimizing KLD due to adversarial losses [32,41].

4.4. Variants of GAN

GAN is difficult to train due to partial gradient loss, thereby requiring careful hyperparameter tuning, but AAE and WGAN are less affected by these factors.

First, AAE is easy to train because adversarial loss applies to low-dimensional, simple distributions. Second, WGAN was designed not to suffer from vanishing gradient [17]. In other words, it is designed to be less sensitive to nonlinear selection without batch normalization. cGAN synthesizes an image with user-specified contents. Vanilla GAN constructs a significant latent space but does not provide an inference model to map samples to latent representation.

Both BiGAN and ALI are algorithms that map images to latent spaces [38] has improved the image quality of ALI than before. This model is an AE similar to VAE, normalizing the latent space by performing adversarial training between pre-encoding and post-encoding samples.

4.5. Latent Space of GAN

GAN learns its own representation through training and generates a structured vector space in various domains. This algorithm is similar to other neural networks including VAE and language models, such as Word2Vec, an algorithm that evaluates word similarities and converts them into vectors [58].

Generally, the domain of the data to be modeled is mapped to a vector space having a dimension lower than such data space. The latent space is generated in G and can be structured, supporting significant operations [20]. For example, if it is a face, mix gender, age, smile, hair, etc.

In many GAN models is an "encoder" that maps back from the space to be modeled to the latent space [36,37]. This is because it is effective in exploring and utilizing GAN's

structured latent space. Using an encoder, it is possible to generate a high-level concept vector by mapping and analyzing a labeled image into a latent space. The concept vector can be applied to the offset in the latent space, affecting *G*.

Gurumurthy proposed a method for modeling latent spaces as a Gaussian distribution and learning mixed components [59]. Mixed components are factors that maximize the potential of the generated data.

5. Applications of GAN

GAN can be quantitatively evaluated for features extracted from unsupervised learning, so it can be applied to image classification. This is the case wherein there are constraints on the generated images, and the condition is how the training object should be achieved. Better super-resolution is possible by adding adversarial losses to the existing approach [60–62]. It can also be applied to image-to-image, which automatically translates an input image to an output image.

5.1. Classification and Regression

The trained GAN model can be used for other downstream tasks. Downstream is data transmitted from the upper medium to the lower medium. For example, the convolutional layer output of the discriminator can be used as a feature extractor, and a linear model, such as SVM (Support Vector Machine), can be combined with it [20,50]. This is a structure wherein a feature vector that has passed through a feature extractor uses a classifier, such as SVM, as new input data. Radford achieved excellent classification performance when this method was applied to all of supervised learning, unsupervised learning, and non-trained datasets [20].

Like ALI, adversarial training can improve image quality when learning inference algorithms simultaneously [36]. The representation vector generated in the last three hidden layers of the ALI encoder records a lower misclassification ratio than DCGAN [35]. Higher performance was achieved when label information was added to the ALI.

With less labeled training data, GAN can be used to generate more training data. Shrivastava improved the synthetic image while maintaining annotation information, achieving state-of-the-art performance in posture and gaze estimation with synthetic images only [63]. Spatiotemporal GAN also reported good results for gaze estimation and prediction [64]. When a model trained as a synthetic image is applied to a real image, on the other hand, it does not always show good results [65].

Bousmalis proposed a method to match the synthetic image of the original domain with the target domain [65]. To synthesize images from different domains, Liu suggested using multiple GANs with combined weights [66].

5.2. Synthesis and Inpainting

A significant part of the recent GAN research topic is to improve the quality and usability of the generated images. LAPGAN (Laplacian Pyramid of Generative Adversarial Networks) introduces a cascade of CNN to generate images schematically [67]. Cascade, once initiated, involves a series of stages each of which is triggered by the previous stage and the results are continued until the end. LAPGGAN extends cGAN wherein generators and discriminators receive additional label information as input. The algorithm has since been extended to solve the problem of natural language processing. The cGAN algorithm was proposed by Huang to work on medium representations rather than low-resolution images [68].

Reed used GAN to synthesize the image with text description [69]. For example, if a text description such as "a bird with black head, orange wings, and white beak" is the input to the network, GAN generates a plausible image.

In the GAWWN(Generative Adversarial What-Where Network) of Figure 19, the position of the image is determined according to the conditions [70]. GAWWN supports an

interactive user interface that allows gradually drawing large images with a description of the object and a bounding box.

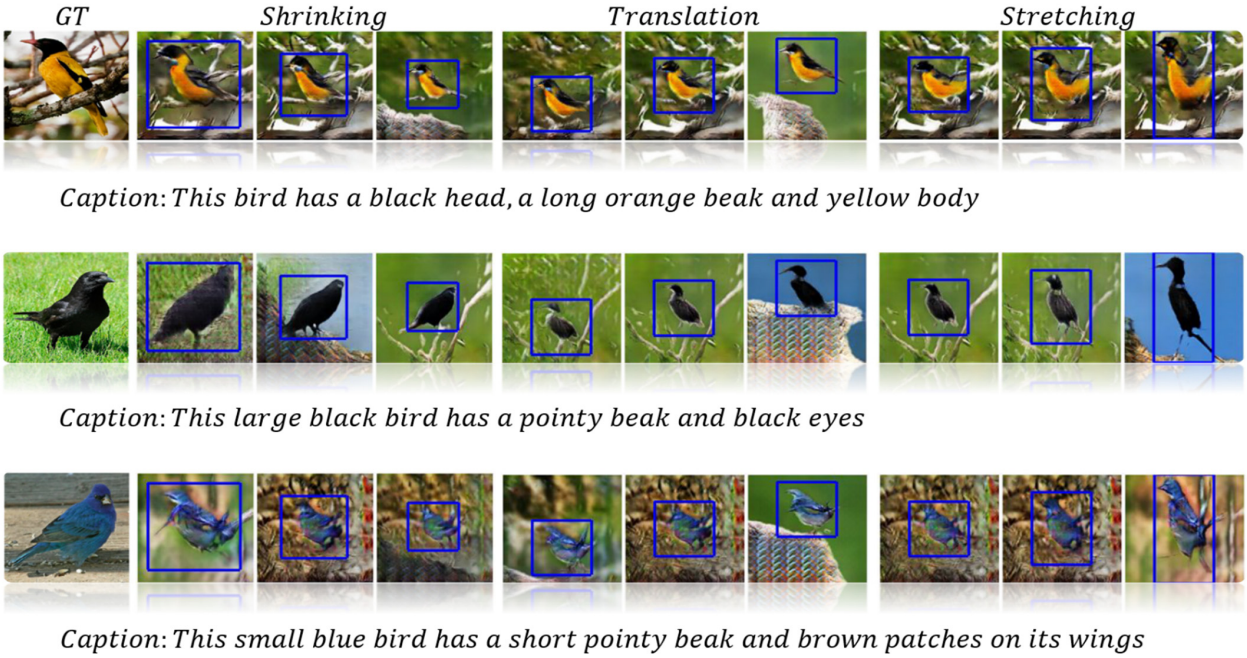


Figure 19. Controlling the bird's position using bounding box coordinates and previously unseen text [70].

cGAN can synthesize new images with specific attributes and intuitively edit images such as changing the hair shape, wearing glasses, and reducing age [58,71,72].

Yurt used GAN to synthesize scarce medical images [73]. The multi-contrast MRI (Magnetic Resonance Imaging) protocol raises the level of morphological discrimination information that can be used for diagnosis. The number and quality of contrasts are limited by several factors such as scan time and patient movement. mustGAN (multi-stream Generative Adversarial Networks) synthesizes insufficient or impaired contrast to alleviate restrictions and improve clinical usefulness. Contrast can be synthesized by one-to-one or many-to-one method. The many-to-one method receives multiple raw images and learns shared latent representation that are more sensitive to common features. mustGAN proposes a multi-stream method that integrates information from multiple raw images. The shared feature map and complementary feature map generated from the stream are combined with the fusion block. The location of the fusion block is adaptively modified to maximize performance for each task. The structure of mustGAN is shown in Figure 20.

In Figure 20, mustGAN's generator first consists of K one-to-one and many-to-one streams. After that, it consists of adaptively deployed fusion blocks and a joint network for recovery. The one-to-one stream independently generates a unique feature map of each source image, and the many-to-one stream generates a shared feature map over the entire source image. The fusion block concatenates the feature map generated in the fusion layer. The joint network synthesizes the target image from the fused feature map. The joint network is classified into early, mid-term, and late-stage, according to the location of fusion.

Figure 20. Structure and principle of mustGAN (multi-stream Generative Adversarial Networks) [73].

Previous works on GAN-based image inpainting focused only on performance improvement and did not consider diversity with a lower priority [23]. Therefore, the diversity is poor. Diversity is especially important in image inpainting. Cai proposed PiiGAN (Pluralistic image inpainting Generative Adversarial Networks), which extracts style vectors from ground truth [74]. Style vectors are latent vectors. PiiGAN has a separate style extractor and a loss of consistency that makes the image to be generated approximate to the ground truth. The style vector and ground truth extracted from the style extractor are input to the generator. Consistency loss allows the generator to learn style mapping corresponding to multiple vector sets. PiiGAN has generated a diversity of realistic images with high quality that match the high-level semantic context of the ground truth images.

5.3. Image-to-Image Translation

The pix2pix model is a technique proposed in 2016 that uses GAN for image translation, not for image generation [75]. pix2pix is a model that maps the input image and the output image and shows very high accuracy. Therefore, the training image also has a pair of inputs and outputs. This model showed excellent results in a variety of computer

vision problems, such as semantic segmentation, map generation in aerial photography, and colorization of black and white images.

pix2pix is supervised learning that requires a pair of input and output images during training. Cycle-consistency Generative Adversarial Networks (CycleGAN) has a greater meaning than anything else in that it can perform image translation without a pair of input and output images [76]. Once trained, images can be translated from one area to another. For example, when training a horse and zebra dataset, CycleGAN can translate only the horse into a zebra without leaving the background when an image with a horse in the foreground is provided.

CycleGAN used two GANs. The generator of each GAN performs image translation from one region to another. If X is an input, the generator of the first GAN performs mapping $G : X \rightarrow Y$. Therefore, the output is $Y = G(X)$. The generator of the second GAN performs reverse mapping $F : Y \rightarrow X$ to become $X = F(Y)$. Each discriminator is trained to distinguish between a real image and a synthetic one. The algorithm is shown in Figure 21.

