Deep Learning

20 Generative Adversarial Network (GAN)

Dr. Konda Reddy Mopuri
Dept. of AI, IIT Hyderabad
Jan-May 2023

Generative Adversarial Networks (GAN)

1. Work by Ian Goodfellow et al. (NeurIPS 2014)

Goal

1. Sampler that draws high quality samples from p_m

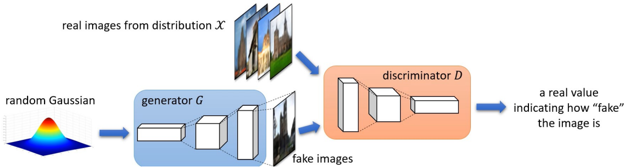
Goal

1. Sampler that draws high quality samples from p_m
2. Without computing p_x and p_m ensures closeness

Goal

1. Sampler that draws high quality samples from p_m
2. Without computing p_x and p_m ensures closeness
3. Draws samples that are similar to the train data (but not exactly them)

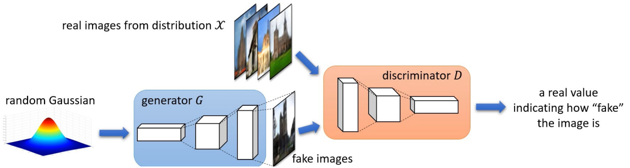
Method



Credit: Microsoft research blog

1. Introduce a latent variable (z) with a simple prior (p_z)

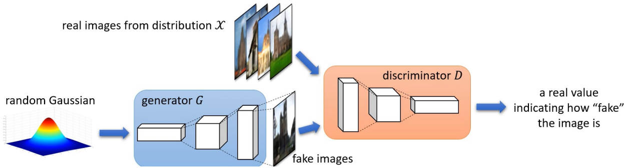
Method



Credit: Microsoft research blog

1. Introduce a latent variable (z) with a simple prior (p_z)
2. Draw $z \sim p_z$, i/p to the generator (G) $\rightarrow \hat{x} \sim p_G$

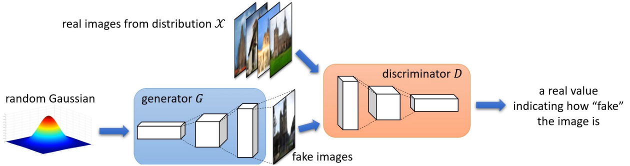
Method



Credit: Microsoft research blog

1. Introduce a latent variable (z) with a simple prior (p_z)
2. Draw $z \sim p_z$, i/p to the generator (G) $\rightarrow \hat{x} \sim p_G$
3. Machinery to ensure $p_G \approx p_{\text{data}}$

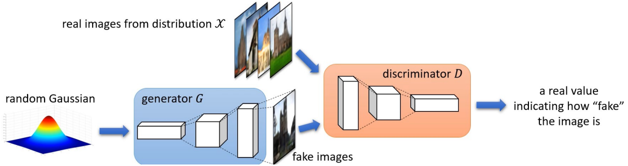
$p_G \approx p_{\textbf{data}}$



Credit: Microsoft research blog

1. Employ a classifier to differentiate between **real** samples $x \sim p_{\text{data}}$ (label 1) and **generated**(fake) ones $\hat{x} \sim p_G$ (label 0)

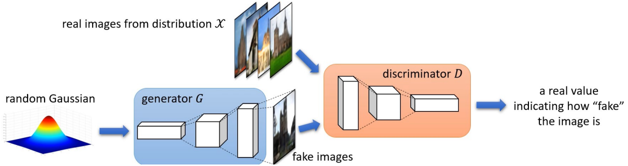
$p_G \approx p_{\textbf{data}}$



Credit: Microsoft research blog

1. Employ a classifier to differentiate between **real** samples $x \sim p_{\text{data}}$ (label 1) and **generated**(fake) ones $\hat{x} \sim p_G$ (label 0)
2. Referred to as the **Discriminator (D)**

$p_G \approx p_{\text{data}}$



Credit: Microsoft research blog

1. Employ a classifier to differentiate between **real** samples $x \sim p_{\text{data}}$ (label 1) and **generated**(fake) ones $\hat{x} \sim p_G$ (label 0)
2. Referred to as the **Discriminator (D)**
3. Train the G such that D misclassifies generated samples $\hat{x}$ into class 1 (can't differentiate b/w $x \sim p_{\text{data}}$ and $\hat{x} \sim p_G$)

Training Objective

$$\text{min}_G \ \text{max}_D \left(\mathbb{E}_{x \sim p_{\text{data}}}[logD(x)] + \mathbb{E}_{z \sim p_z}[log(1 - D(G(z)))] \right)$$

