

Synthetic Image Generation using Generative Adversarial Network (GAN)

Shakti Zanzari¹, Vrushali Bongirwar², Sarang Binalwar³, Varun Mantri⁴ and Parth Tammiwar⁵

¹⁻⁵Shri Ramdeobaba University, Nagpur, India

Email: {zanzariss, bongirwarvk, binalwarv, mantrivv, tammiwarpa}@rknec.edu

Abstract—In recent years, Generative Adversarial Networks (GANs) have emerged as a groundbreaking technique inside the realm of system getting to know, in particular for the technology of synthetic information. Their capacity to research complicated facts distributions and generate high-constancy samples has fueled huge research and innovation in diverse domain names.

This summary delves deeper into the mechanisms of GANs, elucidating how the interaction between the generator and discriminator networks results in the introduction of synthetic images that closely resemble real statistics. GANs continuously enhance the greatness of generated samples, pushing the boundaries of what is viable in the artificial facts' era.

Furthermore, this work explores the potential of visualization data in a variety of industries. Our model achieved more than 90% accuracy in generating the images. The model has wide range of application with a lot of future scope and we are still finding new things every day. It is invented by Ian Goodfellow in 2014 and it is making its place in our day-to-day life. In healthcare, artificial intelligence can facilitate training on medical imaging models without compromising patient privacy. It can also help with risk assessment and fraud detection in the financial sector. It can improve monitoring systems and forensic investigations in security. Moreover, the generation of images has important applications in creative fields such as design and entertainment. This includes creating anime-like visuals, creating realistic faces, and creating unique artwork. By harnessing the potential of GANs, researchers and practitioners can unlock new avenues for innovation and problem solving in a wide range of domains.

I. INTRODUCTION

Image data generation using Generative Adversarial Networks (GANs) is still playing a vital role in Artificial Intelligence. The purpose of this model is to explore the basic principles of synthetic image data processing, including generative models, GANs, and their broader applications. Synthetic image data generation involves the creation of images that closely resemble real-world characteristics and patterns. Instead of taking images directly from the environment, these synthetic images are created using computational techniques. This approach is particularly valuable in situations where obtaining large amounts of real-world data is difficult, expensive, or impossible. Generative models work on a set of machine learning algorithms built to understand and reconstruct the probability distributions that appear in a dataset. Unlike discriminant models, which predict primarily labels or classes for input data, generative models seek to generate data samples that reflect the properties of the original dataset. Generative models play a very important role in generating diverse and realistic data representation.

Generative Adversarial Networks (GANs) are a type of generative version introduced via Ian Goodfellow and his colleagues in 2014. GANs encompass two neural networks: a Generator and a Discriminator. The Generator generates artificial information samples, while the Discriminator evaluates whether or not those samples are real or fake. GANs are trained using an adverse process, where the generator and discriminator compete against each other, leading to the technology of more and more realistic artificial data

The generator in a GAN takes random noise as input and generates synthetic statistics samples, together with snapshots, based totally on this noise. The discriminator network, however, takes each real and synthetic records samples as input and predicts whether or not every sample is actual or fake. During training, the generator aims to provide artificial facts that fool the discriminator into believing it's miles real, whilst the discriminator ambitions to distinguish between real and fake information. These are the adversaries of each other.

GANs regularly employ Convolutional Neural Networks (CNNs) as constructing blocks for both the generator and discriminator networks. CNNs are well-desirable for picture- associated responsibilities due to their potential to capture spatial hierarchies and neighbourhood styles in photographs successfully.

By leveraging CNNs, GANs can learn to generate fairly practical and targeted synthetic images that showcase traits much like real images.

Conditional Generative Adversarial Networks (CGANs) allow focused output generation based totally on particular conditions or enter facts. Both the generator and discriminator are conditioned on extra data, including class labels or auxiliary records, bearing in mind-controlled synthesis. CGANs also can circumstance on examples from the domain, facilitating tasks like image-to-photo translation, together with style switch or altering environmental elements. For instance, in reworking day scenes to night scenes, the discriminator is supplied with actual and generated middle of the night pics, along with actual daylight snap shots. The generator receives a random vector and actual sunlight hours photographs as conditioned input, ensuring coherence and realism within the generated midnight images.

