Volume: 9 Issue: VIII Month of publication: August 2021

DOI: https://doi.org/10.22214/ijraset.2021.37723

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue VIII Aug 2021- Available at www.ijraset.com

Generative Adversarial Networks

Amey Thakur[1], Mega Satish[2]
[1, 2]*Department of Computer Engineering, University of Mumbai, Mumbai, Maharashtra, India*

Abstract: Deep learning's breakthrough in the field of artificial intelligence has resulted in the creation of a slew of deep learning models. One of these is the Generative Adversarial Network, which has only recently emerged. The goal of GAN is to use unsupervised learning to analyse the distribution of data and create more accurate results. The GAN allows the learning of deep representations in the absence of substantial labelled training information. Computer vision, language and video processing, and image synthesis are just a few of the applications that might benefit from these representations. The purpose of this research is to get the reader conversant with the GAN framework as well as to provide the background information on Generative Adversarial Networks, including the structure of both the generator and discriminator, as well as the various GAN variants along with their respective architectures. Applications of GANs are also discussed with examples.
Keywords: Generative Adversarial Networks (GANs), Generator, Discriminator, Supervised and Unsupervised Learning, Discriminative and Generative Modelling, Backpropagation, Loss Functions, Machine Learning, Deep Learning, Neural Networks, Convolutional Neural Network (CNN), Deep Convolutional GAN (DCGAN), Conditional GAN (cGAN), Information Maximizing GAN (InfoGAN), Stacked GAN (StackGAN), Pix2Pix, Wasserstein GAN (WGAN), Progressive Growing GAN (ProGAN), BigGAN, StyleGAN, CycleGAN, Super-Resolution GAN (SRGAN), Image Synthesis, Image-to-Image Translation.

I. INTRODUCTION

In the last decade, the rise of machine learning and later deep learning has taken an essential role in the field of computer science. With this breakthrough, our ability to solve complex problems continues to increase significantly. Deep learning has the potential of discovering comprehensive multilayer models [1] capable of describing probabilities for data used in artificial intelligence applications. Generative Adversarial Networks [2][3] are a type of generative modelling that employs deep learning techniques such as convolutional neural networks. GANs are a revolutionary innovation for both kinds of learning, supervised and unsupervised. GANs [4] are a creative approach to train a generative model by defining it as a supervised learning technique with two sub-models: the generator model, which we train to create new instances, and the discriminator model, which attempts to categorise examples as real or fake i.e. generated. The two models are trained until the discriminator model is tricked approximately half of the time, indicating that the generator model is producing convincing instances. In machine learning, generative modelling is an unsupervised learning problem that entails automatically identifying and trying to learn patterns or similarities in the dataset so that the model may be used to create or produce new instances that might have been drawn from the original dataset. In Generative Adversarial Networks, the generator creates a new sample from input data and the discriminator tries to distinguish whether the generated sample is real or fake. Both networks are in competition with one another and are being trained at the same time. The generator doesn't know how to generate the sample; it learns by interacting with the discriminator. The discriminator basically tries to distinguish between the sample generated by the generator model and the actual data as it has access to both real data and synthesized data. If the output of the generator is considered fake, an error signal is sent to the generator to improve the quality of the results. The discriminator is then modified in the following round to improve its ability to distinguish between real and false data, and the generator is changed depending on how successfully or poorly the produced samples deceived the discriminator. With this paper, we are proposing an explanation of Generative Adversarial Networks background as well as generative modelling and discriminative modelling. We intend to describe the architecture of the Generative Adversarial Network including its working and the data used for this system. We aim to provide an elucidation of the variations of GANs along with their applications in the real world.

A. Supervised and Unsupervised Learning

Predictive modelling, for example, is a common machine learning issue that includes utilising a model to generate a forecast. This necessitates a training dataset, which consists of numerous instances, referred to as samples, each having input variables (X) and output class labels (y). A model is trained by displaying patterns of inputs, having it predict outputs, and then correcting the model so that the outputs are more similar to the predicted outputs [5]. A supervised type of learning, or supervised learning [6], refers to the process of correcting the model. Classification and regression are examples of supervised learning tasks, whereas logistic regression and random forest are examples of supervised learning algorithms.

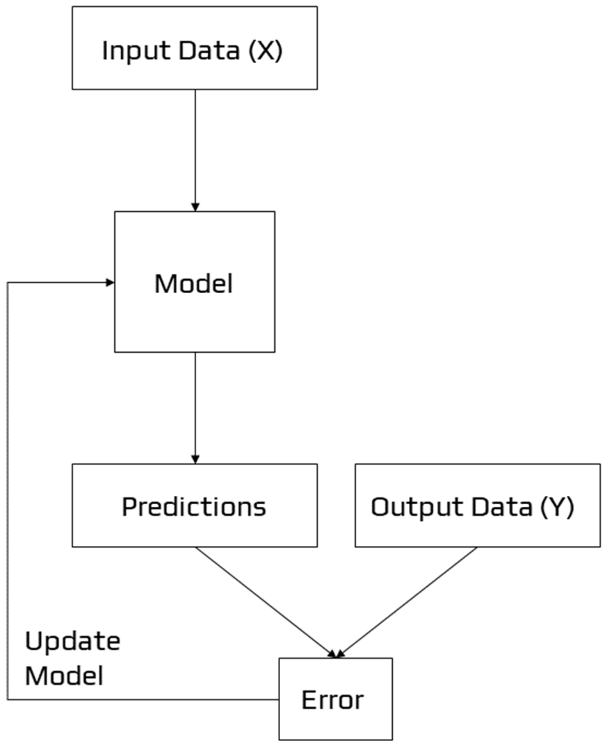


Figure 1: Example of Supervised Learning

Another learning approach is one in which the model is provided with only the input variables (X) and the challenge has no output variables (y). The patterns in the dataset are extracted or summarised to create a model. Because the model isn't forecasting anything, there isn't any need for it to be corrected.

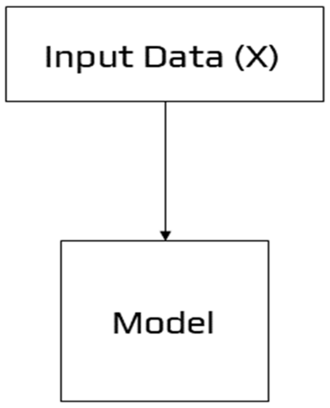


Figure 2: Example of Unsupervised Learning

The unsupervised learning technique [7] is the second primary form of machine learning. The aim is to discover "meaningful findings" in the data, and we are only provided inputs. Because we are not informed what sorts of patterns to look for and there is no clear error measure to utilize, this is a far less well-defined problem unlike supervised learning, where we can compare our predicted y for a given x to the observed value, unsupervised learning does not allow us to compare our predictions to the observed value. Unsupervised learning is the term used to describe this lack of correction. Clustering and generative modelling are examples of unsupervised learning problems, whereas K-means and Generative Adversarial Networks are examples of unsupervised learning algorithms.

B. Discriminative and Generative Modelling

We could be interested in creating a model that predicts a class label given a sample of input variables in supervised learning. This process of predictive modelling is known as classification.

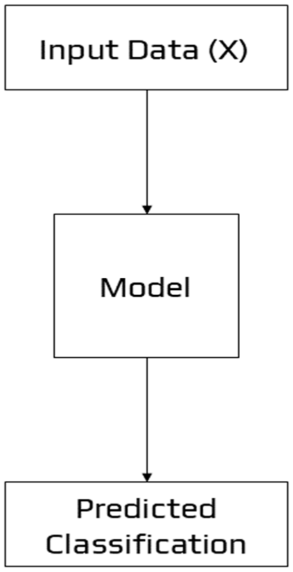


Figure 3: Example of Discriminative Modelling

Discriminative modelling and classification are two terms that have been used interchangeably in the past. We combine the inference and evaluation stages into a single learning problem by using the training data to develop a discriminant function f(x) that maps each x directly onto a class label. This is because a model must distinguish between instances of input variables belonging to different classes; it must pick or decide which class a given example belongs to. A discriminative model overlooks the question of whether or not a particular event is likely, instead of focusing on the likelihood of a label being applied to it.

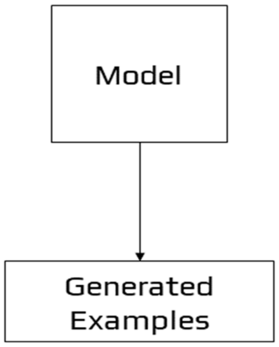


Figure 4: Example of Generative Modelling

Unsupervised models that summarise the distribution of input variables, on the other hand, could be able to produce or generate new examples in the distribution. As a result, generative models are used to describe various sorts of models. A single variable, for example, could have a well-known data distribution, such as a Gaussian distribution or a bell curve. A generative model may be able to adequately describe this data distribution and then be used to create new variables that fit within the input variable's distribution.
Generative models are approaches that explicitly or implicitly describe the distribution of inputs and outputs, allowing synthetic data points to be generated in the input space by sampling from them. A competent generative model may be able to produce new instances that are not just reasonable, but also indistinguishable from real-world examples.
A generative model takes into account the data's distribution and informs how likely a specific occurrence is. Because they can assign a probability to a succession of words, models that predict the next word in a sequence are often generative.

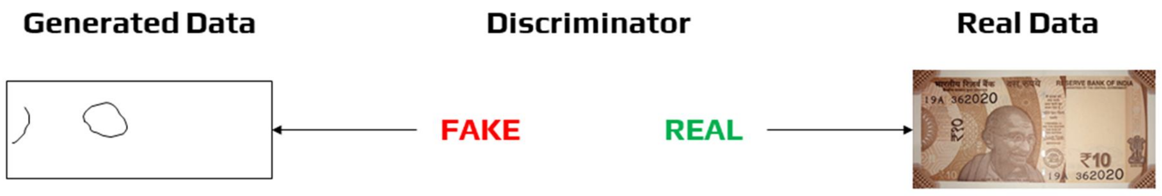
C. Generative models Examples

Naive Bayes [8] is a generative model that is frequently used as a discriminative model. Naive Bayes sums together the probability distributions of each input variable and the output class. When making a prediction, the chance of each potential outcome for each variable is computed, the independent probabilities are added, and the most likely outcome is predicted. Using the probability distributions for each variable in reverse, new reasonable (independent) feature values may be generated. Latent Dirichlet Allocation [9] (LDA) and the Gaussian Mixture Model [10] (GMM) are two more examples of generative models.
As generative models, deep learning approaches can be employed. The Restricted Boltzmann Machine [11] (RBM) and the Deep Belief Network [12] (DBN) are two prominent examples. The Variational Autoencoder [13] (VAE) and the Generative Adversarial Network (GAN) are two examples of deep learning generative modelling techniques that are currently in use.