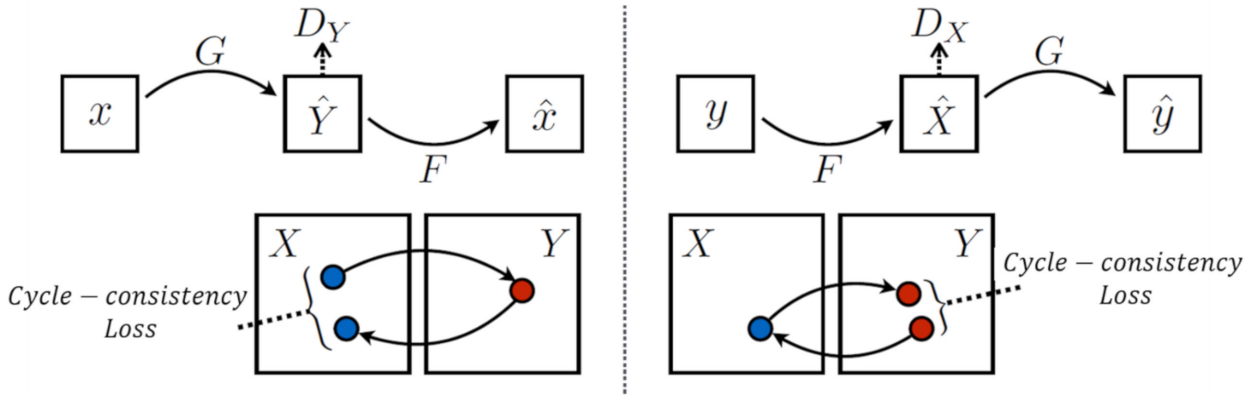


Figure 21. Cycle-consistency loss of CycleGAN (Cycle-consistency Generative Adversarial Networks) [76].

To train the combined GAN, we added forward cycle-consistency loss and reverse cycle-consistency loss. This means that the image obtained after two translations $F(G(X)) \sim X$ for image input X is equal to X. Figure 22 is an example of an image translated by CycleGAN.

CycleGAN translate images between predefined domain sets. CSGAN (Cyclic-Synthesized Generative Adversarial Networks) can also translate images between completely distant domain sets [77]. CSGAN uses cyclic-synthesized loss between synthetic and cyclic images of different domains. The loss of cyclic synthesis improves the quality of the image to be generated by minimizing side effects, such as artifact. Figure 23 is an example of an image translated by CSGAN.

In Figure 23, rows 1 to 2 are samples of CUHK (the Chinese University of Hong Kong) datasets, and rows 3 to 4 are samples of FACADES datasets [78,79]. The first to second columns are the input image and the ground truth image, respectively. Columns 3 to 5 are images translated using DualGAN, CycleGAN, and CSGAN, respectively [76,77,80]. The artifacts of DualGAN and CycleGAN are indicated by red squares in the second and third columns, respectively.

Scott used the pix2pix model to work on the human body [81]. The model was trained with color photographs of dancers and black-and-white images that revealed boundaries. Scott created a new form by generating a line drawing. Again, the model can transform it into a color photo space. Models make pictures similar to possible people. Figure 24 shows Scott's work.

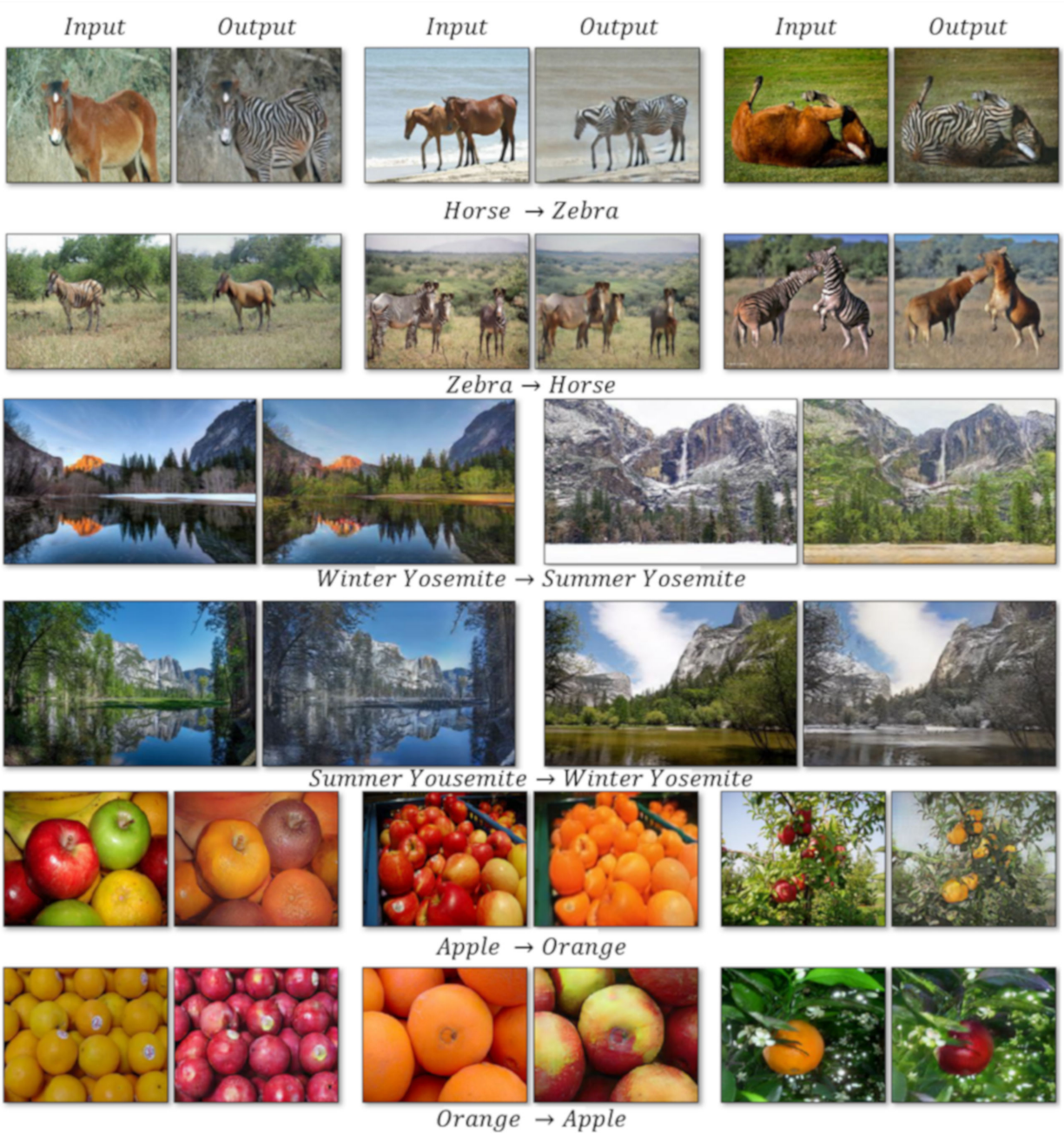


Figure 22. Example of image translation in CycleGAN (Cycle-consistency Generative Adversarial Networks) [76].

Taeyoung succeeded in restoring the solar-side magnetic field image with image-to-image technology based on cGAN [82]. Solar magnetic field images are important sensing data for solar activity and space weather forecasting. The situation of the rotating sun is useful in terms of forecasting. An image of the magnetic field in front of the sun as viewed from Earth can be acquired by SDO (Solar Dynamics Observatory)'s HMI (Helioseismic and Magnetic Imager) sensor. Still, it is difficult to acquire an image because there is no corresponding sensor in the stereo observatory that observes the side of the sun.

SDO's AIA (Atmospheric Imaging Assembly) sensor image and HMI image were trained in pairs on the GAN. SDO is a satellite observing the front of the sun, and HMI is a magnetic field sensor. Subsequently, stereo EUVI (Extreme Ultra Violet Imager) sensor images were inputted as conditions. EUVI is a solar-side observation satellite with the same characteristics as an AIA sensor.

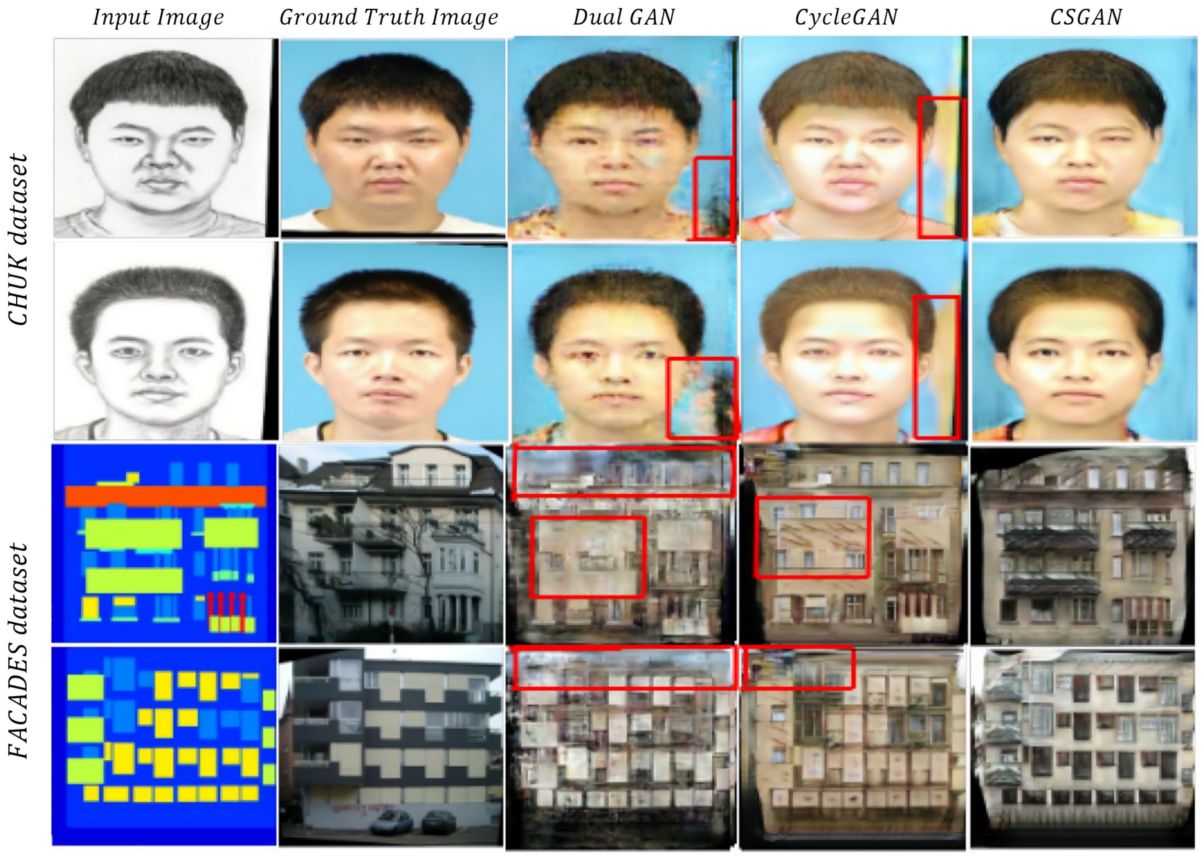


Figure 23. Example of image translation in CSGAN (Cyclic-Synthesized Generative Adversarial Networks) [77].

In Figure 25, a is an extreme ultraviolet image observed by a satellite, c is a magnetic field image observed by a satellite, and b is a side magnetic field image generated by a GAN. a and c were obtained every 3 days. As a result of analyzing the front magnetic field image and the magnetic field image generated by GAN at various time periods, the sunspot is confirmed to have been reproduced properly.

