Deep Learning for Computer Vision

Generative Adversarial Networks

Vineeth N Balasubramanian

Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad

Review: Question

Why does using KL-divergence in finding the generative model simplify to maximum likelihood estimation?

- Recall our problem:

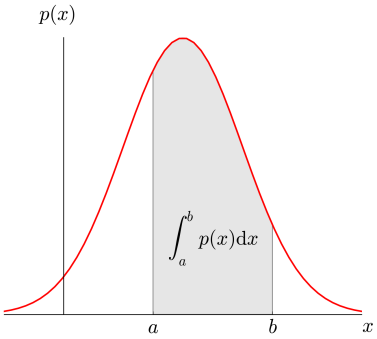
$$\theta^* = \arg\min_{\theta \in M} \text{dist}(p_\theta, p_D)$$

- If distance is KL-divergence:

$$\begin{aligned}
\theta^* &= \arg\min_{\theta \in M} D_{KL}(p_D || p_\theta) \\
&= \arg\min_{\theta \in M} \mathbb{E}_{\mathbf{x} \sim p_D} \left(\log p_D(\mathbf{x}) - \log p_\theta(\mathbf{x}) \right) \\
&= \arg\min_{\theta \in M} \mathbb{E}_{\mathbf{x} \sim p_D} \left(- \log p_\theta(\mathbf{x}) \right) \qquad (\theta \text{ parameters are independent of } \log p_D(\mathbf{x}))
\end{aligned}$$

This is maximum likelihood estimation (minimizing negative log likelihood)!

Recap



- **Generative Models**: Learn probability density function $p(x)$ over images x
- $p(x)$ assigns positive number to each possible image x

  $\implies$ how likely is image x under distribution $p(x)$
- **Applications:**
  - Sample to generate new data
  - Likelihood estimation(Outlier detection)
  - Feature learning (unsupervised)

Density Estimation

Explicit Density Estimation

- Write explicit function $p(x) = f(x, \theta)$
  Input: Image x
  Output: Likelihood value for image
  Parameter: Weights θ
- Assign explicit likelihood to images
- Can enable outlier detection, but compromise generated image quality/sampling speed

Implicit Density Estimation

- Give up estimating explicit form of $p(x)$
- Aim only to sample images from model; no explicit likelihood assignment
- High quality samples generated/fast sampling speed

Generative Adversarial Networks (GANs)[1]

Goal:

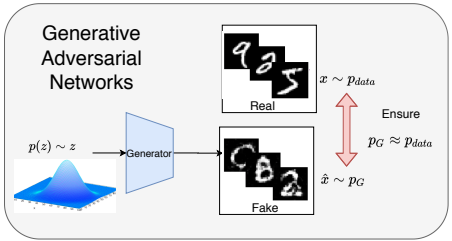
- Build good sampler that allows to draw high quality samples from $p_{model}(x)$
- Not explicitly compute form of $p(x)$ but ensures $p_{model}(x)$ approximates/is close to $p_{data}(x)$
- Output samples similar but not exactly same as train data; how?

[1] Goodfellow et al, Generative Adversarial Nets, NeurIPS 2014

Generative Adversarial Networks (GANs)[1]

Goal:

- Build good sampler that allows to draw high quality samples from $p_{model}(x)$
- Not explicitly compute form of $p(x)$ but ensures $p_{model}(x)$ approximates/is close to $p_{data}(x)$
- Output samples similar but not exactly same as train data; how?



Method:

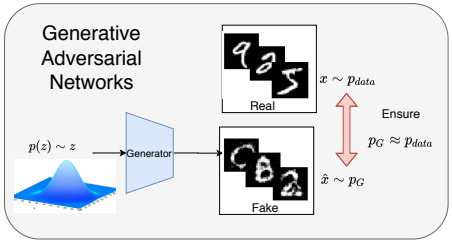
- Introduce latent variable z with simple prior $p(z)$ (e.g. Gaussian)

[1] Goodfellow et al, Generative Adversarial Nets, NeurIPS 2014

Generative Adversarial Networks (GANs)[1]

Goal:

- Build good sampler that allows to draw high quality samples from $p_{model}(x)$
- Not explicitly compute form of $p(x)$ but ensures $p_{model}(x)$ approximates/is close to $p_{data}(x)$
- Output samples similar but not exactly same as train data; how?