II. LITERATURE REVIEW

Researchers have been exploring various methodologies for different types of Generative Adversarial Networks (GANs). One promising approach is the 3D-GAN by Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T. Freeman, and Joshua B. Tenenbaum, which tackles 3D object creation. This method excels in producing high-quality 3D objects without requiring reference data and achieves impressive recognition accuracy, exceeding 80% in tests. It essentially grasps object structure implicitly, allowing for independent generation and recognition.

Another area of study involves Ensembled GANs by Gabriel Eilertsen, Apostolia Tsirikoglou, Claes Lundstrom, and Jonas Unger, which focus on overcoming data limitations, particularly in fields with ethical or privacy concerns, like medical imaging. Here, the research highlights the effectiveness of combining multiple GANs to create a richer data pool. This ensemble approach surpasses traditional boosting strategies and offers a more robust representation.

Furthermore, for tasks involving image generation on common datasets like SVHN and CIFAR-10, ensembled GANs are expected to achieve similar benchmarks as established models, which can range from 90% to 95% accuracy. This suggests that ensembled GANs can not only handle data scarcity but also perform competitively on established datasets.

III. METHODOLOGY

The methodology section outlines the systematic approach adopted to design, develop, and implement the Image Generation System. The process involves data collection, the implementation of machine learning models, testing the model and evaluation. Each step is meticulously planned to ensure the accuracy of generated images, and the seamless user experience.

Fig 1 is the basic flow of the model. First the images are transformed, and the generator and the discriminator are trained using the GPU. They produce an output after each epoch for which loss is calculated and the networks are updated.

A. Data Collection

- **Parameter Selection:** The first step in the data collection process is to select or create a database in which there is not much distortion between images i.e. images do not differ too much from each other. For the dataset which have distinct images we first have to classify the images according to their label and then train the generator and the discriminator.

- **Data Input Mechanism:** To ensure seamless experience we use 'opendatasets' library as od.opendatasets is a library that simplifies the process of downloading datasets from various online sources, including Kaggle. The `od.download(dataset_url)` function call is used to download the dataset. This function will handle the authentication (if required) and download the dataset to the current directory or a specified location.
- **Dataset Used:** The datasets that we have used are Celeb - Face dataset, Anime face dataset, Best artwork of all time, jellyfish-types from Kaggle.

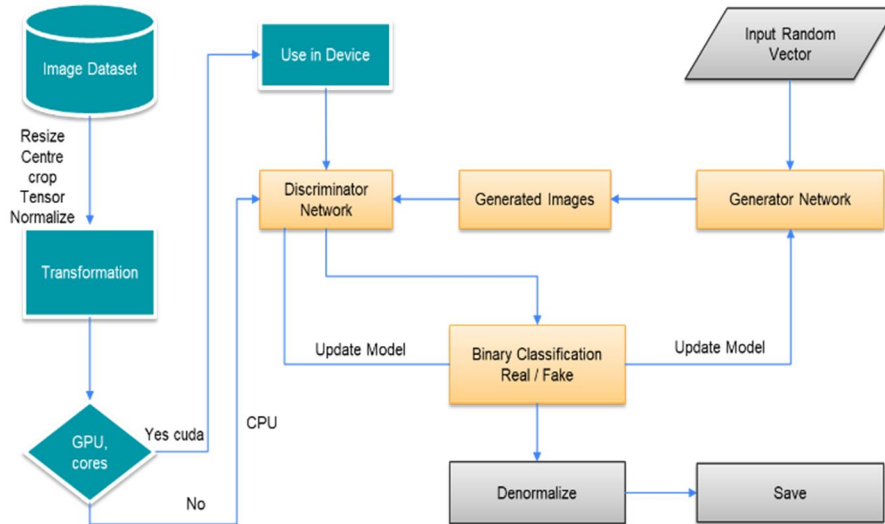


Fig. 1 Model Flow Chart



Fig. 2 Examples of Celeb face dataset [4]

B. Dataset Compilation

Load this dataset using the `ImageFolder` class from `torchvision`. Now resize and crop the images to 64x64 px, and normalize the pixel values with a mean & standard deviation of 0.5 for each channel. To ensure that pixel values fall within the range of (-1, 1), which is more suitable for training the discriminator, we will standardize the data. Additionally, we will create a data

C. GPU

To leverage a GPU seamlessly, if one is available, we'll define a couple of helper functions (`get_default_device` and `to_device`) along with a helper class `DeviceDataLoader` to facilitate the transfer of our model and data to the GPU, if present. Given the computational intensity involved, having access to a GPU is highly recommended.

