Generative Adversarial Networks (GANs)

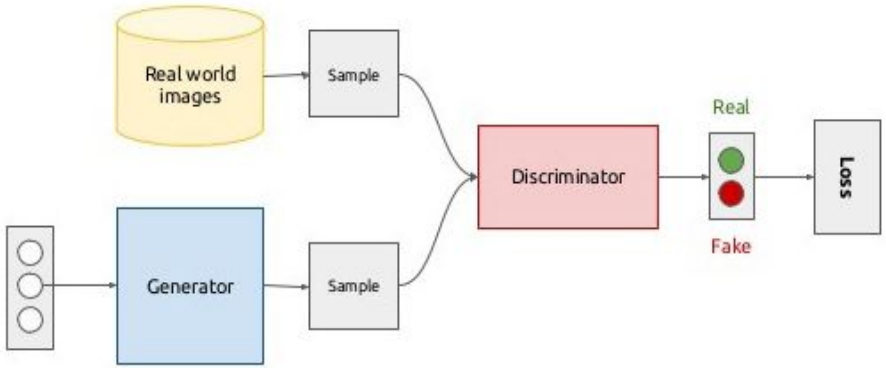
Arin Ghazarian
Chapman University

Generative Models

- A generative model learns to generate data similar to the training data given to it.
  - For instance, a generative model can learn to generate images similar to the images in a dataset used to train the model.
- The new synthetic instances produced by a generative model
  - will have the same statistics as the training set
  - plausibly could have been drawn from the training dataset.

Generative Adversarial Networks (GANs)

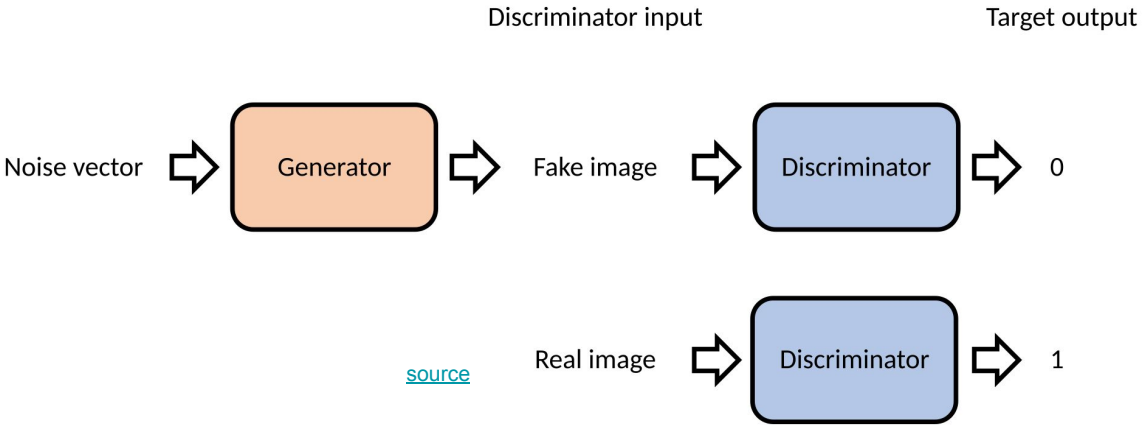
- GANs are a type of generative models based on deep learning neural networks such as convolutional neural networks.
- The main idea behind GANs is the discriminator, another neural network that can predict how realistic and genuine a given input seems.
- The discriminator itself is being updated during the training of a GAN so that it can better discern between the fake synthetic images and the real ones.
- In other words, the generator is not being trained to minimize the distance between its output and a specific image but instead deceives the discriminator model.



source

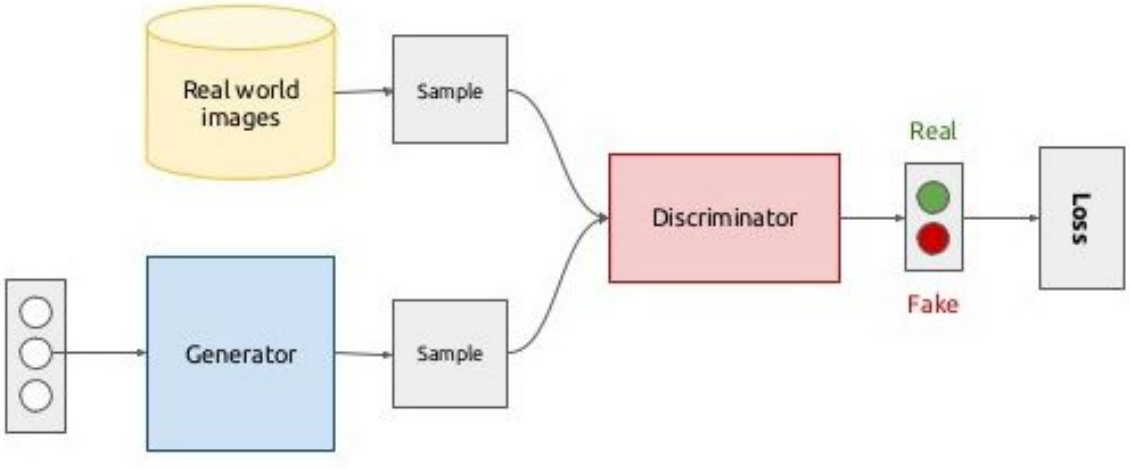
Generative Adversarial Networks (GANs)

- The GAN paper was published in 2014: Ian J. Goodfellow and Jean Pouget-Abadie and Mehdi Mirza and Bing Xu and David Warde-Farley and Sherjil Ozair and Aaron Courville and Yoshua. "Generative Adversarial Networks"



GAN Architecture

- A generator model produces instances from the problem domain, followed by a discriminator model that will classify the instances as real or fake (synthetic)