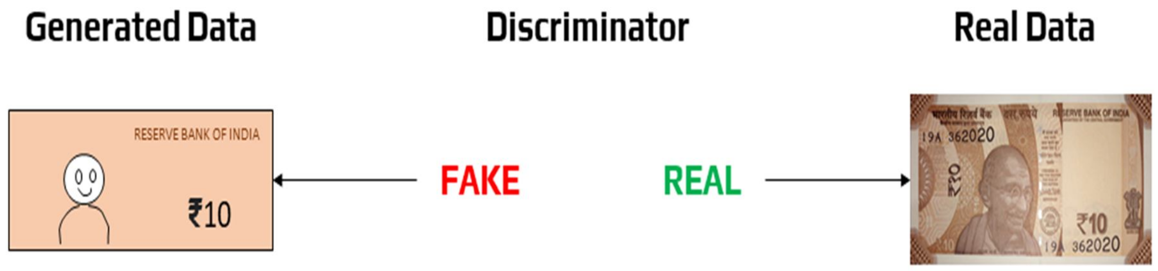
II. OVERVIEW OF GAN STRUCTURE

There are two elements to a generative adversarial network (GAN):
1) The generator improves its ability to create credible data over time. The discriminator uses the produced instances as negative training examples.
2) The discriminator learns to tell the difference between false and real data generated by the generator. The generator is penalised by the discriminator if it produces improbable results.

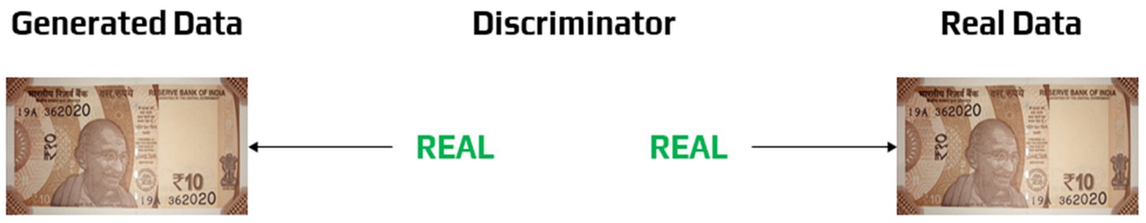
When training begins, the generator generates false data, which the discriminator soon recognises:



The generator comes closer to creating output that can deceive the discriminator as training progresses:



Finally, assuming generator training goes well, the discriminator becomes less capable of distinguishing between genuine and fake objects. It begins to mistakenly identify false data as real, and its accuracy suffers as a result.



The following is a diagram of the entire system:

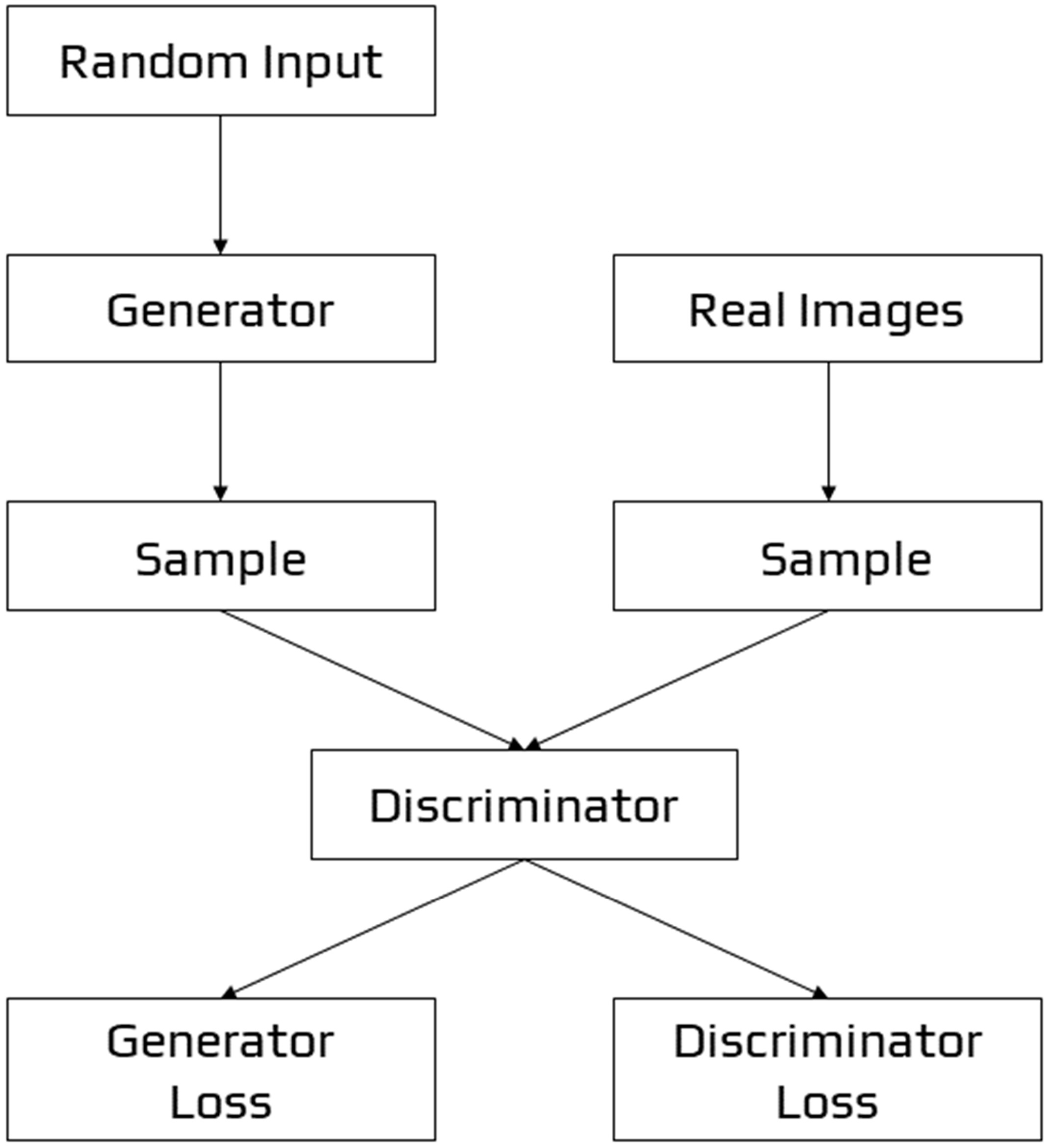


Figure 5: Structure of GANs

III. THE GENERATOR

By integrating input from the discriminator, the generator portion of a GAN learns to produce false data. It learns how to convince the discriminator that its output is real.

Generator training necessitates a greater degree of integration between the generator and the discriminator than discriminator training. The component of the GAN that trains the generator consists of the following:

1) Random Input
2) The random input is transformed into a data instance via the generator network.
3) The produced data is classified using a discriminator network.
4) discriminator output
5) Generator loss is a penalty imposed on the generator when it fails to mislead the discriminator.

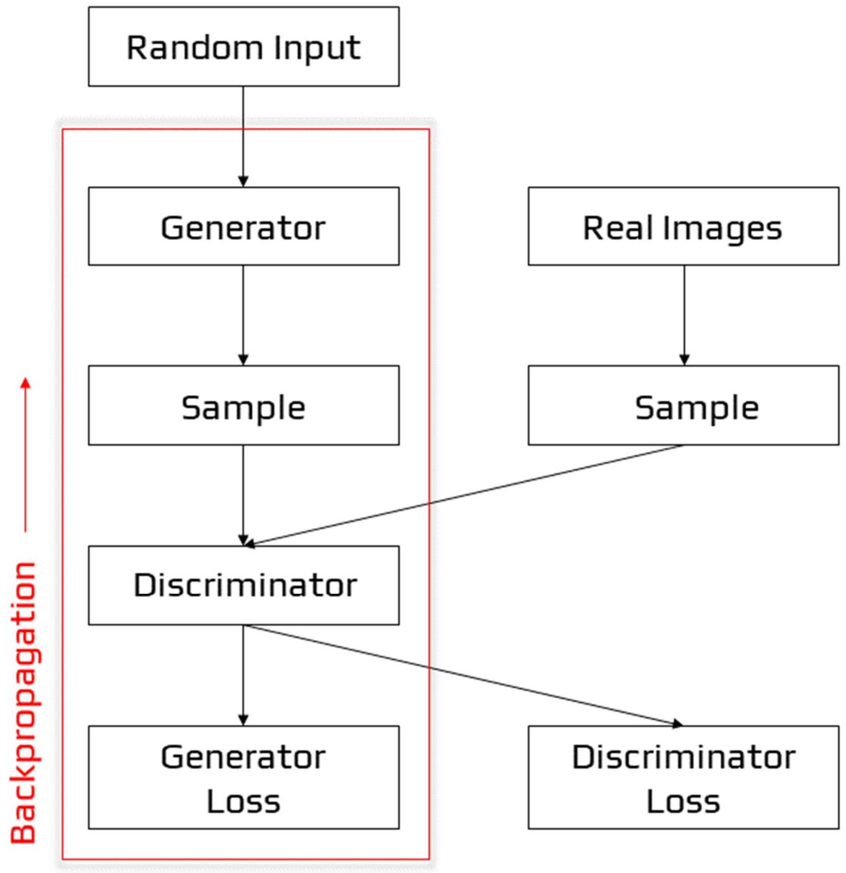


Figure 6: Backpropagation in generator training

A. Random Input

In order to function, neural networks require some type of input. We usually enter data that we wish to do something with, such as a case that we want to categorise or forecast. What, on the other hand, do we feed into a network that generates whole new data instances?

A GAN uses random noise as its input in its most basic form. The generator then converts the noise into something useful. We can get the GAN to create a broad range of data by adding noise and sampling from different points in the target distribution.

