GAN's

Ganapathi Subramanyam Jayam
Balaji D

AGENDA

- About GAN’s
- Working of GAN’s
- Steps to implement GAN’s
- Complete Math of GAN’s
- Types of GAN’s evolved
- Application of GAN’s
- GAN’s using MISI Fashion dataset

Generative Adversarial Network

Ian Goodfellow

- Generative- Unsupervised Learning Approach
- Adversarial- Model is trained in Adversarial
- Network- Neural Networks

GAN's Architecture consists of:

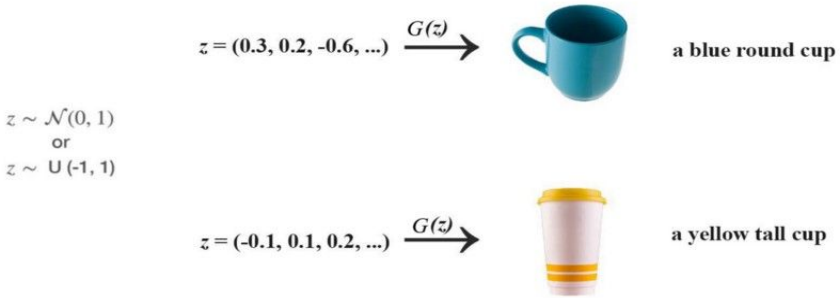
- Generative Network
- Discriminative Network

Working of GAN's

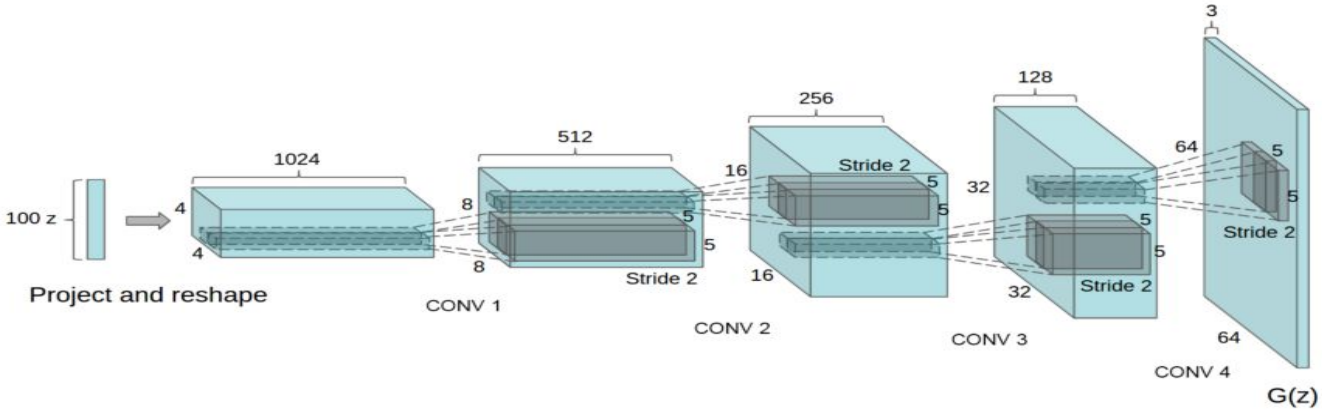
The main focus for GAN (Generative Adversarial Networks) is to generate data from scratch, mostly images but other domains including music have been done.

Generator and Discriminator:

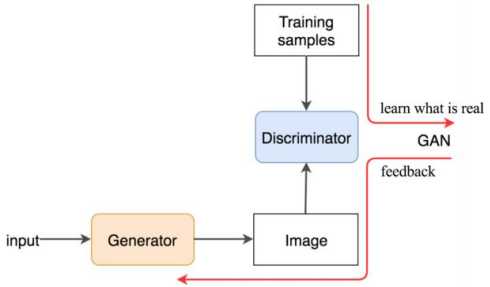
GAN composes of two deep networks one is generator and other is discriminator, Firstly we will talk about how generator works and how it generates images:First we sample some noise Z using either normal (0-1) or uniform (-1 - 1) distribution, with z as input we use a generator G to create an image x (x=G(z))



