Generative Adversarial Networks

Presented by:
Dhaval Devshibhai Kathiriya
Sher Mohammad
Dheeraj Bharat Sethi

Instructor:
Dr. Yasser Alginahi
Nov. 3rd, 2023

From *inferring* in statistics
to *predicting* in machine learning
to *classifying* in deep learning,
AI has evolved to

A neural network that "generates" material the way humans produce

Generative 🧠

Villain 🥷 or Hero 🦸‍♀️ ? perspective decides

But ❤️ of the GAN architecture

Adversarial ♟️

Two Actors Competing against each other 👯‍♀️

By creating "*adverse*" environment

Networks 🧬

Actors are not humans but a *Neural Network*

Provides a plot for both the ***Generator*** and ***Discriminator***

To work "*against*" each other towards one goal 🥅 [1].

Generate Synthetic Data

Generator 🎩

Based on **probabilistic** model

RESPONSIBLE FOR "GENERATING INDISTINGUISHABLE" FAKE DATA FROM THE REAL DATA IT IS TRAINED FOR [1]

SO THAT IT "FOOLS" THE DISCRIMINATOR NETWORK [1]

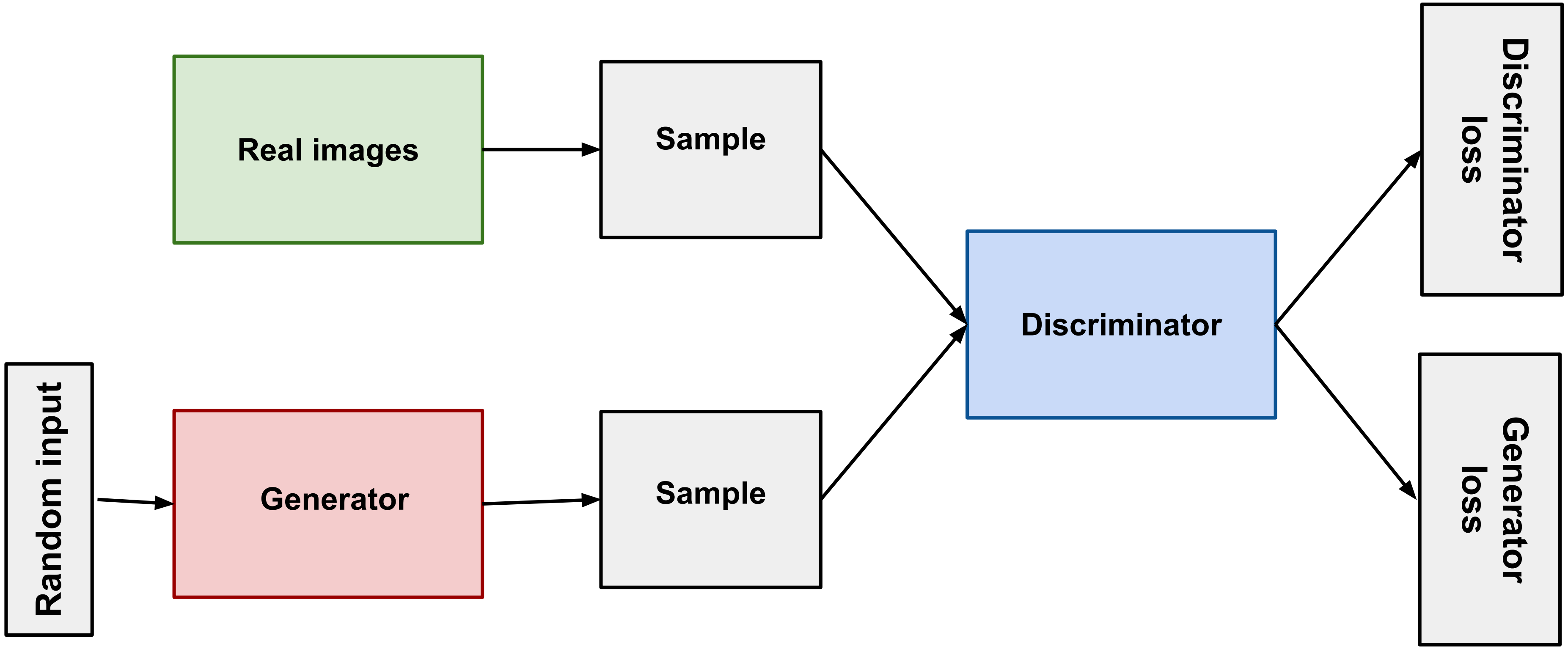
IMPROVING OVERALL PERFORMANCE OF BOTH NETWORKS [1]

Discriminator 🤷‍♂️ 🤷‍♀️

Responsible for "punishing" generator for creating fake data [2]

Catch is...only if the discriminator "catches" the fake data [2]

GAN Architecture 🔬



https://tinyurl.com/2nx5sy9y

GAN Architecture explained[3]

Initially, before training has begun, the generator's fake output is very easy for the discriminator to recognize .

