Abstract Art Generation using Generative Adversarial Networks (GANs)

Final Project Report for Deep Learning

Deep Usdadiya 23PGAI0077
Arpit Tiwari 23PGAI0127
Akash Deshwani 23PGAI0035
Sivaram S 23PGAI0082
Submitted to - Dr. Gaurav Agrawal
7 January 2023

1. Introduction

Generative Adversarial Networks (GANs) are a powerful machine learning technique that allows for the generation of new data samples that are similar to a training dataset. They have been successfully applied in various fields, including image generation, natural language processing, and music generation. In this report, we will explore the use of GANs for generating abstract art. Abstract art is a form of art that does not depict specific objects or scenes, but rather uses color, form, and composition to create visual interest and evoke emotions. It has been a popular artistic medium for centuries and continues to be a source of inspiration for artists and art lovers alike. With the help of GANs, it is now possible to generate abstract art automatically, opening up new possibilities for artistic expression and creativity. In the following sections, we will first review the concept of GANs and how they work. Then, we will discuss the specific architecture and training process used in our GAN model for abstract art generation. We will also present and analyse the results of our experiments, and discuss the potential applications and future directions of GANs for abstract art generation.

2. Aim

The aim of this report is to investigate the use of Generative Adversarial Networks(GANs) for generating abstract art and to evaluate the results of our experiments.

3. Objectives

- Review the concept of Generative Adversarial Networks (GANs) and their potential for abstract art generation.
- Design and implement a GAN model for generating abstract art. Train the GAN model on a dataset of abstract art images and evaluate the quality of the generated images.
- Analyse the results of our experiments and discuss the potential applications and future directions of AI for abstract art generation.

4. Tools and Methods

For our experiment on Abstract Art Generation using Generative Adversarial Networks (GANs), we used the following tools and methods:

1. Tensorflow: We used Tensorflow, an open-source machine learning library developed by Google, to implement and train our GAN model. Tensorflow provides a flexible and efficient platform for building and training neural networks, as well as for analysing and visualising the results.

2. Abstract Art Dataset: We used a dataset of abstract art images to train our GAN model. The dataset consists of a collection of digital images in JPG format, each representing a unique abstract art piece. The images have a resolution of 168x189 pixels and are in color (RGB).
   Source - https://www.kaggle.com/datasets/bryanb/abstract-art-gallery
3. GAN Architecture: We designed a GAN model with a generator network and a discriminator network, as described in the previous sections. The generator network consists of three fully-connected (dense) layers, while the discriminator network consists of four fully-connected layers. Each layer has a leaky ReLU activation function, except for the last layer of the generator and the last layer of the discriminator, which have linear activation functions.
4. RMSprop (Root Mean Squared Propagation) optimisation algorithm: can be used to update the weights of both the generator and discriminator networks. The generator network is trained to generate new sample that are similar to a training set, while the discriminator network is trained to distinguish between real samples from the training set and fake samples generated by the generator.
5. Binary Cross Entropy Loss function : Binary cross-entropy loss is a loss function commonly used in binary classification tasks, such as in a GAN where the discriminator is trying to classify whether an image is real or fake.
6. Evaluation Metrics: We evaluated the quality of the generated images using several metrics. We used two metrics Inception Score (IS) and Fréchet Inception Distance (FID).

The project was done on Google Colab with the following deep learning frameworks -

1. TensorFlow
2. Keras

5. Input Data Source

There are many different sources of data that can be used as input for training a GAN. Some common sources include:

Publicly available datasets: There are many datasets that are freely available online and can be used as input for training a GAN. Examples include image datasets such as the MNIST handwriting dataset, the CIFAR-10 image dataset, and the ImageNet dataset.