1. minmax optimization (or, zero-sum game)

Training Objective

$$\min_G \max_D \left(\mathbb{E}_{x \sim p_{\text{data}}}[log D(x)] + \mathbb{E}_{z \sim p_z}[log(1 - D(G(z)))] \right)$$

1. minmax optimization (or, zero-sum game)
2. With a sigmoid o/p neuron, $D(\cdot) \rightarrow$ probability that the i/p is real

Training Objective

$$\min_G \max_D \left(\mathbb{E}_{x \sim p_{\text{data}}}[log D(x)] + \mathbb{E}_{z \sim p_z}[log(1 - D(G(z)))] \right)$$

1. minmax optimization (or, zero-sum game)
2. With a sigmoid o/p neuron, $D(\cdot) \rightarrow$ probability that the i/p is real
3. Expectation in practice is average over a batch of samples

Training Strategy

1. Natural idea is to go for training D first and then to train G

Training Strategy

1. Natural idea is to go for training D first and then to train G
2. Issue here would be poor gradients for training G.

Training Strategy

1. Natural idea is to go for training D first and then to train G
2. Issue here would be poor gradients for training G.
3. $\min_G \left(\mathbb{E}_{z \sim p_z}[log(1 - D(G(z)))] \right)$

Training Strategy

1. Natural idea is to go for training D first and then to train G
2. Issue here would be poor gradients for training G.
3. $\min_G \left(\mathbb{E}_{z \sim p_z}[log(1 - D(G(z)))] \right)$
4. For an x generated by G, $\frac{\partial \log(1-\sigma(x))}{\partial x} = \frac{\sigma(x)\cdot(\sigma(x)-1)}{(1-\sigma(x))} = -\sigma(x)$

Training Strategy

1. Natural idea is to go for training D first and then to train G
2. Issue here would be poor gradients for training G.
3. $\min_G \left(\mathbb{E}_{z \sim p_z}[log(1 - D(G(z)))] \right)$
4. For an x generated by G, $\frac{\partial \log(1-\sigma(x))}{\partial x} = \frac{\sigma(x) \cdot (\sigma(x)-1)}{(1-\sigma(x))} = -\sigma(x)$
5. Which would be ≈ 0 for a confident $D \rightarrow$ (no gradients to train G!)

Training Strategy

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.

Idea of convergence

1. Adversarial components → nontrivial convergence for the training

Idea of convergence

1. Adversarial components $\rightarrow$ nontrivial convergence for the training
2. In other words, objective is not to push the loss/objective towards 0

Optimality

$$\min_G \max_D \left(\mathbb{E}_{x \sim p_{\text{data}}}[log D(x)] + \mathbb{E}_{z \sim p_z}[log(1 - D(G(z)))] \right)$$

$$\rightarrow \min_G \max_D \int_x \left(p_{\text{data}}(x) \cdot log D(x) + p_G(x) \cdot log(1 - D(x)) \right) dx$$

$$\rightarrow \min_G \int_x \max_D \left(p_{\text{data}}(x) \cdot log D(x) + p_G(x) \cdot log(1 - D(x)) \right) dx$$

let $y = D(x)$, $a = p_{\text{data}}$, and $b = p_G$

$\rightarrow f(y) = a \cdot \log y + b \cdot \log(1 - y)$

f exhibits local maximum at $y = \frac{a}{a+b}$

Optimal discriminator $D_G^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + P_G(x)}$

Optimality

$$\min_G \int_X \left(p_{\text{data}}(x) \cdot log D_G^*(x) + p_G(x) \cdot log(1 - D_G^*(x)) \right) dx$$

$$\min_G \int_X \left(p_{\text{data}}(x) \cdot \left[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + P_G(x)} \right] + p_G(x) \cdot log(1 - \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + P_G(x)}) \right) dx$$

$$\min_G \int_X \left(p_{\text{data}}(x) \cdot \left[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + P_G(x)} \right] + p_G(x) \cdot log(\frac{p_G(x)}{p_{\text{data}}(x) + P_G(x)}) \right) dx$$

$$\min_G \left(\mathbb{E}_{x \sim p_{\text{data}}} \left[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + P_G(x)} \right] + \mathbb{E}_{x \sim p_G} \cdot log(\frac{p_{\mathsf{G}}(x)}{p_{\text{data}}(x) + P_G(x)}) \right)$$