Since the output of the generator is fed directly into the discriminator as input, this means that when the discriminator classifies an output of the generator, we can apply the backpropagation algorithm through the whole system and update the generator's weights.

Over time, the generator's output becomes more realistic and the generator gets better at fooling the discriminator. Eventually, the generator's outputs are so realistic, that the discriminator is unable to distinguish them from the real examples.

Discriminator – the real hero

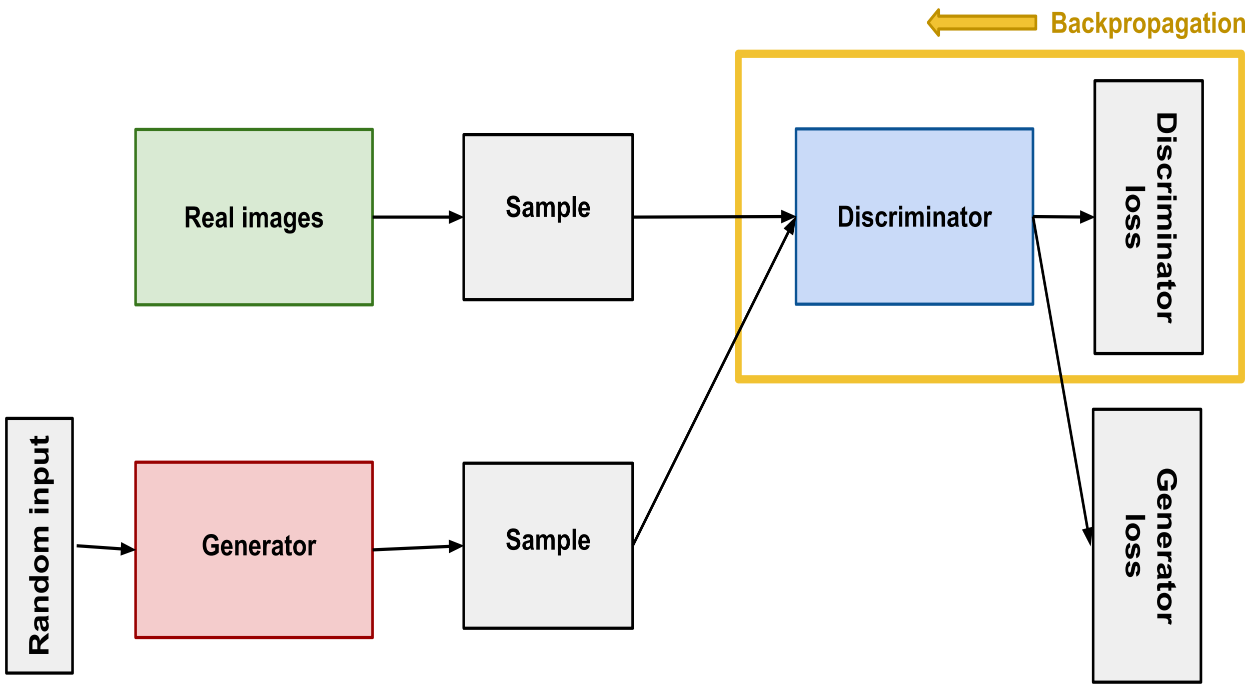
Binary **classifier neural network** - "supervised" learning model

Takes real or generated sample as input.

Output: An array where two numbers indicate the Discriminator's estimate of probability of input example being fake or real.

Discriminator *training and learning*[3]

- While we are training the discriminator, *we do not train the generator*, but hold the generator's weights constant and use it to produce negative examples for the discriminator.
- Pass some real examples, and some fake examples from the generator, into the discriminator as input.
- Calculate the discriminator *loss* using a suitable function such as the cross-entropy loss.
- Update the discriminator's *weights* through backpropagation.