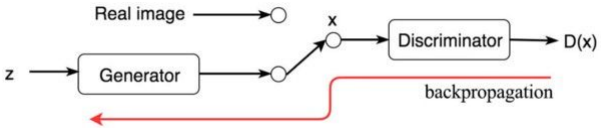
Here, as we know that in deep learning classification we don't control the features of the model is learning, the same thing is applicable to GAN we let the training process to learn it. Now coming to the structure of the generator there can be many design structures one can create as many as hidden layers they wish to perform the training on depending on their requirements, but one of the most popular designs of the generator is DCGAN(deep convolutional generative adversarial networks).it mainly composes convolution layers without any max pooling or fully connected layers.



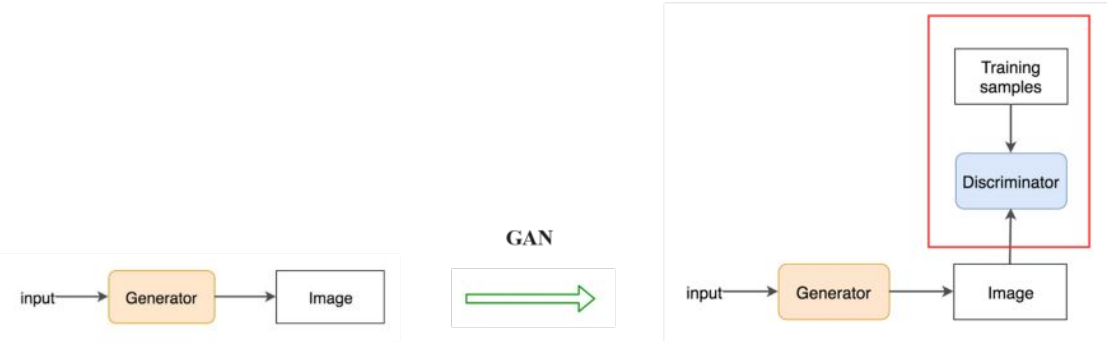
Generator alone creates just a random noise we need discriminator that providence guidance to the generator on what images to create.



The discriminator looks at real images and generated images separately, it distinguishes the image is generated or real. We train the discriminator just like the neural networks, if the input is real we want D(x)=1,if it is generated it should be zero. On the other hand we want generator to create images with D(X)=1, so we train the generator by back propagating the value back to the generator from discriminator.



We train both the networks in alternating steps to improve both the networks, eventually the discriminator finds a tiny difference between generated and real images, and the GAN model generates eventually real images.



Simple steps to implement GAN's

1. Define the problem
2. Choose Architecture of GAN
3. Train Discriminator of Real Data
4. Generate fake input by generator
5. Train Discriminator on fake data
6. Train generator with the output of discriminator

Math behind GAN's:

When coming to maths behind GAN's we need to focus on one main topic that is backpropagation, let the discriminator outputs the value D(x) indicating the chance of image being real, our objective is maximize the chance of real image being real and generated image being fake, to measure the loss we generally use cross entropy i.e *p log(q)*. For real images this has to be 1, and for generated images this has to be reverse, so we reverse the label, so the objective becomes:

$$\max_D V(D) = \underbrace{\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})]}_{\text{recognize real images better}} + \underbrace{\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]}_{\text{recognize generated images better}}$$

On the generator side, its objective function wants the model to generate images with the highest possible value of D(x) to fool the discriminator.

