1. Info on GANs :

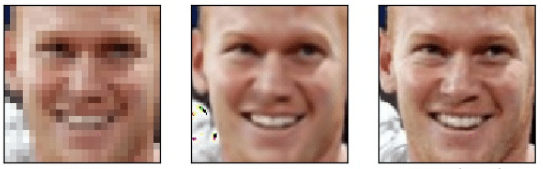
GANs stands for Generative Adversarial (conflict or opposition) Network.

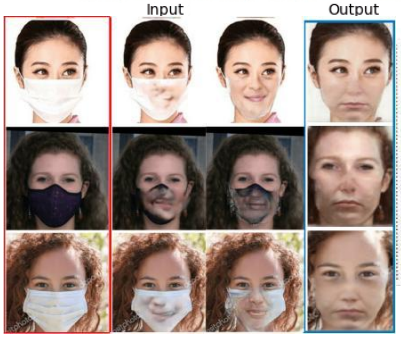
Types of GANs :

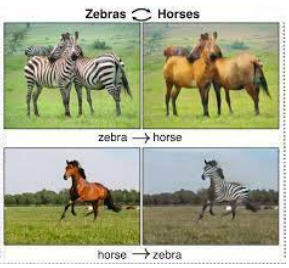
- FCGAN : Fully Connected
- DCGAN : Deep Convolutional
- SRGAN : Super Resolution
- StyleGAN
- Many more

Application of GANs:

- Image generation
- Style Transfer
- Increasing resolution of blurred images
- Facial Reconstruction
- Many More





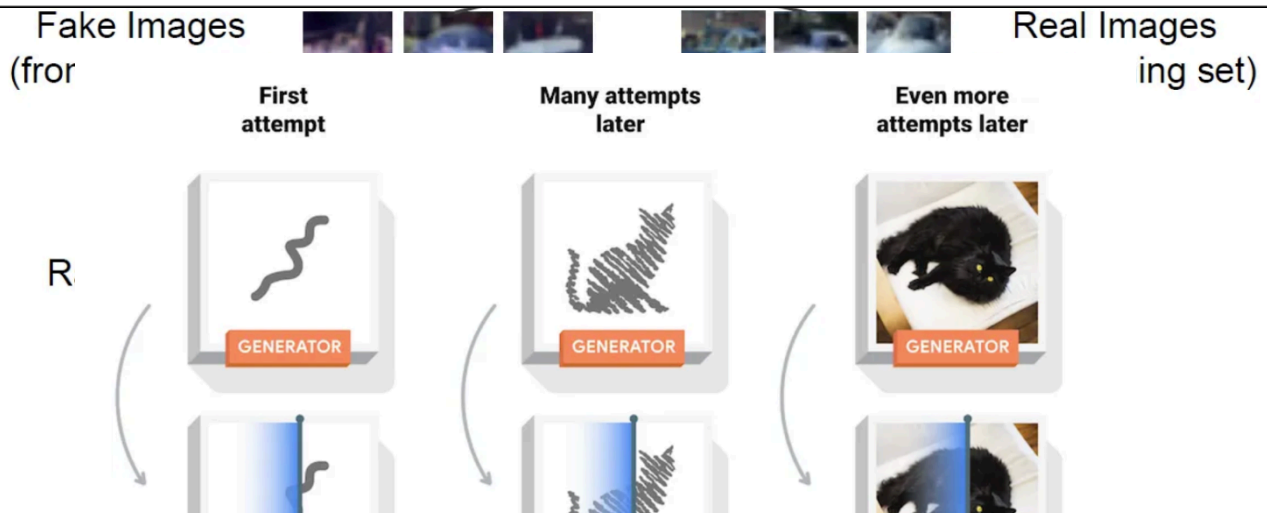


A generator network takes a random normal distribution (z), and outputs a generated sample that's targeted to be close to the original distribution.

A discriminator tries to evaluate the output **generated by the generator** with the original sample, and outputs a value between 0 and 1. If the value is close to 0, then the generated sample is said to be fake, and if the value is close to 1 then the generated sample is said to be real.