Source: https://tinyurl.com/2nx5sy9y

GAN Code Setup and Discriminator 💻

- Generative adversarial networks can also generate high-dimensional samples such as images. In this example, you're going to use a GAN to generate images of handwritten digits. For that, you'll train the models using the MNIST dataset of handwritten digits, which is included in the torchvision package[10].
- MNIST is a dataset of 10K grayscale images of handwritten digits from 0 to 9 digits which is used in code.
- Source code link: GAN-Group4-Colaboratory (google.com) [10]

Generator What is it & what's inside?

- An "unsupervised" learning model.
- Takes input values from random noise or latent space [4].
- This input is then passed through a series of layers, typically consisting of neural networks to transform the noise into a meaningful output.
- CNN, RNN, NLP variations are developed in generative models per use-case [4].

Generator training and generation

- Trained to produce fake data [5].
- Takes random noise as input and generates synthetic data that resemble realistic-looking data.

University of Windsor

How training in Generator works

1. **Data Understanding: Know your target data type and characteristics.**
2. **Input: Decide on the input (often random noise). Standard Distribution – latent space**
3. **Architecture: Choose the neural network structure (e.g., CNN for images).**
4. **Output Format: Define the output to match real data (e.g., image dimensions).**
5. Activation Functions: **Apply appropriate activations in networks**

How training in Generator works...part 2

1. Loss Function: Create a suitable loss function (e.g., binary cross-entropy) $Z = -\sum y \log(\hat{y}) + (1-y) \log(1-\hat{y})$.
2. Regularization: Use techniques like dropout and batch normalization.
3. Optimization: Optimize the generator via adversarial training with the discriminator.
4. Hyperparameters: Experiment with settings (e.g., learning rate, architecture).
5. Quality Evaluation: Continuously assess generated quality.

Common Activation Functions

ReLU (Rectified Linear Unit):

- Sets negative input values to zero and keeps positive values unchanged.
- It helps to prevent the vanishing gradient problem.
- Mathematical function: $f(x) = \max(0, x)$.

LeakyReLU (Leaky Rectified Linear Unit):

- An extension of ReLU that addresses the "dying ReLU" problem [6].
- Prevents complete suppression of negative values.
- Mathematical function: $f(x) = \max(\alpha x, x)$, where α is a small constant.

Common Activation Functions cont...

Tanh (Hyperbolic Tangent):

- Commonly used in the output layer of the generator.
- Produce data within the range of -1 to 1.
- Mathematical function:
  tanh(x) =(exp(x) - exp(-x))/(exp(x)+exp(-x))

Sigmoid:

- Used in the output layer to generate data between 0 and 1.
- Mathematical function : σ(x) = 1/(1 + exp(-x))