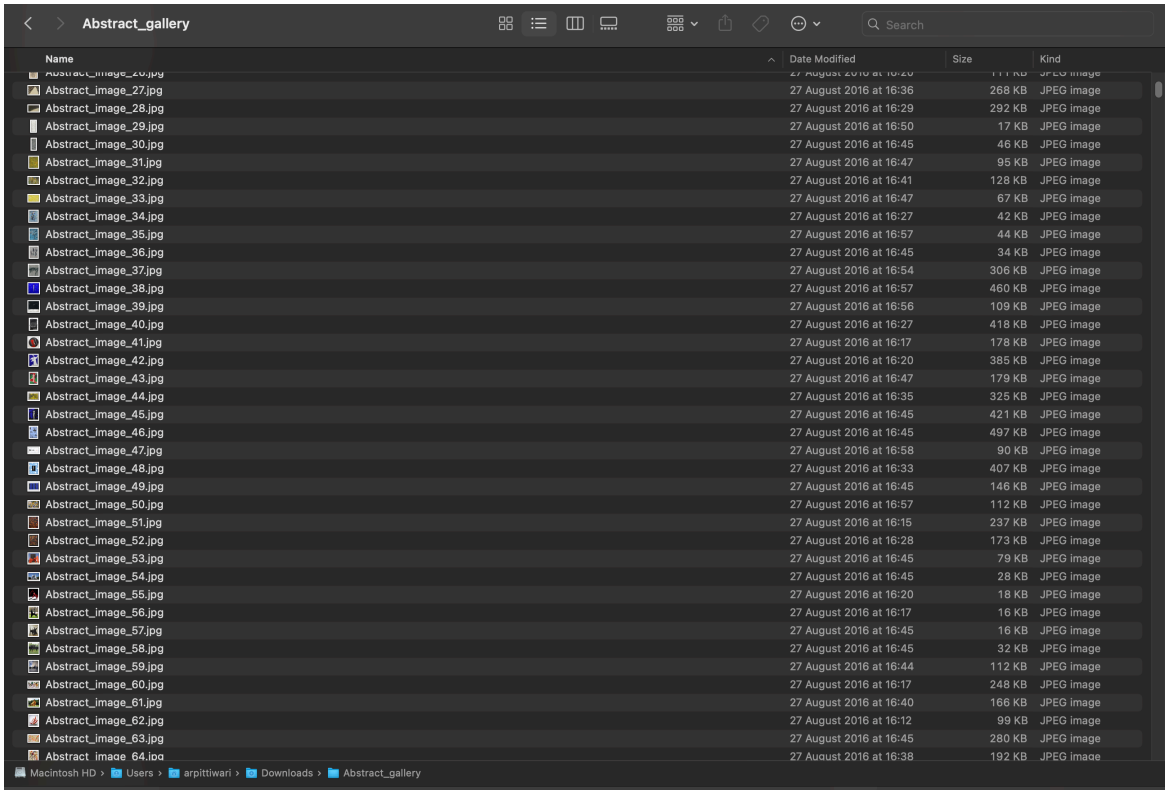
Private datasets: It is also possible to use privately held data as input for training a GAN. This may be particularly useful if the data is proprietary or sensitive, and cannot be shared publicly.

Synthetic data: In some cases, it may be useful to generate synthetic data as input for training a GAN. This can be done using tools such as computer graphics software or simulations. Synthetic data can be useful for testing the performance of a GAN on data that is artificially generated, and may be easier to obtain than real-world data

For our purpose and project, we chose a public dataset available on gaggle for the work (link as given below): -

https://www.kaggle.com/code/mixmore/abstract-art-gallery/data.