Experiments show that the noise distribution is unimportant, therefore we may select a distribution that is simple to sample from, such as a uniform distribution. The space from which the noise is sampled is generally less in size than the dimensionality of the output space for practical reasons.

B. Training the Generator with the Discriminator

We change the weights of a neural net to decrease the inaccuracy or loss of its output when training it. The generator in our GAN, on the other hand, is not directly related to the loss we are attempting to reduce. The generator feeds into the discriminator net, which generates the output we are seeking to influence. The generator is penalised by the generator loss if the discriminator network classifies a sample as fraudulent.

Backpropagation [14] must incorporate this additional network segment. Backpropagation corrects each weight by estimating the weight's influence on the output, or how the output would change if the weight were altered. The impact of a generator weight, on the other hand, is determined by the impact of the discriminator weights into which it feeds. As a result, backpropagation begins at the output and flows back into the generator via the discriminator.

During generator training, however, we do not want the discriminator to alter. Attempting to strike a moving target would make an already difficult task considerably more difficult for the generator.

As a result, we use the technique below to train the generator:

1) Sample random noise.
2) Using sampled random noise, generate generator output.
3) For generator output, use the discriminator to classify it as "Real" or "Fake".
4) Calculate the loss incurred as a result of discriminator classification.
5) Gradients may be obtained by backpropagation through the discriminator and generator.
6) To just modify the generator weights, use gradients.

This is one iteration of generator training.

IV. THE DISCRIMINATOR

In a GAN, the discriminator is essentially a classifier. It tries to tell the difference between actual data and data generated by the generator. Any network architecture suited to the sort of data it's categorising might be used.

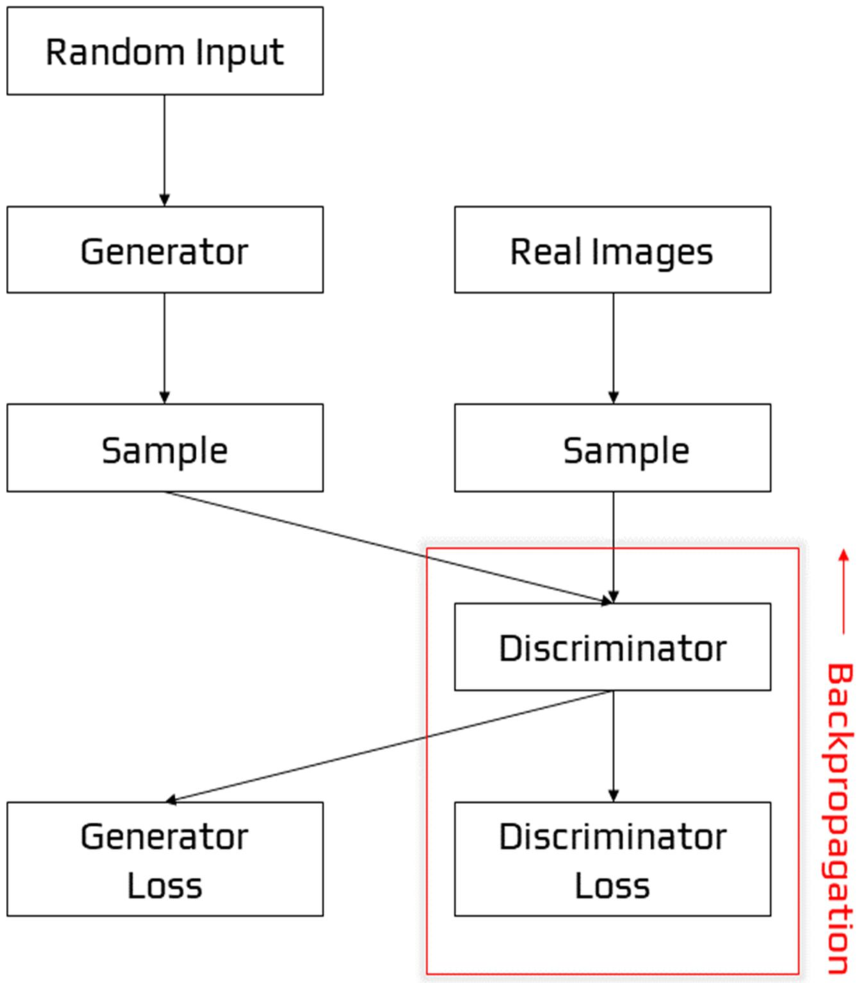


Figure 7: Backpropagation in discriminator training

A. Discriminator Training Data

The training data for the discriminator originates from two places:

1) Real-world data examples, such as real-world portraits of people. During training, the discriminator utilises these examples as positive examples.
2) The generator generates fake data objects. During training, the discriminator utilises these events as negative examples.

The generator does not train during discriminator training. It keeps its weights constant while creating samples for the discriminator to learn from.

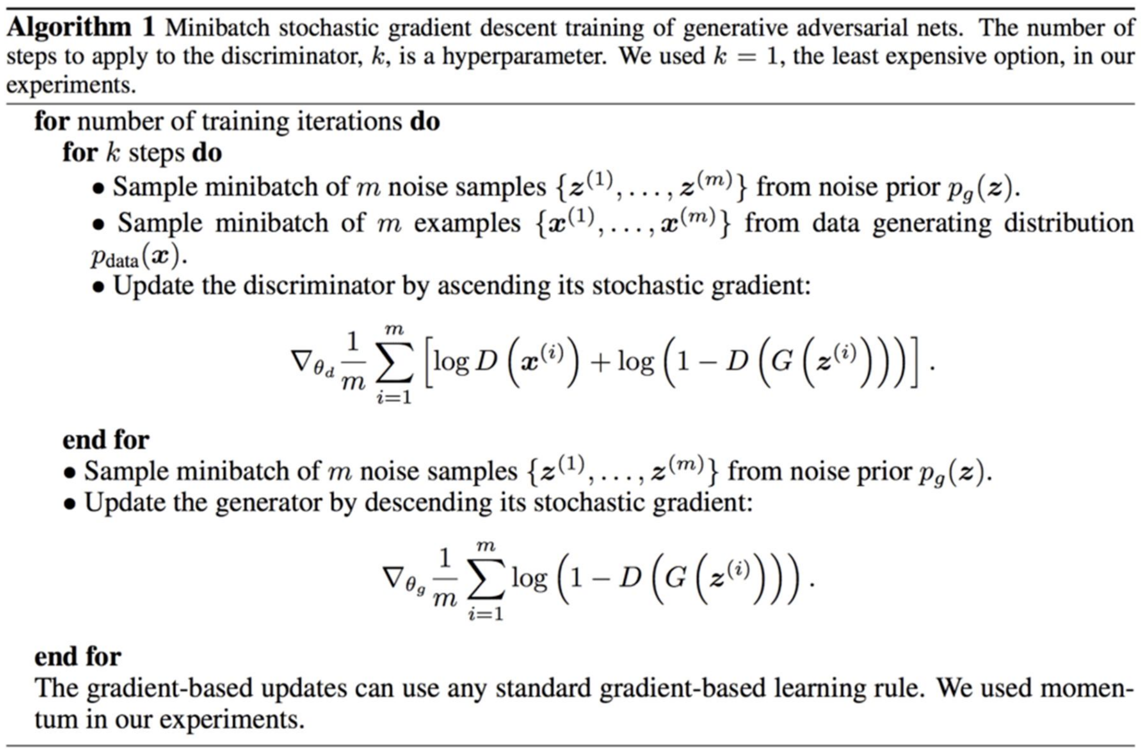
B. Training the Discriminator

Two loss functions are connected to the discriminator. The discriminator ignores the generator loss and only utilises the discriminator loss during training. During generator training, we utilise the generator loss.

During discriminator training, the discriminator sorts the generator's actual and false data. The discriminator suffers a loss if it incorrectly classifies a genuine instance as fake or a fake instance as real. Backpropagation from the discriminator loss across the discriminator network updates the weights of the discriminator.

V. TRAINING OF GAN

The GAN training technique comprises concurrent training of both the discriminator and generator models. The algorithm is summarised in the image below, which is borrowed from Ian Goodfellow and his colleagues 2014 publication titled "Generative Adversarial Networks."



Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

```
for number of training iterations do
    for $k$ steps do
        • Sample minibatch of $m$ noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
        • Sample minibatch of $m$ examples $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
        • Update the discriminator by ascending its stochastic gradient:
            $\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[ \log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$
    end for
    • Sample minibatch of $m$ noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
    • Update the generator by descending its stochastic gradient:
        $\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$
end for
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.
```

A summary of the Generative Adversarial Network Training Algorithm.

The algorithm's outer loop iterates over steps to train the models in the architecture. One cycle through this loop is not an epoch; rather, it is a single update consisting of particular batch modifications to the discriminator and generator models. An epoch is defined as one cycle through a training dataset in which the samples are utilised to update the model weights in mini-batches. A training dataset of 100 samples used to train a model with a mini-batch size of 10 samples, for example, would result in 10 mini-batch updates for each epoch. The model would be appropriate for a specific number of epochs, such as 500. This is frequently concealed from you by automating model training by calling the fit() method and setting the number of epochs and the size of each mini-batch.

A. Convergence

Because the discriminator can't detect the difference between genuine and false, as the generator improves with training, the discriminator's performance deteriorates. The discriminator has a 50% accuracy if the generator succeeds flawlessly. To make its forecast, the discriminator essentially flips a coin. This development causes difficulty for the GAN's overall convergence: the discriminator feedback becomes less useful over time. If the GAN continues to train after the discriminator has given totally random feedback, the generator will begin to train on garbage feedback, and its own quality will deteriorate. Convergence is generally a transitory rather than a permanent condition for a GAN.