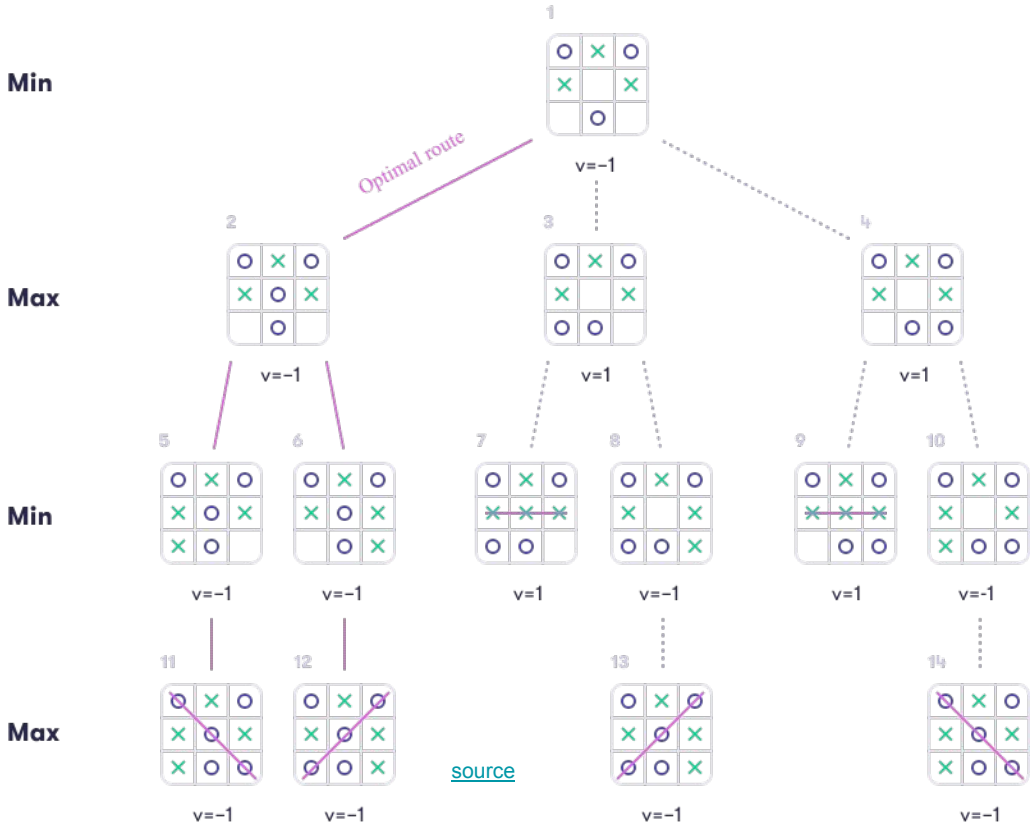


source

GANs: Game-theoretic View

- Generative adversarial networks are based on a game theoretic scenario in which the generator network must compete against an adversary. The generator network directly produces samples $x=g(z;\theta^{(g)})$.
- Its adversary, the discriminator network, attempts to distinguish between samples drawn from the training data and samples drawn from the generator. The discriminator emits a probability value given by $d(x;\theta^{(d)})$, indicating the probability that x is a real training example rather than a fake sample drawn from the model

MiniMax Algorithm in zero sum Games



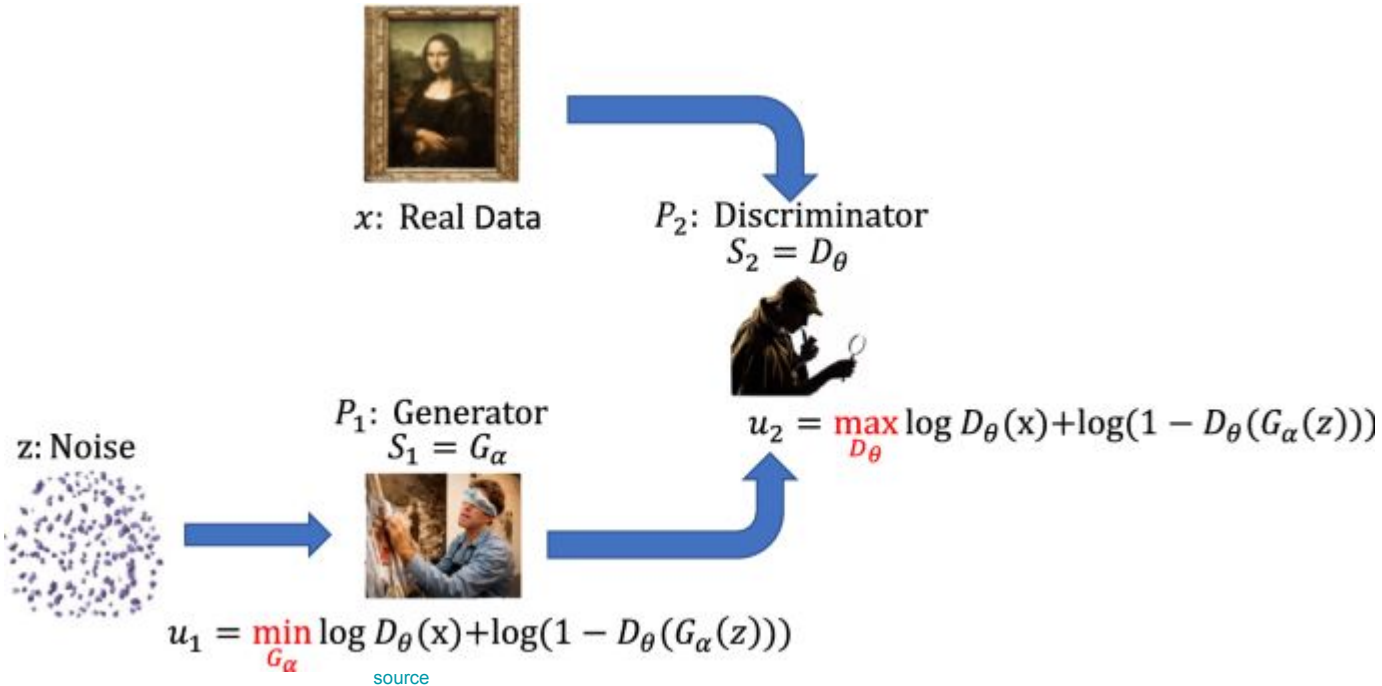
source

GANs: Game-theoretic View

- The simplest way to formulate learning in generative adversarial networks is as a zero-sum game, in which a function $v(\theta^{(g)}, \theta^{(d)})$ determines the payoff of the discriminator.
  - Zero-sum game is a mathematical representation in game theory and economic theory of a situation that involves two sides, where the result is an advantage for one side and an equivalent loss for the other
- The generator receives $-v(\theta^{(g)}, \theta^{(d)})$ as its own payoff. During learning, each player attempts to maximize its own payoff, so that at convergence:

$$g^* = \arg\min_g \max_d v(g, d).$$

minimizing the opponent's maximum payoff.


