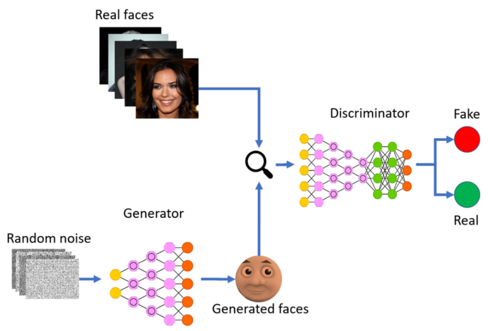
Generative Adversarial Networks (GANs)

Pradip Das
pradipdas_2021@iitkalumni.org

1 Architecture

Generative Adversarial Networks (GANs) are a class of deep learning models that generate new data samples that resemble a given dataset. GANs consist of two neural networks, a generator and a discriminator, that compete in a **minimax** game.



The job of generator is to produce an image that looks so natural that the discriminator thinks that the image comes from a real data distribution. The job of discriminator is to get better and better at distinguishing between the true image and the generated (fake) image.

Generator (G_ϕ)**:**

- A neural network that maps a random noise vector $z \sim p_z(z)$ to a data distribution $G_\phi(z) = X$, generally $p_z = \mathcal{N}(0, I)$.
- $G_\phi : \mathbb{R}^d \to \mathbb{R}^n$, where d is the dimension of the latent space and n is the dimension of the generated data.
- Objective: Generate realistic-looking samples that can fool the discriminator.

Discriminator (D_θ)**:**

- A neural network that takes an input X and outputs a probability $D_\theta(X) \in (0, 1)$ indicating whether X is real (from the true data distribution) or fake (generated by G_ϕ).
- $D_\theta : \mathbb{R}^n \to (0, 1)$, where n is the dimension of the true data.
- Objective: Correctly distinguish between real and fake samples.

2 Objective Function:

Let's look the objective function of generator first. Given an image generated by the generator as $G_\phi(z)$ the discriminator assigns a score $D_\theta(G_\phi(z))$ to it. This score is between 0 and 1 and will tell us the probability that the image is real or fake.

For a given z, the generator would want to maximize $logD_\theta(G_\phi(z))$ (log likelihood) or minimize $log(1 - D_\theta(G_\phi(z)))$. This is just for a single z and the generator would like to do this for all possible z.

For example z is discrete and drawn from a uniform distribution (*i.e.* $p(z) = \frac{1}{N}\ \forall z$ then the generator objective function would be

$$\min_\phi \sum_{i=1}^{N} \frac{1}{N} log(1 - D_\theta(G_\phi(z)))$$

As in our case z is continuous and not uniform ($z \sim \mathcal{N}(0, I)$) so the objective function would be

$$\min_\phi \int p(z) log(1 - D_\theta(G_\phi(z))) \tag{1}$$

$$\min_\phi \mathbb{E}_{z\sim p(z)}[log(1 - D_\theta(G_\phi(z)))] \tag{2}$$

Now let's look at discriminator. The task of discriminator is to assign a high score to real images and a low score to fake images. In other words, it should try to maximize the following objective function:

$$\max_\theta \mathbb{E}_{x\sim p_{data}}[logD_\theta(x)] + \mathbb{E}_{z\sim p(z)}[log(1 - D_\theta(G_\phi(z)))] \tag{3}$$

If we put the objective function of generator and discriminator together we get a minimax game

$$\min_\phi \max_\theta [\mathbb{E}_{x\sim p_{data}}\ logD_\theta(x) + \mathbb{E}_{z\sim p(z)}\ log(1 - D_\theta(G_\phi(z)))] \tag{4}$$

The discriminator wants to maximize the second term whereas the generator wants to minimize it, hence it is a two-player game. *This is the objective function of vanilla GAN.*

3 Training

The over-all training proceeds by alternating between two steps.

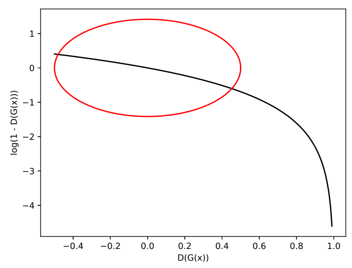
- **Step 1:** Gradient ascent on Discriminator

$$\max_\theta [\mathbb{E}_{x\sim p_{data}}\ logD_\theta(x) + \mathbb{E}_{z\sim p(z)}\ log(1 - D_\theta(G_\phi(z)))]$$

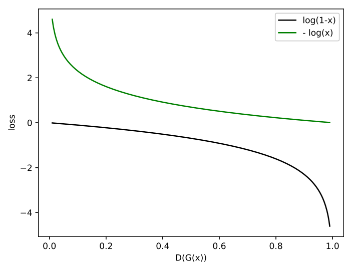
- **Step 2:** Gradient descent on Generator

$$\min_\phi \mathbb{E}_{z\sim p(z)}\ log(1 - D_\theta(G_\phi(z)))]$$

In practice, the above gradient descent does not work well and uses a slightly modified objective. Let us see why?



(a)



(b)

When the sample is likely to fake, we want to give a feedback to the generator (using gradient descent). However in the region (marked red in Fig. (a)) where $D_\theta(G_\phi(z))$ is close to 0, the curve of the loss function is very flat and the gradient is close to 0. Instead of minimizing the likelihood of discriminator being correct, maximize the likelihood of discriminator being wrong (green graph in Fig. (b)). In effect, the objective remains the same, but the gradient signal becomes better.