VI. GAN LOSS FUNCTIONS

The discriminator has been trained to distinguish between authentic and false pictures. This is accomplished by averaging over each mini-batch of samples the log of the anticipated probability of actual pictures and the log of the inverted probability of false images. Remember that adding log probabilities is the same as multiplying probabilities, but without the possibility of disappearing into small values. As a result, we may interpret this loss function as seeking probability near 1.0 for actual pictures and probabilities near 0.0 for false images, inverted to produce bigger numbers. The combination of these numbers indicates that lower average values of this loss function result in higher discriminator performance.
The loss of the generator model is defined by the GAN method as reducing the log of the inverted probability of the discriminator's prediction of false pictures over a mini-batch. This is simple, but the authors claim that it is ineffective in practice when the generator is weak and the discriminator is skilled at rejecting false pictures with high confidence. The loss function saturates rather than providing appropriate gradient information for the generator to change weights.
Instead, the authors suggest increasing the log of the discriminator's anticipated likelihood of detecting false pictures. The transformation is slight. In the first example, the generator is taught to reduce the likelihood that the discriminator would be right. With this modification to the loss function, the generator is trained to maximise the likelihood that the discriminator will be wrong. This loss function's sign may then be reversed to provide a familiar minimising loss function for training the generator. As a result, this is also known as the -log D technique for training GANs.

A. Loss Functions

GANs are algorithms that attempt to duplicate a probability distribution. As a result, they should utilise loss functions that represent the distance between the GAN-generated data distribution and the distribution of the real data.
In GAN loss functions, how can you represent the difference between two distributions? This is a hotly debated topic, and a variety of techniques have been offered. Here, we'll look at two typical GAN loss functions that are both implemented in TF-GAN:
1) The loss function employed in the study that first described GANs was called minimax loss.
2) The default loss function for TF-GAN Estimators is the Wasserstein loss. A 2017 publication was the first to describe it.
TF-GAN also provides a variety of additional loss functions.

B. Minimax Loss

The generator seeks to reduce the following function, whereas the discriminator strives to maximise it, according to the study that first proposed GANs:

$$E_x [\log(D(x))] + E_z[\log(1-D(G(z)))]$$

In this function:
1) D(x) is the discriminator's assessment of the likelihood of whether a genuine data instance x exists.
2) E_x is the average of all real-world data occurrences.
3) G(z) is the outcome of the generator for input noise z.
4) D(G(z)) is the discriminator's estimation of the likelihood that a false example is genuine.
5) E_z is the estimated value across all provided by particular inputs (the expected value across all produced false instances G(z)).
6) The calculation is based on the difference in entropy between both the real and produced distribution.
7) The generator has no significant influence on the log(D(x)) term.

VII. GANs and Convolution Neural Networks

Convolutional Neural Networks [15], or CNNs, are used as the generator and discriminator models in GANs, which generally deal with picture data. This could be due to the fact that the technique was first described in the field of computer vision and used CNNs and image data, as well as the remarkable progress made in recent years using CNNs more broadly to achieve state-of-the-art results on a variety of computer vision tasks like object detection and face recognition. When the latent space, the generator's input, is used to model picture data, it offers a compressed representation of the collection of images or photographs used to train the model. It also implies that the generator creates fresh pictures or photographs, resulting in an output that developers or users of the model can readily examine and evaluate. It's possible that this characteristic, above all others, the capacity to visually judge the quality of the produced output, has both led to the emphasis of computer vision applications with CNNs and to the huge jumps in capabilities of GANs when compared to other generative models, deep learning-based or not.

VIII. VARIATIONS OF GANS

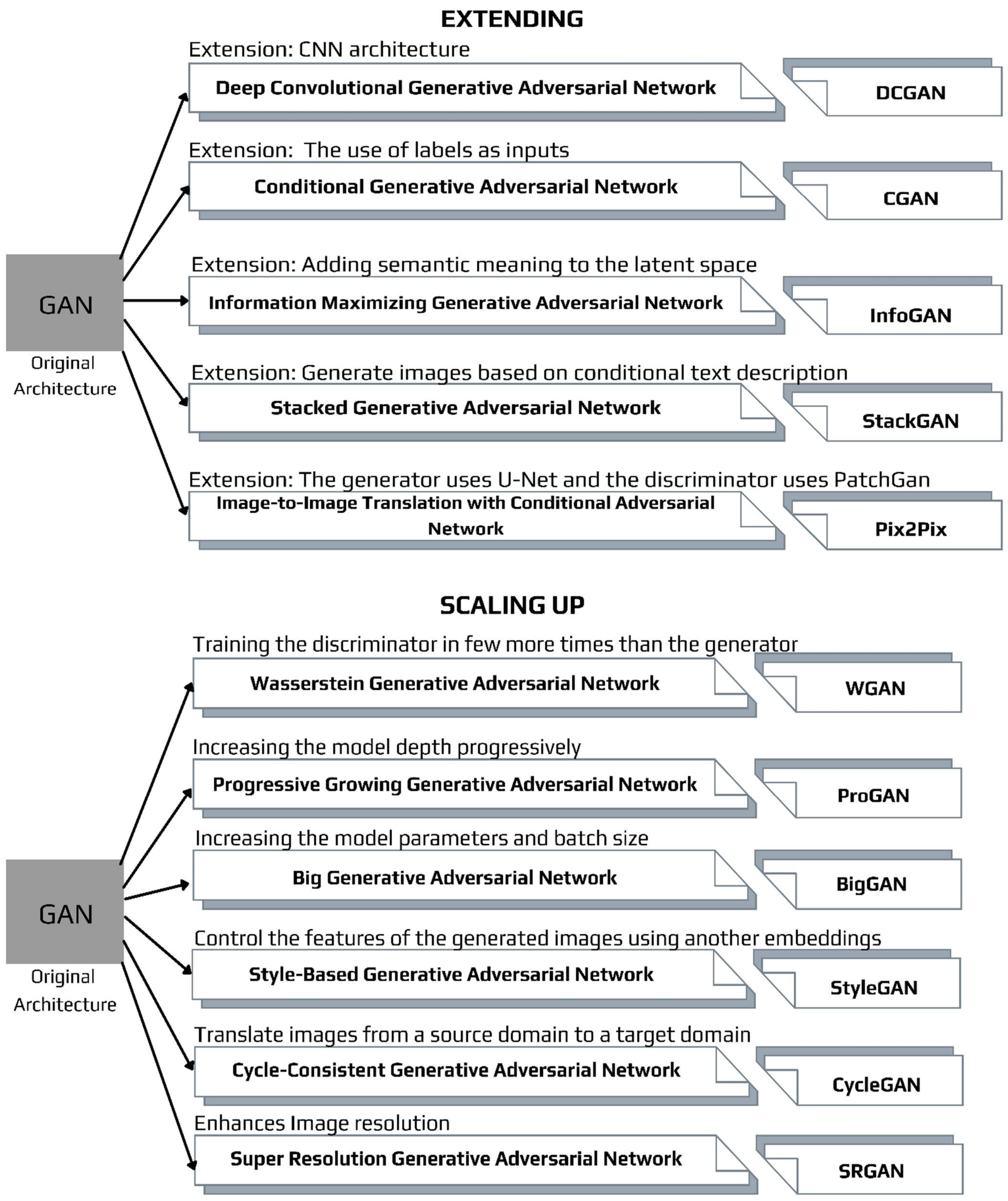

EXTENDING
GAN
Original Architecture
Extension: CNN architecture
Deep Convolutional Generative Adversarial Network
DCGAN
Extension: The use of labels as inputs
Conditional Generative Adversarial Network
CGAN
Extension: Adding semantic meaning to the latent space
Information Maximizing Generative Adversarial Network
InfoGAN
Extension: Generate images based on conditional text description
Stacked Generative Adversarial Network
StackGAN
Extension: The generator uses U-Net and the discriminator uses PatchGan
Image-to-Image Translation with Conditional Adversarial Network
Pix2Pix
SCALING UP
GAN
Original Architecture
Training the discriminator in few more times than the generator
Wasserstein Generative Adversarial Network
WGAN
Increasing the model depth progressively
Progressive Growing Generative Adversarial Network
ProGAN
Increasing the model parameters and batch size
Big Generative Adversarial Network
BigGAN
Control the features of the generated images using another embeddings
Style-Based Generative Adversarial Network
StyleGAN
Translate images from a source domain to a target domain
Cycle-Consistent Generative Adversarial Network
CycleGAN
Enhances Image resolution
Super Resolution Generative Adversarial Network
SRGAN

Figure 8: Types of GAN Model architectures and extensions

A. Deep Convolutional GAN

DCGAN [16] is an expansion of the GAN system that uses deep convolutional neural networks for both the generator and discriminator models, as well as model and training settings that result in robust training of a generator model. DCGANs employ stride and fractionally stride convolutions which are basic blocks of convolution that are used in CNN. This enables the model to learn about the operators used in up sampling and down sampling networks all through the training. Up sampling is expanding the conceptual images representations through multiple techniques to keep the spatial dimensions comparable to the input data. Down sampling is the loss of spatial resolution yet maintaining the same two-dimensional image representation.

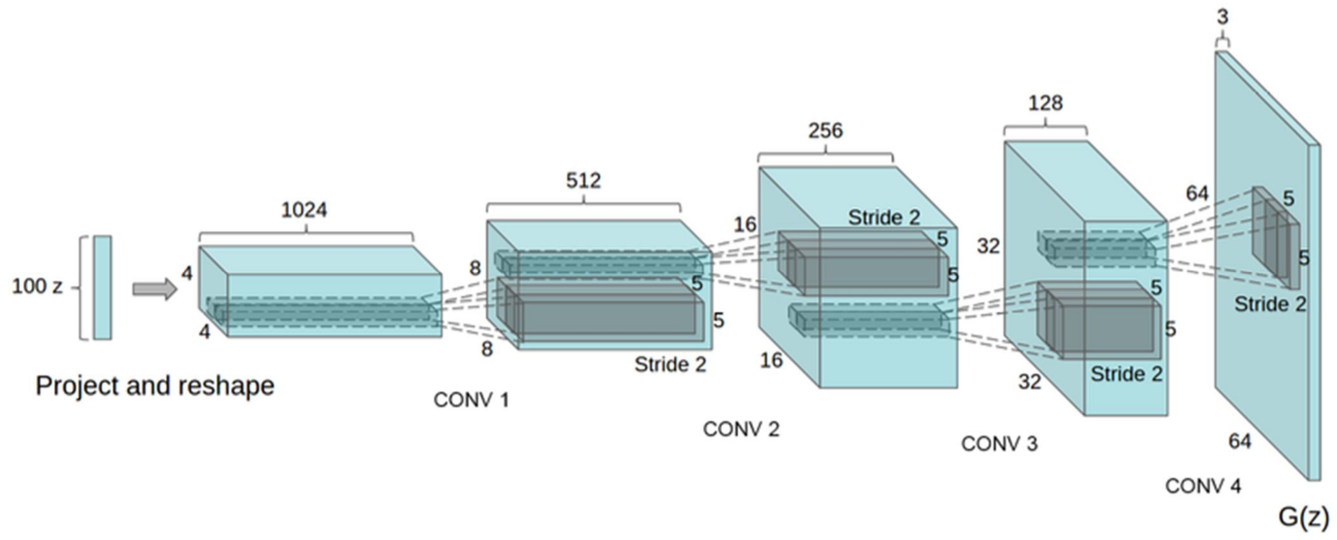


Figure 9: The DCGAN generator was used to simulate the LSUN scenario. A 100-dimensional uniform distribution Z is projected to a limited spatial extent convolutional representation with numerous feature maps. This high-level representation is then converted into a 64 × 64-pixel picture via a sequence of four fractionally-strided convolutions (in some recent studies, they are incorrectly called deconvolutions). It is worth noting that no completely linked or pooling layers are employed.