x: Real Data
P_2: Discriminator
$S_2 = D_\theta$
P_1: Generator
$S_1 = G_\alpha$
z: Noise
$u_2 = \max_{D_\theta} \log D_\theta(\mathrm{x}) + \log(1 - D_\theta(G_\alpha(z)))$
$u_1 = \min_{G_\alpha} \log D_\theta(\mathrm{x}) + \log(1 - D_\theta(G_\alpha(z)))$

source

GANs Loss Function: Game-theoretic View

- The default choice for v is

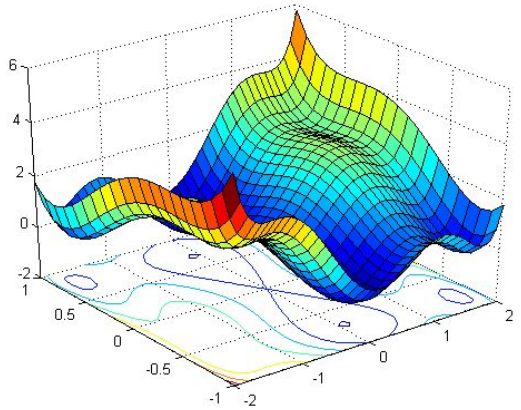
$$v(\boldsymbol{\theta}^{(g)}, \boldsymbol{\theta}^{(d)}) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \log d(\boldsymbol{x}) + \mathbb{E}_{\boldsymbol{x} \sim p_{\text{model}}} \log (1 - d(\boldsymbol{x})) .$$

- This drives the discriminator to attempt to learn to correctly classify samples as real or fake. Simultaneously, the generator attempts to fool the classifier into believing its samples are real.
- At convergence, the generator's samples are indistinguishable from real data, and the discriminator outputs ½ everywhere. The discriminator may then be discarded

$$\min_{\theta_g} \max_{\theta_d} \underbrace{\mathbb{E}_{p_{\text{data}}} [\log D_{\theta_d}(\mathbf{x})]}_{\text{likelihood of true data}} + \underbrace{\mathbb{E}_{p_z(\mathbf{z})} \left[\log \left(1 - D_{\theta_d}(G_{\theta_g}(\mathbf{z}))\right)\right]}_{\text{likelihood of generated data}}$$

Nonconvergence Issue with GANs

- Unfortunately, learning in GANs can be difficult in practice when g and d are represented by neural networks and $\max_d v(g, d)$ is not convex.
- In general, simultaneous gradient descent on two players' costs is not guaranteed to reach an equilibrium.



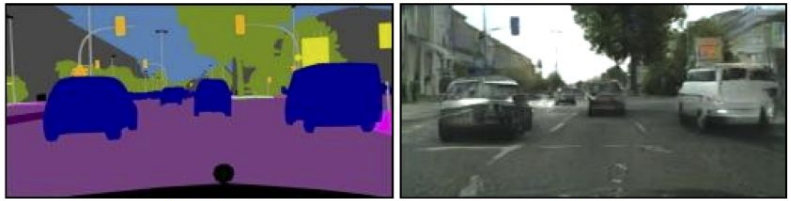
https://www.youtube.com/shorts/Pre8SbO0Qek

GAN Applications

- **Data augmentation:** generating more data samples for training neural networks. This also helps with the model's regularization.
- **Image-to-image translation:** Translating photos from summer to winter
- **Denoising**
- **Face aging**
- **Text-to-image**