$$\min_G V(G) = \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

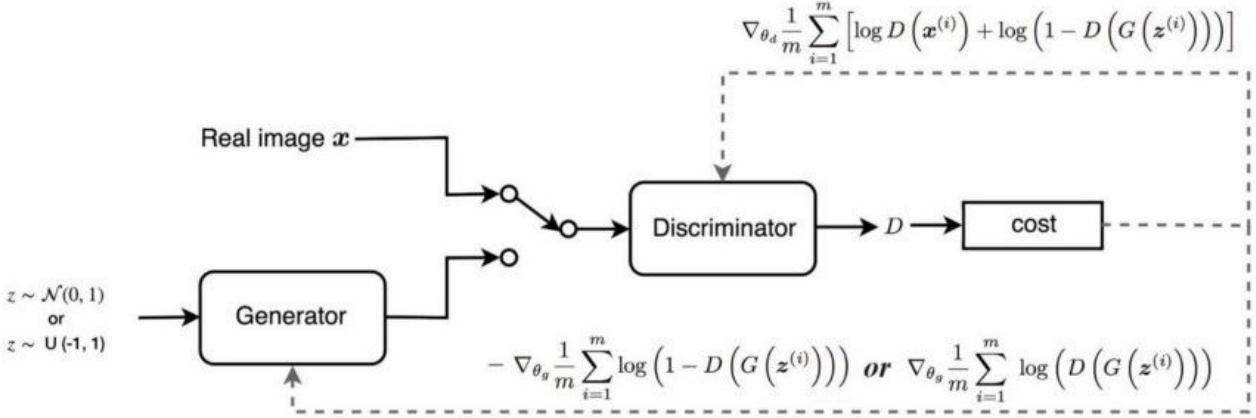
Optimize G that can fool the discriminator the most.

We can simply tell it as a minmax game where G wants to minimize v while D wants to maximize it.

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

Once the both objective are decided they are learned jointly by the alternating gradient descent.

We fix the generator models parameters and perform a single iteration of gradient descent on the discriminator using the real and generated images, the we switch sides fix the discriminator and train the generator for another single iteration. Similarly in this way we train both the networks in alternating steps until the generator produces good quality images, the gradients used for back propagation are:



Pseudo code:

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Generator diminished gradient:

There will be a gradient descent problem for generator, it is always easier to distinguish the generated images at earlier training, that makes v approaches 0. i.e. -*log(1 -D(G(z))) → 0.* The gradient for generator will also vanish which makes the gradient descent optimization very slow. To improve that GAN provides alternative function to backpropagate the gradient to the generator.

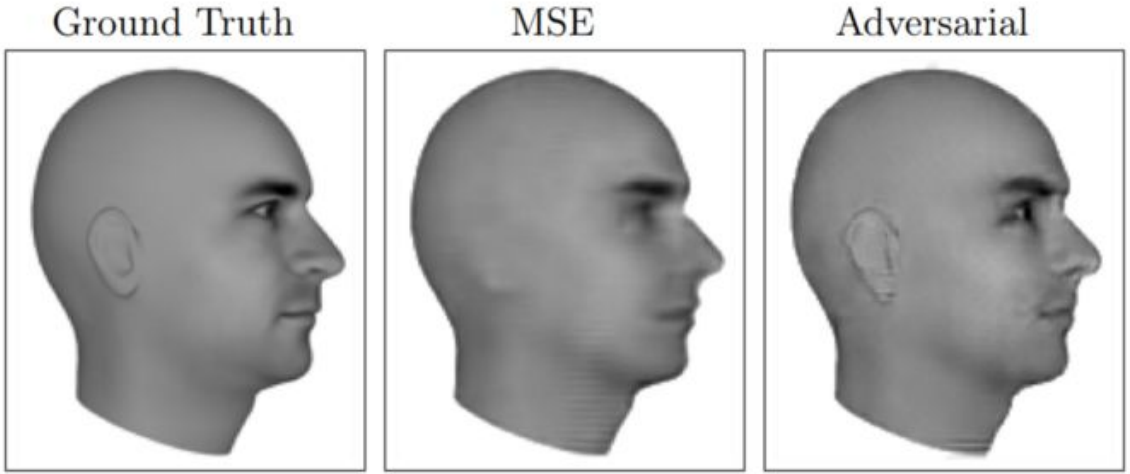
$$-\nabla_{\theta_g} \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right) \rightarrow 0 \quad \textit{change to} \quad \nabla_{\theta_g} \log\left(D\left(G\left(z^{(i)}\right)\right)\right)$$