B. Conditional GAN

Conditional GANs [17] (cGANs) are a type of network where both the generator and discriminator are conditioned by extra information during training. cGANs modify the network by introducing the label y as an extra argument to the generator so that the generator can produce matching pictures. Labels are also fed to the discriminator to improve the discrimination between real and fake samples. Conditional GANs use a labelled data set to train and allow to choose the label for each produced instance. An unconditional MNIST GAN, for example, will create random digits, but a conditional MNIST GAN will allow us to select which digit the GAN should generate. Conditional GANs represent the conditional probability P (X | Y) instead of the joint probability P (X, Y).

The use of the GAN for conditionally producing an output is a significant extension. The generative model may be trained to create new instances from the input domain, where the input, a random vector from the latent space, is supplemented (conditioned by) some extra information. In the case of producing photos of handwritten numbers, the extra input may be a class value, such as male or female in the case of generating photographs of humans, or even a figure in the event of photographing handwritten characters. If both the generator and the discriminator are conditioned on some additional information y, such as class labels or input from other modalities, generative adversarial networks can be expanded to a conditional model. We may do the conditioning by adding y as an additional input layer to both the discriminator and the generator.

The discriminator is additionally conditioned, which means it is given both a genuine and a false input picture as well as the additional input. The discriminator would therefore anticipate the input to be of that class in the case of a classification label type conditional input, training the generator to create instances of that class in order to mislead the discriminator. A conditional GAN may be used in this way to produce examples from a domain of a specific type.

GAN models can be conditioned on a domain exemplar, such as a picture, if taken a step further. This enables GANs to be used in applications such as text-to-image or image-to-image translation. This enables GANs to do some of their most spectacular tasks, such as style transfer, photo colourization, and photo transformations from summer to winter or day to night, among others.
When using conditional GANs for image-to-image translation, such as converting day to night, the discriminator is fed samples of actual and produced night-time pictures, as well as (conditioned on) real daytime photos. A random vector from the latent space, as well as (conditioned on) real daytime photographs, are fed into the generator.

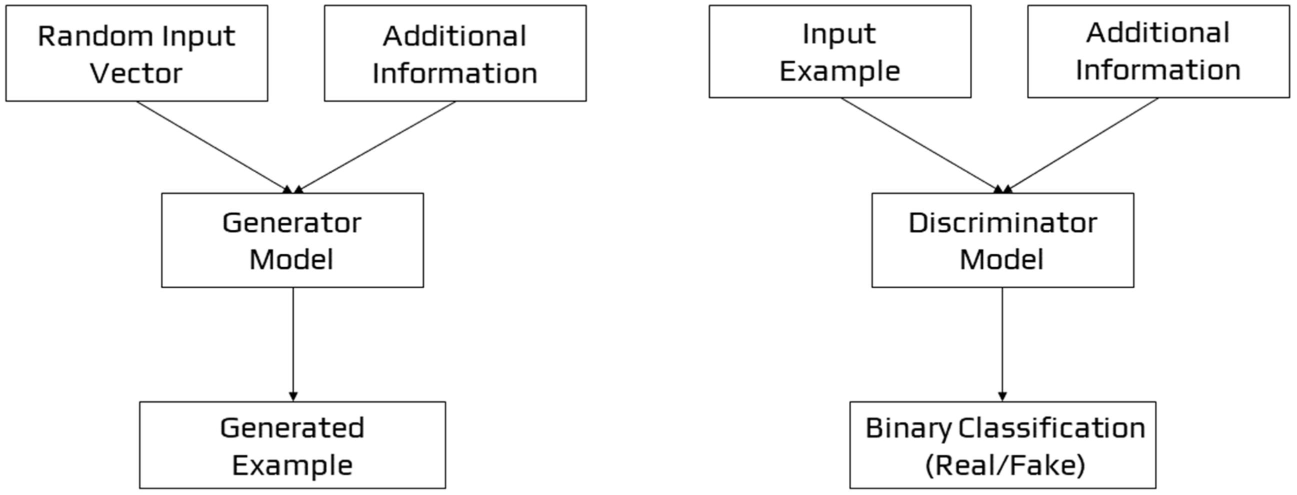


Figure 10: Conditional GAN Model Architecture

C. Information Maximizing GAN

Information Maximizing GAN [18] is a GAN derivative. It's an algorithm for learning unsupervised representations. This network magnifies the mutual information between the input noise vector and the latent code, which are basically variables that aren't observed during the training phase and the test phases. InfoGan solves the problem of entangled representations and gives a disentangled one. It separates the Generator input noise vector in two: the conventional noise vector and a new 'latent code' vector. Later, by maximising the mutual information between the code and the generator output, the latent code is made significant and are used to condition or control specific semantic properties in the generated image. The addition of the component which maximises the mutual information among the generative model's latent code input and its output, leads to the disentanglement of the significant features and assigns them to the enforced latent code space.

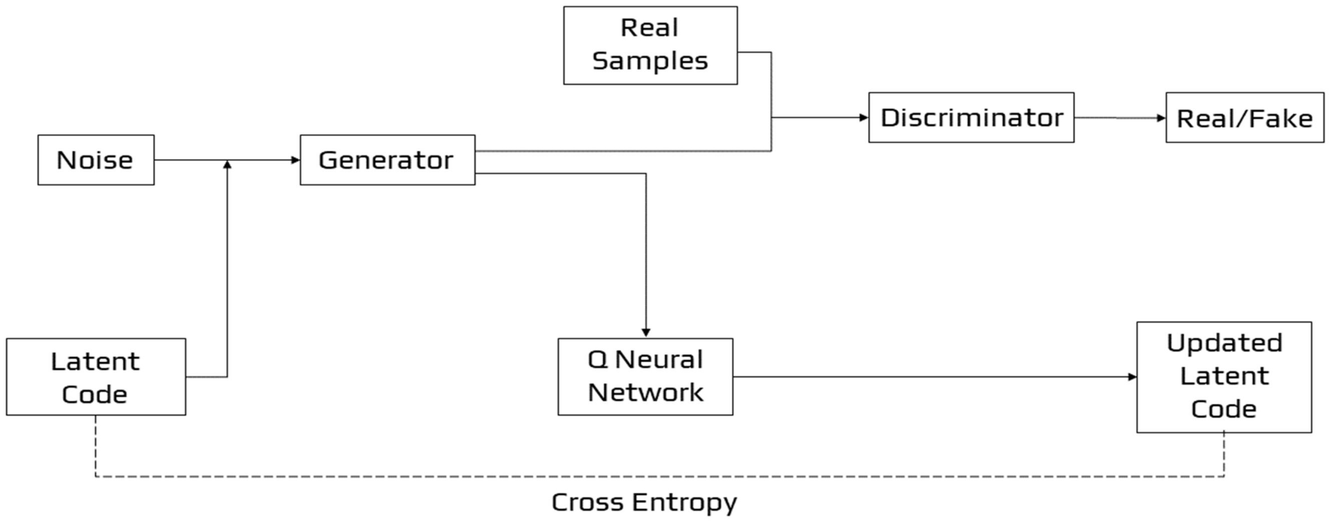


Figure 11: InfoGAN

D. Stacked GAN

The stacked generative adversarial network, or StackGAN [19][20], is a GAN modification that uses a hierarchical stack of conditional GAN networks to create pictures from words. The structure is made up of conditional GAN models. There are two generators, the first one is text conditioned and produces a poor resolution image. The second one is regulated on the text and the output of the first generator; it produces a high-resolution picture. Our SGAN breaks down variations into several layers and gradually eliminates uncertainties in the top-down generating process, unlike the original GAN, which utilises a single noise vector to represent all variations.