Method:

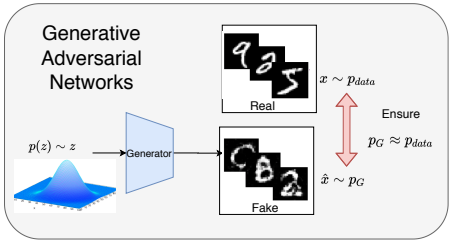
- Introduce latent variable z with simple prior $p(z)$ (e.g. Gaussian)
- Sample $z \sim p(z)$, pass it through Generator $\hat{x} = G(z)$; where $\hat{x} \sim p_G$

[1] Goodfellow et al, Generative Adversarial Nets, NeurIPS 2014

Generative Adversarial Networks (GANs)[1]

Goal:

- Build good sampler that allows to draw high quality samples from $p_{model}(x)$
- Not explicitly compute form of $p(x)$ but ensures $p_{model}(x)$ approximates/is close to $p_{data}(x)$
- Output samples similar but not exactly same as train data; how?

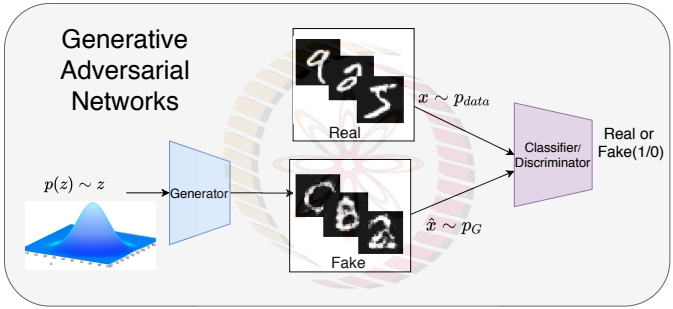


Method:

- Introduce latent variable z with simple prior $p(z)$ (e.g. Gaussian)
- Sample $z \sim p(z)$, pass it through Generator $\hat{x} = G(z)$; where $\hat{x} \sim p_G$
- Introduce mechanism to ensure $p_G \approx p_{data}$

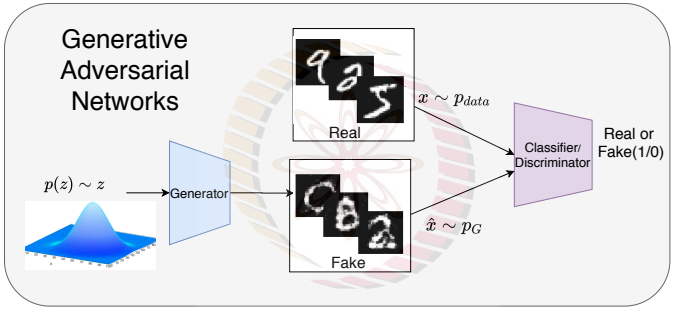
[1] Goodfellow et al, Generative Adversarial Nets, NeurIPS 2014

GANs: Ensuring $p_G \approx p_{data}$



- **Idea**: Use a classifier (called **discriminator**) that differentiates between real samples $x \sim p_{data}$ (class 1) and $\hat{x} \sim p_G$ (class 0)

GANs: Ensuring $p_G \approx p_{data}$



- **Idea**: Use a classifier (called **discriminator**) that differentiates between real samples $x \sim p_{data}$ (class 1) and $\hat{x} \sim p_G$ (class 0)
- Train generator G such that discriminator misclassifies generated sample $\hat{x}$ into class 1
  $\implies$ can no more differentiate between $x \sim p_{data}$ and $\hat{x} \sim p_G$

GANs: How to train?

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

(**zero-sum game**)

(Assume sigmoid activation in discriminator $\implies$ $D(.) =$ probability that input sample is real)

GANs: How to train?

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

(**zero-sum game**)

(Assume sigmoid activation in discriminator $\implies$ $D(.)$ = probability that input sample is real)

$$O_1 = \max_D \mathbb{E}_{x \sim p_{data}}[\log D(x)]$$

Train discriminator, D, such that if sample belongs to p_{data}, maximize the log probability of it being a real sample

GANs: How to train?

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

(**zero-sum game**)

(Assume sigmoid activation in discriminator $\implies D(.) =$ probability that input sample is real)

$$O_1 = \max_D \mathbb{E}_{x\sim p_{data}}[\log D(x)]$$

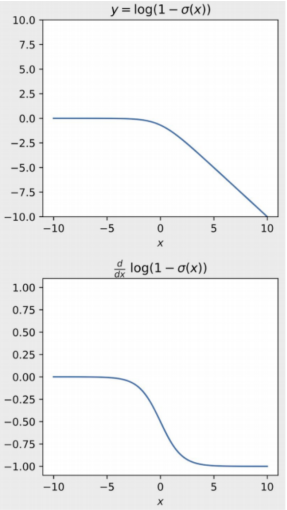
Train discriminator, D, such that if sample belongs to p_{data}, maximize the log probability of it being a real sample

$$O_2 = \min_G \mathbb{E}_{z\sim p_z}[\log(1 - D(G(z))]$$

Train Generator, G such that if sample belongs to p_G(i.e $G(z)$), maximize the log probability of it being a real sample

