Generative Networks

STAT991

Wenting Zhan

Claim

This slides are modified and integrated from:

- CS231n Stanford University
- STAT991 University of Pennsylvania (Fall 2018 Lecture 4)
- CIS700-004 University of Pennsylvania (Spring 2019 Lecture 7M & 7W)

Agenda

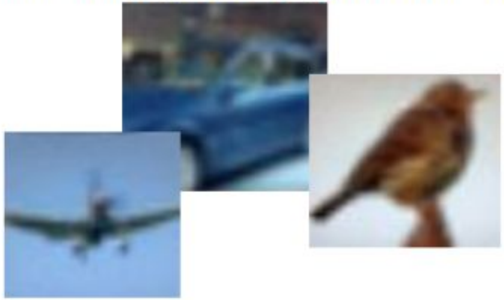
- Generative Models
- Variational Autoencoders (VAE)
- Generative Adversarial Networks (GAN)
    - Basic Mechanism: Generator and Discriminator
    - Dual Loss Function
    - Major Issues
        - Mode Collapse
        - Why are GANs so hard to train
- Types of GAN
    - Deep Convolution GAN
    - Cycle GAN
    - Etc.
- Applications of GAN

Agenda

- Generative Models
- Variational Autoencoders (VAE)
- Generative Adversarial Networks (GAN)
  - Basic Mechanism: Generator and Discriminator
  - Dual Loss Function
  - Major Issues
    - Mode Collapse
    - Why are GANs so hard to train
- Types of GAN
  - Deep Convolution GAN
  - Cycle GAN
  - Etc.
- Applications of GAN

Generative Models

- Given training data, generate new samples from same distribution



Training data $\sim p_{data}(x)$

Generated samples $\sim p_{model}(x)$

Want to learn $p_{model}(x)$ similar to $p_{data}(x)$

- Addresses density estimation, a core problem in unsupervised learning
- **Several flavors:**
  - Explicit density estimation: explicitly define and solve for $p_{model}(x)$
  - Implicit density estimation: learn model that can sample from $p_{model}(x)$ w/o explicitly defining it

Agenda

- Generative Models
- Variational Autoencoders (VAE)
- Generative Adversarial Networks (GAN)
  - Basic Mechanism: Generator and Discriminator
  - Dual Loss Function
  - Major Issues
    - Mode Collapse
    - Why are GANs so hard to train
- Types of GAN
  - Deep Convolution GAN
  - Cycle GAN
  - Etc.
- Applications of GAN

Autoencoders

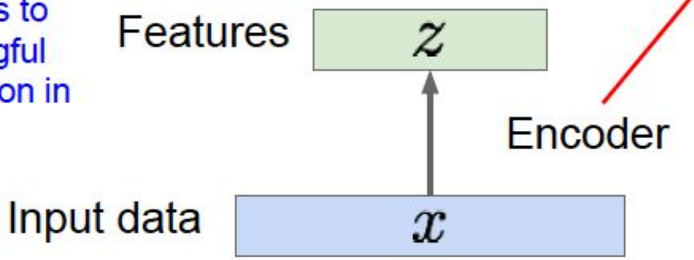
Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data

z usually smaller than **x**
(dimensionality reduction)

Q: Why dimensionality reduction?

A: Want features to capture meaningful factors of variation in data

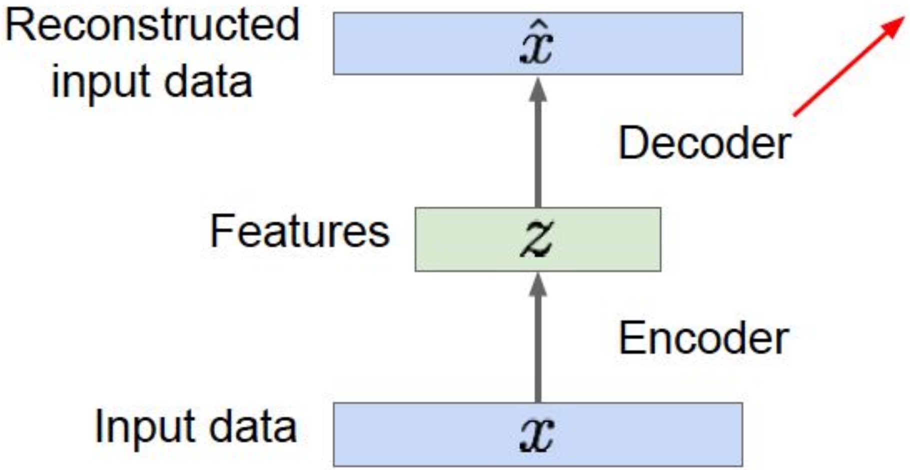
Originally: Linear + nonlinearity (sigmoid)
Later: Deep, fully-connected
Later: ReLU CNN