Addressing Training Instability

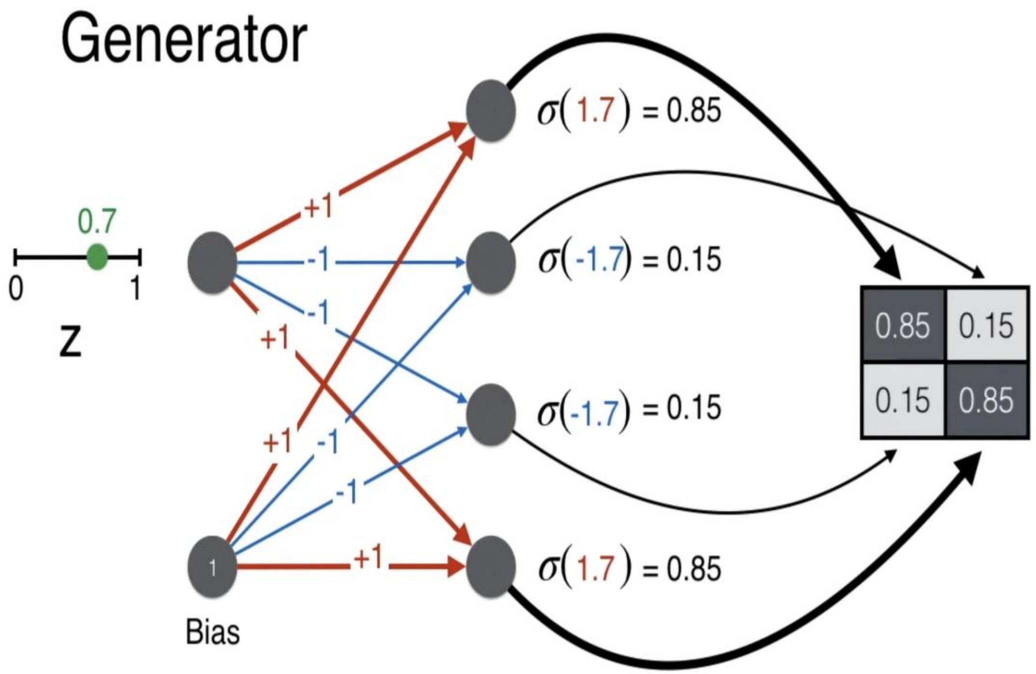
Causes

- Oscillation and Divergence in performance
- Generator and Discriminator struggle to find stable equilibrium

Possible Solutions

- Adjust learning rate.
- Use gradient penalties.
- Implement regularization techniques.

Generator Prediction Example



https://tinyurl.com/5w9zyu9y

Weighted sum: z = w * x + b

z1=0.7*1+1=1.7 =>y = sigmoid(1.7)=0.85

z2=-0.7*1-1=-1.7 =>y = sigmoid(1.7)=0.15

z3=-0.7*1+1=-1.7 =>y = sigmoid(-1.7)=0.15

z4=0.7*1+1=1.7 =>y = sigmoid(1.7)=0.85

G(z)=(G1,G2,G3,G4)

=(σ(v1*z+b1), σ(v2*z+b2), σ(v3*z+b3), σ(v4*z+b4))

G(z)=(0.85, 0.15, 0.15, 0.85)

GAN Code Generator 💻

[GAN-Group4-Generator (google.com) [10]](#)

Training GAN seems simple, right? 🥸

Let's *review*

1) Generator generates randomly first

2) Discriminator trains on real and random samples from generator

3) Discriminator provides feedback by back propagating the gradients to Generator as loss function – real vs fake

4) Generator takes feedback to regenerate new fake data

5) And this goes on until Generator is the winner...

In the Clash of the Networks…when

Generator Loss

becomes less than

Discriminator Loss

Nash Equilibrium 🧘‍♀️

The point of *Convergence*

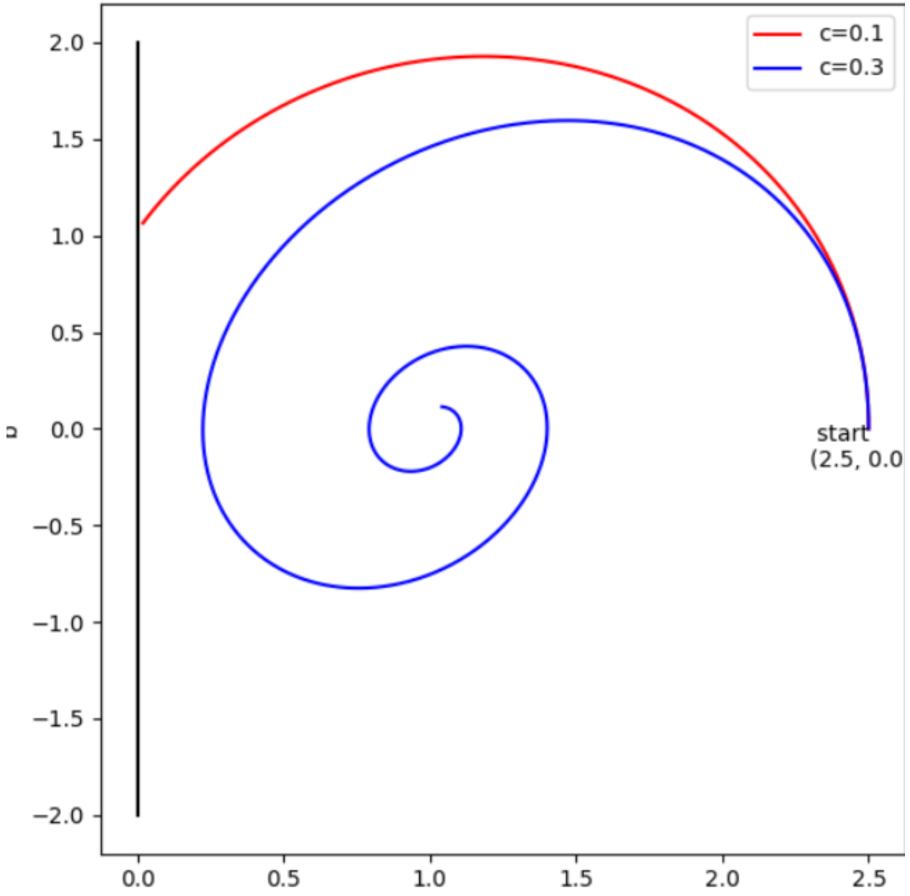
...an "optimal Discriminator" is required for it