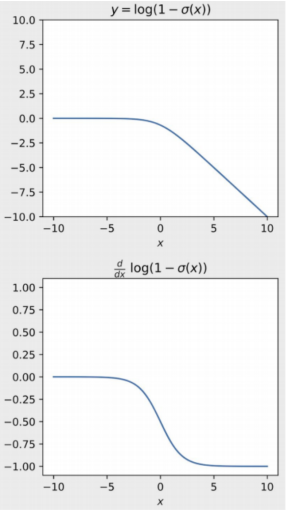
Note: the expectation in the objective function simply implies that losses are averaged over a batch of samples

GANs: Training Strategy



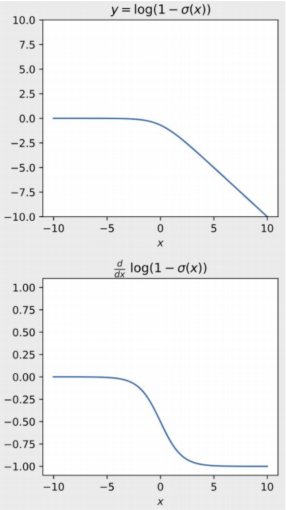
- **Idea:** First train D completely to optimize O_1; then train G to optimize O_2

GANs: Training Strategy



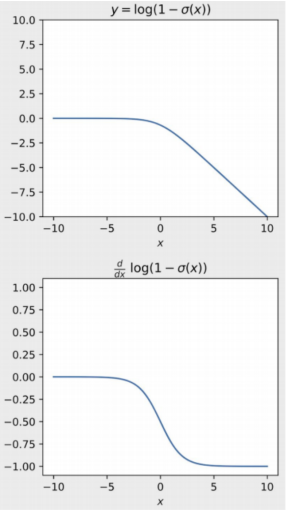
- **Idea:** First train D completely to optimize O_1; then train G to optimize O_2
- **Problem:** If D is initially very confident that samples from G are fake, when x is obtained from G: $D(x)$ or $\sigma(x) = 0 \implies \log(1 - \sigma(x)) = 0$

GANs: Training Strategy



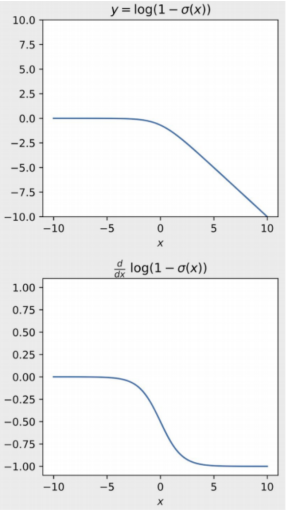
- **Idea:** First train D completely to optimize O_1; then train G to optimize O_2
- **Problem:** If D is initially very confident that samples from G are fake, when x is obtained from G:
  $D(x)$ or $\sigma(x) = 0 \implies \log(1 - \sigma(x)) = 0$
  $\implies \frac{\partial(\log(1-\sigma(x))}{\partial x} = 0$ (as shown in graph)

GANs: Training Strategy



- **Idea:** First train D completely to optimize O_1; then train G to optimize O_2
- **Problem:** If D is initially very confident that samples from G are fake, when x is obtained from G:
  $D(x)$ or $\sigma(x) = 0 \implies \log(1 - \sigma(x)) = 0$
  $\implies \frac{\partial(\log(1-\sigma(x))}{\partial x} = 0$ (as shown in graph)
  $\implies$ G gets no gradients to train!

GANs: Training Strategy



- **Idea:** First train D completely to optimize O_1; then train G to optimize O_2
- **Problem:** If D is initially very confident that samples from G are fake, when x is obtained from G:
  $D(x)$ or $\sigma(x) = 0 \implies \log(1 - \sigma(x)) = 0$
  $\implies \frac{\partial(\log(1-\sigma(x)))}{\partial x} = 0$ (as shown in graph)
  $\implies$ G gets no gradients to train!
- **Solution:** Alternate between Discriminator objective O_1 and Generator objective O_2

GANs: Algorithm[2]

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.