D. Discriminator Network

An image is passed in the discriminator, and it tries to classify it as "real" or "generated". In this sense, it's like any other neural network. A Convolutional Neural Network (CNN) is utilized, which produces a single numerical output for each image. We'll employ a stride of 2 to gradually decrease the dimensions of the output feature map. The architecture of the Discriminator is designed to learn a function that maps the input image to a probability value between 0 and 1, indicating the likelihood that the input image is real. The convolutional layers are used to extract features from the input image, while the batch normalization layers help to stabilize and speed up training. The model is a sequential neural network with five convolutional layers, each followed by a batch normalization layer and a LeakyReLU activation function. The convolutional layers decrease the spatial dimensions (height and width) of the input while increasing the number of channels (feature maps). The final convolutional layer reduces the spatial dimensions to 1x1, effectively flattening the output. A sigmoid activation function is applied to the flattened output to produce a probability value between 0 and 1, indicating the likelihood that the input image is real.

`nn.Conv2d(3, 64, kernel_size=4, stride=2, padding=1, bias=False)`: This is a 2D convolutional layer with 64 filters (kernels), each of size 4x4. The stride is set to 2, meaning the filter moves two pixels at a time, and padding is set to 1 to preserve the spatial dimensions. The input has 3 channels (RGB images), and the bias term is set to False to exclude it from the computation.

The channel map is increased in the convolutional layers to extract more complex features from the input image. Each convolutional layer applies a set of filters to the input, and each filter extracts a specific feature from the input. By increasing the number of filters (and thus the number of channels), the model can learn to extract more complex and abstract features from the input image.

Flattening the output of the final convolutional layer is necessary to convert the 2D feature map into a 1D vector that can be passed through the sigmoid activation function. This is because the sigmoid activation function is applied to each element of the input vector independently and produces a probability value between 0 and 1 for each element. By flattening the output of the final convolutional layer, we can apply the sigmoid activation function to each feature map independently and produce a probability value for each feature map.

A feature map is the output of a convolutional layer, and represents the features extracted from the input image by the filters in that layer. Each feature map is associated with a particular filter, and the values within the feature map indicate how that filter responds to the input image. By stacking multiple feature maps together, the model can learn to extract multiple features from the input image simultaneously.

In essence, the Discriminator is pivotal in a GAN, offering a training signal for the Generator. This setup enables the Generator to craft images capable of deceiving the Discriminator into perceiving them as real. By teaching the Discriminator to differentiate between real and fake images, we leverage its output as a gauge for the quality of the generated images. As the Generator advances, the Discriminator's capacity to distinguish between real and fake diminishes, creating a feedback loop that propels the Generator towards generating progressively lifelike images.

To sum up, the Discriminator plays a vital role in a GAN, furnishing a training signal for the Generator and empowering it to learn the art of producing increasingly realistic images. The Discriminator's architecture is tailored to extract features from input images and generate a probability value indicating the likelihood that the input image is real.

E. Generator Network

The generator typically takes a random vector or matrix of numbers (known as a latent tensor) as input, serving as a seed to generate an image. It transforms a latent tensor of shape $(128, 1, 1)$ into an image tensor of shape $3 \times 64 \times 64$. This transformation is achieved using the `ConvTranspose2d` layer from PyTorch, which performs a transposed convolution (also called deconvolution).

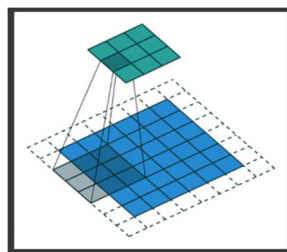


Fig.3 Convolutional layer string and padding [7]

The output of the final layer undergoes a hyperbolic tangent (Tanh) activation function, scaling the output values to the range $[-1, 1]$.