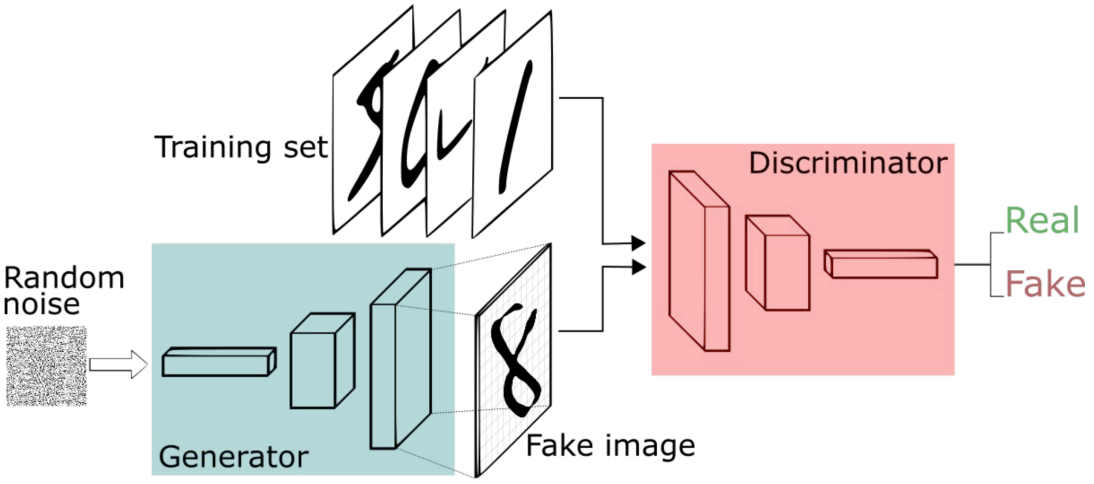
View of the data : -



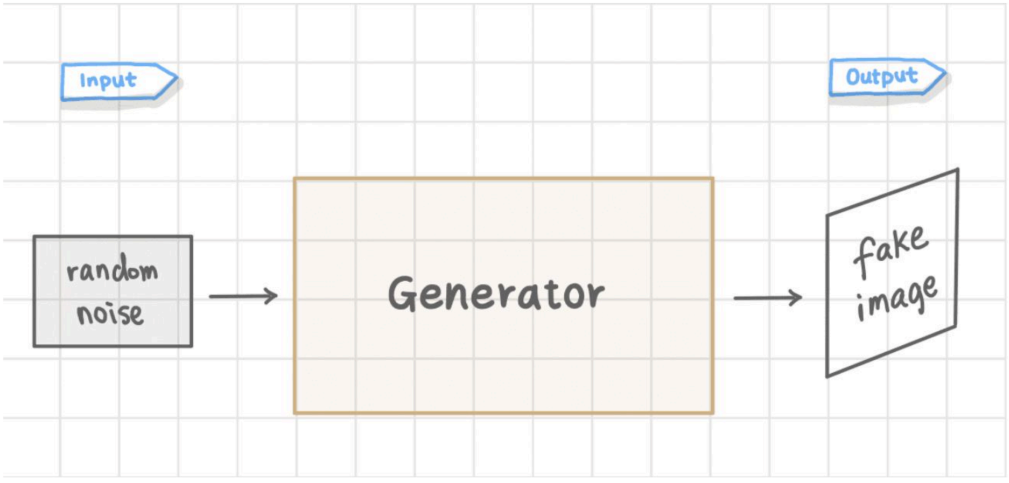
This dataset contains 2782 files of abstract images.