Optimality

$$\min_G \left(\mathbb{E}_{x\sim p_{\text{data}}} \left[\log \frac{2 * p_{\text{data}}(x)}{2 * (p_{\text{data}}(x) + P_G(x))} \right] + \mathbb{E}_{x\sim p_G} \cdot log(\frac{2 * p_G(x)}{2 * (p_{\text{data}}(x) + P_G(x)})) \right)$$

$$\min_G \left(\mathbb{E}_{x\sim p_{\text{data}}} \left[\log \frac{2 * p_{\text{data}}(x)}{(p_{\text{data}}(x) + P_G(x))} \right] + \mathbb{E}_{x\sim p_G} \cdot log(\frac{2 * p_G(x)}{(p_{\text{data}}(x) + P_G(x)}) - \log 4) \right)$$

$$\min_G \left(\mathbf{KL}(\mathbf{p}_{\text{data}}(\mathbf{x}), \frac{\mathbf{p}_{\text{data}}(\mathbf{x}) + \mathbf{P_G}(\mathbf{x})}{\mathbf{2}}) + \mathbf{KL}(\mathbf{p}_{\mathsf{G}}(\mathbf{x}), \frac{(\mathbf{p}_{\text{data}}(\mathbf{x}) + \mathbf{P_G}(\mathbf{x})}{\mathbf{2}}) - \log \mathbf{4}) \right)$$

$$\min_G \left(2 * \mathbf{JSD}(\mathbf{p}_{\text{data}}, \mathbf{p_G}) - \log \mathbf{4} \right)$$

$\rightarrow$ minimized when $p_{\text{data}} = p_G$

Optimality

1. $D_G^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x)+p_G(x)}$ (Optimal Discriminator for any G)

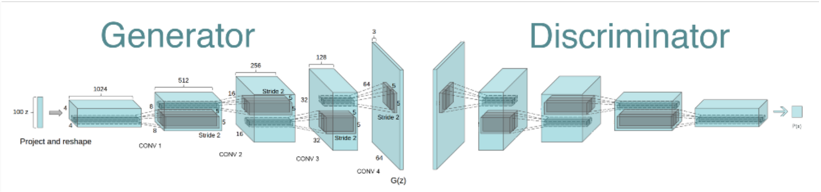
Optimality

1. $D_G^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x)+p_G(x)}$ (Optimal Discriminator for any G)
2. $p_{\text{data}} = p_G$ (Optimal Generator for any D)

Optimality

1. $D_G^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x)+p_G(x)}$ (Optimal Discriminator for any G)
2. $p_{\text{data}} = p_G$ (Optimal Generator for any D)
3. $D_G^*(x) = \frac{1}{2}$

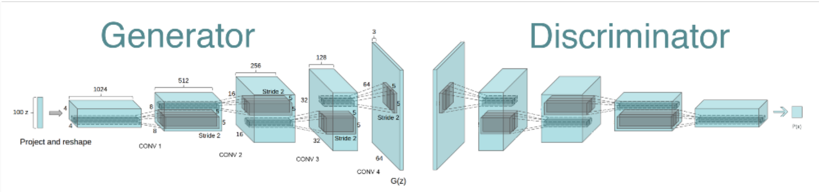
Deep Convolutional GAN (DC-GAN)



Radford et al. ICLR 2016

1. Combined the developments of CNNs with the generative modeling

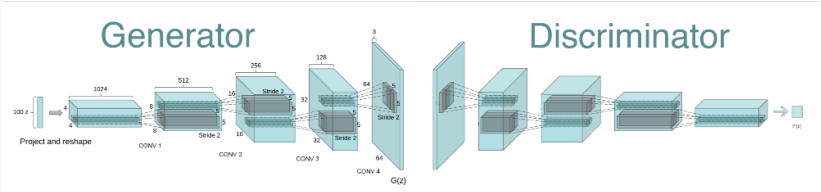
Deep Convolutional GAN (DC-GAN)



Radford et al. ICLR 2016

1. Combined the developments of CNNs with the generative modeling
2. Demonstrated some of the best practices for stable training of deep GAN architectures

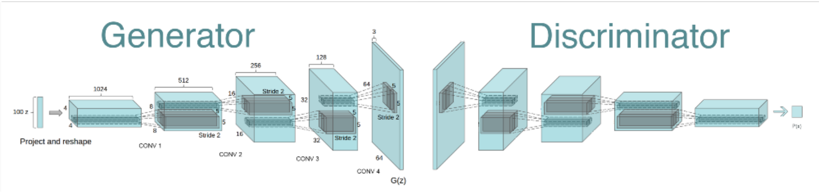
Deep Convolutional GAN (DC-GAN)