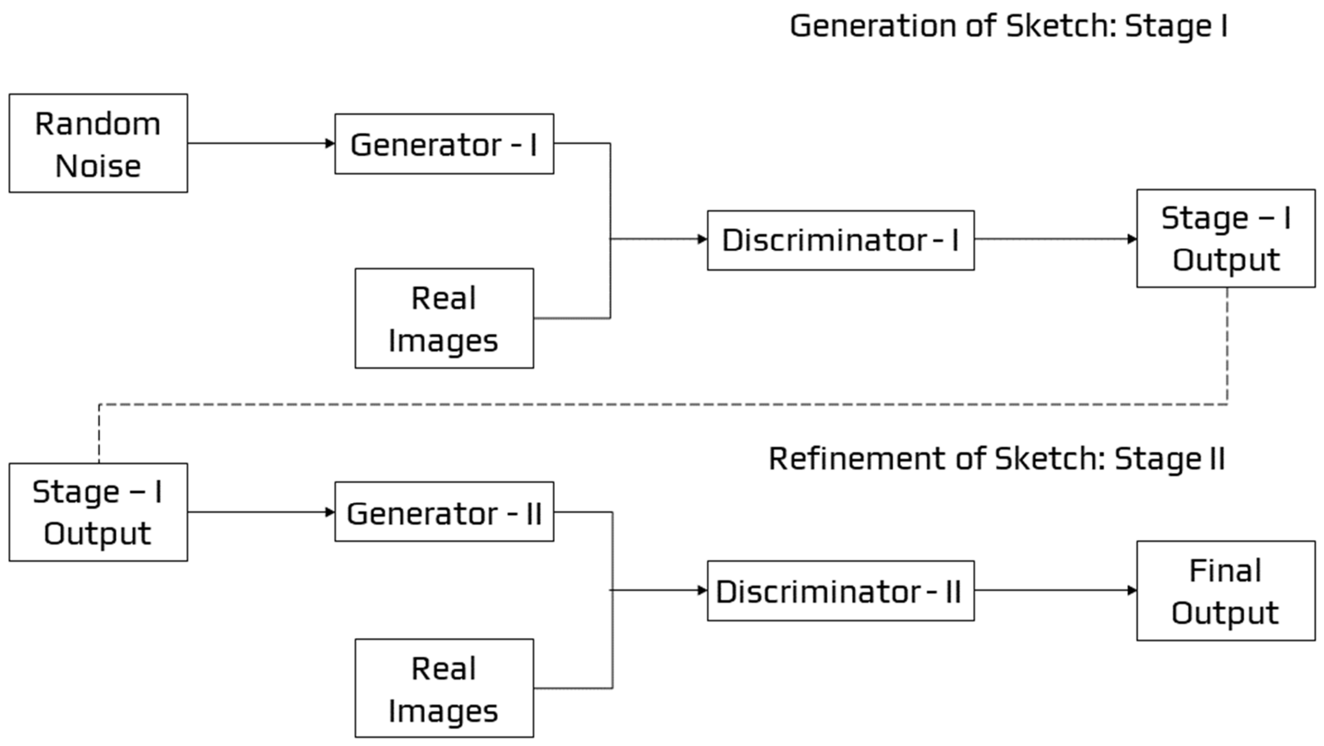


Figure 12: StackGAN

E. Pix2Pix

The Pix2Pix [21] GAN is a method for teaching convolutional neural networks to do an image-to-image conversion. When compared to previous GAN models (e.g., 256256 pixels), the precise setup of architecture as a sort of image-conditional GAN provides for both the creation of big pictures and the capacity to perform well on a range of image-to-image machine translation. Pix2Pix is a form of conditional GAN, or cGAN, in which the output picture creation is dependent on input, in this instance a source image. An original picture and a target image are supplied to the discriminator, who must decide if the target is a reasonable translation of the input images.

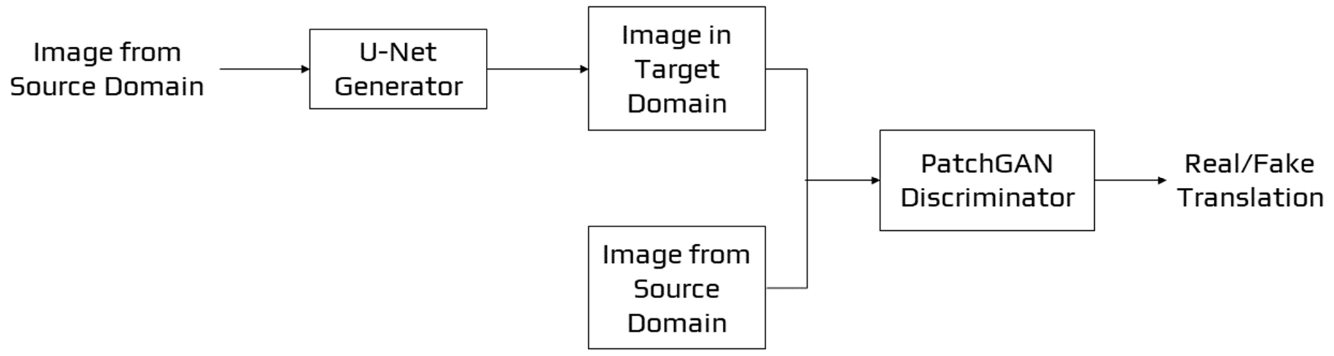


Figure 13: Pix2Pix

F. Wasserstein GAN

Wasserstein GAN [22] or WGAN is a GAN extension that looks for a different approach to train the generator model so that it can better mimic the distribution of data seen in a particular training dataset. For each iteration, it modifies the training method to update the discriminator model, several times more than the generating model. The discriminator is revised to produce a real-value (linear activation) rather than binary forecasting with a sigmoid function. Both the generator and the discriminator are trained with "Wasserstein loss". It is the average of the product of real and estimated values from the discriminator to give linear gradients useful for updating the model.

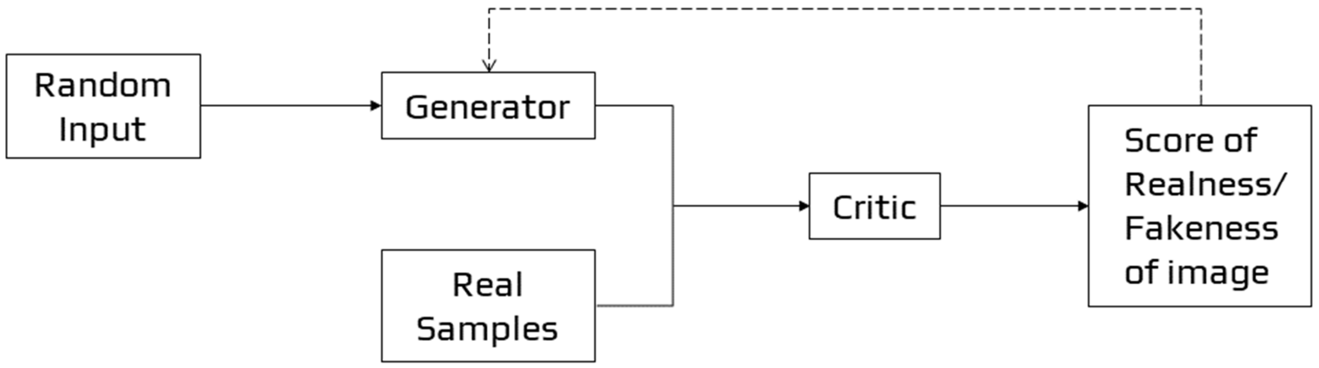


Figure 14: Wasserstein GAN

G. Progressive Growing GAN

Progressive Growing GAN [23] is a GAN training method enhancement that enables for the steady training of generator networks capable of producing huge, high-quality pictures. The process consists of increasing the size of the model gradually for a very small picture. This will result in a rise in the output size of the generator and the input size of the discriminator. It will continue till the required picture size is obtained. The earliest layers of a progressive GAN create relatively low-quality pictures, but later layers add details. This method allows the GAN to train faster than non-progressive GANs while also producing higher-resolution pictures.

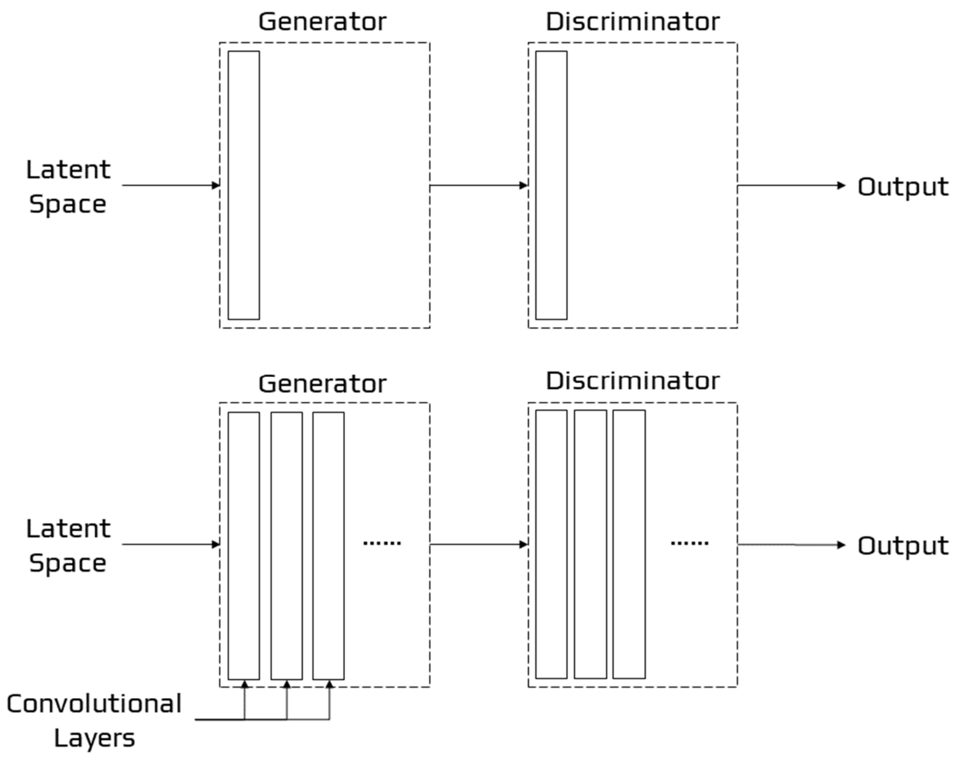


Figure 15: ProGAN

H. BigGAN

Big Generative Adversarial Networks [24] is a method for demonstrating how existing class conditional models may be scaled up to produce high-quality output pictures. BigGAN is a GAN extension; the idea is to increase the size of the batches while also increasing the amount of parameters. As a result, high-quality, high-resolution images are generated. Modifying the architecture and training procedures, for example, will allow GANs to be scaled up.

I. StyleGAN

The StyleGAN [25][26], or style-based generative adversarial network, is a generator modification that enables the latent code to be utilised as input at various stages during the model to influence aspects of the produced picture. Rather than using the latent space point as input, the point is passed via a deep embedding system before it can be used as input at numerous places in the generator model. Along with the embedding network's output, noise is also included.