Real or Fake

In short, **the discriminator's job is to identify whether the generated sample is real or fake** by comparing it with the original sample. The **generator's job is to fool the discriminator** by generating samples that are close to the original sample.



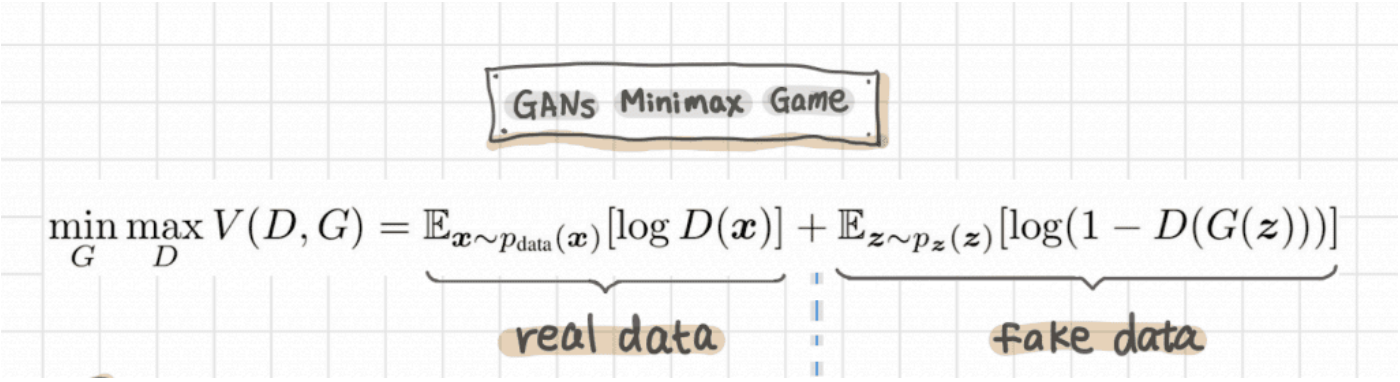
Discriminator's POV

In the Figure above, **generator generated some random noise at first which gets easily detected by the discriminator** as fake by saying the loss is big , which makes generator to learn the image better so generator makes another attempt to draw the image such to minimize the loss.

So the discriminator tries to maximize the loss by learning good classification of real or fake, and the generator tries to minimize the loss by generating close to real image. This is called **mini-max game**.

Both generator and discriminator fights again each other **until generator starts generating images which discriminator is not able to classify**. Discriminator reaches a stage where each image has 50% chance to be classified as fake or real.

2. Mathematically: the two-player



*We will get into this later, don't get overwhelmed

Random noise Dim = 100 → [Generator: 100 → conv. → 28×28×3] → (R G B) each 28×28

For a B & W scale → (28 × 28 × 1)

Generated Image (28 × 28 × 3) → fed as real (label 1) → [Discriminator] → 0.2 (somewhat fake)

(Assumed to be trained already)

real → 1
fake → 0

(faking part)