Types Of GAN's

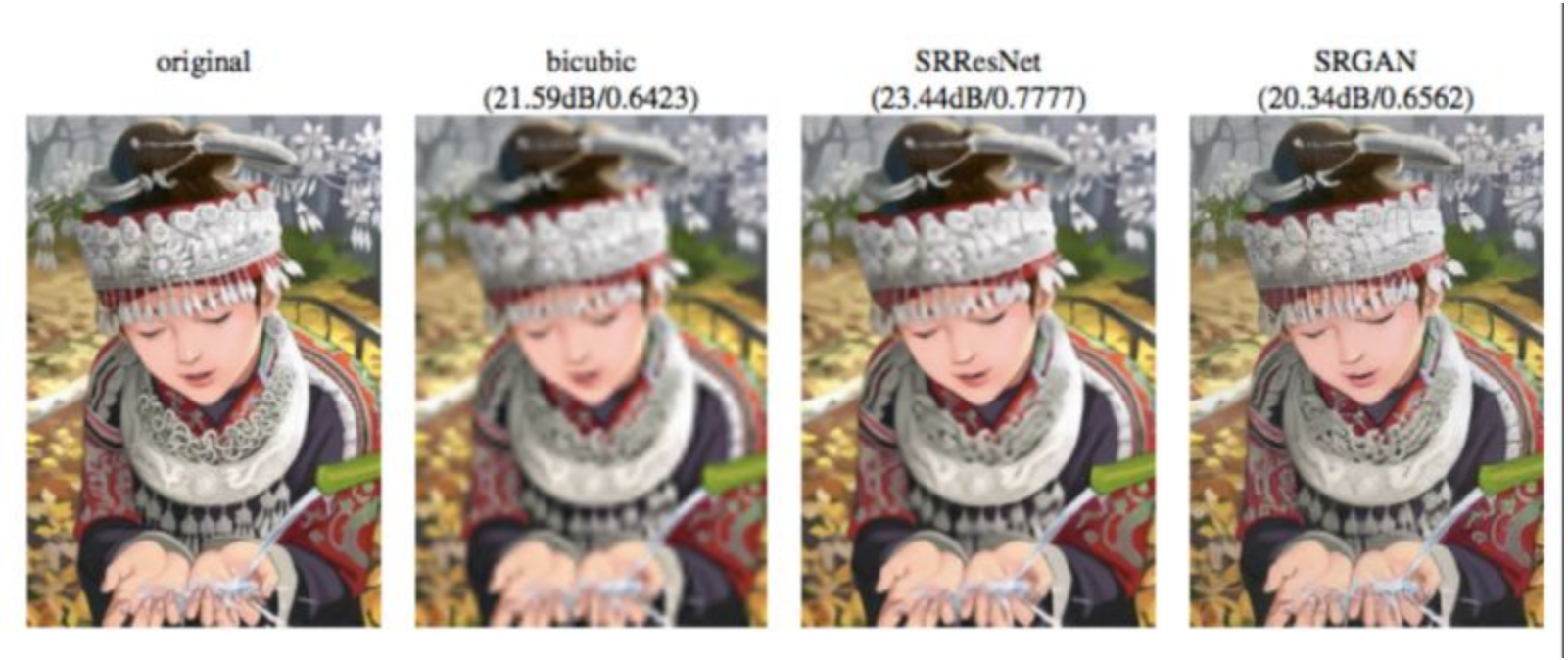
- ★ Deep Convolutional GANs (DCGANs)
- ★ Conditional GANs (cGANs)
- ★ StackGAN
- ★ InfoGANs
- ★ Wasserstein GANs(WGAN)
- ★ Discover Cross-Domain Relations with Generative Adversarial Networks(Disco GANS)

Application of GAN's

- Predicting the next frame in a video.



- Increasing Resolution of an image.



- Text-to-Image Generation.

description	with white petals and purple and white anthers.	flower are maroon in colour and have green leaves.	petals that are pink and has yellow stamen	pink in colour, and has petals that are curled upward.	yellow and white in colour, with petals that are pointed at the tips.
DC - GAN					

- Image to Image Translation

GAN's using fashion dataset

Code in spyder

References:

https://medium.com/@jonathan_hui/gan-whats-generative-adversarial-networks-and-its-application-f39ed278ef09

https://heartbeat.fritz.ai/introduction-to-generative-adversarial-networks-gans-35ef44f21193