6. Methodology

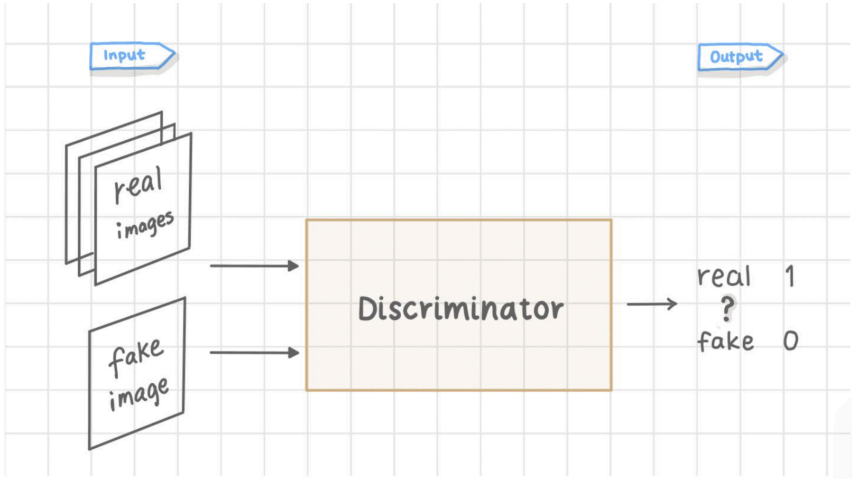
Architecture

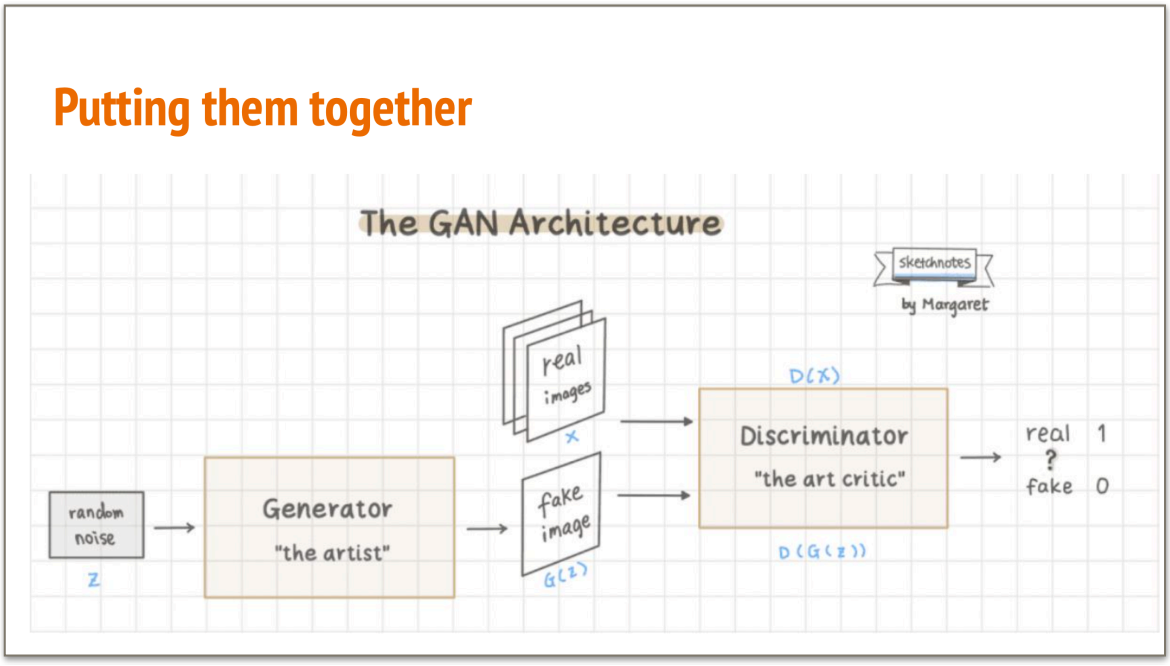


Generator



Discriminator





Generative models were designed by Goodfellow et al. in 2014. It consists of two differentiable functions, represented by neural networks, are locked in a game. The two players, the generator and the discriminator, have different roles in this framework. The generator tries to produce data that come from some probability distribution .The discriminator, acts like a judge. It gets to decide if its input comes from the generator or from the true training set.

In the context of abstract art generation, the generator would be responsible for generating synthetic images that are similar to a training dataset of abstract art images. The discriminator would then be trained to distinguish real abstract art images from the synthetic images generated by the generator.

As the training process continues, the generator and discriminator will become increasingly adept at their respective tasks. The generator will learn to generate increasingly realistic and diverse synthetic abstract art images, while the discriminator will become better at distinguishing real abstract art images from synthetic ones.

Once training is complete, the generator can be used to generate new abstract art images that are similar to the training dataset, while the discriminator can be used to identify which images are real and which are synthetic

7. Implementation

There are several code checkpoints which play a critical role in the entire GAN network. Hyper parameters and parameters play a very essential role in the overall performance of the model.

1. **Generator** - The generator network is defined using the Sequential model from Keras. It starts with a dense layer that takes as input a noise vector with shape [noise_shape], which will be used to generate the images. The output of this layer is then reshaped

into a 4x4x512 tensor, which is passed through a series of Conv2DTranspose layers with increasing number of filters. These layers use strided convolutions to upsample the input tensor and generate an image with higher resolution. Each of these layers is followed by a LeakyReLU activation layer and a BatchNormalization layer. Finally, the last layer is a Conv2DTranspose layer with 3 filters and a sigmoid activation function, which generates the final output image with shape 128x128x3.

2. **Discriminator** - The discriminator network is defined using the Sequential model from Keras, and it consists of a series of Conv2D layers with increasing number of filters, followed by a LeakyReLU activation layer and a BatchNormalization layer. The output of the last Conv2D layer is flattened and passed through a dropout layer and a dense layer with a single output and a sigmoid activation function. This discriminator network takes as input an image with shape 128x128x3 and outputs a single scalar value indicating whether the image is real or generated.