- GAN achieves an equilibrium state
- When the further training is stopped
- Generator is *ready* to be used to generate fake samples

What if GAN has *no convergence*? 🙆🏽‍♂️

Due to…

- ❑ Discriminator is not fixed, always fluctuating
- ❑ Generator and Discriminator can't improve each other
- ❑ Generator doesn't get feedbacks from its counterpart
- ❑ Discriminator matures itself sooner...
- ❑ Generator has found the weakness of discriminator, generating similar images once labelled as real



https://tinyurl.com/4ux9kz8z

The *Challenges* of GAN "research" areas

Vanishing Gradient

Mode Collapse

University of Windsor

Mode Collapse [7] - by a perfect discriminator

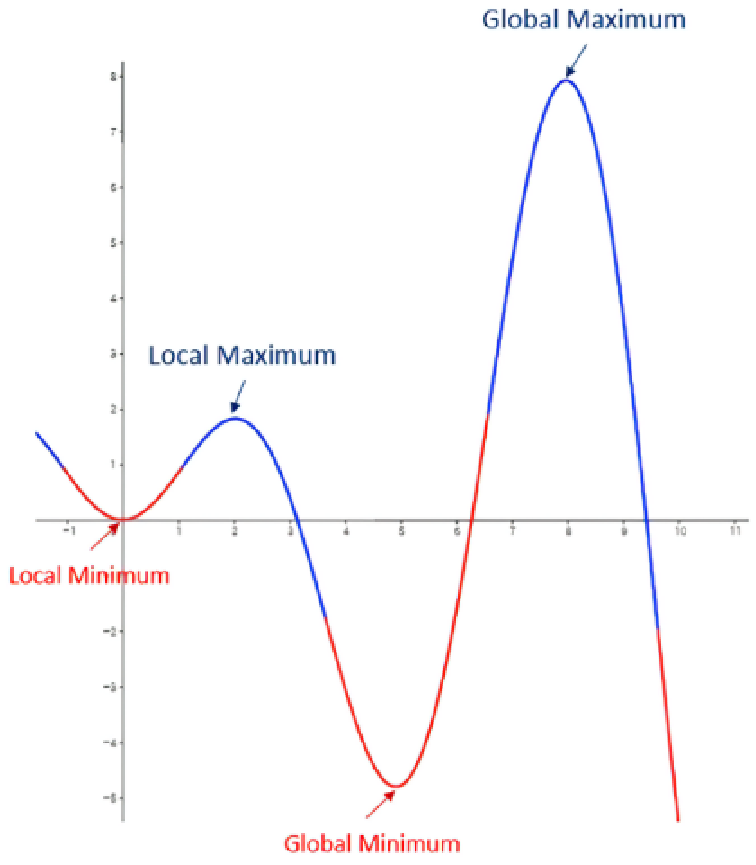
- Generates few "identical" samples regardless of its inputs
- Generator only learns the *highest frequency* among the sample space
- ...to induce the discriminator not to recognize generated samples

In simple words...

Generator is trying hard to fool the *superb discriminator*, is stuck around same *"local minimum"* of input

Mode Collapse – local minimum problem

- ❑Happens "later" in the training
- ❑Discriminator is perfect
- ❑Gradients for the training of the discriminator can explode
- ❑ Fails to cover whole sample space
- ❑Not a fault of Discriminator, its "trapped" as well due to the same generation