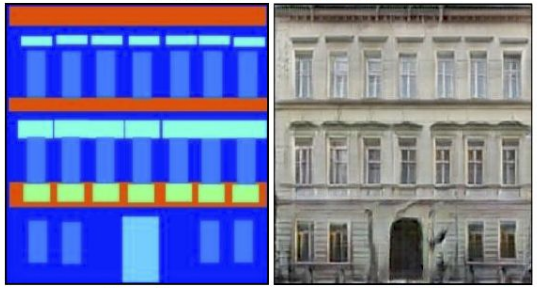
Image-to-image translation

Labels to Street Scene



input output

Labels to Facade



input output

BW to Color

input output

Aerial to Map

input output

Day to Night

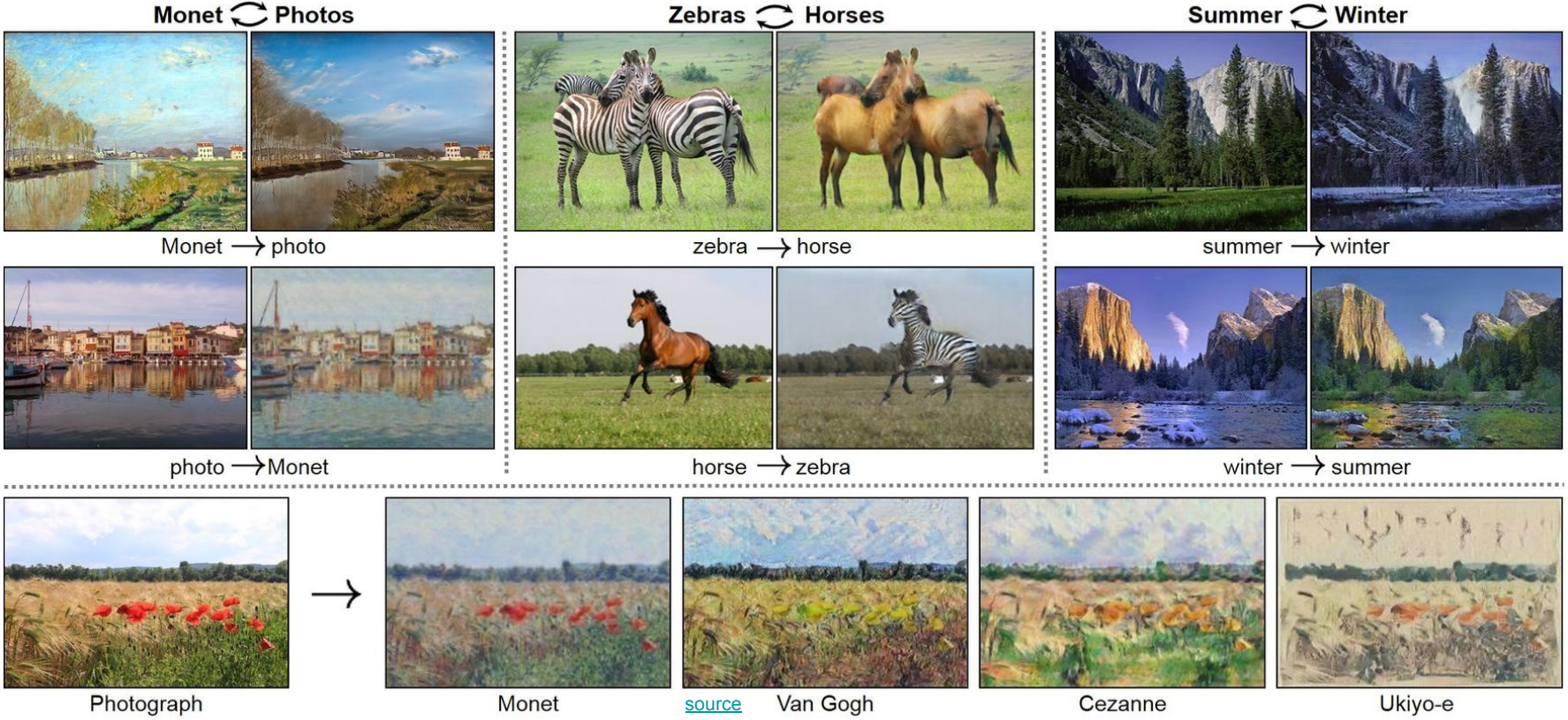
input output

Edges to Photo

input output

source

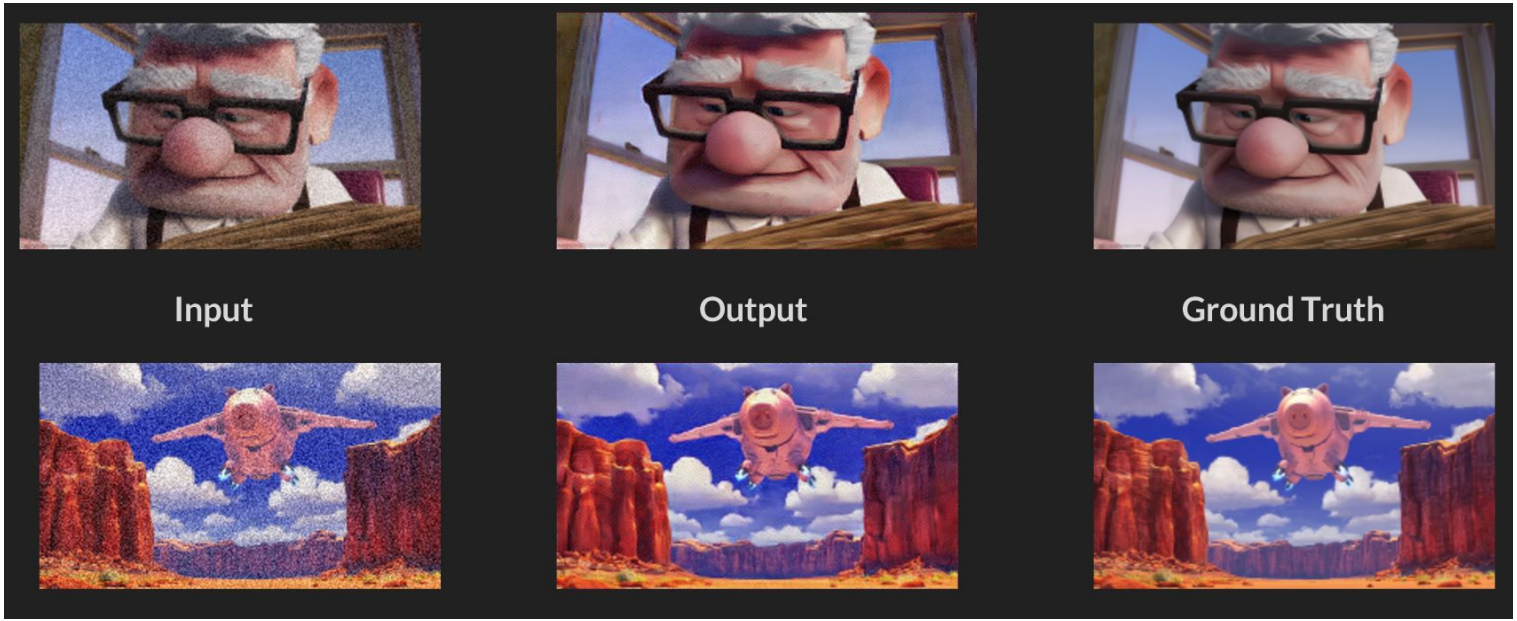
Image-to-image translation



Face Aging

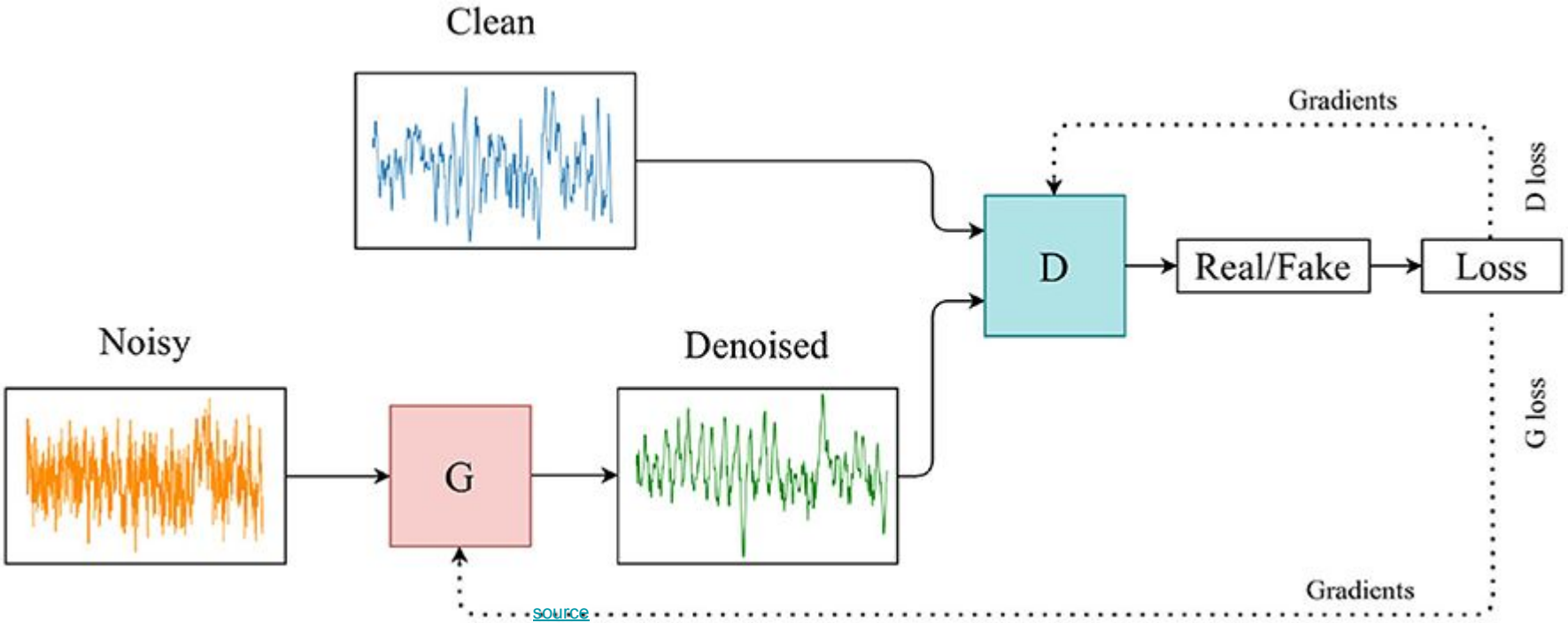
source

Denoising



source

Denoising


Clean
Gradients
D loss
D
Real/Fake
Loss
Noisy
Denoised
G
G loss
Gradients
source

The Adversarial Relationship in GANs

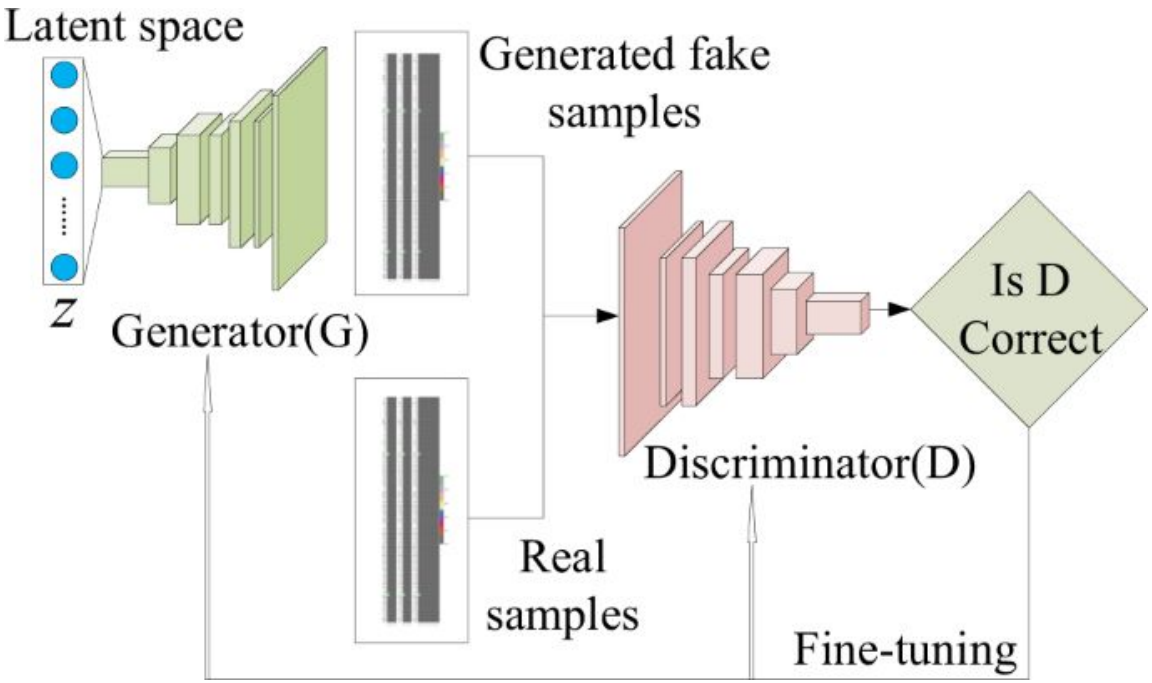
- There is an adversarial relationship between the generator and the discriminator.
  - If the discriminator distinguishes the fake instances from the real ones, then the generator is updated to generate more realistic outputs to better deceive the discriminator.
  - in contrast, when the discriminator performs poorly, the generator model will update its weights less.
- Ideally, the generator will eventually produce such a perfect instances so that the discriminator generates a 0.5 probability for both real and fake instances.

GANs as Unsupervised Models

- GANs **do not** try to generate an image and compare it to a real perfect image which is the actual answer
- GAN models are unsupervised techniques (no need for any labeled data), even though the discriminator model in a GAN is a supervised model
- We are usually only interested in the generator. After the GAN training is done, we can get rid of the discriminator