3. **DCGAN (Deep Convolutional Generative Adversarial Network)**- Deep Convolutional Generative Adversarial Network (DCGAN) for generating abstract art images. A DCGAN is a type of GAN that uses convolutional neural networks (CNNs) as the generator and discriminator networks. The generator and discriminator networks are then combined into a single GAN model using the Sequential model, with the generator being the first layer and the discriminator being the second layer. The discriminator is then compiled with an RMSprop optimizer and a binary cross-entropy loss function, and its trainable parameter are set to False. The GAN model is then compiled with an RMSprop optimizer and a binary cross-entropy loss function.

4. **Optimizer RMSprop:** RMSprop (Root Mean Square Propagation) is a gradient descent optimization algorithm that helps to speed up the training process of a neural network. It is often used in deep learning models to optimize the parameters of the network. One of the main differences between RMSprop and other optimization algorithms such as stochastic gradient descent (SGD) and Adam is the way it handles the learning rate. In SGD and Adam, the learning rate is a hyperparameter that is set by the user and remains constant throughout training. However, in RMSprop, the learning rate is adaptively adjusted for each parameter based on the historical gradient information of that parameter. This allows RMSprop to use a larger learning rate for parameters that are changing slowly and a smaller learning rate for parameters that are changing rapidly, which can help to speed up the training process. RMSprop also has a momentum term, similar to SGD with momentum and Adam, which helps to smooth out the updates to the parameters and avoid oscillations. Overall, RMSprop is a popular choice for optimization in deep learning due to its ability to adaptively adjust the learning rate and smooth out the updates to the parameters. It is often used in conjunction with other techniques such as batch normalization and dropout to further improve the training process.

5. **Loss function Binary cross-entropy**: Binary cross-entropy loss is a loss function commonly used in binary classification tasks, such as in a GAN where the discriminator is trying to classify whether an image is real or fake. It is defined as follows:
   Loss = -(y * log(y_pred) + (1 - y) * log(1 - y_pred))

where y is the true label (either 0 or 1), y_pred is the predicted probability of the label being 1, and log is the natural logarithm.

In the context of a GAN, the binary cross-entropy loss function is used to train the discriminator to distinguish between real and fake images. The labels for the real images are set to 1, and the labels for the fake images are set to 0. The discriminator is then trained to predict a probability close to 1 for real images and a probability close to 0 for fake images. The generator is trained to generate images that can fool the discriminator into predicting a probability close to 1 for fake images.

There are other available : - Mean squared error (MSE), Categorical cross-entropy loss, Hinge loss, Kullback-Leibler divergence (KL divergence) etc.

Binary cross entropy loss function is better because it is often a good choice for binary classification tasks, where the goal is to predict a label that can take on only two values (e.g., 0 or 1). This is because it is a measure of the difference between the predicted probability distribution and the true label, and it has the convenient property of being convex, which means that it has only one global minimum that can be found efficiently using gradient descent.

8. Evaluation Metrics

There are several ways to evaluate the quality of the images generated by a Generative Adversarial Network (GAN):

1. Visual inspection: One of the simplest and most intuitive ways to evaluate the quality of generated images is to look at them and see if they look realistic and visually appealing. This can be subjective and can vary depending on the criteria used to define "realistic" and "visually appealing".
2. Human evaluation: Another way to evaluate the quality of generated images is to ask human evaluators to rate the images on different dimensions, such as realism, diversity, and overall quality. This can be done through surveys or other types of user studies.
3. Quantitative measures: There are several quantitative measures that can be used to evaluate the quality of generated images. For example, you can use metrics like the

Inception Score (IS) or the Fréchet Inception Distance (FID) to measure the diversity and fidelity of the generated images. You can also use classification or regression models to evaluate the quality of the generated images, by training the models on real images and testing them on the generated images.

Inception Score (IS)

The Inception Score (IS) is a metric used to evaluate the quality of images produced by a generative model, such as a Generative Adversarial Network (GAN). It was introduced in the paper "Improved Techniques for Training GANs" by Salimans et al.

The Inception Score is based on the idea that images generated by a good generative model should be both diverse and visually coherent. The metric uses the Inception network, a pre-trained image classification model, to evaluate the diversity and quality of the generated images.