Radford et al. ICLR 2016

1. Strided convolution in place of spatial pooling (learn spatial downsampling)

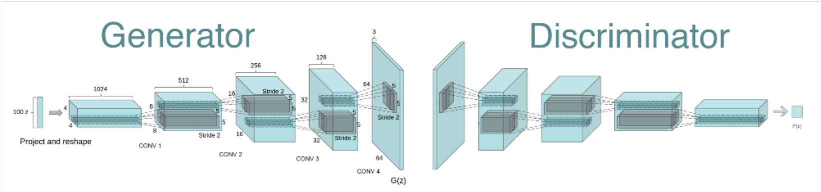
Deep Convolutional GAN (DC-GAN)



Radford et al. ICLR 2016

1. Strided convolution in place of spatial pooling (learn spatial downsampling)
2. No dense layers

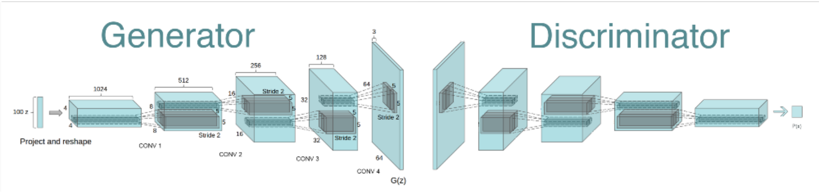
Deep Convolutional GAN (DC-GAN)



Radford et al. ICLR 2016

1. Strided convolution in place of spatial pooling (learn spatial downsampling)
2. No dense layers
3. Batchnorm in G and D

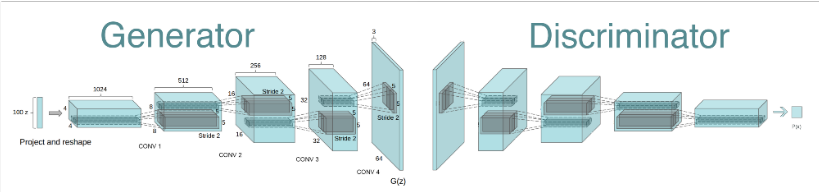
Deep Convolutional GAN (DC-GAN)



Radford et al. ICLR 2016

1. Strided convolution in place of spatial pooling (learn spatial downsampling)
2. No dense layers
3. Batchnorm in G and D
4. ReLU (tanh for the o/p layer) for G and Leaky-ReLU (sigmoid for the o/p layer) for D

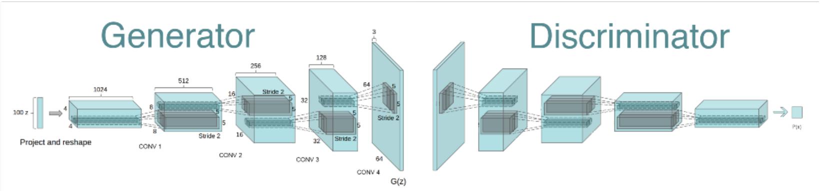
Deep Convolutional GAN (DC-GAN)



Radford et al. ICLR 2016

1. Smooth interpolation in the latent space and Vector arithmetic

Deep Convolutional GAN (DC-GAN)

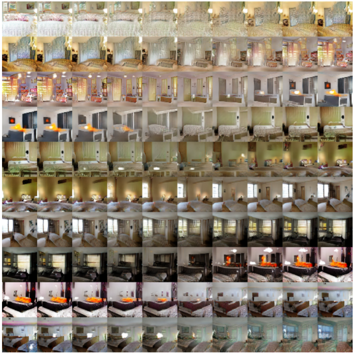


Radford et al. ICLR 2016

1. Smooth interpolation in the latent space and Vector arithmetic
2. Unsupervised feature learning (via the Discriminator)

Moving in the latent space

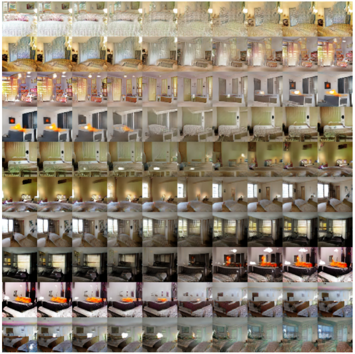
1. Interpolate between two points in the latent space and visualize