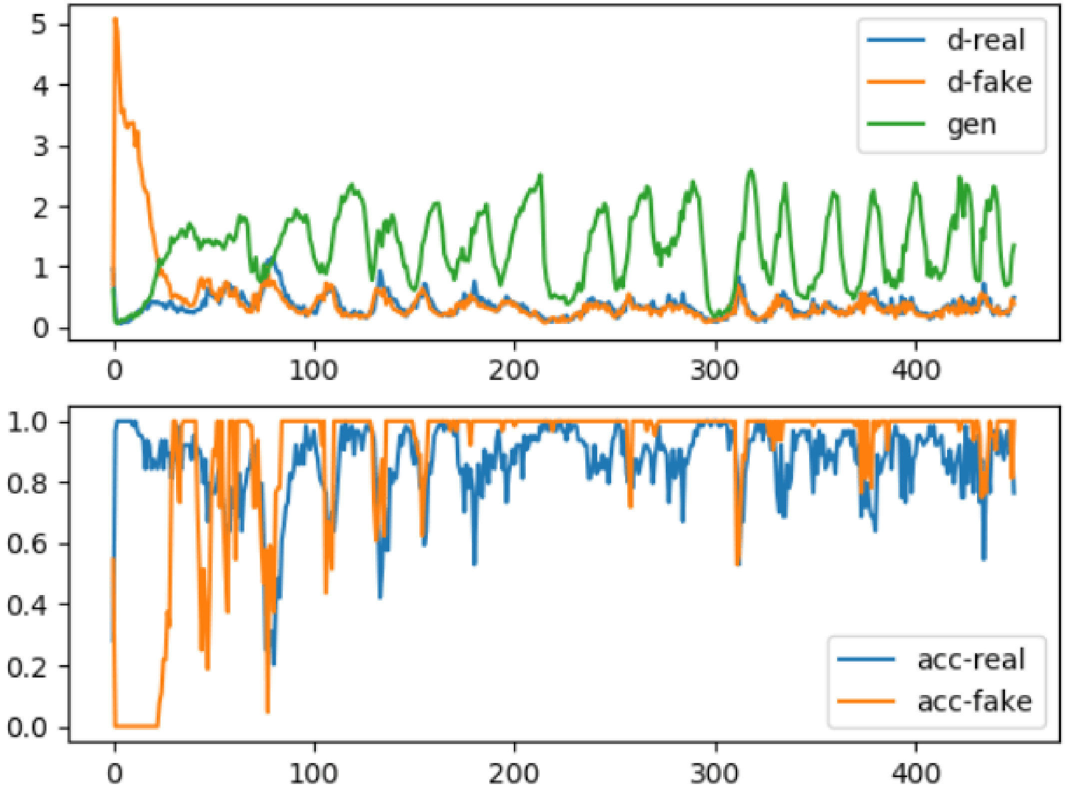
https://tinyurl.com/4ux9kz8z

Mini batch discrimination - Regularization

- Generator generates samples in *mini batches*
- Encode the info about "**uniqueness**" of the batch
- discriminator gets hints about how different each fake sample is from others within that batch

It acts as a regularization to incentivize coverage of the multi-modal data distribution, rather than collapsing to certain clusters

Vanishing Gradients



https://tinyurl.com/4yx9kz8z

- Occurs in "early" stages of training
- When Discriminator is still learning
- As it gets better, gradients to the Generator decrease
- Either the updates to the discriminator are inaccurate, or they disappear.
- Generator gets heavily penalized, which leads to saturation in the value post-activation function, and the eventual gradient vanishing.

Proposed Solution – modify Discriminator's activation function

Sigmoid → Linear

Discriminator feedbacks linear "score" instead of probability to the Generator.
Discriminator is a mere "critique", not a judge

Evaluation Metrics [8]

Inception Score (IS):

- Measures quality of generated images.
- Calculated by feeding the generated images into an Inception model and then measuring the entropy of the model's predictions.
- A higher IS score --> generated images are more realistic.

Fréchet Inception Distance (FID):

- measures the similarity between two sets of images.
- calculated by feeding the images into an Inception model and then measuring the distance between the distributions of the generated sets.
- lower FID score --> generated images are more similar to the real images.

Insert Web Page

This app allows you to insert secure web pages starting with https:// into the slide deck. Non-secure web pages are not supported for security reasons.

Please enter the URL below.