Autoencoders

How to learn this feature representation?
Train such that features can be used to reconstruct original data
"Autoencoding" - encoding itself

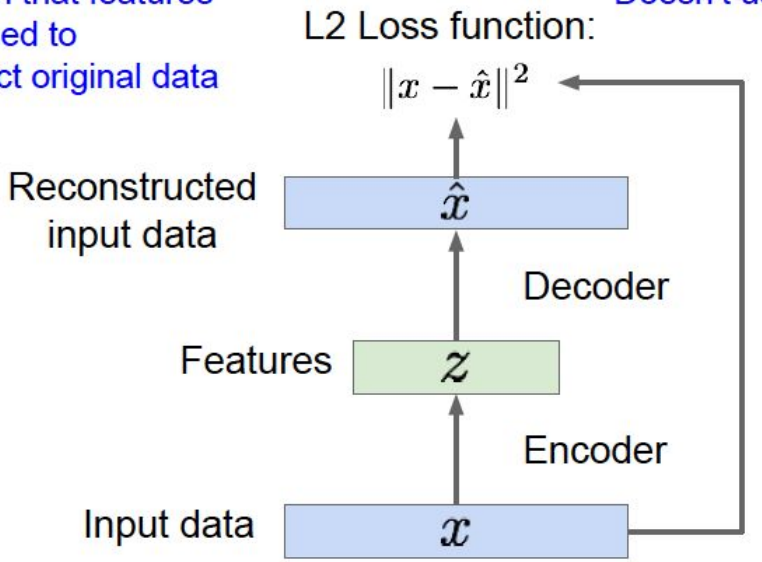
Originally: Linear + nonlinearity (sigmoid)
Later: Deep, fully-connected
Later: ReLU CNN (upconv)

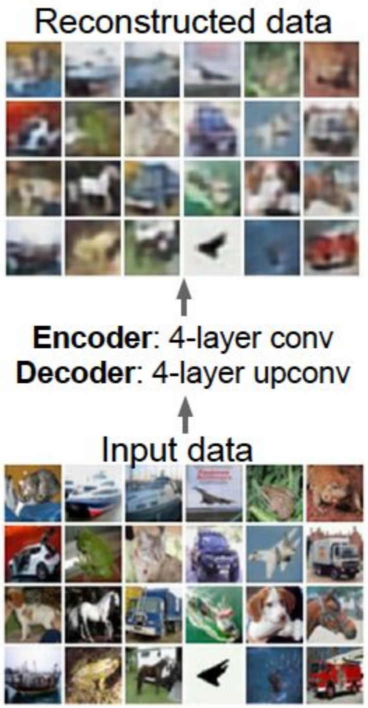


Autoencoder

Train such that features can be used to reconstruct original data

Doesn't use labels!





VAE

Probabilistic spin on autoencoders - will let us sample from the model to generate data!

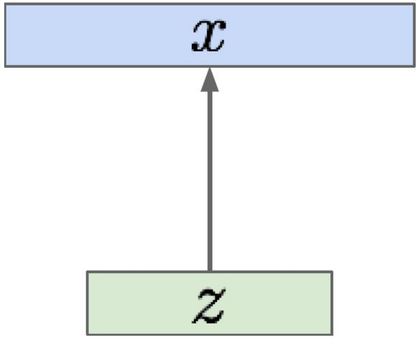
Assume training data $\{x^{(i)}\}_{i=1}^{N}$ is generated from underlying unobserved (latent) representation **z**

Sample from true conditional

$p_{\theta^*}(x \mid z^{(i)})$

Sample from true prior

$p_{\theta^*}(z)$

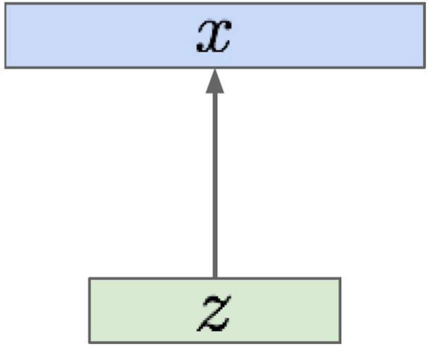


Intuition (remember from autoencoders!): **x** is an image, **z** is latent factors used to generate **x:** attributes, orientation, etc.