Training GANs

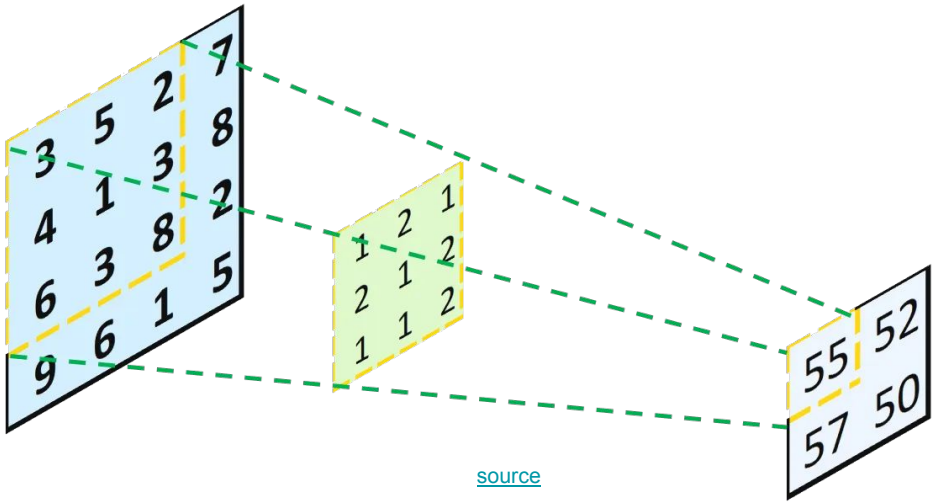
- GAN trainings can happen in two phases
  - Discriminator training
  - Generator Training



https://www.nature.com/articles/s41524-020-00352-0

Transposed Convolution: GANs For Image Generation

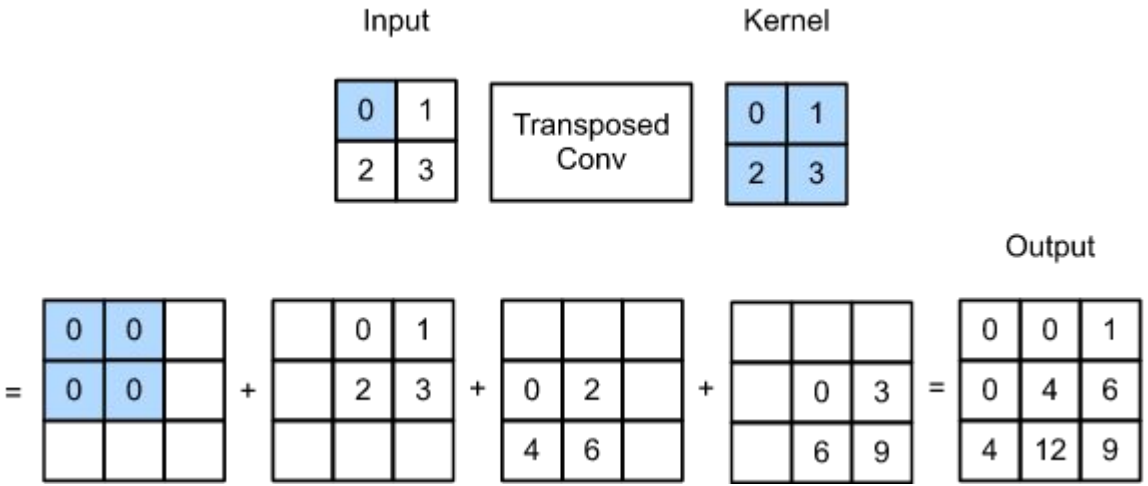
- In the generator, we use transpose conv2d operator to upsample the input
- Transposed convolution as the opposite of the convolution



source

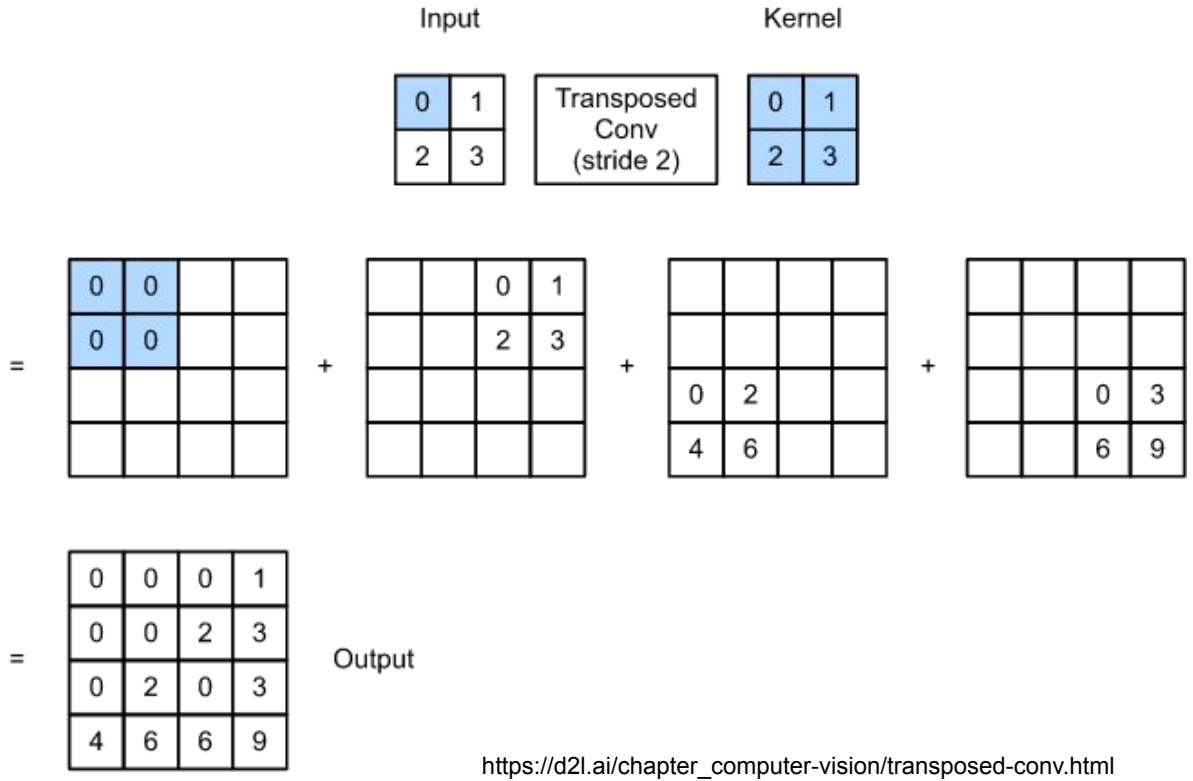
Transposed Convolution

- Multiply each value in the input layer by the 3x3 kernel to yield a 3x3 matrix. Then, we just combine all of them together according to the initial positions in the input layer, and sum the overlapped values together



https://d2l.ai/chapter_computer-vision/transposed-conv.html

Transposed Convolution: Stride



https://d2l.ai/chapter_computer-vision/transposed-conv.html

Discriminator’s Training

- Discriminator’s training data comes from both the actual training data and the generator:
  1. In the first step, it will get the real data and try to predict them as real (probability 1)
  2. Then, it will try to detect the fake samples from the generator.
- If the discriminator makes a wrong prediction, its loss function will penalize the model and the model weights will be updated using backpropagation