[2] Goodfellow et al, Generative Adversarial Nets, NeurIPS 2014

GANs: Evaluating Optimality

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

GANs: Evaluating Optimality

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

$$\implies \min_G \max_D \int_x \big(p_{data}(x) \log D(x) + p_G(x) \log(1 - D(x)\big) dx$$

(change of variable, expanding expectation)

GANs: Evaluating Optimality

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

$$\implies \min_G \max_D \int_x \big(p_{data}(x) \log D(x) + p_G(x) \log(1 - D(x)\big) dx$$

(change of variable, expanding expectation)

$$\implies \min_G \int_x \max_D \big(p_{data}(x) \log D(x) + p_G(x) \log(1 - D(x)\big) dx$$ **(taking $\max$ inside)**

GANs: Evaluating Optimality

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

$$\implies \min_G \max_D \int_x \big(p_{data}(x)\log D(x) + p_G(x)\log(1 - D(x)\big)dx$$

(change of variable, expanding expectation)

$$\implies \min_G \int_x \max_D \big(p_{data}(x)\log D(x) + p_G(x)\log(1 - D(x)\big)dx$$ **(taking $\max$ inside)**

Let $y = D(x); a = p_{data}; b = p_G$

GANs: Evaluating Optimality

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

$$\implies \min_G \max_D \int_x \big(p_{data}(x)\log D(x) + p_G(x)\log(1 - D(x)\big)dx$$

(change of variable, expanding expectation)

$$\implies \min_G \int_x \max_D \big(p_{data}(x)\log D(x) + p_G(x)\log(1 - D(x)\big)dx$$ **(taking** $\max$ **inside)**

Let $y = D(x); a = p_{data}; b = p_G$

Then, $f(y) = a\log y + b\log(1 - y)$

GANs: Evaluating Optimality

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

$$\implies \min_G \max_D \int_x \big(p_{data}(x)\log D(x) + p_G(x)\log(1 - D(x)\big)dx$$

(change of variable, expanding expectation)

$$\implies \min_G \int_x \max_D \big(p_{data}(x)\log D(x) + p_G(x)\log(1 - D(x)\big)dx$$ **(taking max inside)**

Let $y = D(x); a = p_{data}; b = p_G$

Then, $f(y) = a\log y + b\log(1 - y)$

$$f'(y) = \frac{a}{y} - \frac{b}{1-y}; f'(y) = 0 \implies y = \frac{a}{a+b}$$ **(local max)**

GANs: Evaluating Optimality

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

$$\implies \min_G \max_D \int_x \big(p_{data}(x)\log D(x) + p_G(x)\log(1 - D(x)\big)dx$$

(change of variable, expanding expectation)

$$\implies \min_G \int_x \max_D \big(p_{data}(x)\log D(x) + p_G(x)\log(1 - D(x)\big)dx$$ **(taking** $\max$ **inside)**

Let $y = D(x); a = p_{data}; b = p_G$

Then, $f(y) = a\log y + b\log(1 - y)$

$$f'(y) = \frac{a}{y} - \frac{b}{1-y}; f'(y) = 0 \implies y = \frac{a}{a+b}$$ **(local max)**

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

GANs: Evaluating Optimality

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

GANs: Evaluating Optimality

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

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

GANs: Evaluating Optimality

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

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

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

GANs: Evaluating Optimality

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

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

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

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

GANs: Evaluating Optimality

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

GANs: Evaluating Optimality

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

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

GANs: Evaluating Optimality

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

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

$$\min_G \left(\mathbf{KL}(p_{data}(x), \frac{p_{data}(x) + p_G(x)}{2}) + \mathbf{KL}(p_G(x), \frac{p_{data}(x) + p_G(x)}{2}) - \log 4 \right)$$

GANs: Evaluating Optimality

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

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

$$\min_G \left(\mathbf{KL}(p_{data}(x), \frac{p_{data}(x) + p_G(x)}{2}) + \mathbf{KL}(p_G(x), \frac{p_{data}(x) + p_G(x)}{2}) - \log 4 \right)$$

Kullback–Leibler Divergence

$\mathbf{KL(p,q)} = \mathbb{E}_{x \sim p}[\frac{\log p(x)}{\log q(x)}]$

Jenson-Shannon Divergence

$\mathbf{JSD}(p_{data}, p_G) = \frac{KL(p_{data}, \frac{p_{data}+p_G}{2})}{2} + \frac{KL(p_G, \frac{p_{data}+p_G}{2})}{2}$

$\min_G(2 * JSD(p_{data}, p_G) - \log 4) \implies$ expression is minimized when $p_{data} = p_G$

(Recall $JSD \geq 0$ by definition)