5.4. Super-Resolution

SRGAN generates high-resolution images from low-resolution ones [60]. The SRGAN structure consists of three neural networks: generator, discriminator, and pretrained VGG-16 (Visual Geometry Group) neural network using the residual module [83,84]. SRGAN uses a perceptual loss function.

The difference in feature map activity at the upper layer of the VGG network between the network output and the high-resolution portion becomes a perceptual loss function. In addition to perceptual loss, the authors added content loss and adversarial loss to make the generated image more natural and more artistic in detail. Perceptual loss is defined as the weighted sum of content loss and adversarial loss, as shown in Equation (8).

$$l^{SR} = l_X^{SR} + 10^{-3} \times l_{Gen}^{SR} \tag{8}$$

The first term on the right-hand side is content loss obtained using a feature map generated by pretrained VGG-16. This is the Euclidean distance between the feature map of the mathematically reconstructed image and the original high-resolution reference image. The second term on the right-hand side is adversarial losses, designed to allow the generator-generated image to fool the discriminator. Figure 26 shows that the image generated by SRGAN is much closer to the original high-resolution image.

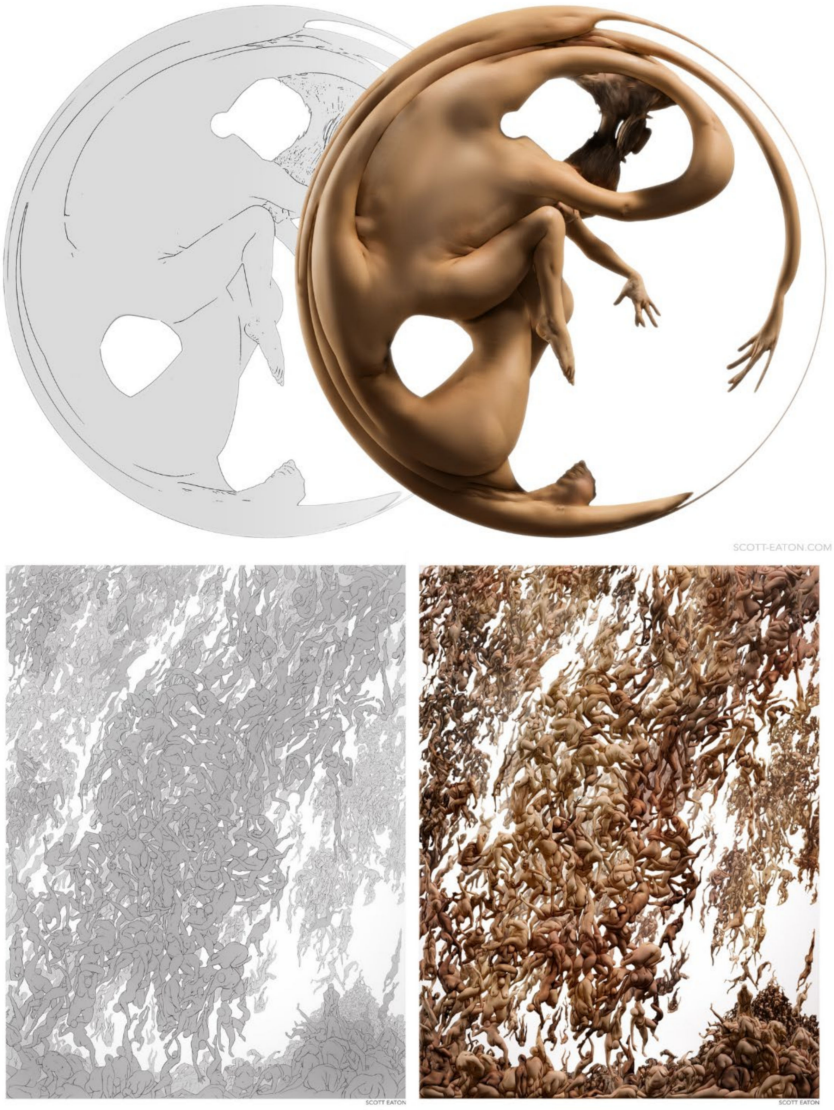


Figure 24. Artist, Creative AI (Artificial Intelligence) [81].

5.5. Point Registration

GAN is a learning-based technology that has also recently been applied to point registration. Point registration is mainly used in the fields of computer vision, pattern recognition, and robotics, and is also called point cloud registration or scan matching. Point registration is the process of finding a spatial transformation that can align two point clouds.

Mahapatra used GAN to register medical images [85]. In the paper, CAE encodes a given pair of input images into latent space vectors. The encoded latent space vector was used to generate the matched image with the help of GAN. This feature transformation guarantees greater constancy to the input image type. As a result, GAN contributed to show better performance than the existing method in experiments on Chest X-ray, Retinal, and Brain Magnetic Resonance images.

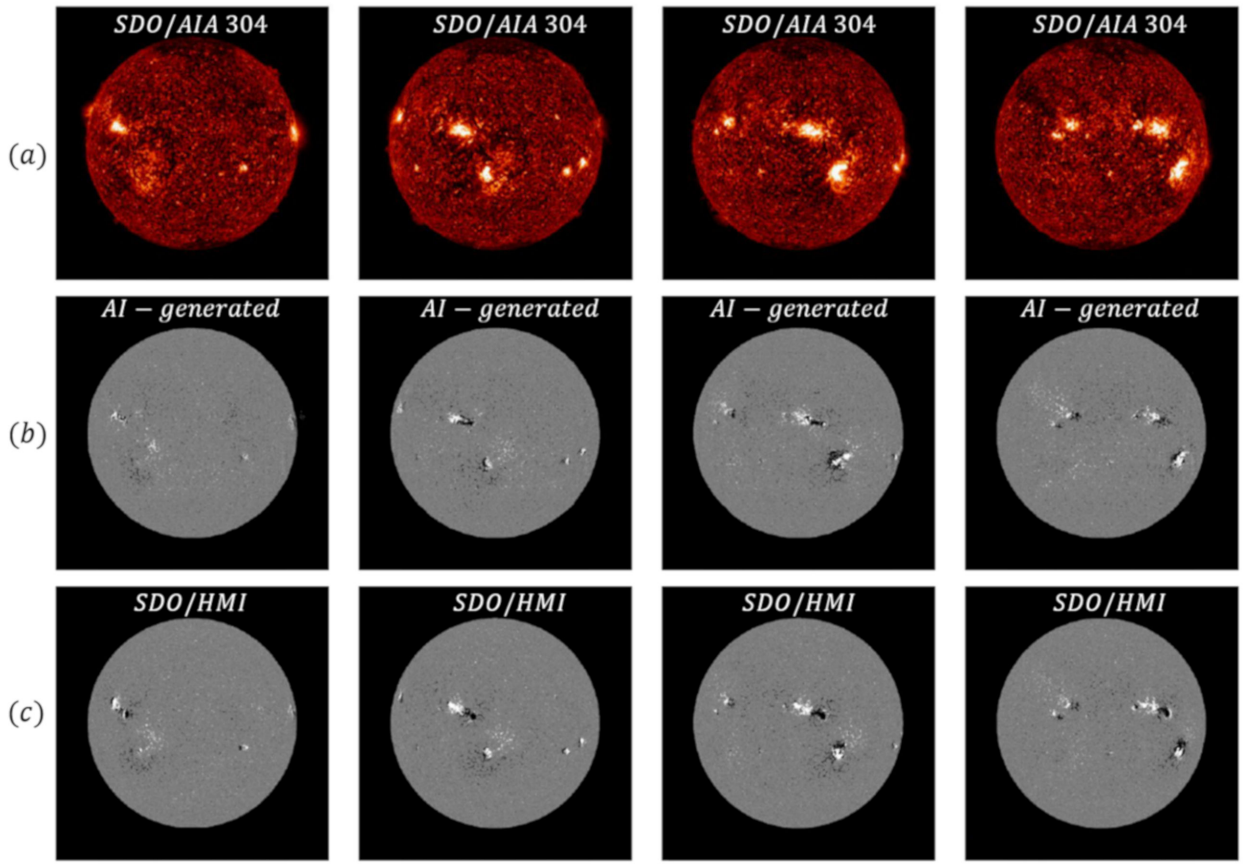


Figure 25. A lateral magnetic field image of the sun restored to the GAN [82]. (**a**) is an extreme ultraviolet image observed by a satellite, (**c**) is a magnetic field image observed by a satellite, and (**b**) is a side magnetic field image generated by a GAN.

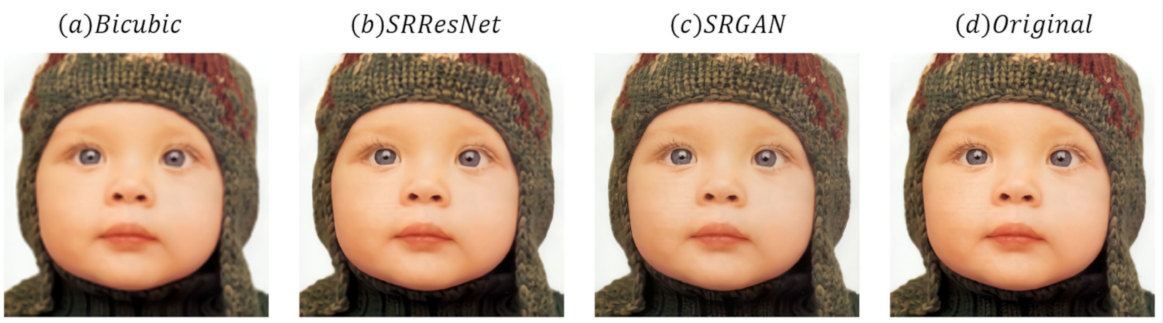


Figure 26. Comparison of super-resolution performance between SRGAN (Super Resolution Generative Adversarial Networks) and other models [60].

6. Discussion

Several major advances in recent years have greatly improved the overall stability of the GAN model. Some challenges remain in GAN.

6.1. Mode Collapse

First is mode collapse, which occurs when the generator finds a small number of samples that deceive the discriminator. Therefore, no other sample can be generated apart from this limited sample. Generators tend to find one sample that deceives the discriminator, i.e., a mode, and they can map any point in the latent space to this sample. This means that the gradient of the loss function collapses to a value close to zero. Even if the discriminator is trained again to avoid being tricked into a single point, the generator

will easily find another mode to deceive the discriminator. This is because the generator is already insensitive to input, and there is no reason to generate various outputs [51,86].

One of the mode collapse mitigation methods is to change the distance measurement algorithm used for statistical distribution comparison. Arjovsky suggests Wasserstein distance rather than DCGAN's KLD and EBGAN (Energy-Based Generative Adversarial Networks)'s TV (Total Variance) distances [15,20,87]. Metz updated A by applying minimax to the accumulated loss values in several steps [88]. G knows how D will update, and it can generate a better image.