Radford et al. ICLR 2016

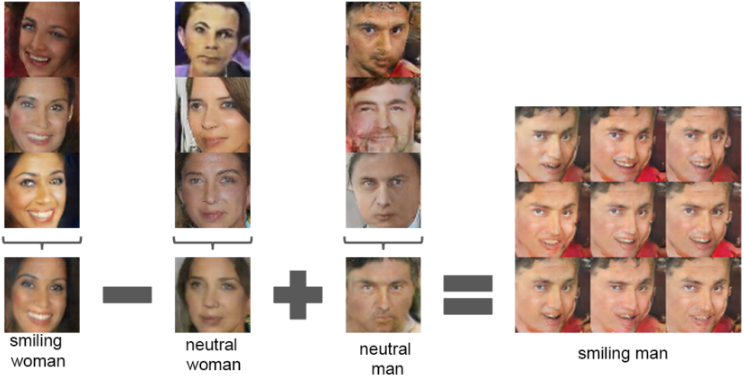
Moving in the latent space

1. Interpolate between two points in the latent space and visualize
2. Smooth transition in the generated image is a sign of good model



Radford et al. ICLR 2016

Vector arithmetic



Radford et al. ICLR 2016

Pose Transformation

Radford et al. ICLR 2016

Representation learning

Table 1: CIFAR-10 classification results using our pre-trained model. Our DCGAN is not pre-trained on CIFAR-10, but on Imagenet-1k, and the features are used to classify CIFAR-10 images.

Model	Accuracy	Accuracy (400 per class)	max # of features units
1 Layer K-means	80.6%	63.7% (±0.7%)	4800
3 Layer K-means Learned RF	82.0%	70.7% (±0.7%)	3200
View Invariant K-means	81.9%	72.6% (±0.7%)	6400
Exemplar CNN	84.3%	77.4% (±0.2%)	1024
DCGAN (ours) + L2-SVM	82.8%	73.8% (±0.4%)	512

Radford et al. ICLR 2016

Evaluating GANs

భారతీయ సాంకేతిక విజ్ఞాన సంస్థ హైదరాబాద్
भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

1. Open research problem

Evaluating GANs

భారతీయ సాంకేతిక విజ్ఞాన సంస్థ హైదరాబాద్
भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

1. Open research problem
2. Humans judgement!

Evaluating GANs

1. Open research problem
2. Humans judgement!
3. In case of images
   - **Recognizable objects**: accurate and high-confidence predictions by a classifier
   - **Semantic diversity**: samples should be drawn evenly from all categories of train data

Inception Score (IS)

1. Consider the pretrained Inception classifier $\rightarrow p(y/x)$

Inception Score (IS)

1. Consider the pretrained Inception classifier $\rightarrow p(y/x)$
2. label distribution of the generated samples $\rightarrow p(y)$

Inception Score (IS)

1. Consider the pretrained Inception classifier $\rightarrow p(y/x)$
2. label distribution of the generated samples $\rightarrow p(y)$
3. Desired: low entropy for $p(y/x)$ (distinctly recognizable) and high entropy for $p(y)$ (semantic diversity)

Inception Score (IS)

1. Consider the pretrained Inception classifier $\rightarrow p(y/x)$
2. label distribution of the generated samples $\rightarrow p(y)$
3. Desired: low entropy for $p(y/x)$ (distinctly recognizable) and high entropy for $p(y)$ (semantic diversity)
4. Inception score (IS) $= \exp\left(H(y) - H(y/x)\right)$

Inception Score (IS)

1. Consider the pretrained Inception classifier $\rightarrow p(y/x)$
2. label distribution of the generated samples $\rightarrow p(y)$
3. Desired: low entropy for $p(y/x)$ (distinctly recognizable) and high entropy for $p(y)$ (semantic diversity)
4. Inception score (IS) $= \exp\left(H(y) - H(y/x)\right)$
5. Higher is better

Inception Score (IS)

1. Based completely on the generated data (real data is not considered)

Frechet Inception Distance (FID)

1. Attempts to find the distance b/w p_{data} and p_G

Frechet Inception Distance (FID)

1. Attempts to find the distance b/w p_{data} and p_G
2. In the feature space (inception model, pool3 layer)

Frechet Inception Distance (FID)

1. Attempts to find the distance b/w p_{data} and p_G
2. In the feature space (inception model, pool3 layer)
3. Frechet distance between two multi-variate Gaussians

$$d^2((m, C), (m_d, C_d)) = |m - m_d|^2 + Tr(C + C_d - 2(C \cdot C_d)^2)$$

(m_d, C_d are mean and covariance of the original data)
(m, C are mean and covariance of the generated data)

Frechet Inception Distance (FID)

1. Attempts to find the distance b/w p_{data} and p_G
2. In the feature space (inception model, pool3 layer)
3. Frechet distance between two multi-variate Gaussians

   $$d^2((m, C), (m_d, C_d)) = |m - m_d|^2 + Tr(C + C_d - 2(C \cdot C_d)^2)$$

   (m_d, C_d are mean and covariance of the original data)
   (m, C are mean and covariance of the generated data)
4. lower is better