Optimality of GANs: Conclusions

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

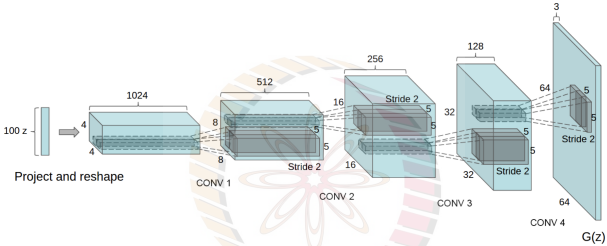
Optimality of GANs: Conclusions

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

Optimality of GANs: Conclusions

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

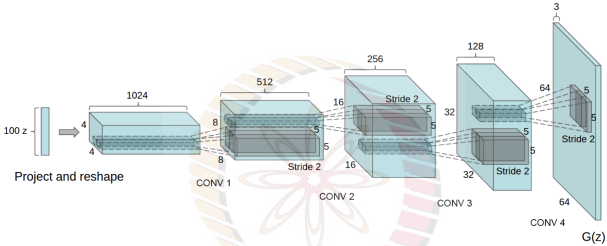
Deep Convolution GAN (DCGAN)[3]



- Bridge the gap between success of CNNs for supervised learning and unsupervised generative models

[3]Radford et al, Unsupervised Representation learning with deep convolutional GANs, NeurIPS 2016

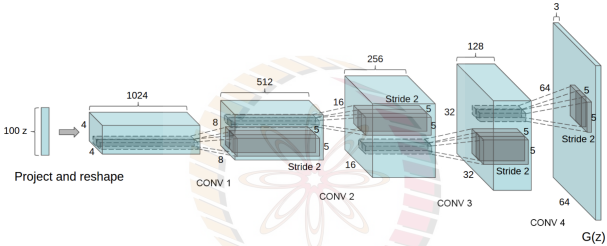
Deep Convolution GAN (DCGAN)[3]



- Bridge the gap between success of CNNs for supervised learning and unsupervised generative models
- Best practices to enable stable training of GANs with deep architectures

[3]Radford et al, Unsupervised Representation learning with deep convolutional GANs, NeurIPS 2016

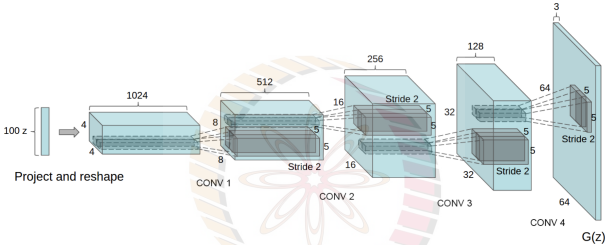
Deep Convolution GAN (DCGAN)[3]



- Bridge the gap between success of CNNs for supervised learning and unsupervised generative models
- Best practices to enable stable training of GANs with deep architectures
- Smooth high-dimensional Interpolations; Vector arithmetic

[3]Radford et al, Unsupervised Representation learning with deep convolutional GANs, NeurIPS 2016

Deep Convolution GAN (DCGAN)[3]

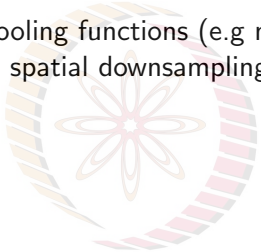


- Bridge the gap between success of CNNs for supervised learning and unsupervised generative models
- Best practices to enable stable training of GANs with deep architectures
- Smooth high-dimensional Interpolations; Vector arithmetic
- Demonstrate unsupervised feature learning capabilities of GANs and its applications **(Representation Learning)**

[3] Radford et al, Unsupervised Representation learning with deep convolutional GANs, NeurIPS 2016

DCGAN: Training Practices for Deeper GANs[4]

- Replaces deterministic spatial pooling functions (e.g maxpooling) with **strided convolutions** $\rightarrow$ allows to learn spatial downsampling



[4] Radford et al, Unsupervised Representation Learning with Deep Convolutional GANs, NeurIPS 2016

DCGAN: Training Practices for Deeper GANs[4]

- Replaces deterministic spatial pooling functions (e.g maxpooling) with **strided convolutions** $\rightarrow$ allows to learn spatial downsampling
- **Remove fully connected hidden layers** for deeper architectures

[4] Radford et al, Unsupervised Representation Learning with Deep Convolutional GANs, NeurIPS 2016

DCGAN: Training Practices for Deeper GANs[4]

- Replaces deterministic spatial pooling functions (e.g maxpooling) with **strided convolutions** $\rightarrow$ allows to learn spatial downsampling
- **Remove fully connected hidden layers** for deeper architectures
- **Batchnorm in G and D** $\rightarrow$ prevent generator collapse/enable gradient flow in deeper architectures; not applied at output of G and input of D

[4] Radford et al, Unsupervised Representation Learning with Deep Convolutional GANs, NeurIPS 2016

DCGAN: Training Practices for Deeper GANs[4]

- Replaces deterministic spatial pooling functions (e.g maxpooling) with **strided convolutions** $\rightarrow$ allows to learn spatial downsampling
- **Remove fully connected hidden layers** for deeper architectures
- **Batchnorm in G and D** $\rightarrow$ prevent generator collapse/enable gradient flow in deeper architectures; not applied at output of G and input of D
- **Non-Linearity:** `ReLU` for generator, `Leaky-ReLU (0.2)` for discriminator