In essence, the generator network takes a random latent vector as input and produces a generated image resembling the training data. It employs transposed convolutional layers to increase the spatial dimensions of the input tensor and batch normalization layers to stabilize the training process. ReLU activation is used in all layers except the final layer, where Tanh activation ensures output values fall within the range $[-1, 1]$.

We opt for ReLU activation (Nair & Hinton, 2010) in the generator, except for the output layer which utilizes the Tanh function. This approach with bounded activation aids the model in faster learning to saturate and cover the color space of the training distribution. In the discriminator, we found that leaky rectified activation (Maas et al., 2013; Xu et al., 2015) worked well, particularly for higher resolution modeling.

It's worth noting that since Tanh activation outputs lie within the range $[-1, 1]$, a similar transformation is applied to the images in the training dataset.

F. Discriminator Training

Given that the discriminator functions as a binary classification model, we employ the binary cross-entropy loss function to evaluate its ability to distinguish between real and generated images.

Here's the breakdown of the steps involved in training the discriminator:

- We anticipate the discriminator to output 1 for images drawn from the real MNIST dataset and 0 for those generated by the generator network.
- Initially, we feed a batch of real images into the discriminator, computing the loss while setting the target labels to 1.
- Subsequently, we pass a batch of fake images (generated by the generator) through the discriminator, calculating the loss with target labels set to 0.
- Finally, we combine the losses from both steps and utilize the overall loss to perform gradient descent, thereby adjusting the weights of the discriminator

It's crucial to emphasize that during the discriminator's training phase, we refrain from altering the weights of the generator model. The optimization process (`opt_d`) exclusively affects the parameters of the discriminator (`discriminator.Parameters()`).

G. Generator Training

The challenge in training the generator lies in its output being images. To address this, we employ a clever strategy by incorporating the discriminator into the loss function. Here's how it operates:

- We generate a batch of images using the generator and pass them through the discriminator.
- By setting the target labels to 1 (representing real images), we compute the loss. This approach aligns with the generator's objective of deceiving the discriminator.
- We utilize this loss to execute gradient descent, thereby adjusting the weights of the generator. This process enhances the generator's capability to produce images that closely resemble real ones, tricking the discriminator in the process.

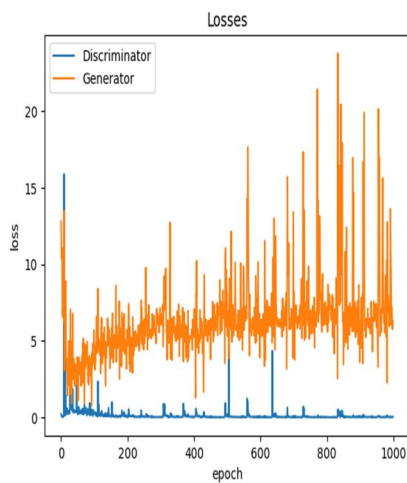


Fig.4 Losses Vs epoch

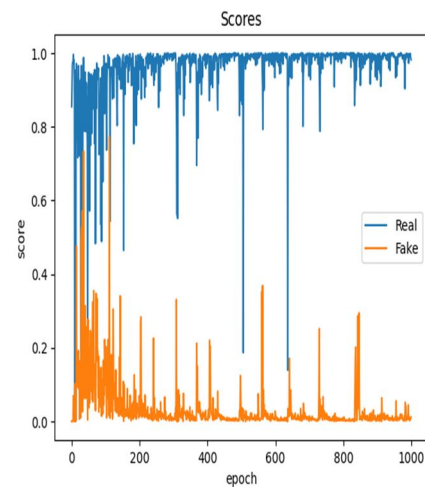


Fig.5 Scores Vs Epoch

H. Full Training Loop

We establish a `fit` function to train both the discriminator and generator simultaneously for each batch of training data. Employing the Adam optimizer with customized parameters (betas), which are known to be effective for GANs, ensures smooth optimization. Additionally, we periodically save some sample generated images for inspection.

```
`opt_d = torch.optim.Adam(discriminator.parameters(), lr=lr, betas= (0.5, 0.999))`
```

We define a small learning rate (`lr=0.0002`) to aid the model in converging towards an optimal solution without encountering divergence or oscillation.