6.2. Training Instability

Second is training instability. The hessian of the GAN loss function is infinite. Therefore, the optimal solution is to find a saddle point rather than the local minimum. In the case of the saddle point, the loss rate of the discriminator is high, and that of the generator is formed at the low optimal point. Many optimizers rely on the first derivative of the loss function. This is because initialization is important to find the saddle point.

Lee argues that the nonlinear manifold theorem makes it difficult for GD to reach the target, and that the initial algorithm should be randomly selected [89,90].

Mescheder claims that GAN's convergence is difficult because of the Jacobian matrix error and the eigenvalue of the imaginary part [55]. A second-order optimizer like the newton method is promising, but it needs to be expanded by a second- or a third-order formula depending on the parameter dimension.

Arora uses a nonparametric Bayesian to associate a finite mixture of neural networks with an equilibrium [86]. This means that there is no equilibrium when the network is below a certain capacity. Even if the GAN's training appears to converge, true learning distribution may still be far from the target distribution. To improve this problem, Arora proposed a new measurement method called neural network distance [86].

The optimal distance measurement method depends on the solution space. In order to solve the instability of training, we propose the EM (Earth Mover) distance as a method for measuring the distance between the real and synthetic sample distributions in the following situations: (1) There are two parallel lines, the probability distribution P and Q. (2) One straight line is fixed to $x = 0$, and the other straight line can move along the x axis at $x = \theta$. (3) $\theta > 0$. KL, TV, and JS distance are each $KL(P \parallel Q) = +\infty$, $TV(P, Q) = 1$, $JS(P, Q) = log2$. None of these distance measurements are a function of parameter θ. Therefore, the probability distributions P and Q cannot be differentiated with respect to θ so that they are similar to each other. On the other hand, the EM distance is $EM(P, Q) = |\theta|$. Therefore, there is a slope for θ, and we can move Q toward P. As a result, it can be differentiated without overlap of the two distributions. For this reason, EM distance can improve training performance.

6.3. Evaluation Matrixs

Evaluation matrixs, such as how to evaluate the accuracy of the generated image and whether the designed model can be compared with other models, can vary depending on the training purpose [56]. In GAN, most of the objective functions of generators and discriminators are measured by comparing how well each plays a role. For example, a specific objective function measures the degree to which a generator deceives a discriminator. IS (Inception Score) and FID (Frechet Inception Distance) are methods of comparing the results of various GAN models [91,92].

IS uses two criteria when measuring GAN performance. First is the quality of the generated image, and second is diversity.

A good result is the ease of predicting conditional probability $P(y|x)$. In other words, if an image is given, it should be easy to identify the type of object. IS classifies the generated image and predicts $P(y|x)$ using the InceptionV3 model [93]. Here, y is a label, and x is

a generated image, reflecting the quality of the image. $P(y)$ is the marginal probability calculated as Equation (9).

$$\int_z p(y|x = G(z))dz \tag{9}$$

Marginal probability is the probability distribution of X or Y when two random variables X and Y get paired and have a combined probability distribution as (X, Y). Equation (9) removes the remaining probability through integral or summation. If the images generated in Equation (9) are varied, the data distribution for y should be uniform. In other words, it should have high entropy. To combine the two criteria, KLD is calculated, with IS derived using Equation (10).

$$IS(G) = exp(E_{X \sim p_{data}} D_{KL}(p(y|x)||p(y))) \tag{10}$$

The disadvantage of IS is that it can incorrectly express performance when generating only one image per class. Then, even with low diversity, $P(y)$ is still uniform.

FID uses an inception module and extracts features from the middle layer. The data distribution of the extracted features is then modeled using multivariate normal distribution with mean μ and covariance $\sum$. The FID between real image x and generated image g is shown in Equation (11).

$$FID(x, g) = ||\mu_x - \mu_g||_2^2 + Tr(\Sigma_x + \Sigma_g - 2(\Sigma_x \Sigma_g)^{\frac{1}{2}}) \tag{11}$$

In Equation (11), Tr summarizes all the elements of the diagonal. Lowering the FID value improves image quality and diversity. FID is sensitive to mode collapse; thus, the more similar the images are, the larger the FID value.

FID is more resistant to noise than IS. If the network generates only one image per class, the distance value increases. For this reason, it is a better measurement than IS for image diversity. FID has a large bias but small variance. Due to the large bias, the value may vary depending on the size of the minibatch. When calculating the FID between the training data and the test data, the FID will be "0" because both are real images. When testing with multiple training samples, but, the FID will not be zero.

Since FID and IS both use an inception model, it is a method based on feature extraction. If the model is not good at extracting features, the same score can be given regardless of which image is generated.

Precision, recall, and f1 score are also used as evaluation matrixs [94]. The more similar the generated image is to the real image, the higher the precision. With higher recall, the generator generates a sample of the training dataset without duplication. Recall is also called sensitivity, hit rate, and true positive rate. The f1 score is a harmonic mean of precision and recall. Harmonic mean is the reciprocal of the arithmetic mean of the inverses for n positive numbers.

In addition to cost optimization, studies on end-to-end evaluation matrixs that can detect and prevent GAN problems such as mode collapse earlier are ongoing. However, it is difficult to find a satisfactory solution. Therefore, we propose to find and apply an evaluation matrix suitable for the model to be used through a lot of trial and error.

6.4. Performance Comparisons

6.4.1. Qualitative Comparisons

Research on GAN is still actively underway. Academia and industry experts praise GAN as an innovation. The image inserted in Tables 1–6 is not a real image. All are synthetic images generated by GAN.

There are six comparison datasets: Modified National Institute of Standards and Technology (MNIST), LSUN Bedroom, ImageNet, CelebA, CelebA-HQ, and Flickr Faces High Quality (FFHQ). The comparison model is the model introduced in III and V and its application model. Application models that added and removed Multi-Scale Gradients (MSG), Feature Quantization (FQ) and instance normalization techniques from StyleGAN

were also added as comparison items [95–97]. Tables 1–6 shows the performance of GAN that can synthesize images at the same level as a real image.

Table 1. Visual results of different models trained with the MNIST dataset. More results are presented in that paper.

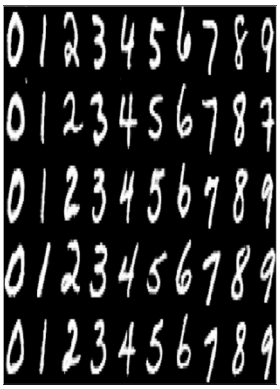
	cGAN (Mirza et al., 2014) [33]	DCGAN (Radford et al., 2015) [20]	InfoGAN (Chen et al., 2016) [32]	BigGAN (Brock et al., 2018) [10]
MNIST				

Table 2. Visual results of different models trained with the LSUN bedroom dataset. More results are presented in that paper.

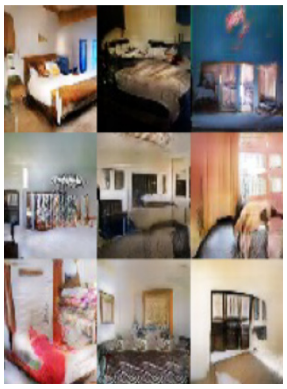
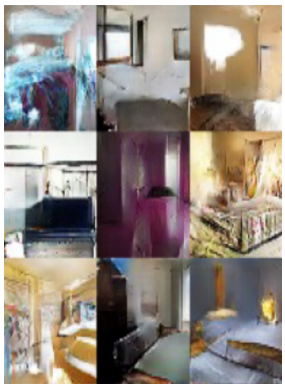
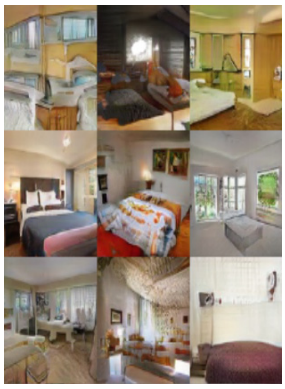
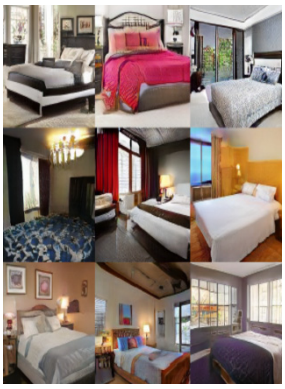
	DCGAN (Radford et al., 2015) [20]	WGAN (Arjovsky et al., 2017) [15]	WGAN-GP (Gulrajani et al., 2017) [17]	PGGAN (Karras et al., 2018) [8]
LSUN Bedroom				

Table 3. Visual results of different models trained with the ImageNet dataset. More results are presented in that paper.

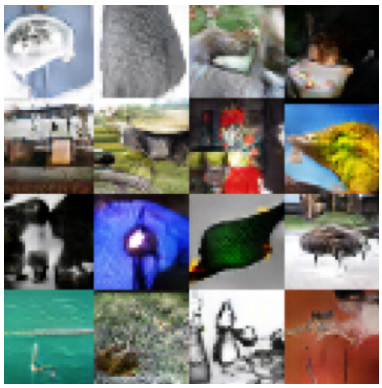
	DCGAN (Radford et al., 2015) [20]	SAGAN (Zhang et al., 2018) [9]	BigGAN (Brock et al., 2018) [10]
ImageNet			

Table 4. Visual results of different models trained with the CelebA dataset. More results are presented in that paper.

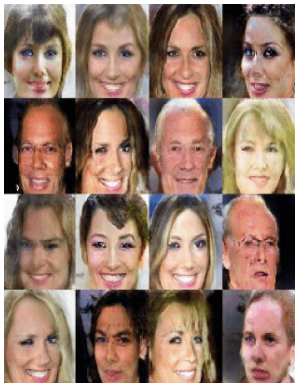
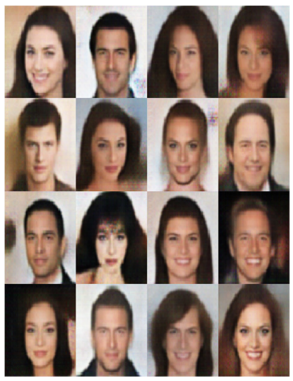
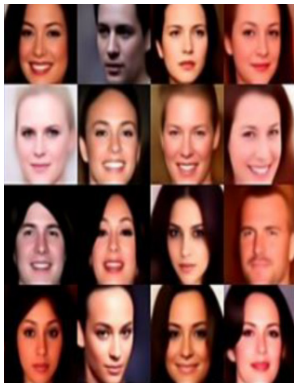
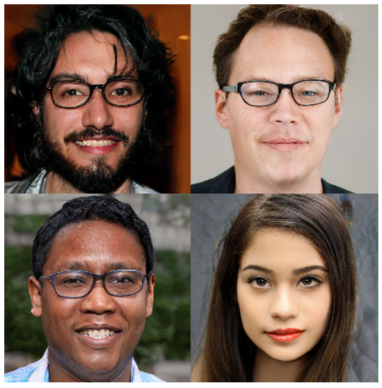
	ALI (Dumoulin et al., 2016) [36]	AVB (Mescheder et al., 2017) [41]	ALICE (Li et al., 2017) [38]	BEGANv3 (Park et al., 2020) [27]
CelebA				

Table 5. Visual results of different models trained with the CelebA-HQ higher resolution image dataset. More results are presented in that paper.