[4] Radford et al, Unsupervised Representation Learning with Deep Convolutional GANs, NeurIPS 2016

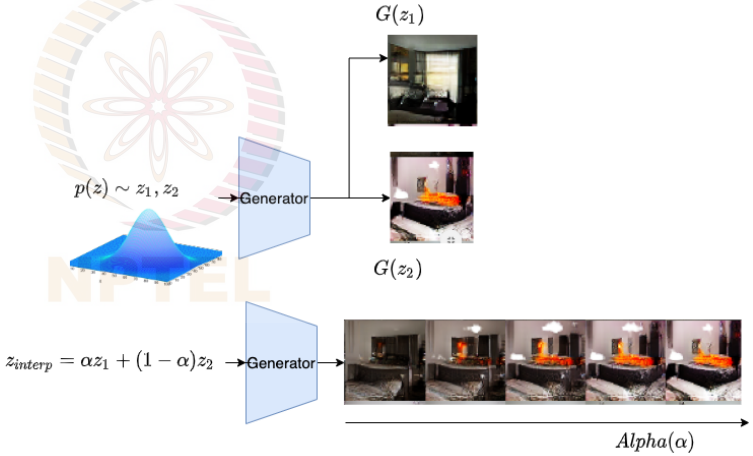
DCGAN: Training Practices for Deeper GANs[4]

- Replaces deterministic spatial pooling functions (e.g maxpooling) with **strided convolutions** $\rightarrow$ allows to learn spatial downsampling
- **Remove fully connected hidden layers** for deeper architectures
- **Batchnorm in G and D** $\rightarrow$ prevent generator collapse/enable gradient flow in deeper architectures; not applied at output of G and input of D
- **Non-Linearity:** `ReLU` for generator, `Leaky-ReLU (0.2)` for discriminator
- **Output Non-Linearity:** `tanh` for generator, `sigmoid` for discriminator

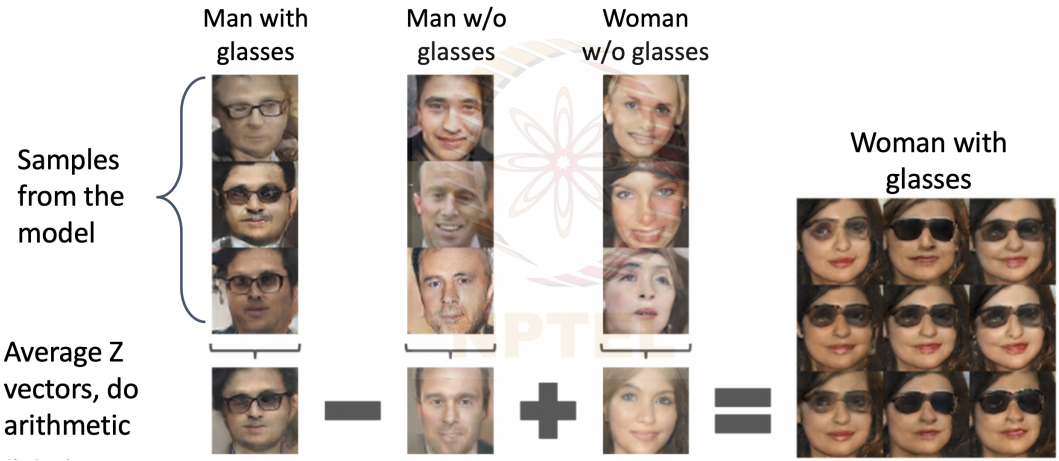
[4] Radford et al, Unsupervised Representation Learning with Deep Convolutional GANs, NeurIPS 2016

Walking the Latent Space

- We can interpolate between two points in latent space and visualize
- Smooth transition in generated image space by changing latent (perceptual) features is a sign of a good model



Vector Arithmetic in Latent Space[5]



[5] Radford et al, Unsupervised Representation Learning with Deep Convolutional GANs, NeurIPS 2016

Pose Transformation[6]

- z_{left}: averaged latent vector of faces looking left
  z_{right}: average latent vector of faces looking right
- $z_{\text{turn}} = z_{\text{right}} - z_{\text{left}}$
  $z_{\text{new}} = z + \alpha z_{\text{turn}}$
- $G(z_{\text{new}}) \implies$ Transformed image

[6] Radford et al, Unsupervised Representation Learning with Deep Convolutional GANs, NeurIPS 2016

Feature Learning[7]

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

- Train DCGAN on ImageNet-1k

[7] Radford et al, Unsupervised Representation Learning with Deep Convolutional GANs, NeurIPS 2016