I. Graphs

Though not much can be Inferred from the losses graph, it shows us the way the training has happened after each epoch and how the generator and the discriminator performed against each other. The sharp spikes in the losses shows that the discriminator became quite good in identifying the images or the generator became good in producing one.

IV. RESULTS

The Synthetic Data Generation Using Generative Adversarial Networks (GAN) work has yielded significant insights and outcomes across various domains, including human faces, anime faces, and artworks. The dataset should not differ from each other like in artworks as for it first we have to classify it among different categories and then train every category otherwise the output will not be up to mark.

- Result with Human Faces Dataset: When utilizing the human faces dataset, the work demonstrated remarkable success in generating high-quality synthetic images. Leveraging advanced GAN models, the system accurately captured facial features and textures and some pictures reaching 90% accuracy, producing realistic images suitable for applications such as face recognition
- Result with Anime Faces Dataset: By training the GAN model on anime face datasets, the system captured the distinct characteristics and styles prevalent in anime artwork. Users benefited from the comprehensive data synthesis capabilities, accessing a wide range of synthetic images for creative endeavors and character generation.



Fig. 6 Output for Human Faces dataset after 15 epochs



Fig. 7 Output for anime dataset after 15 epoch

V. CONCLUSION AND FUTURE SCOPE

Wrapping things up, our Synthetic Data Generation Using Generative Adversarial Networks (GAN) work is a big step forward in making artificial intelligence more practical and useful. We've built a system that can make all sorts of realistic images, like human faces and different styles of art, using smart computer programs. Our work has been about pushing boundaries, always striving to make things better, and solving real problems. The future of synthetic data generation using GAN holds immense potential for further advancements. Research efforts can focus on refining GAN architectures, exploring novel functions, and incorporating additional constraints to improve the quality and diversity of synthesized data. Techniques such as self-attention

mechanism and progressive growing GANs can be investigated to generate even more realistic and high-resolution images across diverse domains.

REFERENCES

- [1] Jason Brownlee on July 19, 2019, in Generative Adversarial Networks <https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/> J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
- [2] Mckinsey666 dataset scraped from www.getchu.com <https://www.kaggle.com/datasets/splcher/animefacedataset> K. Elissa, “Title of paper if known,” unpublished.
- [3] CelebFaces Attributes (CelebA) Dataset Over 200k images of celebrities with 40 binary attribute annotation <https://www.kaggle.com/datasets/jessicali9530/celeba-dataset/data>
- [4] International Journal of Information Management Data Insights Volume 1, Issue 1, April 2021, 100004
- [5] <https://www.sciencedirect.com/science/article/pii/S2667096820300045>
- [6] Generative Adversarial Networks for Synthetic Data Generation: A Comparative Study Claire Little, Mark Elliot, RicharAllmendinger, Sahel Shariati Samani <https://www.sciencedirect.com/science/article/pii/S2667096821000471>
- [7] High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs Ting-Chun Wang¹ Ming-Yu Liu¹ Jun-Yan Zhu² Andrew Tao¹ Jan Kautz¹ Bryan Catanzaro¹ NVIDIA Corporation ²UC Berkeley <https://arxiv.org/pdf/1711.11585.pdf>
- [8] Zhang, S., et al. (2020a). An Image Denoising Method Based on BM4D and GAN in 3D Shearlet Domain. Mathematical Problems in Engineering. 10.1155/2020/ 1730321.
- [9] Zhang, X., Jian, W., Chen, Y., & Yang, en S. (2020b). Deform-GAN:AN unsupervised learning model for deformable registration. Toegang verkry. [Online]. Available at: <http://arxiv.org/abs/2002.11430>.
- [10] Zhou, H., et al. (2020b). 3D high resolution generative deep-learning network for fluorescence microscopy imaging. Optics Letters. 10.1364/ol.387486.