Fréchet Inception Distance (FID)

Fréchet Inception Distance (FID) is a metric that is often used to evaluate the quality of images produced by a Generative Adversarial Network (GAN). It was introduced in the paper "Improved Techniques for Training GANs" by Salimans et al.

The Fréchet Inception Distance is based on the idea that images produced by a good generative model should be similar to a set of reference images. To compute the FID, the GAN is first used to generate a set of images, and the Inception network, a pre-trained image classification model, is used to extract features from both the generated images and the reference images

9. Output

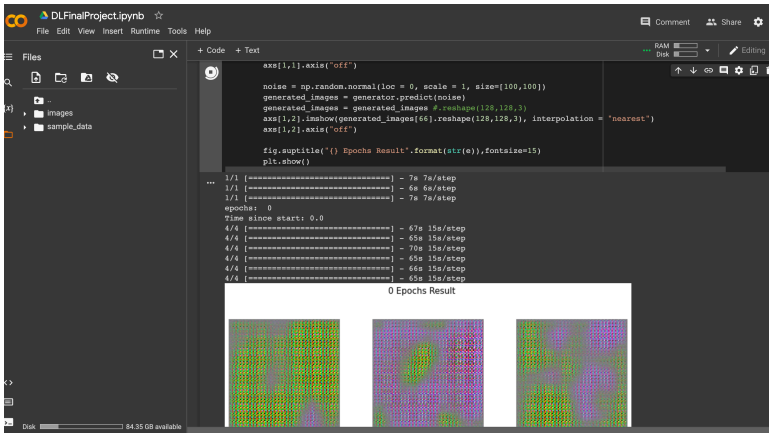


Fig.1. Training under process

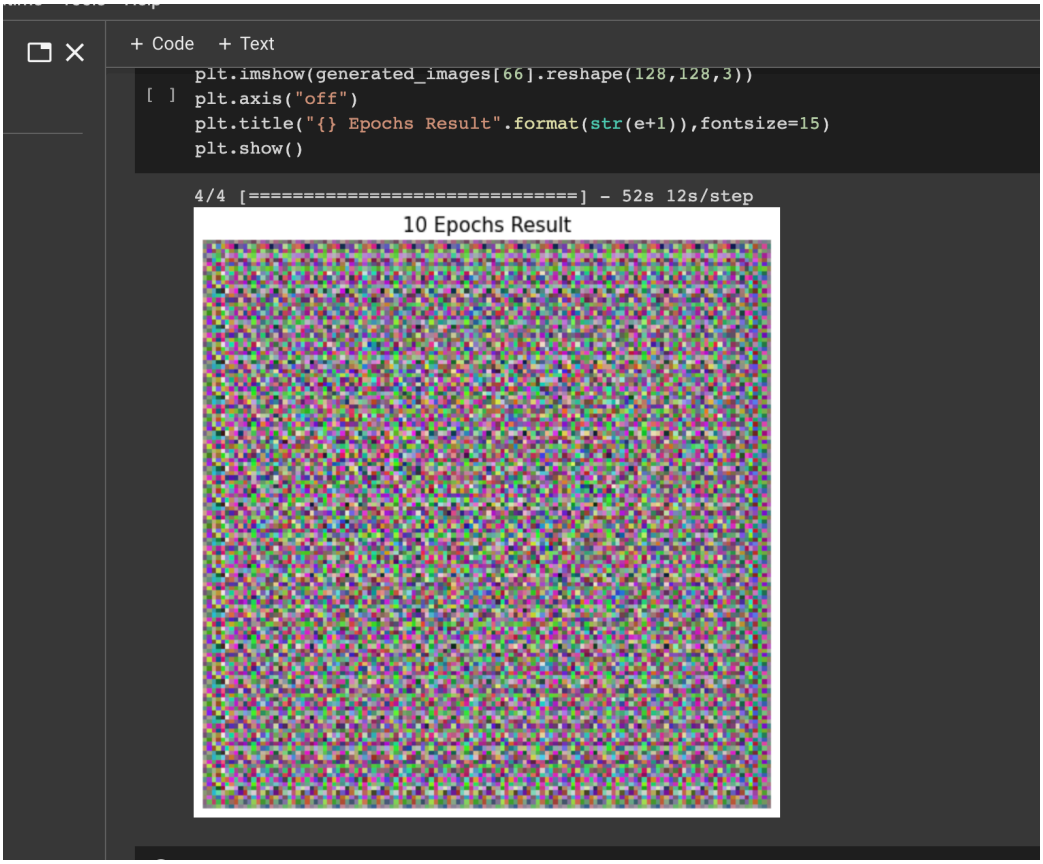


Fig.2. Output#1

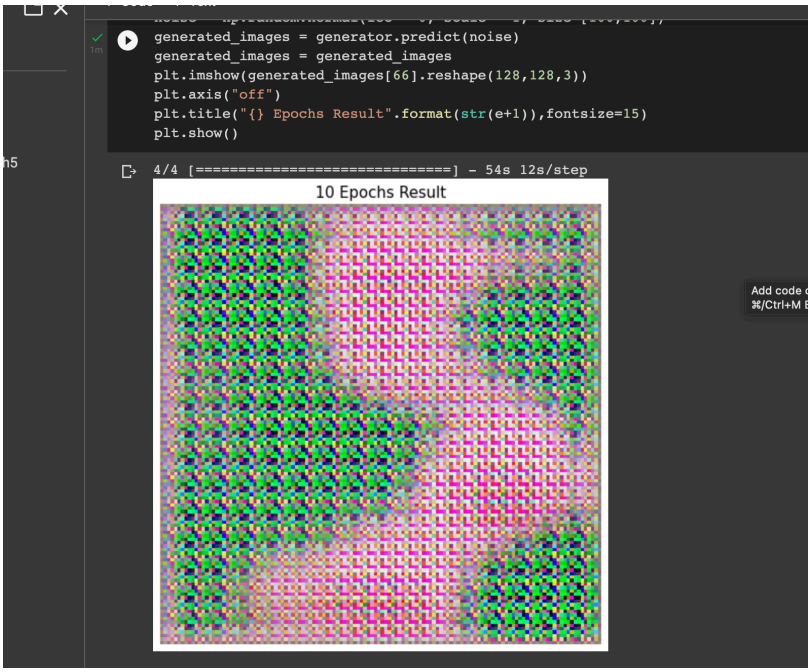


Fig.3. Output#2

10. Findings/Summary

The generator and discriminator are then trained together in an adversarial manner, where the generator tries to generate realistic images that can fool the discriminator, and the discriminator tries to correctly distinguish real from generated images.

The generator and discriminator networks are then combined into a single GAN model using the Sequential model, with the generator being the first layer and the discriminator being the second layer. The discriminator is then compiled with an RMSprop optimizer and a binary cross-entropy loss function, and its trainable parameter are set to False. The GAN model is then compiled with an RMSprop optimizer and a binary cross-entropy loss function.

The GAN is then trained using the train_on_batch function from Keras. In each training iteration, a batch of noise vectors is generated and passed through the generator to generate a batch of fake images.

11. Limitations and Conclusions