Generator’s training

- During the generator's training, we only update the generator model and we freeze the discriminator layers (we do not update them):

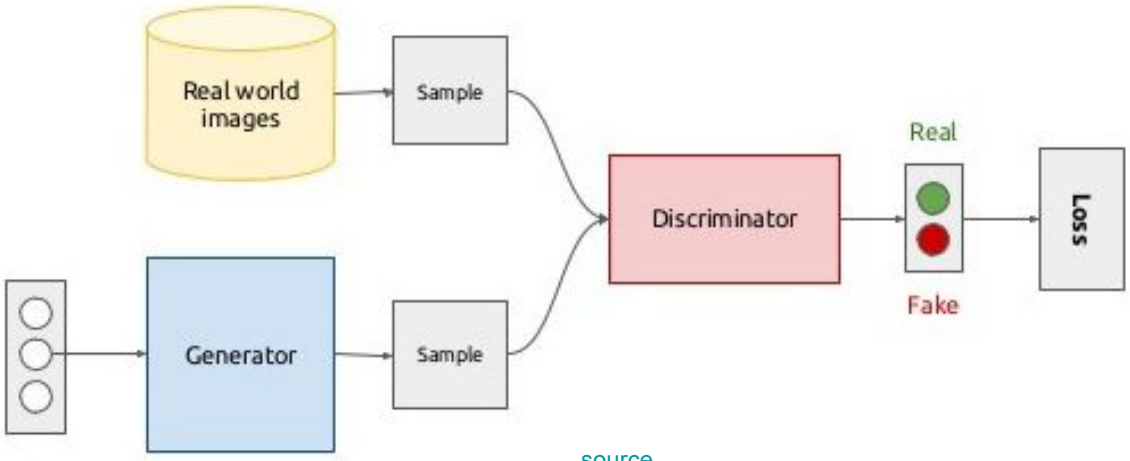
1. The generator will use a random seed to produce fake samples.
2. Then, this synthetic sample will be fed to the discriminator for classification. If the discriminator classifies correctly (as fake), then the generator loss function will penalize the generator. The generator will update its weight using backpropagation from the generator loss function.
    a. **Note:** We assign 1 to the fake image labels.

The Latent Space

- To generate an instance, the generator model takes a random seed vector as input drawn randomly from a Gaussian distribution.
- These points in this multidimensional vector space eventually create a compressed representation of the training data distribution, representing the points in the training dataset.
- This vector space is known as the latent space

Robust to Overfitting

- Generator never sees the training data.



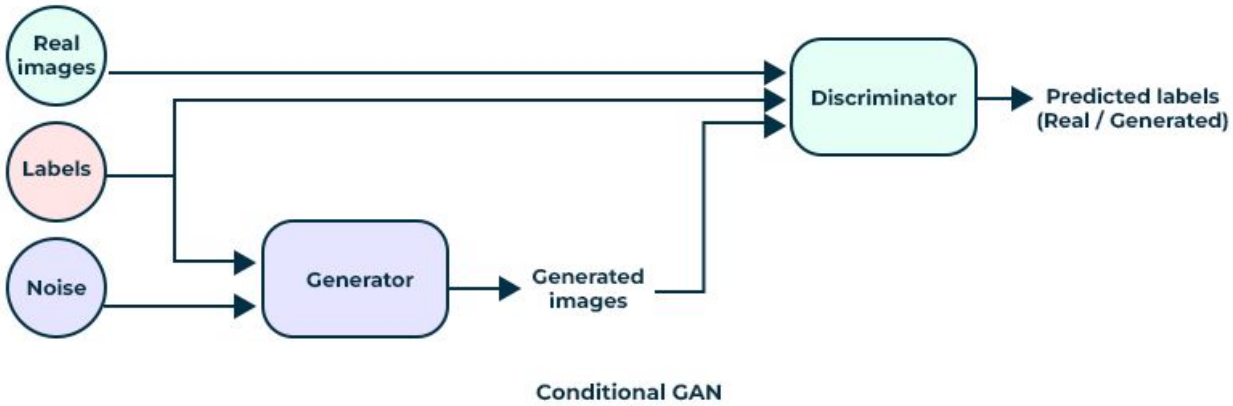
source

Training Problems

- Non-Convergence
  - SGD was not designed to find the Nash equilibrium of a game
  - It might never converge to the Nash equilibrium at all
- Mode-Collapse
  - Generator fails to output diverse samples
  - Generator produces only a small set of outputs but good
  - Mini-Batch GANs and Supervision with labels techniques help to address this issue

Conditional GANs

- If we have access to labels, we can inject it to both generator and discriminator



Conditional GAN

source

Images generated by GANs trained on the LSUN dataset. (Left)Images of bedrooms generated by a DCGAN model, reproduced, Radfordet al. (2015). (Right)Images of churches generated by a LAPGAN model, Denton et al. (2015)

Demo:Training GAN on CIFAR-10 Using Keras

- http://localhost:8888/notebooks/jupyter-notebooks/chapman-generative-AI/GAN.ipynb

Demo: GAN on MNIST Using PyTorch

- https://github.com/SimpleSchwarz/GAN/blob/main/DCGAN_MNIST/DCGAN_MNIST.ipynb

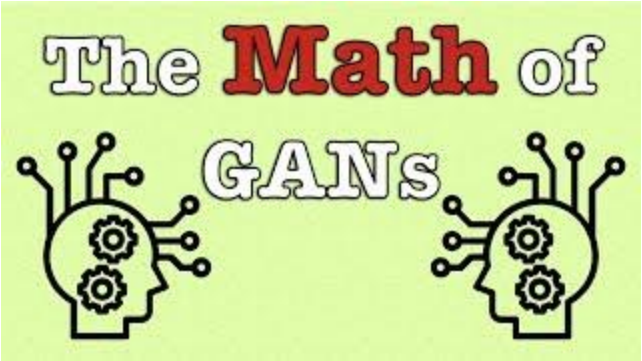
Demo: Conditional GAN

https://pub.towardsai.net/a-beginners-guide-to-building-a-conditional-gan-d261e4d94882