Feature Learning[7]

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

- Train DCGAN on ImageNet-1k
- Use discriminator's convolution features for supervised tasks i.e. classify images from CIFAR-10 dataset

[7] Radford et al, Unsupervised Representation Learning with Deep Convolutional GANs, NeurIPS 2016

Feature Learning[7]

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

- Train DCGAN on ImageNet-1k
- Use discriminator's convolution features for supervised tasks i.e. classify images from CIFAR-10 dataset
- DCGAN never trained on CIFAR-10 $\implies$ representation power and domain robustness of learned features

[7] Radford et al, Unsupervised Representation Learning with Deep Convolutional GANs, NeurIPS 2016

Feature Learning[7]

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% ($\pm$0.7%)	4800
3 Layer K-means Learned RF	82.0%	70.7% ($\pm$0.7%)	3200
View Invariant K-means	81.9%	72.6% ($\pm$0.7%)	6400
Exemplar CNN	84.3%	77.4% ($\pm$0.2%)	1024
DCGAN (ours) + L2-SVM	82.8%	73.8% ($\pm$0.4%)	512

- Train DCGAN on ImageNet-1k
- Use discriminator's convolution features for supervised tasks i.e. classify images from CIFAR-10 dataset
- DCGAN never trained on CIFAR-10 $\implies$ representation power and domain robustness of learned features
- Competitive results when compared with other feature learning methods

[7] Radford et al, Unsupervised Representation Learning with Deep Convolutional GANs, NeurIPS 2016

GANs: How to evaluate?

- **Human judgement:** Good generator $\implies$ images with distinctly recognizable objects; Semantically diverse samples

GANs: How to evaluate?

- **Human judgement:** Good generator $\implies$ images with distinctly recognizable objects; Semantically diverse samples
  - **Recognizable objects** $\implies$ classifier predicts class of generated sample correctly with high confidence
  - **Semantic diversity** $\implies$ generated samples must be evenly from all classes in train set

GANs: How to evaluate?

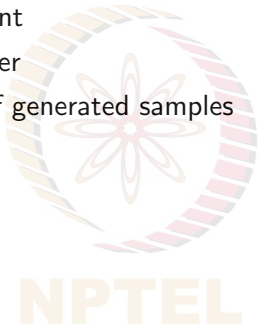
- **Human judgement:** Good generator $\implies$ images with distinctly recognizable objects; Semantically diverse samples
  - **Recognizable objects** $\implies$ classifier predicts class of generated sample correctly with high confidence
  - **Semantic diversity** $\implies$ generated samples must be evenly from all classes in train set
- **Prediction power**: How ImageNet pre-trained Inception Network V3 performs on these generated images

GANs: How to evaluate?

- **Human judgement:** Good generator $\implies$ images with distinctly recognizable objects; Semantically diverse samples
  - **Recognizable objects** $\implies$ classifier predicts class of generated sample correctly with high confidence
  - **Semantic diversity** $\implies$ generated samples must be evenly from all classes in train set
- **Prediction power**: How ImageNet pre-trained Inception Network V3 performs on these generated images
- Still an open research problem...

Inception Score[8]

- Correlates with human judgement
- $p(y|x)$: Inception model/classifier
- $p(y)$: Label/class distribution of generated samples



[8] Radford et al, Unsupervised Representation Learning with Deep Convolutional GANs, NeurIPS 2016

Inception Score[8]

- Correlates with human judgement
- $p(y|x)$: Inception model/classifier
- $p(y)$: Label/class distribution of generated samples
- **Inception Score(IS):** $= \exp(\mathbb{E}_{x\sim p_g}[D_{KL}[p(y|x)||p(y)])$

$$= \exp(\mathbb{E}_{x\sim p_g, y\sim p(y|x)}[\log(p(y|x)) - \log(p(y)])$$
$$= \exp(H(y) - H(y|x))$$

$H(y)$: entropy of generated sample class labels (**Semantic diversity = high** $H(y)$)

$H(y/x)$: entropy of class labels from classifier (**Distinctly Classifiable = low** $H(y|x)$)

[8] Radford et al, Unsupervised Representation Learning with Deep Convolutional GANs, NeurIPS 2016

Inception Score[8]

- Correlates with human judgement
- $p(y|x)$: Inception model/classifier
- $p(y)$: Label/class distribution of generated samples
- **Inception Score(IS):** $= \exp(\mathbb{E}_{x\sim p_g}[D_{KL}[p(y|x)||p(y)])$

$$= \exp(\mathbb{E}_{x\sim p_g, y\sim p(y|x)}[\log(p(y|x)) - \log(p(y)])$$
$$= \exp(H(y) - H(y|x))$$