Generative Adversarial Networks (GANs) are a type of deep learning model that can be used to generate synthetic data that is similar to a given dataset. They consist of two neural networks, a generator and a discriminator, that are trained simultaneously in an adversarial manner. The generator tries to produce synthetic data that is indistinguishable from the real data, while the discriminator tries to differentiate between the real and synthetic data.

One limitation of GANs is that they can be difficult to train, and can suffer from instability and mode collapse. Mode collapse is when the generator produces a limited range of outputs and fails to capture the diversity of the real data. This can be mitigated by using techniques such as mini-batch discrimination and using a more powerful generator network.

Another limitation of GANs is that they can produce synthetic data that is not always of high quality. The quality of the generated data depends on the complexity and diversity of the real data, as well as the capacity and training of the GAN. Improving the quality of the generated data may require using more advanced GAN architectures or larger and more diverse datasets.

In conclusion, GANs are a powerful tool for generating synthetic data, but they have limitations that need to be considered when using them. Careful design and training of the GAN is required to produce high-quality synthetic data that is representative of the real data.

12. References

1. https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215
2. https://www.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html%20
3. https://www.freecodecamp.org/news/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050/
4. https://www.kaggle.com/code/mixmore/abstract-art-gallery
5. https://www.kaggle.com/getting-started/150948
6. https://sthalles.github.io/intro-to-gans/
7. Class Lectures and PPTs