Demo: Denoising Images Using GANs

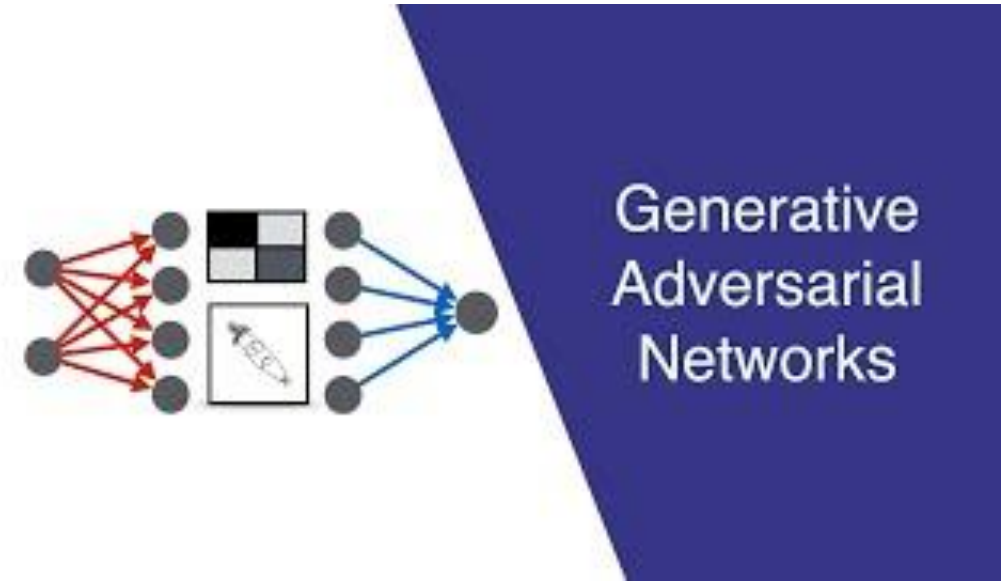
- https://github.com/rr250/Image-Denoising-Using-GAN/blob/master/Image%20Denoising%20Using%20GAN.ipynb

https://www.youtube.com/watch?v=OXWvrRLzEaU



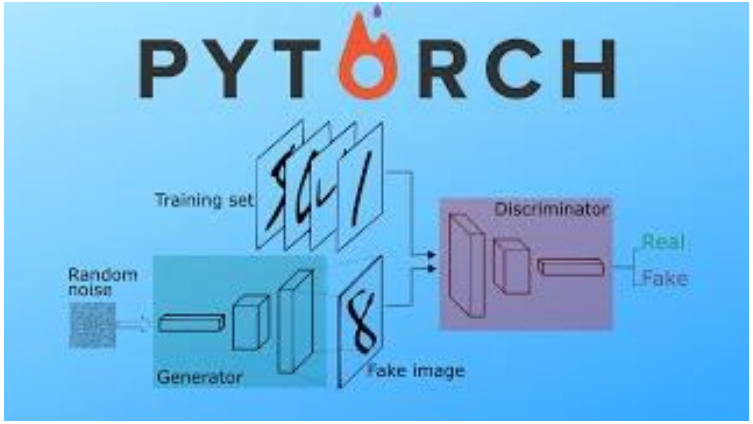
https://www.youtube.com/watch?v=J1aG12dLo4I

https://www.youtube.com/watch?v=xBX2VlDgd4I

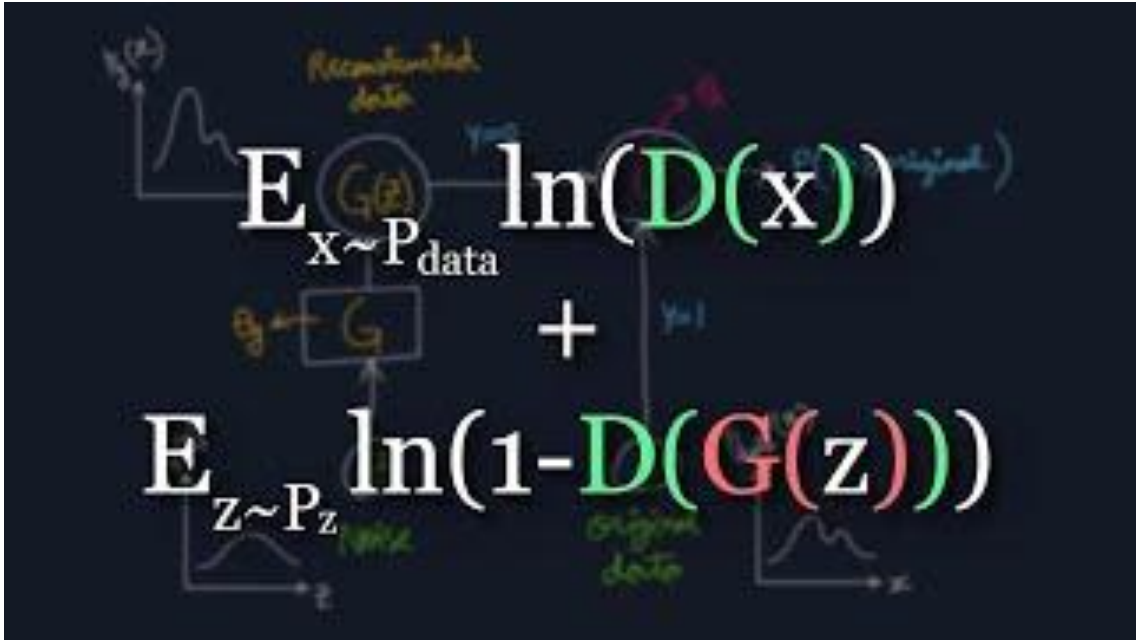


https://www.youtube.com/watch?v=8L11aMN5KY8&t=674s

Simple GAN using Pytorch



https://www.youtube.com/watch?v=OljTVUVzPpM



https://www.youtube.com/watch?v=Gib_kiXgnvA

Additional Readings

- https://slazebni.cs.illinois.edu/spring17/lec11_gan.pdf