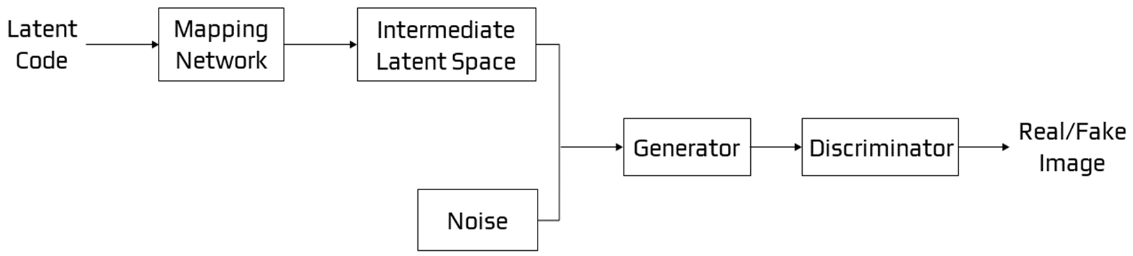


Figure 16: StyleGAN

J. CycleGAN

CycleGANs [27] are generative adversarial networks with two generators and two discriminators. Every generator has its discriminator that aims to differentiate its generated pictures from genuine ones.

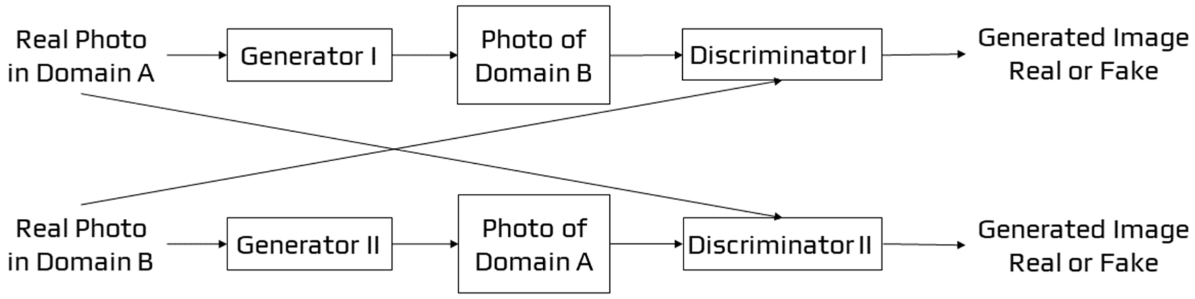
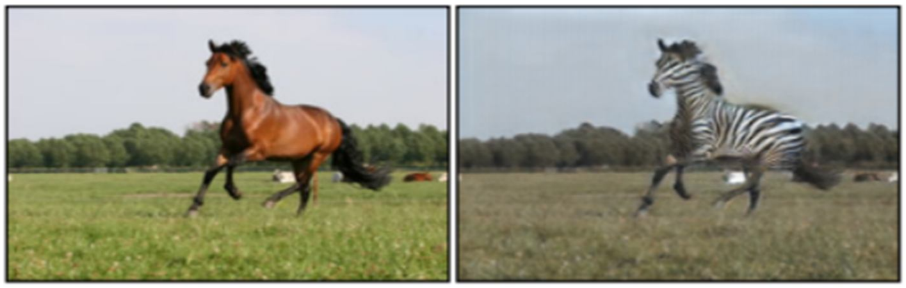


Figure 17: CycleGAN

CycleGANs learn to convert pictures from one set into images that may be from a different collection. When given the left-hand picture as input, a CycleGAN created the right-hand image below. It took a horse image and transformed it into a zebra image. The CycleGAN's training data consists of just two picture sets (in this case, a set of horse images and a set of zebra images). No labels or pairwise picture correspondences are required by the system.



(A) Input Image (B) Output Image

Figure 18

K. Super-Resolution GAN

Super-Resolution GAN [28], like other GAN designs, is separated into two sections: the generator and the discriminator. The generator generates data based on a probability distribution, and the discriminator attempts to predict if the data came from the input samples or generated samples.

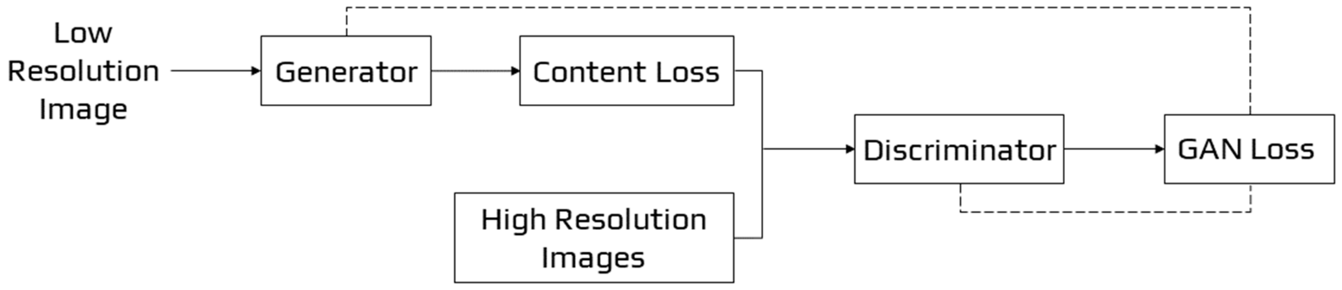
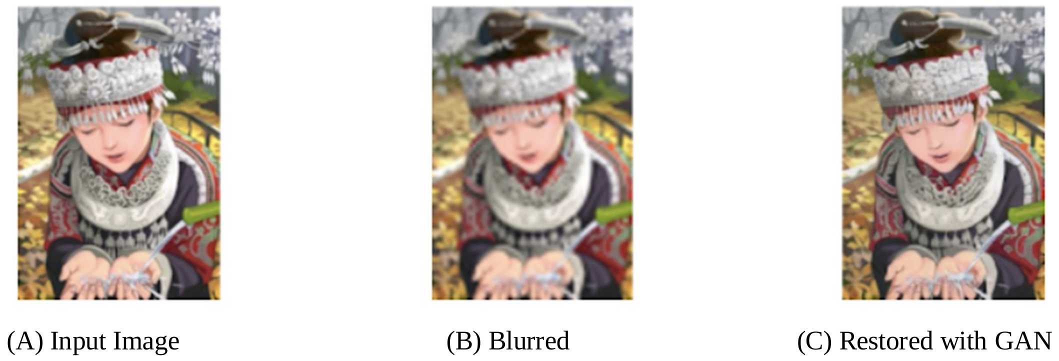


Figure 19: SRGAN

To generate better quality pictures, SRGAN employs a network model in association with an adversarial network. GANs with super-resolution boost picture resolution by adding detail where it's needed to fill up hazy regions. The fuzzy middle picture on the right, for example, is a down-sampled reproduction of the original image on the left. A GAN created the clearer picture on the right from the fuzzy image:



(A) Input Image (B) Blurred (C) Restored with GAN

Figure 20

Although the GAN-generated picture appears to be extremely similar to the original, a careful examination of the headband reveals that the GAN did not replicate the original's starburst pattern. Instead, it created its own convincing pattern to replace the down-sampled pattern.

IX. APPLICATIONS OF GANS

Exploring recent advances for adversarial deep network training is an ongoing focus of study. Some examples of applications were selected to demonstrate some diverse methods to employ GAN-based representations for picture modification, analysis, or characterisation, and thus do not completely represent the possible range of use of GANs.

Image synthesis [29] is one of the core application GAN functions and it is used when there are some existing conditions for the produced images. Image to Image translation [30] shows the potential of GANs by translating input data into an output image. Super-resolution using GAN demonstrates the use of an existing technique that can be enhanced with supplement loss functions to generate high-quality outcomes.

A. Image Synthesis

Recently, the research regarding image synthesis with GAN [31] mostly focuses on the quality of the generated images. We can distinguish three approaches for image synthesis: direct, hierarchical, and iterative.

Direct methods are methods in which the models contain a generator G and discriminator D and have a simple architecture with fewer connections. Of these, DCGAN is one of the most traditional, with a framework that is adopted by several subsequent models.

In DCGAN, G adopts transposed convolution, batch-normalization and ReLU activation whereas D employs convolution, batch-normalization and Leaky ReLU activation. As its name indicates, this method is very direct for designing and implementing. DCGAN was implemented for the generation of apparel pictures for inclusion in the Fashion MNIST dataset. In this case, the noise was given as input to the generator, then moulded to produce a low image resolution. To obtain the final picture, this was up-sampled using Conv2DTranspose. The discriminator had an architecture based on CNN supervised learning for classifying the generated image. Compared to the direct method, the hierarchical approach uses two pairs of generators and discriminators. Both the generators and discriminators have different purposes. The two generators' relationship might be either parallel or sequential. In the case of Structure-Style GAN, it contains two GANs, one GAN to create a surface map from latent space and the other one to take both the created surface normal map and a noise vector as input and produce an image.
The Iterative method is different from the Hierarchical method. This method uses several generators with similar structures and the generators produce images from rough to detailed, with the generators improving the previous output each and every time. Furthermore, for the architecture of the generators, shared weights can be employed between the generators. This however is not possible in the case of hierarchical models. For example, Text-to-Image synthesis, the text is used as input and the model generates pictures that are believable and accurately represented by the text. For text-to-image synthesis, StackGAN suggests using two distinct generators. The first generator produces low-quality images with rough forms and colors of objects. The second one takes as input the outcome of the previous generator and generates high-quality images. The floral image below, for example, was created by feeding a written description to a GAN.

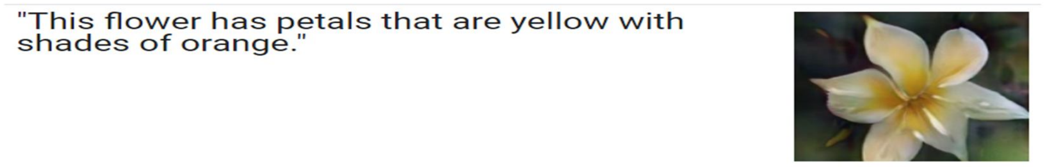
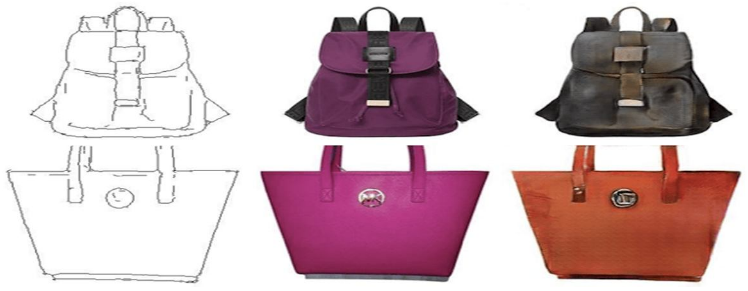

"This flower has petals that are yellow with shades of orange."