	PGGAN (Karras et al., 2017) [8]	StarGANv2 (Choi et al., 2019) [44]	MSG-StyleGAN (Karnewar et al., 2020) [95]
CelebA-HQ			

Table 6. Visual results of different models trained with the FFHQ higher resolution image dataset. More results are presented in that paper.

	StyleGAN (Karras et al., 2018) [11]	StyleGAN-No Instance Norm (Kynkaanniemi et al., 2019) [97]	FQ-StyleGAN (Zhao et al., 2020) [96]
FFHQ			

6.4.2. Quantitative Comparisons

In this section, the quality of the generated image is evaluated probabilistically. There are five comparison datasets: Canadian Institute for Advanced Research-10 (CIFAR-10), ImageNet, LSUN, CelebA-HQ, and FFHQ. The comparison method is IS and FID, and the comparison model is the model introduced in Sections 3 and 5 and its application model. IS and FID, described in Section 6.3, related to the realism and visual appeal of images and provide useful properties related to quality. Performance is evaluated by looking at the IS or FID output from a model trained on a given dataset.

Table 7 is the IS of 5 models trained on the CIFAR-10 dataset. Among the five models in Table 7, BigGAN showed the highest performance with IS 9.22 points, and ALI showed the lowest performance with 5.34 points for a performance difference of 3.88.

Table 7. The IS of different models trained on CIFAR-10 dataset. The higher the IS value, the better the performance.

Dataset	Model		IS (↑)
CIFAR-10 (64×64)	ALI	(Dumoulin et al., 2016) [36]	5.34
	BEGAN	(Berthelot et al., 2017) [24]	5.62
	WGAN-GP	(Gulrajani et al., 2017) [17]	7.86
	PGGAN	(Karras et al., 2018) [8]	8.80
	BigGAN	(Brock et al., 2018) [10]	9.22

Table 8 is the FID of 9 models trained on the CIFAR-10 dataset. Application models incorporating the latest techniques such as Adaptive Discriminator Augmentation (ADA), Consistency Regularization (CR), Differentiable Augmentation (DiffAugment), training MIXture (MIX), Latent Transformation (LT) and Adversarial Lipschitz Regularization (ALP) with StyleGAN, BigGAN, and WGAN were also added to the comparison items [98–103].

Table 8. The FIDs of different models trained on the CIFAR-10 dataset. The lower the FID value, the better the performance.

Dataset	Model		FID (↓)
CIFAR-10 (64×64)	StyleGAN2+ADA+Tuning	(Karras et al., 2020) [98]	2.92
	CR-BigGAN+DiffAugment		4.30
	BigGAN+DiffAugment	(Zhao et al., 2020) [100]	4.61
	StyleGAN2+DiffAugment		5.79
	BigGAN+MIX	(Tang et al., 2020) [101]	8.17
	BigGAN+CR+LT	(Patel et al., 2020) [102]	9.80
	WGAN-ALP	(Terjek et al., 2019) [103]	12.96
	BigGAN	(Brock et al., 2018) [10]	14.73
	WGAN-GP	(Gulrajani et al., 2017) [17]	29.30

Among the 9 models in Table 8, StyleGAN2+ADA+Tuning showed the highest performance with FID 2.92 and WGAN-GP had the lowest performance at 29.30 for a performance difference of 26.38. The FIDs of BigGAN+DiffAugment, BigGAN+MIX, and BigGAN+CR+LT were 10.12, 6.56, and 4.93 lower than those of BigGAN, respectively. In the case of the StyleGAN2 application model, StyleGAN2+ADA+Tuning has a performance difference of 2.87 lower FID than StyleGAN2+DiffAugment. In the BigGAN application model, BigGAN+DiffAugment has the best performance, and in the StyleGAN2 application model, StyleGAN2+ADA+Tuning has the best performance.

Table 9 is the FID for each training data usage of three models trained with the CIFAR-10 dataset. When 100% of the training data were used, CR-BigGAN+DiffAugment showed the highest performance with FID 4.30. StyleGAN2+DiffAugment showed the lowest performance at 5.79, for a performance difference of 1.49. When 20% and 10% of the training data were used, StyleGAN2+DiffAugment showed the highest performance at 12.15 and 14.5, respectively. BigGAN+DiffAugment had the lowest performance at

14.04 and 22.4, respectively, with a respective difference of 1.89 and 7.9. Among the three models, StyleGAN2+DiffAugment showed the lowest performance decrease when the training data was reduced to 20% and 10%, and BigGAN+DiffAugment showed the highest performance decrease.

Table 9. FID of different models trained with the CIFAR-10 dataset. StyleGAN2 performs better than CR-BigGAN and BigGAN models when there is little training data. FIDs are measured using 100%, 20%, and 10% training data of the CIFAR-10 dataset. The lower the FID value, the better the performance.

Dataset	Model	FID (↓)		
		100% Training Data	20% Training Data	10% Training Data
CIFAR-10 (64×64)	StyleGAN2+DiffAugment [100]	5.79	12.15	14.50
	CR-BigGAN+DiffAugment [100]	4.30	12.84	18.70
	BigGAN+DiffAugment [100]	4.61	14.04	22.40

Table 10 is the IS and FID of the four models trained on the ImageNet dataset. Among the four models in Table 10, BigGAN-deep showed the highest performance with FID 5.7, and BigGAN showed the lowest performance at 8.7 for a performance difference of 3.0. In IS, BigGAN-deep scored the highest with 124.5 points, and BigGAN showed the lowest performance with 98.8 points for a performance difference of 25.7 points. CR-BigGAN was not used in the experiment. As a result, when generating an ImageNet 128×128 sized image, the BigGAN-deep model had the best performance.

Table 10. The IS and FID of different models trained on ImageNet dataset. The higher the IS value, the better the performance. The lower the FID value, the better the performance.

Dataset	Model		IS (↑)	FID (↓)
ImageNet (128×128)	BigGAN-deep	(Brock et al., 2019) [10]	124.5	5.7
	CR-BigGAN	(Zhang et al., 2020) [99]	-	6.7
	BigGAN+DiffAugment	(Zhao et al., 2020) [100]	100.8	6.8
	BigGAN	(Brock et al., 2019) [10]	98.8	8.7

Table 11 is the FID of 4 models trained on the LSUN bedroom dataset. A PG-SWGAN model that combines Sliced Wasserstein (SW) technique with PGGAN was also added to the comparison items [104]. Among the four models in Table 11, StyleGAN showed the highest performance with FID 2.65, and StackGAN2 (Stacked Generative Adversarial Networks2) showed the lowest performance at 35.61, for a performance difference of 32.96 [105]. The application model of PGGAN, PG-SWGAN FID, was 8.0, for a 0.34 higher performance than PGGAN.

Table 11. The FID of different models trained on the LSUN bedroom dataset. The lower the FID value, the better the performance.

Dataset	Model		Category	FID (↓)
LSUN (256 × 256)	StyleGAN	(Karras et al., 2019) [11]	Bedroom	2.65
	PG-SWGAN	(Wu et al., 2019) [104]		8.00
	PGGAN	(Karras et al., 2018) [8]		8.34
	StackGAN2	(Zhang et al., 2017) [105]		35.61

Table 12 is the FID of the four models trained on the LSUN churches dataset. The MSG-StyleGAN model, which combines the MSG technique with StyleGAN, was also

added to the comparison items. Among the four models in Table 12, StyleGAN2 showed the highest performance with FID 3.86, and PGGAN showed the lowest performance at 2.56 for a performance difference of 1.3 [106]. StyleGAN2 FID, an application model of StyleGAN, was 3.86, for a performance difference of 0.35 higher than StyleGAN. On the other hand, the MSG-StyleGAN FID, an application model of StyleGAN, was 5.2, for a performance difference 0.99 lower than StyleGAN.

Table 12. The FID of different models trained on LSUN churches dataset. The lower the FID value, the better the performance.

Dataset	Model		Category	FID (↓)
LSUN (256×256)	StyleGAN2	(Karras et al., 2020) [106]	Churches	3.86
	StyleGAN	(Karras et al., 2019) [11]		4.21
	MSG-StyleGAN	(Karras et al., 2018) [95]		5.20
	PGGAN	(Zhang et al., 2018) [8]		6.42

Table 13 is the FID for each LSUN dataset category of the StyleGAN2 model. Among the four categories in Table 13, Car showed the highest performance with FID 2.32, and Cat showed the lowest performance at 6.93 for a performance difference of 4.61. When the Car image is generated in size 512×384, the FID is 2.32. It has the same FID value as when it is generated in the 256×256 size. When Cat, Churches, and Horse images were generated in the 512×384 size, FID was excluded from Table 13. Therefore, it is difficult to conclude that there will be no difference in FID in the Cat, Churches, and Horse categories when the image size is enlarged.

Table 13. FID of StyleGAN2 model trained with LSUN category and size-specific datasets. The lower the FID value, the better the performance.

Dataset	Model	Category	Image Size	FID (↓)
LSUN	StyleGAN2 [106]	Car	256×256	2.32
		Cat		6.93
		Churches		3.86
		Horse		3.43
		Car	512×384	2.32

Table 14 is the FID of 4 models trained on the CelebA-HQ dataset. Among the four models in Table 14, StyleGAN showed the highest performance with FID 5.06, and PGGAN showed the lowest performance at 7.3 for a performance difference of 2.24. The MSG-StyleGAN FID, an application model of StyleGAN, was 6.37, for a 1.31 lower performance than StyleGAN. PGGAN's application model, PG-SWGAN FID, was 5.5, for a 1.8 higher performance than PGGAN.

Table 14. The FID of different models trained on CelebA-HQ higher resolution image dataset. The lower the FID value, the better the performance.

Dataset	Model		FID (↓)
CelebA-HQ (1024 × 1024)	StyleGAN	(Karras et al., 2019) [11]	5.06
	PG-SWGAN	(Wu et al., 2019) [104]	5.50
	MSG-StyleGAN	(Karnewar et al., 2020) [95]	6.37
	PGGAN	(Karras et al., 2018) [8]	7.30

Table 15 is the FID of 5 models trained on the FFHQ dataset. Among the five models in Table 15, StyleGAN2 showed the highest performance with FID 2.84, and PGGAN showed the lowest performance at 8.4, for a performance difference of 5.56. The MSG-StyleGAN FID, an application model of StyleGAN, was 5.8, or 1.37 lower performance than StyleGAN.

Looking at only Table 12, Table 14, and Table 15, MSG-StyleGAN is inferior to the StyleGAN. In other words, advances in technology do not necessarily bring only good points. On the other hand, when the instance normalization of StyleGAN was removed, the FID was 4.16, or 0.27 higher performance than StyleGAN.

Table 15. The FID of different models trained on FFHQ higher resolution image dataset. The lower the FID value, the better the performance.