$H(y)$: entropy of generated sample class labels (**Semantic diversity = high** $H(y)$)

$H(y/x)$: entropy of class labels from classifier (**Distinctly Classifiable = low** $H(y|x)$)

- Higher IS $\implies$ better generative model

[8] Radford et al, Unsupervised Representation Learning with Deep Convolutional GANs, NeurIPS 2016

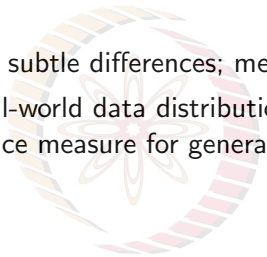
Fréchet Inception Distance (FID)[9]

- **Drawback of IS:** Statistics of real-world samples are not used to compare with statistics of synthetic samples
- FID enables us to capture more subtle differences; measures diversity better

[9] Heusel et al, GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium, NeurIPS 2017

Fréchet Inception Distance (FID)[9]

- **Drawback of IS:** Statistics of real-world samples are not used to compare with statistics of synthetic samples
- FID enables us to capture more subtle differences; measures diversity better
- **Intution:** Distance between real-world data distribution and generating model's data distribution serves as performance measure for generative models; how?



[9] Heusel et al, GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium, NeurIPS 2017

Fréchet Inception Distance (FID)[9]

- **Drawback of IS:** Statistics of real-world samples are not used to compare with statistics of synthetic samples
- FID enables us to capture more subtle differences; measures diversity better
- **Intution:** Distance between real-world data distribution and generating model's data distribution serves as performance measure for generative models; how?
- Embed image x into feature space (activations of Inception-v3 pool3 layer), and compute the Fréchet distance between two multivariate Gaussians:

$$d^2((m, C), (m_w, C_w)) = ||m - m_w||_2^2 + Tr(C + C_w - 2(C \cdot C_w)^2)$$

where m is mean, C is covariance of generated image features; m_w is mean, C_w is covariance of original image features

[9] Heusel et al, GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium, NeurIPS 2017

Fréchet Inception Distance (FID)[9]

- **Drawback of IS:** Statistics of real-world samples are not used to compare with statistics of synthetic samples
- FID enables us to capture more subtle differences; measures diversity better
- **Intution:** Distance between real-world data distribution and generating model's data distribution serves as performance measure for generative models; how?
- Embed image x into feature space (activations of Inception-v3 pool3 layer), and compute the Fréchet distance between two multivariate Gaussians:

$$d^2((m, C), (m_w, C_w)) = ||m - m_w||_2^2 + Tr(C + C_w - 2(C \cdot C_w)^2)$$

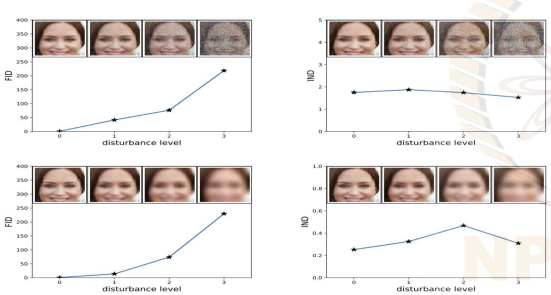
where m is mean, C is covariance of generated image features; m_w is mean, C_w is covariance of original image features

- Lower FID $\implies$ better generative model

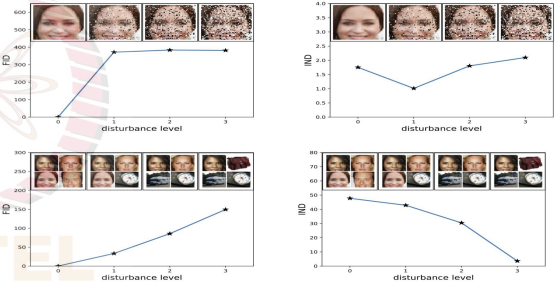
[9] Heusel et al, GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium, NeurIPS 2017

FID vs Inception Score[10]

CelebA dataset (Columns 1,3: FID Score; Columns 2,4: Inception Score)



(Top:) Gaussian noise added to images
(Bottom:) Gaussian blur added to images



(Top:) Salt-and-pepper noise added to images
(Bottom:) ImageNet crops added to images

[10] Heusel et al, GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium, NeurIPS 2017

Homework

Readings

- Dive into Deep Learning, Chapter 17
  https://d2l.ai/chapter_generative-adversarial-networks/gan.html
- Play with Generative Adversarial Networks (GANs) in your browser
  https://poloclub.github.io/ganlab/
- Deep Learning With Python - François Chollet, Section 8.5
  https://github.com/veritone/ds-transcription-capstone