Figure 21: Caption - The floral image above was created by feeding a written description to a GAN.

B. *Image-to-Image Translation*

The challenge of converting a potential representation of one image into the other, such as transforming black and white pictures into RGB images, or the other way around, is characterised as image-to-image translation. Image-to-image is not limited only to translating the images but also changing and modifying the characteristics of the images. GANs take an input image and map it to a produced output image with various characteristics. Based on the data, we can say that there are two types of translation, supervised and unsupervised.
In Supervised translation, there are paired images in several domains. It means that for each image of the source domain, there exists a corresponding image in the target domain. Pix2Pix merges the loss of a cGAN with the loss of L1 regularization for the generator to fool the discriminator but also produce actual, real images. Though Pix2Pix generates highly amazing synthetic pictures, its main restriction is that it must take coupled photos as supervision, because the data pair (x, y) is taken from the joint distribution p (x, y). Another image-to-image translation case is the White Box Cartoonization using extended GAN framework [32][33]. This network outputs a cartooned image from a given photo. The image is separated in three representations: surface, structure and texture representations. The first one smoothens the surface of the image, the second one segments the image and create a segmentation map. The third one focuses on the colors and the texture as it name indicates. The output is a high resolution and high-quality cartooned image which is very similar to the original photo, yet holds the characteristics of a cartooned picture.



(A) Input Images (B) Real Images (C) Output Images
Figure 22

In an Unsupervised setup [34][35], there are two independent datasets: one has images from one domain while the other has images from another. Consequently, there is no paired data. This poses a challenge as there is no way of showing how the image could be translated into its respective image of another domain. The goal of unsupervised image-to-image translation is to learn a joint distribution of pictures in distinct domains by taking pictures from the probability distribution in each domain. Adversarial Open Domain Adaptation for sketch-to-photo synthesis [36] is a framework that has dealt with unsupervised translation. It uses two generators, G1 translates photo to sketch and G2 translates sketch to photo depending on the input tag, and two discriminators D1 and D2 for drawing and photo domains.

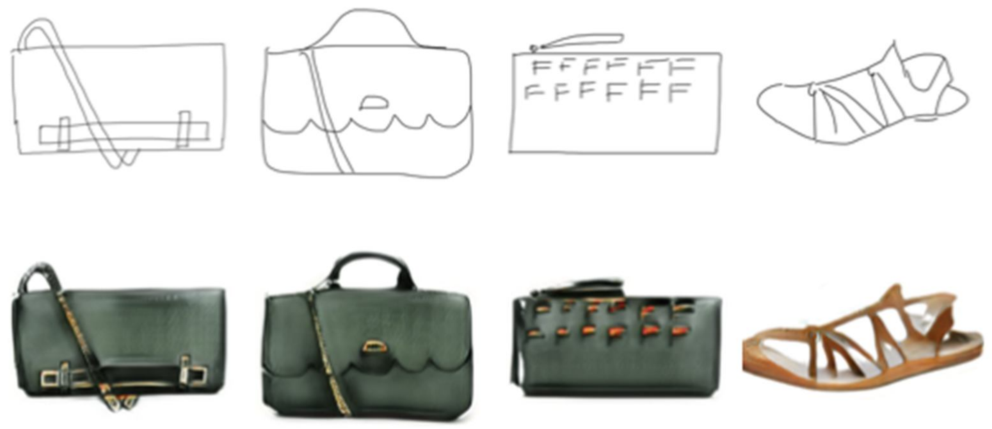


Figure 23: Sketch-to-Photo Synthesis

X. CONCLUSIONS

GANs are a type of game-theoretic generative model. They've had a lot of success in the field of creating realistic data, particularly pictures. Training them is still a challenge. It will be required to create models, prices, or training algorithms that can consistently and rapidly identify appropriate Nash equilibria for GANs in order for them to become a more dependable technology. GANs have inspired a surge of interest due to their capacity to learn deep, highly non-linear mappings from a latent space to data space and back, as well as their ability to handle huge amounts of unlabelled picture data that are inaccessible to deep representation learning. There are several chances for theoretical and algorithmic advancements within the intricacies of GAN training, and with the strength of deep networks, there are numerous opportunities for new applications. By creating their own representations of the data they are trained on, GANs produce organised geometric vector spaces for a variety of domains. GANs may train representations that can be used for a variety of tasks, including image synthesis, semantic image editing, style transfer, image super-resolution, and classification, to name a few applications.

REFERENCES

[1] Thakur, Amey, and Archit Konde. "Fundamentals of Neural Networks", Volume 9, Issue VIII, International Journal for Research in Applied Science and Engineering Technology (IJRASET) Page No: 407-426. https://doi.org/10.22214/ijraset.2021.37362
[2] Goodfellow, Ian, et al. "Generative adversarial nets." Advances in neural information processing systems 27 (2014).
[3] Goodfellow, Ian. "Nips 2016 tutorial: Generative adversarial networks." arXiv preprint arXiv:1701.00160 (2016).
[4] CRESWELL, Antonia, WHITE, Tom, DUMOULIN, Vincent, et al. Generative adversarial networks: An overview. IEEE Signal Processing Magazine, 2018, vol. 35, no 1, p. 53-65.
[5] Liu, Ming-Yu, and Oncel Tuzel. "Coupled generative adversarial networks." Advances in neural information processing systems 29 (2016): 469-477.
[6] Jiang, Tammy, Jaimie L. Gradus, and Anthony J. Rosellini. "Supervised machine learning: a brief primer." Behavior Therapy 51.5 (2020): 675-687.
[7] Gentleman, Robert, and Vincent J. Carey. "Unsupervised machine learning." Bioconductor case studies. Springer, New York, NY, 2008. 137-157.
[8] Leung, K. Ming. "Naive bayesian classifier." Polytechnic University Department of Computer Science/Finance and Risk Engineering 2007 (2007): 123-156.
[9] Rasiwasia, Nikhil, and Nuno Vasconcelos. "Latent dirichlet allocation models for image classification." IEEE transactions on pattern analysis and machine intelligence 35.11 (2013): 2665-2679.
[10] Reynolds, Douglas A. "Gaussian mixture models." Encyclopedia of biometrics 741 (2009): 659-663.
[11] Larochelle, Hugo, et al. "Learning algorithms for the classification restricted boltzmann machine." The Journal of Machine Learning Research 13.1 (2012): 643-669.
[12] Krizhevsky, Alex, and Geoff Hinton. "Convolutional deep belief networks on cifar-10." Unpublished manuscript 40.7 (2010): 1-9.
[13] Mescheder, Lars, Sebastian Nowozin, and Andreas Geiger. "Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks." International Conference on Machine Learning. PMLR, 2017.

[14] Adigun, Olaoluwa, and Bart Kosko. "Training generative adversarial networks with bidirectional backpropagation." 2018 17th IEEE international conference on machine learning and applications (ICMLA). IEEE, 2018.
[15] Albawi, Saad, Tareq Abed Mohammed, and Saad Al-Zawi. "Understanding of a convolutional neural network." 2017 International Conference on Engineering and Technology (ICET). Ieee, 2017.
[16] Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv preprint arXiv:1511.06434 (2015).
[17] Mirza, Mehdi, and Simon Osindero. "Conditional generative adversarial nets." arXiv preprint arXiv:1411.1784 (2014).
[18] Chen, Xi, et al. "Infogan: Interpretable representation learning by information maximizing generative adversarial nets." Proceedings of the 30th International Conference on Neural Information Processing Systems. 2016.
[19] Huang, Xun, et al. "Stacked Generative Adversarial Networks." CVPR. Vol. 2. 2017.
[20] Zhang, Han, et al. "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks." Proceedings of the IEEE international conference on computer vision. 2017.
[21] Isola, Phillip, et al. "Image-to-image translation with conditional adversarial networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.
[22] Arjovsky, M., Chintala, S., & Bottou, L. (2017, July). Wasserstein generative adversarial networks. In International conference on machine learning (pp. 214-223). PMLR.
[23] Karras, Tero, et al. "Progressive growing of gans for improved quality, stability, and variation." arXiv preprint arXiv:1710.10196 (2017).
[24] Brock, Andrew, Jeff Donahue, and Karen Simonyan. "Large scale GAN training for high fidelity natural image synthesis." arXiv preprint arXiv:1809.11096 (2018).
[25] X. Wang and A. Gupta, "Generative image modelling using style and structure adversarial networks," arXiv preprint arXiv:1603.05631, 2016.
[26] Karras, T., Laine, S., & Aila, T. (2019). A style-based generator architecture for generative adversarial networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 4401-4410).
[27] Zhu, Jun-Yan, et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks." Proceedings of the IEEE international conference on computer vision. 2017.
[28] Ledig, Christian, et al. "Photo-realistic single image super-resolution using a generative adversarial network." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.
[29] Huang, He, Philip S. Yu, and Changhu Wang. "An introduction to image synthesis with generative adversarial nets." arXiv preprint arXiv:1803.04469 (2018).
[30] Isola, P., Zhu, J. Y., Zhou, T., & Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1125-1134).
[31] Shamsolmoali, Pourya, et al. "Image synthesis with adversarial networks: A comprehensive survey and case studies." Information Fusion (2021).
[32] Thakur, Amey, Hasan Rizvi, and Mega Satish. "White-Box Cartoonization Using An Extended GAN Framework." arXiv preprint arXiv:2107.04551 (2021)
[33] Thakur, Amey, Hasan Rizvi, and Mega Satish. "WHITE-BOX CARTOONIZATION USING AN EXTENDED GAN FRAMEWORK.", Volume 5, Issue 12, International Journal of Engineering Applied Sciences and Technology (IJEAST), Page No: 294-298, 2021.
[34] Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv preprint arXiv:1511.06434 (2015).
[35] Liu, Ming-Yu, Thomas Breuel, and Jan Kautz. "Unsupervised image-to-image translation networks." Advances in neural information processing systems. 2017.
[36] Xiang, Xiaoyu, et al. "Adversarial Open Domain Adaption for Sketch-to-Photo Synthesis." arXiv preprint arXiv:2104.05703(2021).