Dataset	Model		FID (↓)
FFHQ (1024 × 1024)	StyleGAN2	(Karras et al., 2020) [106]	2.84
	StyleGAN+No Instance Normalization	(Kynkaanniemi et al., 2019) [97]	4.16
	StyleGAN	(Karras et al., 2019) [11]	4.43
	MSG-StyleGAN	(Karnewar et al., 2020) [95]	5.80
	PGGAN	(Karras et al., 2018) [8]	8.40

Table 16 shows the model that showed the highest performance in the six datasets in Tables 7–15. The six datasets are CIFAR-10, ImageNet, LSUN Bedroom, LSUN Churches, CelebA-HQ, and FFHQ. In the CIFAR-10 dataset, StyleGAN2+ADA+Tuning showed the highest performance, BigGAN-deep in the ImageNet dataset, StyleGAN in the LSUN Bedroom dataset, StyleGAN2 in the LSUN Churches dataset, StyleGAN in the CelebA-HQ dataset, and StyleGAN2 in the FFHQ dataset showed the highest performance. Except for the ImageNet dataset, they are all StyleGAN and its application models, and you can see that they work well for relatively multiple datasets.

Table 16. The model that achieved the highest FID performance in each dataset. The lower the FID value, the better the performance.

Dataset	Model		FID (↓)
CIFAR-10 (64 × 64)	StyleGAN2+ADA+Tuning	(Karras et al., 2020) [98]	2.92
ImageNet (128 × 128)	BigGAN-deep	(Brock et al., 2019) [10]	5.70
LSUN Bedroom (256 × 256)	StyleGAN	(Karras et al., 2019) [11]	2.65
LSUN Churches (256 × 256)	StyleGAN2	(Karras et al., 2020) [106]	3.86
CelebA-HQ (1024 × 1024)	StyleGAN	(Karras et al., 2019) [11]	5.06
FFHQ (1024 × 1024)	StyleGAN2	(Karras et al., 2020) [106]	2.84

Several implications can be found in the performance evaluation section. First, out of Tables 7–16, there are only two tables in which IS is used as an evaluation matrix. As such, it is difficult to find papers using IS as an evaluation matrix. This is because when the IS generates only one image per class, the performance can be incorrectly expressed. Although FID has its drawbacks, it is most widely used as an evaluation matrix because it is more accurate than IS.

Second, in Table 8, when the latest techniques such as CR, LT, MIX, and DiffAugment were applied to BigGAN, the performance was higher than when they were not. However, grafting the latest techniques does not always yield good results. It has been grafted, but the performance may be lower. In Table 10, BigGAN-deep had a difference in the performance of 0.96 and 1.1 from CR-BigGAN and BigGAN+DiffAugment. BigGAN-deep is a more sophisticated model with deeper layers from the original BigGAN. In Table 12, StyleGAN had a difference of 0.99 performance from MSG-StyleGAN. In Tables 14 and 15, StyleGAN had a performance difference of 1.31 and 1.37 from MSG-StyleGAN. Different results may be generated depending on the nature of the data types used for training. This means that if the generation target is significantly different from now, the optimization configuration can also be changed accordingly. For this reason, it is judged that the optimal technique for the type of task to be solved must be confirmed through experimentation. In Table 9, the CR technique was applied to BigGAN to reduce the performance reduction

rate that occurs when the dataset size is reduced, but it is not higher than the performance of StyleGAN2. We can expect performance improvement by grafting StyleGAN2's CR technique, but it remains to be seen whether it will be better or worse like above.

Finally, when the Cat, Churches, and Horse images in Table 13 were generated in 512×384 size, FID was excluded. It is difficult to conclude that there will be no difference in FID when the image size is enlarged because the reason for excluding it from the paper has not been clarified [106]. In other words, it is difficult to conclude that the image size does not affect the FID.

7. Conclusions

Deep learning technology, which has led to the development of discriminative modeling, has also been used in generative modeling. Among them, GAN is capable of nonlinear mapping from latent space to data space, and it can utilize large datasets without labels. GAN is useful for image generation work. Currently, GAN research related to computer vision includes high-resolution image generation, image synthesis using text, style simulation, and image-to-image translation.

To date, GAN's performance has been remarkable. GAN showed excellent results in tasks that were difficult to perform in the traditional way. Previously, it was difficult to convert low-resolution images into high-resolution ones. The SRGAN and pix2pix models showed GAN's potential for this task. The StackGAN2 model was useful for text and image synthesis.

GAN improved the existing deep learning method. Training deep learning models with supervised learning requires large amounts of data, including considerable time and money to collect. There may be no public data and even low volumes, if any. GAN can be used to generate the necessary data in this situation. CEGAN (Classification Enhancement Generative Adversarial Network) synthesized images by approximating the real data distribution in an unbalanced data environment, and significantly improved the classification performance [107].

GAN can also be used as a commercial application. Many commercial GAN applications have been developed, receiving good reviews. One example is Prisma, a mobile application [108].

Additionally, there is a field of text-based infographic generation. Infographics are useful for marketing and social promotion, but infographic design requires a lot of time, effort, and skill, which can lengthen the working period. Designers can use GAN to shorten the working time. Designing a similar website requires a lot of time, skill, and creativity, but designing the initial design as a GAN would save significant cost and time.

Second is data compression. Using GAN enables increasing the resolution of image and video. We can connect to the Internet and send large amounts of data all over the world. Using GAN when transmitting low-resolution image and video can increase data quality and reduce bandwidth.

Third is drug discovery and development. GAN can be used to generate molecular structures, taking into account the desired chemical and biological properties. Insilico medicine designed the corona virus disease-2019 3c-like protease inhibitory molecule [109]. The MIT technology review selected Insilico's AI design technology as 2020 Innovation 10.

Fourth is reinforcement learning [110]. If the agent can simulate the environment with GAN, there is no need to test the strategy in computer or reality because agents learn in their own environment.

In addition, GAN is already applied to music production, video games, game design, and movie industry. MuseNet has seen the power to transform the music industry. The process of creating music must be creative and complex, but MuseNet generated music in custom style. It can also extend to areas like news and novels but can give rise to ethical questions. Modern neuroscience theory argues that a person's ability to perceive reality is not a discriminative model [111].

GAN appeared in 2014; six years have passed since then, but GAN training instability still remains. Sometimes, GAN does not converge at all because the two neural networks diverge during the training process. Many researchers have tried to stabilize GAN's training. For example, solutions such as one-sided label smoothing, instance normalization, and minibatch discrimination have been proposed. It is believed that, as GAN develops, this stabilization will mature, and we will be able to train the model without any problems in the near future.

Author Contributions: Conceptualization, S.-W.P. and J.-S.K.; methodology, J.-H.H.; validation, J.-C.K., S.-W.P. and J.-S.K.; formal analysis, J.-H.H.; investigation, S.-W.P.; resources, J.-S.K.; writing—original draft preparation, S.-W.P.; writing—review and editing, S.-W.P.; visualization, S.-W.P.; supervision, J.-H.H., and J.-C.K.; project administration, J.-H.H., and J.-C.K.; funding acquisition, J.-C.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Yann, L.; Yoshua, B.; Geoffrey, H. Deep Learning. *Nature* **2015**, *521*, 436–444.
2. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Bengio, Y. Generative adversarial nets. *arXiv* **2020**, arXiv:1406.2661. Available online: https://arxiv.org/abs/1406.2661 (accessed on 21 April 2020).
3. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Polosukhin, I. Attention is all you need. *arXiv* **2020**, arXiv:1706.03762. Available online: https://arxiv.org/abs/1706.03762 (accessed on 21 April 2020).
4. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. Bert: Pre-Training of Deep Bidirectional Transformers for Language Understanding. *arXiv* **2020**, arXiv:1810.04805. Available online: https://arxiv.org/abs/1810.04805 (accessed on 22 April 2020).
5. Brown, T.B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Amodei, D. Language Models Are Few-Shot Learners. *arXiv* **2020**, arXiv:2005.14165. Available online: https://arxiv.org/abs/2005.14165 (accessed on 22 August 2020).
6. Payne, C. MuseNet. Available online: https://openai.com/blog/musenet (accessed on 23 April 2020).
7. Yamamoto, R.; Song, E.; Kim, J.M. Parallel WaveGAN: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram. In *Proceedings of the ICASSP 2020–2020 IEEE International Conference on Acoustics, Speech and Signal Processing, Barcelona, Spain, 4–8 May 2020*; IEEE: Piscataway, NJ, USA, 2020; pp. 6199–6203.
8. Karras, T.; Aila, T.; Laine, S.; Lehtinen, J. Progressive Growing of Gans for Improved Quality, Stability, and Variation. *arXiv* **2020**, arXiv:1710.10196. Available online: https://arxiv.org/abs/1710.10196 (accessed on 16 May 2020).
9. Zhang, H.; Goodfellow, I.; Metaxas, D.; Odena, A. Self-Attention Generative Adversarial Networks. *arXiv* **2020**, arXiv:1805.08318. Available online: https://arxiv.org/abs/1805.08318 (accessed on 17 May 2020).
10. Brock, A.; Donahue, J.; Simonyan, K. Large Scale Gan Training for High Fidelity Natural Image Synthesis. *arXiv* **2020**, arXiv:1809.11096. Available online: https://arxiv.org/abs/1809.11096 (accessed on 20 May 2020).
11. Karras, T.; Laine, S.; Aila, T. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 16–20 June 2019*; IEEE: Piscataway, NJ, USA, 2019; pp. 4401–4410.
12. Nwankpa, C.; Ijomah, W.; Gachagan, A.; Marshall, S. Activation Functions: Comparison of Trends in Practice and Research for Deep Learning. *arXiv* **2020**, arXiv:1811.03378. Available online: https://arxiv.org/abs/1811.03378 (accessed on 22 May 2020).
13. LeCun, Y.; Touresky, D.; Hinton, G.; Sejnowski, T. A theoretical framework for back-propagation. In Proceedings of the 1988 Connectionist Models Summer School, San Mateo, CA, USA, 1 June 1988; Volume 1, pp. 21–28.
14. Triola, M.F. Bayes' Theorem. Available online: http://faculty.washington.edu/tamre/BayesTheorem.pdf (accessed on 10 January 2020).
15. Arjovsky, M.; Chintala, S.; Bottou, L. Wasserstein Gan. *arXiv* **2020**, arXiv:1701.07875. Available online: https://arxiv.org/abs/1701.07875 (accessed on 25 May 2020).
16. Gouk, H.; Frank, E.; Pfahringer, B.; Cree, M. Regularisation of Neural Networks by Enforcing Lipschitz Continuity. *arXiv* **2020**, arXiv:1804.04368. Available online: https://arxiv.org/abs/1804.04368 (accessed on 26 May 2020).
17. Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; Courville, A.C. Improved training of wasserstein gans. In *Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017*; Curran Associates Inc.: Red Hook, NY, USA, 2018; pp. 5767–5777.
18. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv* **2020**, arXiv:1502.03167. Available online: https://arxiv.org/abs/1502.03167 (accessed on 28 May 2020).
19. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In *Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012*; Curran Associates Inc.: Red Hook, NY, USA, 2013; pp. 1097–1105.