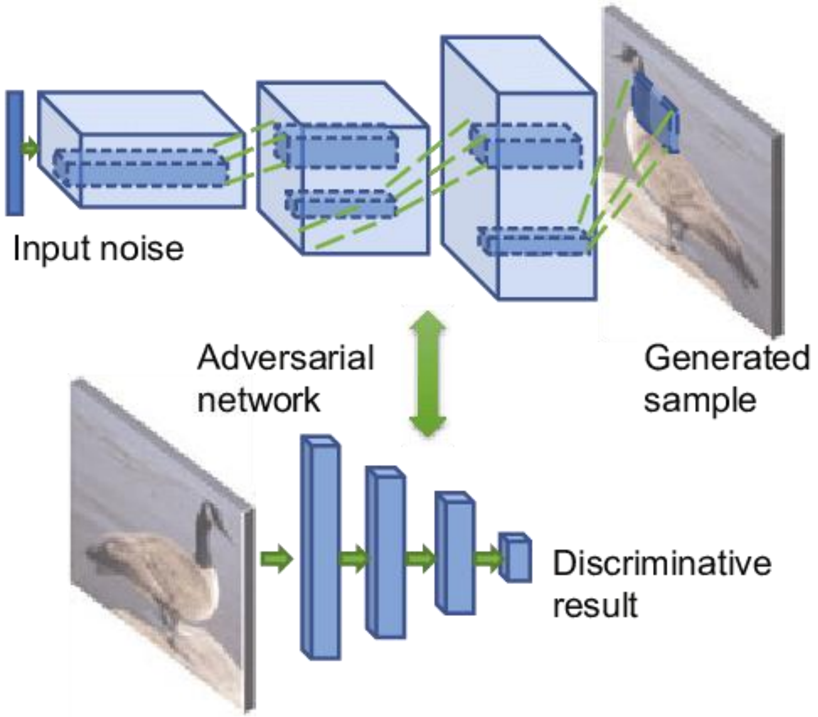
https:// poloclub.github.io/ganlab/

Note: Many popular websites allow secure access. Please click on the preview button to ensure the web page is accessible.

Web Viewer Terms | Privacy & Cookies Preview

Evolution of GAN 🚀

GAN Type [9]	Description
Vanilla GAN	The *original* GAN architecture, consisting of a generator and discriminator network.
Conditional GAN (CGAN)	A GAN that takes in *additional information*, such as a label, along with the random noise vector. This additional information can be used to control the output of the generator network.
Deep Convolutional GAN (DCGAN)	A GAN that uses convolutional neural networks (CNNs) for both the generator and discriminator networks.



https://tinyurl.com/5n6db5bw

GAN Code Training & Loss 💻

[GAN-Group4-Evaluation (google.com) [10]]

References

[1] “Generative adversarial network,” *Wikipedia*, Aug. 23, 2023. https://en.wikipedia.org/wiki/Generative_adversarial_network#/media/File:Generative_Adversarial_Network_illustration.svg (accessed Oct. 13, 2023).

[2] “Overview of GAN Structure | Generative Adversarial Networkttps://developers.google.com/machine-learning/gan/gan_structure

[3] T. Wood, “Generative Adversarial Network,” *DeepAI*, Jul. 22, 2020. https://deepai.org/machine-learning-glossary-and-terms/generative-adversarial-network

[4] R. Python, “Generative Adversarial Networks: Build Your First Models – Real Python,” realpython.com. https://realpython.com/generative-adversarial-networks/

[5] J. Hui, “GAN — What is Generative Adversarial Networks GAN?,” *Medium*, Dec. 26, 2019. https://jonathan-hui.medium.com/gan-whats-generative-adversarial-networks-and-its-application-f39ed278ef09

[6] P. Kishore, “Art of Generative Adversarial Networks (GAN),” *Medium*, Mar. 17, 2019. https://towardsdatascience.com/art-of-generative-adversarial-networks-gan-62e96a21bc35 (accessed Nov. 02, 2023).

[7] Y. Kossale, M. Airaj and A. Darouichi, "Mode Collapse in Generative Adversarial Networks: An Overview," 2022 8th International Conference on Optimization and Applications (ICOA), Genoa, Italy, 2022, pp. 1-6, doi: 10.1109/ICOA55659.2022.9934291.

[8] S. Guan and M. Loew, "Evaluation of Generative Adversarial Network Performance Based on Direct Analysis of Generated Images," 2019 IEEE Applied Imagery Pattern Recognition Workshop (AIPR), Washington, DC, USA, 2019, pp. 1-5, doi: 10.1109/AIPR47015.2019.9174595.

[9] J. Brownlee, “A Tour of Generative Adversarial Network Models,” *Machine Learning Mastery*, Jul. 09, 2019.

https://machinelearningmastery.com/tour-of-generative-adversarial-network-models/

[10] R. Python, “Generative Adversarial Networks: Build Your First Models – Real Python,” realpython.com. https://realpython.com/generative-adversarial-networks/

Thank you 🤟