Algorithm 1 GAN Training: Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter

```
for number of training iteration do
    for $k$ step do
        • sample minibatch of $m$ noise samples $\{z_1, z_2, .., z_m\}$ from the noise prior $p_z(z)$
        • sample minibatch of $m$ examples $\{x_1, x_2, .., x_m\}$ from the prior data generating distribution $p_{data}(x)$
        • Update the discriminator by ascending its stochastic gradient:
              $$\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} [logD_\theta(x_i) + log(1 - D_\theta(G_\phi(z_i)))]$$
    end for
    • sample minibatch of $m$ noise samples $\{z_1, z_2, .., z_m\}$ from the noise prior $p_z(z)$
    • Update the generator by descending its stochastic gradient:
          $$\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} [log\ D_\theta(G_\phi(z_i))]$$
end for
```

GAN Architecture

Now look at one of the popular neural networks used for generator and discriminator (Deep Convolutional GANs). For any discriminator, any CNN based classifier with one class (real) at the output can be used (e.g. VGG-Net, ResNet, etc.). For generator in first layer project and reshape are used to get a matrix from noise vector and then the transpose convolution is used to get more higher dimensional matrices.

Architecture guidelines for stable deep-convolutional GANs:

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use $batchnorm$ in both generator and discriminator.
- Remove fully connected hidden layers for deeper architecture.
- Use $ReLU$ activation in generator for all layer except for the output, which uses $tanh$.
- Use $LeakyReLU$ activation in the discriminator for all layers.

4 Math Behind It

Let us denote the true data distribution by $p_{data}(x)$ and the distribution of data generated by the model as $p_G(x)$. At the end of the training we want $p_{data}(x) = p_G(x)$
Let define,

$$\begin{aligned} V(D,G) &= \mathbb{E}_{x\sim p_{data}}\, logD(x) + \mathbb{E}_{z\sim p(z)}\, log(1 - D(G(z))) && (5)\\ &= \mathbb{E}_{x\sim p_{data}}\, logD(x) + \mathbb{E}_{z\sim p_G}\, log(1 - D(z)) && (6)\end{aligned}$$

Then the objective is $\min_G \max_D V(D,G)$.

Theorem: For fixed G, the optimal discriminator D is

$$D^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_G(x)}$$

Proof. For a given G, the optimal $D^* = \arg\max_D V(D,G)$

$$\begin{aligned} V(D,G) &= \mathbb{E}_{x\sim p_{data}}\, logD(x) + \mathbb{E}_{x\sim p_G}\, log(1 - D(x))\\ &= \int_x p_{data}(x)\, logD(x)dx + \int_x p_G(x)\, log(1 - D(x))dx\\ &= \int_x [p_{data}(x)\, logD(x) + p_G(x)\, log(1 - D(x))]dx\end{aligned}$$

For a given x, optimal D^* maximizing

$$f(D) = p_{data}(x)\, logD(x) + p_G(x)\, log(1 - D(x))$$

Then,

$$\frac{df(D)}{dD} = 0 \implies \frac{p_{data}(x)}{D(x)} - \frac{p_G(x)}{1 - D(x)} = 0$$
$$\implies D^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_G(x)}$$

$\square$

Now, with optimal D the minimax game in Eq. 6 can be reformulated as

$$\begin{aligned}
C(G) &= \max_D V(D, G) \\
&= V(D^*, G) \\
&= \mathbb{E}_{x\sim p_{data}}\ logD^*(x) + \mathbb{E}_{z\sim p_G}\ log(1 - D^*(G(z))) \\
&= \mathbb{E}_{x\sim p_{data}}\ logD^*(x) + \mathbb{E}_{x\sim p_G}\ log(1 - D^*(x)) \\
&= \mathbb{E}_{x\sim p_{data}}\ \left[log\frac{p_{data}(x)}{p_{data}(x) + p_G(x)}\right] + \mathbb{E}_{x\sim p_G}\ \left[log\frac{p_G(x)}{p_{data}(x) + p_G(x)}\right]
\end{aligned}$$

Theorem: The global minimum of the criterion $C(G)$ is achieved if and only if $p_G = p_{data}$.

Proof.

$$\begin{aligned}
C(G) &= \mathbb{E}_{x\sim p_{data}}\ \left[log\frac{p_{data}(x)}{p_{data}(x) + p_G(x)}\right] + \mathbb{E}_{x\sim p_G}\ \left[log\frac{p_G(x)}{p_{data}(x) + p_G(x)}\right] \\
&= \int_x p_{data}(x)log\frac{p_{data}(x)}{p_{data}(x) + p_G(x)}dx + \int_x p_G(x)log\frac{p_G(x)}{p_{data}(x) + p_G(x)}dx \\
&= \int_x p_{data}(x)log\frac{\frac{1}{2}p_{data}(x)}{\frac{p_{data}(x)+p_G(x)}{2}}dx + \int_x p_G(x)log\frac{\frac{1}{2}p_G(x)}{\frac{p_{data}(x)+p_G(x)}{2}}dx \\
&= -2log2 + \int_x p_{data}(x)log\frac{p_{data}(x)}{\frac{p_{data}(x)+p_G(x)}{2}}dx + \int_x p_G(x)log\frac{p_G(x)}{\frac{p_{data}(x)+p_G(x)}{2}}dx \\
&= -2log2 + KL\left(p_{data}||\frac{p_{data} + p_g}{2}\right) + KL\left(p_G||\frac{p_{data} + p_g}{2}\right) \\
&= -log4 + 2 \cdot JS(p_{data}||p_G)
\end{aligned}$$

Since the Jensen–Shannon divergence between two distributions is always non-negative and zero only when they are equal, we have shown that $C^* = -log4$ is the global minimum of $C(G)$ and that the only solution is $p_G = p_{data}$. $\square$

So in vanilla GAN from cross-entropy loss, we get Jensen-Shannon divergence.

Ref

- Vanilla GAN paper