20. Radford, A.; Metz, L.; Chintala, S. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv* **2020**, arXiv:1511.06434. Available online: https://arxiv.org/abs/1511.06434 (accessed on 29 May 2020).
21. Kingma, D.P.; Welling, M. Auto-Encoding Variational Bayes. *arXiv* **2020**, arXiv:1312.6114. Available online: https://arxiv.org/abs/1312.6114 (accessed on 1 June 2020).
22. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2020**, arXiv:1412.6980. Available online: https://arxiv.org/abs/1412.6980 (accessed on 2 June 2020).
23. Yeh, R.; Chen, C.; Lim, T.Y.; Hasegawa-Johnson, M.; Do, M.N. Semantic Image Inpainting with Perceptual and Contextual Losses. *arXiv* **2020**, arXiv:1607.07539. Available online: https://arxiv.org/abs/1607.07539 (accessed on 3 June 2020).
24. Berthelot, D.; Schumm, T.; Metz, L. Began: Boundary Equilibrium Generative Adversarial Networks. *arXiv* **2020**, arXiv:1703.10717. Available online: https://arxiv.org/abs/1703.10717 (accessed on 4 June 2020).
25. Baldi, P. Autoencoders, unsupervised learning, and deep architectures. In Proceedings of the ICML Workshop on Unsupervised and Transfer Learning; Proceedings of Machine Learning Research, New York City, NY, USA, 19–24 June; 2016; pp. 37–49.
26. Chang, C.C.; Hubert Lin, C.; Lee, C.R.; Juan, D.C.; Wei, W.; Chen, H.T. Escaping from collapsing modes in a constrained space. In Proceedings of the European Conference on Computer Vision, Munich, Germany, 8–14 September 2018; Springer: New York, NY, USA, 2019; pp. 204–219.
27. Park, S.W.; Huh, J.H.; Kim, J.C. BEGAN v3: Avoiding Mode Collapse in GANs Using Variational Inference. *Electronics* **2020**, *9*, 688. [CrossRef]
28. Clevert, D.A.; Unterthiner, T.; Hochreiter, S. Fast and Accurate Deep Network Learning by Exponential Linear Units (Elus). *arXiv* **2020**, arXiv:1511.07289. Available online: https://arxiv.org/abs/1511.07289 (accessed on 5 June 2020).
29. Maas, A.L.; Hannun, A.Y.; Ng, A.Y. Rectifier nonlinearities improve neural network acoustic models. *Proc. ICML* **2013**, *30*, 3.
30. Pérez-Cruz, F. Kullback-Leibler divergence estimation of continuous distributions. In *Proceedings of the 2008 IEEE International Symposium on Information Theory, Toronto, ON, Canada, 8 August 2008*; IEEE: New York, NY, USA, 2008; pp. 1666–1670.
31. Yu, F.; Seff, A.; Zhang, Y.; Song, S.; Funkhouser, T.; Xiao, J. Lsun: Construction of a Large-Scale Image Dataset Using Deep Learning with Humans in the Loop. *arXiv* **2020**, arXiv:1506.03365. Available online: https://arxiv.org/abs/1506.03365 (accessed on 6 June 2020).
32. Chen, X.; Duan, Y.; Houthooft, R.; Schulman, J.; Sutskever, I.; Abbeel, P. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Proceedings of the Advances in Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016*; Curran Associates Inc.: Red Hook, NY, USA, 2016; pp. 2172–2180.
33. Mirza, M.; Osindero, S. Conditional Generative Adversarial Nets. *arXiv* **2020**, arXiv:1411.1784. Available online: https://arxiv.org/abs/1411.1784 (accessed on 9 June 2020).
34. Creswell, A.; Bharath, A.A. Inverting the Generator of a Generative Adversarial Network. *IEEE Trans. Neural Networks Learn. Syst.* **2019**, *30*, 1967–1974. [CrossRef] [PubMed]
35. Lipton, Z.C.; Tripathi, S. Precise Recovery of Latent Vectors from Generative Adversarial Networks. *arXiv* **2020**, arXiv:1702.04782. Available online: https://arxiv.org/abs/1702.04782 (accessed on 10 June 2020).
36. Dumoulin, V.; Belghazi, I.; Poole, B.; Mastropietro, O.; Lamb, A.; Arjovsky, M.; Courville, A. Adversarially Learned Inference. *arXiv* **2020**, arXiv:1606.00704. Available online: https://arxiv.org/abs/1606.00704 (accessed on 11 June 2020).
37. Donahue, J.; Krähenbühl, P.; Darrell, T. Adversarial Feature Learning. *arXiv* **2020**, arXiv:1605.09782. Available online: https://arxiv.org/abs/1605.09782 (accessed on 12 June 2020).
38. Li, C.; Liu, H.; Chen, C.; Pu, Y.; Chen, L.; Henao, R.; Carin, L. Alice: Towards understanding adversarial learning for joint distribution matching. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Curran Associates Inc.: Red Hook, NY, USA, 2017; pp. 5495–5503.
39. Bengio, Y.; Yao, L.; Alain, G.; Vincent, P. Generalized Denoising Auto-Encoders as Generative Models. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 5–10 December 2013; Curran Associates Inc.: Red Hook, NY, USA, 2013; pp. 899–907.
40. Makhzani, A.; Shlens, J.; Jaitly, N.; Goodfellow, I.; Frey, B. Adversarial Autoencoders. *arXiv* **2020**, arXiv:1511.05644. Available online: https://arxiv.org/abs/1511.05644 (accessed on 15 June 2020).
41. Mescheder, L.; Nowozin, S.; Geiger, A. Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks. In Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; Volume 70, pp. 2391–2400.
42. Goodfellow, I. NIPS 2016 tutorial: Generative Adversarial Networks. *arXiv* **2020**, arXiv:1701.00160. Available online: https://arxiv.org/abs/1701.00160 (accessed on 18 June 2020).
43. Choi, Y.; Choi, M.; Kim, M.; Ha, J.W.; Kim, S.; Choo, J. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018*; Curran Associates Inc.: Red Hook, NY, USA, 2018; pp. 8789–8797.
44. Choi, Y.; Uh, Y.; Yoo, J.; Ha, J.W. Stargan v2: Diverse image synthesis for multiple domains. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Online, 14–19 June 2020; IEEE: New York, NY, USA, 2020; pp. 8188–8197.
45. He, K.; Zhang, X.; Ren, S.; Sun, J. Identity Mappings in Deep Residual Networks. In *Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 8–16 October 2016*; Springer: Cham, Switzerland, 2016; pp. 630–645.

46. Liu, Z.; Luo, P.; Wang, X.; Tang, X. Deep Learning Face Attributes in the Wild. In Proceedings of the International Conference on Computer Vision, Araucano Park, Las Condes, Chile, 11–18 December 2015; IEEE Computer Society: Washington, DC, USA, 2015; pp. 3730–3738.
47. Khan, M.H.; McDonagh, J.; Khan, S.; Shahabuddin, M.; Arora, A.; Khan, F.S.; Tzimiropoulos, G. AnimalWeb: A Large-Scale Hierarchical Dataset of Annotated Animal Faces. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Online, 14–19 June 2020; IEEE: New York, NY, USA, 2020; pp. 6939–6948.
48. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami Beach, FL, USA, 20–26 June 2009*; Curran Associates Inc.: Red Hook, NY, USA, 2009; pp. 248–255.
49. Gatys, L.A.; Ecker, A.S.; Bethge, M. A neural Algorithm of Artistic Style. *arXiv* **2020**, arXiv:1508.06576. Available online: https://arxiv.org/abs/1508.06576 (accessed on 1 July 2020). [CrossRef]
50. Salimans, T.; Goodfellow, I.; Zaremba, W.; Cheung, V.; Radford, A.; Chen, X. Improved techniques for training gans. In *Proceedings of the Advances in Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016*; Curran Associates Inc.: Red Hook, NY, USA, 2016; pp. 2234–2242.
51. Arjovsky, M.; Bottou, L. Towards Principled Methods for Training Generative Adversarial Networks. *arXiv* **2020**, arXiv:1701.04862. Available online: https://arxiv.org/abs/1701.04862 (accessed on 2 July 2020).
52. Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; IEEE: New York, NY, USA, 2016; pp. 3431–3440.
53. Nair, V.; Hinton, G.E. Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th International Conference on Machine Learning, Haifa, Israel, 21–24 June 2010; Omnipress: Madison, WI, USA, 2010; pp. 807–814.
54. Sønderby, C.K.; Caballero, J.; Theis, L.; Shi, W.; Huszár, F. Amortised Map Inference for Image Super-Resolution. *arXiv* **2020**, arXiv:1610.04490. Available online: https://arxiv.org/abs/1610.04490 (accessed on 21 July 2020).
55. Mescheder, L.; Nowozin, S.; Geiger, A. The Numerics of Gans. In *Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017*; Curran Associates Inc.: Red Hook, NY, USA, 2017; pp. 1825–1835.
56. Theis, L.; Oord, A.V.D.; Bethge, M. A Note on the Evaluation of Generative Models. *arXiv* **2020**, arXiv:1511.01844. Available online: https://arxiv.org/abs/1511.01844 (accessed on 24 April 2020).
57. Siddharth, N.; Paige, B.; Van de Meent, J.W.; Desmaison, A.; Goodman, N.; Kohli, P.; Torr, P. Learning disentangled representations with semi-supervised deep generative models. In *Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017*; Curran Associates Inc.: Red Hook, NY, USA, 2017; pp. 5925–5935.
58. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. *arXiv* **2020**, arXiv:1301.3781. Available online: https://arxiv.org/abs/1301.3781 (accessed on 18 May 2020).
59. Gurumurthy, S.; Kiran Sarvadevabhatla, R.; Venkatesh Babu, R. Deligan: Generative adversarial networks for diverse and limited data. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; IEEE: New York, NY, USA, 2017; pp. 166–174.
60. Ledig, C.; Theis, L.; Huszár, F.; Caballero, J.; Cunningham, A.; Acosta, A.; Shi, W. Photo-realistic single image super-resolution using a generative adversarial network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; IEEE: New York, NY, USA, 2017; pp. 4681–4690.
61. Yu, X.; Porikli, F. Ultra-resolving face images by discriminative generative networks. In *Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 8–16 October 2016*; Springer: Cham, Switzerland, 2016; pp. 318–333.
62. Yu, X.; Porikli, F. Hallucinating very low-resolution unaligned and noisy face images by transformative discriminative autoencoders. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; IEEE: New York, NY, USA, 2017; pp. 3760–3768.
63. Shrivastava, A.; Pfister, T.; Tuzel, O.; Susskind, J.; Wang, W.; Webb, R. Learning from simulated and unsupervised images through adversarial training. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; IEEE: New York, NY, USA, 2017; pp. 2107–2116.
64. Zhang, M.; Teck Ma, K.; Hwee Lim, J.; Zhao, Q.; Feng, J. Deep future gaze: Gaze anticipation on egocentric videos using adversarial networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; IEEE: New York, NY, USA, 2017; pp. 4372–4381.
65. Bousmalis, K.; Silberman, N.; Dohan, D.; Erhan, D.; Krishnan, D. Unsupervised pixel-level domain adaptation with generative adversarial networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; IEEE: New York, NY, USA, 2017; pp. 3722–3731.
66. Liu, M.Y.; Tuzel, O. Coupled generative adversarial networks. In *Proceedings of the Advances in Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016*; Curran Associates Inc.: Red Hook, NY, USA, 2016; pp. 469–477.
67. Denton, E.L.; Chintala, S.; Fergus, R. Deep generative image models using a laplacian pyramid of adversarial networks. In *Proceedings of the Advances in Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016*; Curran Associates Inc.: Red Hook, NY, USA, 2016; pp. 1486–1494.
68. Huang, X.; Li, Y.; Poursaeed, O.; Hopcroft, J.; Belongie, S. Stacked generative adversarial networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, Hawaii, USA, 21–26 July 2017; IEEE: New York, NY, USA, 2017; pp. 5077–5086.