Loss in GANs

$\log(1 - D(G(z)))$

basically, a MinMax game

(z) Input to Generator

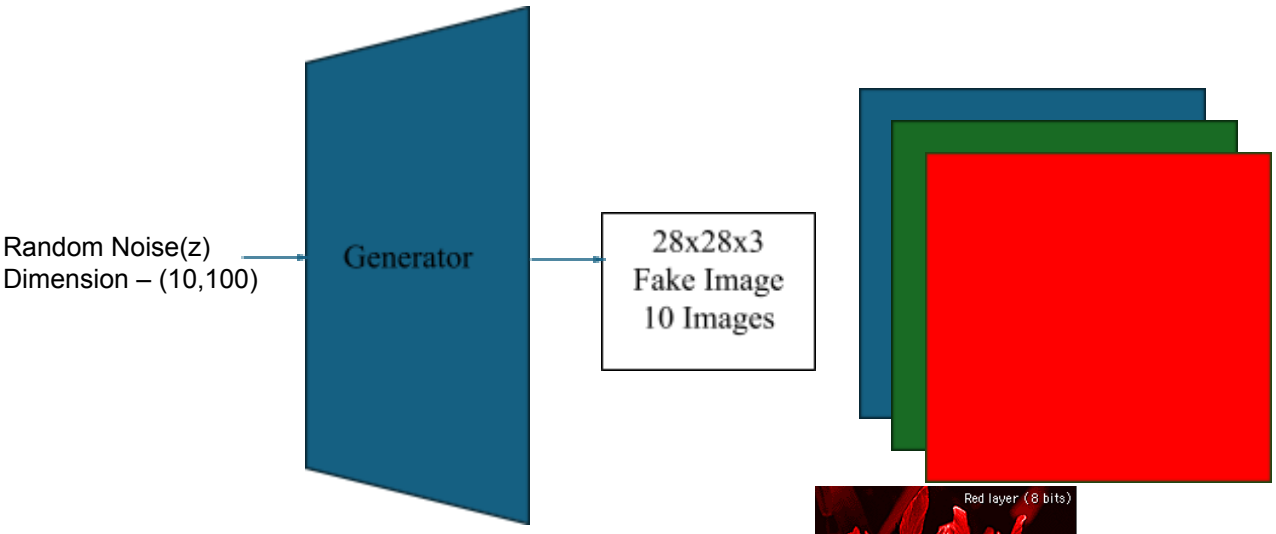
D is trying to increase the loss function, while G is trying to decrease it.

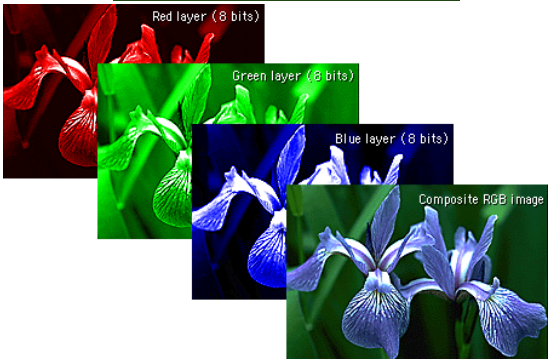
Loss in GANs (real ing part)

Real Img (28×28×3) → Discriminator → 0.9 (most probably real)

Loss = $\log(D(x))$ → real image

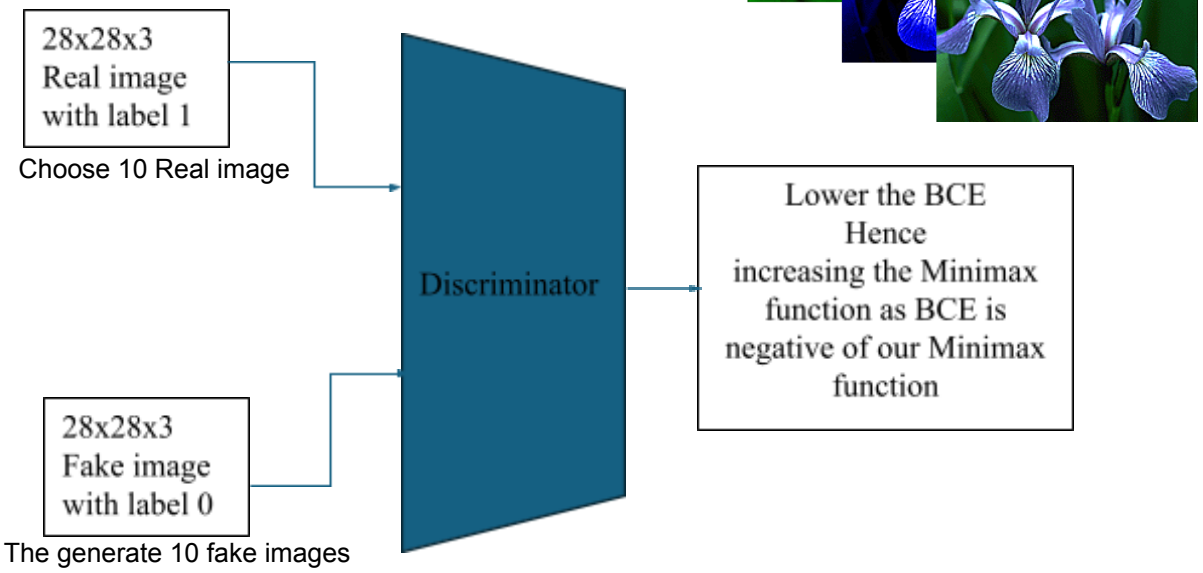
D tries to bring D(x) as close as possible to "1", maximising the loss.