69. Reed, S.; Akata, Z.; Yan, X.; Logeswaran, L.; Schiele, B.; Lee, H. Generative Adversarial Text to Image Synthesis. *arXiv* **2020**, arXiv:1605.05396. Available online: https://arxiv.org/abs/1605.05396 (accessed on 23 May 2020).
70. Reed, S.E.; Akata, Z.; Mohan, S.; Tenka, S.; Schiele, B.; Lee, H. Learning what and where to draw. In *Proceedings of the Advances in Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016*; Curran Associates Inc.: Red Hook, NY, USA, 2016; pp. 217–225.
71. Zhu, J.Y.; Krähenbühl, P.; Shechtman, E.; Efros, A.A. Generative visual manipulation on the natural image manifold. In *Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 8–16 October 2016*; Springer: Cham, Switzerland, 2016; pp. 597–613.
72. Brock, A.; Lim, T.; Ritchie, J.M.; Weston, N. Neural Photo Editing with Introspective Adversarial Networks. *arXiv* **2020**, arXiv:1609.07093. Available online: https://arxiv.org/abs/1609.07093 (accessed on 27 May 2020).
73. Yurt, M.; Dar, S.U.; Erdem, A.; Erdem, E.; Oguz, K.K.; Çukur, T. mustGAN: Multi-Stream Generative Adversarial Networks for MR Image Synthesis. *Med. Image Anal.* **2021**, *70*, 101944. [CrossRef]
74. Cai, W.; Wei, Z. PiiGAN: Generative Adversarial Networks for Pluralistic Image Inpainting. *IEEE Access* **2020**, *8*, 48451–48463. [CrossRef]
75. Isola, P.; Zhu, J.-Y.; Zhou, T.; Efros, A.A. Image-to-Image Translation with Conditional Adversarial Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 5967–5976.
76. Zhu, J.Y.; Park, T.; Isola, P.; Efros, A.A. Unpaired image-to-image translation using cycle-consistent adversarial networks. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; IEEE: New York, NY, USA, 2017; pp. 2223–2232.
77. Babu, K.K.; Dubey, S.R. CSGAN: Cyclic-Synthesized Generative Adversarial Networks for image-to-image transformation. *Expert Syst. Appl.* **2021**, *169*, 114431. [CrossRef]
78. Wang, X.; Tang, X. Face Photo-Sketch Synthesis and Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2009**, *31*, 1955–1967. [CrossRef]
79. Tyleček, R.; Šára, R. Spatial pattern templates for recognition of objects with regular structure. In *Proceedings of the German Conference on Pattern Recognition, Saarbrücken, Germany, 3–6 September 2013*; Springer: Cham, Switzerland, 2013; pp. 364–374.
80. Yi, Z.; Zhang, H.; Tan, P.; Gong, M. Dualgan: Unsupervised dual learning for image-to-image translation. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; IEEE: New York, NY, USA, 2017; pp. 2849–2857.
81. Scott, E. Artist, Creative AI, Scott's Blog. Available online: http://www.scott-eaton.com/ (accessed on 10 March 2020).
82. Kim, T.; Park, E.; Lee, H.; Moon, Y.-J.; Bae, S.-H.; Lim, D.; Jang, S.; Kim, L.; Cho, I.-H.; Choi, M.; et al. Solar farside magnetograms from deep learning analysis of STEREO/EUVI data. *Nat. Astron.* **2019**, *3*, 397–400. [CrossRef]
83. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA,, 26 June–1 July 2016; Curran Associates Inc.: Red Hook, NY, USA, 2016; pp. 770–778.
84. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2020**, arXiv:1409.1556. Available online: https://arxiv.org/abs/1409.1556 (accessed on 7 June 2020).
85. Mahapatra, D.; Ge, Z. Training data independent image registration using generative adversarial networks and domain adaptation. *Pattern Recognit.* **2020**, *100*, 107109. [CrossRef]
86. Arora, S.; Ge, R.; Liang, Y.; Ma, T.; Zhang, Y. Generalization and equilibrium in generative adversarial nets (gans). In Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; Volume 70, pp. 224–232.
87. Zhao, J.; Mathieu, M.; LeCun, Y. Energy-Based Generative Adversarial Network. *arXiv* **2020**, arXiv:1609.03126. Available online: https://arxiv.org/abs/1609.03126 (accessed on 8 June 2020).
88. Metz, L.; Poole, B.; Pfau, D.; Sohl-Dickstein, J. Unrolled Generative Adversarial Networks. *arXiv* **2020**, arXiv:1611.02163. Available online: https://arxiv.org/abs/1611.02163 (accessed on 13 June 2020).
89. Lee, J.D.; Simchowitz, M.; Jordan, M.I.; Recht, B. Gradient descent only converges to minimizers. In Proceedings of the Conference on Learning Theory, New York, NY, USA, 23–26 June 2016; pp. 1246–1257.
90. Pemantle, R. Nonconvergence to Unstable Points in Urn Models and Stochastic Approximations. *Ann. Probab.* **1990**, *18*, 698–712. [CrossRef]
91. Barratt, S.; Sharma, R. A Note on the Inception Score. *arXiv* **2020**, arXiv:1801.01973. Available online: https://arxiv.org/abs/1801.01973 (accessed on 14 June 2020).
92. Heusel, M.; Ramsauer, H.; Unterthiner, T.; Nessler, B.; Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017*; Curran Associates Inc.: Red Hook, NY, USA, 2017; pp. 6626–6637.
93. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the inception architecture for computer vision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; Curran Associates Inc.: Red Hook, NY, USA, 2016; pp. 2818–2826.
94. Sajjadi, M.S.; Bachem, O.; Lucic, M.; Bousquet, O.; Gelly, S. Assessing generative models via precision and recall. In *Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 3–8 December 2018*; Curran Associates Inc.: Red Hook, NY, USA, 2018; pp. 5228–5237.

95. Karnewar, A.; Wang, O. Msg-gan: Multi-scale gradients for generative adversarial networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Online, 14–19 June 2020; IEEE: New York, NY, USA, 2020; pp. 7799–7808.
96. Zhao, Y.; Li, C.; Yu, P.; Gao, J.; Chen, C. Feature Quantization Improves GAN Training. *arXiv* **2020**, arXiv:2004.02088. Available online: https://arxiv.org/abs/2004.02088 (accessed on 15 August 2020).
97. Kynkäänniemi, T.; Karras, T.; Laine, S.; Lehtinen, J.; Aila, T. Improved Precision and Recall Metric for Assessing Generative models. *arXiv* **2020**, arXiv:1904.06991. Available online: https://arxiv.org/abs/1904.06991 (accessed on 16 June 2020).
98. Karras, T.; Aittala, M.; Hellsten, J.; Laine, S.; Lehtinen, J.; Aila, T. Training Generative Adversarial Networks with Limited Data. *arXiv* **2020**, arXiv:2006.06676. Available online: https://arxiv.org/abs/2006.06676 (accessed on 7 November 2020).
99. Zhang, H.; Zhang, Z.; Odena, A.; Lee, H. Consistency Regularization for Generative Adversarial Networks. *arXiv* **2020**, arXiv:1910.12027. Available online: https://arxiv.org/abs/1910.12027 (accessed on 17 June 2020).
100. Zhao, S.; Liu, Z.; Lin, J.; Zhu, J.Y.; Han, S. Differentiable Augustmentation for Data-Efficient Gan Training. *arXiv* **2020**, arXiv:2006.10738. Available online: https://arxiv.org/abs/2006.10738 (accessed on 7 January 2021).
101. Tang, S. Lessons Learned from the Training of GANs on Artificial Datasets. *IEEE Access* **2020**, *8*, 165044–165055. [CrossRef]
102. Patel, P.; Kumari, N.; Singh, M.; Krishnamurthy, B. LT-GAN: Self-Supervised GAN with Latent Transformation Detection. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, Waikoloa, HI, USA, 5–9 January 2021; pp. 3189–3198.
103. Terjék, D. Adversarial Lipschitz Regularization. *arXiv* **2020**, arXiv:1907.05681. Available online: https://arxiv.org/abs/1907.05681 (accessed on 3 July 2020).
104. Wu, J.; Huang, Z.; Acharya, D.; Li, W.; Thoma, J.; Paudel, D.P.; Gool, L.V. Sliced wasserstein generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 16–20 June 2019*; IEEE: Piscataway, NJ, USA, 2019; pp. 3713–3722.
105. Zhang, H.; Xu, T.; Li, H.; Zhang, S.; Wang, X.; Huang, X.; Metaxas, D.N. Stackgan++: Realistic image synthesis with stacked generative adversarial networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *41*, 1947–1962. [CrossRef]
106. Karras, T.; Laine, S.; Aittala, M.; Hellsten, J.; Lehtinen, J.; Aila, T. Analyzing and improving the image quality of stylegan. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Online, 14–19 June 2020; IEEE: New York, NY, USA, 2020; pp. 8110–8119.
107. Suh, S.; Lee, H.; Lukowicz, P.; Lee, Y.O. CEGAN: Classification Enhancement Generative Adversarial Networks for unraveling data imbalance problems. *Neural. Networks* **2021**, *133*, 69–86. [CrossRef] [PubMed]
108. Prisma Labs, Prisma Photo Editor. Available online: http://www.prisma-ai.com/index.html (accessed on 12 May 2020).
109. Zhavoronkov, A.; Zagribelnyy, B.; Zhebrak, A.; Aladinskiy, V.; Terentiev, V.; Vanhaelen, Q.; Bishop, M. Potential Non-Covalent SARS-CoV-2 3C-Like Protease Inhibitors Designed Using Generative Deep Learning Approaches and Reviewed by Human Medicinal Chemist in Virtual Reality. *ChemRxiv* **2020**. [CrossRef]
110. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
111. Goertzel, B. *Artificial General Intelligence*; Springer: Berlin/Heidelberg, Germany, 2007; Volume 2.