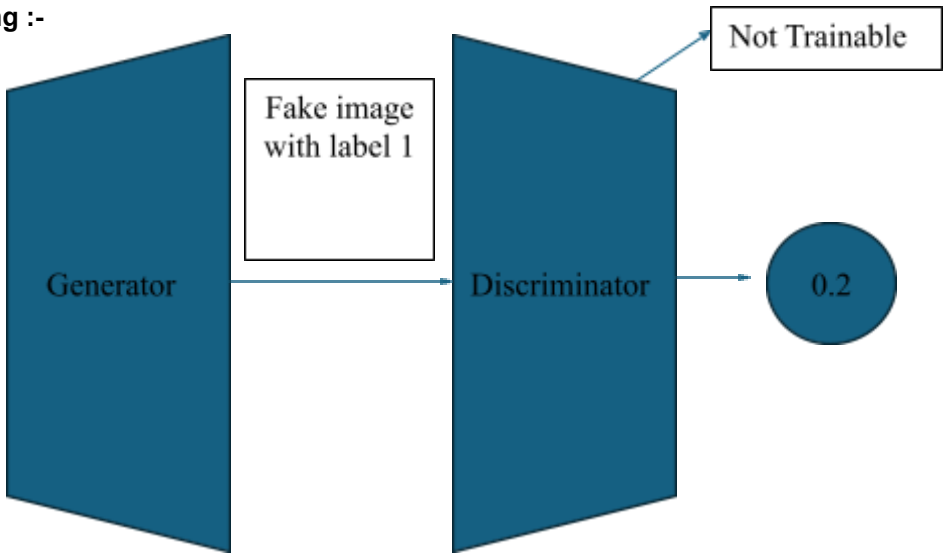


Sequence of Training GANs

1) **Discriminator Training :-**



2) **Generator Training :-**



Let's take a look at binary cross entropy loss function,

$$L_{BCE} = -\frac{1}{n}\sum_{i=1}^{n}(Y_i \cdot \log \hat{Y}_i + (1 - Y_i) \cdot \log (1 - \hat{Y}_i))$$

Here Y_i is the ground truth and $\hat{Y}_i$ is the predicted label.

If Y_i = 1, $L_{BCE} = -\frac{1}{n}\sum_{i=1}^{n} \log \hat{Y}_i$ then

If Y_i = 0,then $L_{BCE} = -\frac{1}{n}\sum_{i=1}^{n} \log (1 - \hat{Y}_i))$

Doesn't this look familiar to our GANs loss function, here's how:

$\hat{Y}_i$ is the predicted label which will be represented by output of discriminator { D(x) or D(G(z)) } in GANs.

If Y_i = 1 (image is real), then Loss = $\log (D(x))$

If Y_i = 0 (image is generated),the $\log (1 - D(G(z))$ Loss =

x represents the real image data, whereas **z** represent the random noise which we feed into the generator.

Instead of summation of log loss over all data, **we take expectation values of the loss and add them together to form the minimax loss function**. Hence we get this :

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

Where x represents the real image batch and z represents the noise.

Discriminator maximizing:

You can see here if image is real, Discriminator will produce D(x) near to 1 hence increasing the loss and if image is fake, D(G(z)) will be near to 0 hence again increasing the loss. Hence gradient ascent happens while training the Discriminator to maximize the loss functions.

Generator Minimizing:

You can see that is G(z) { generated image) is closer to x { real image }, discriminator will likely give D(G(z)) near 1 hence minimizing the loss. Hence gradient descent happens while training the Generator to minimize the loss functions.

Here you can see the Generator minimizing the loss and Discriminator Maximizing the loss.

3. Let's Dive into Code :

Vanilla Old TF : Kaggle
Vanilla GANS : Main_GANs.ipynb