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

VAE

Sample from
true conditional
$p_{\theta^*}(x \mid z^{(i)})$



Sample from
true prior
$p_{\theta^*}(z)$

We want to estimate the true parameters θ^* of this generative model.

How should we represent this model?

Choose prior p(z) to be simple, e.g. Gaussian. Reasonable for latent attributes, e.g. pose, how much smile.

$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

Data likelihood: $p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$

Intractible to compute
p(x|z) for every z!

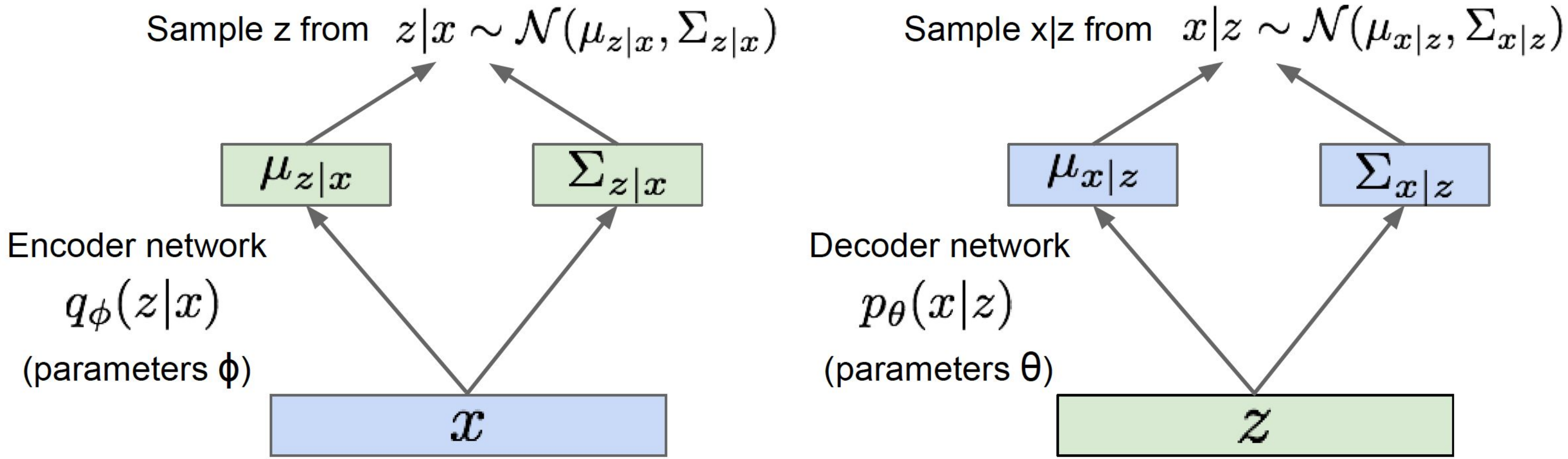
Posterior density also intractable: $p_\theta(z|x) = p_\theta(x|z) p_\theta(z) / p_\theta(x)$

Intractable data likelihood

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

VAE

Solution: In addition to decoder network modeling $p_\theta(x|z)$, define additional encoder network $q_\phi(z|x)$ that approximates $p_\theta(z|x)$



Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

VAE Loss Function

Now equipped with our encoder and decoder networks, let's work out the (log) data likelihood:

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \qquad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \qquad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \qquad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \qquad \text{(Logarithms)}$$

$$= \boxed{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right]} - \boxed{D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))} + D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))$$

Make approximate posterior distribution close to prior

Reconstruct Input data

Decoder network gives $p_\theta(x|z)$, can compute estimate of this term through sampling. (Sampling differentiable through reparam. trick, see paper.)

This KL term (between Gaussians for encoder and z prior) has nice closed-form solution!

$p_\theta(z|x)$ intractable (saw earlier), can't compute this KL term :(But we know KL divergence always >= 0.

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

VAE

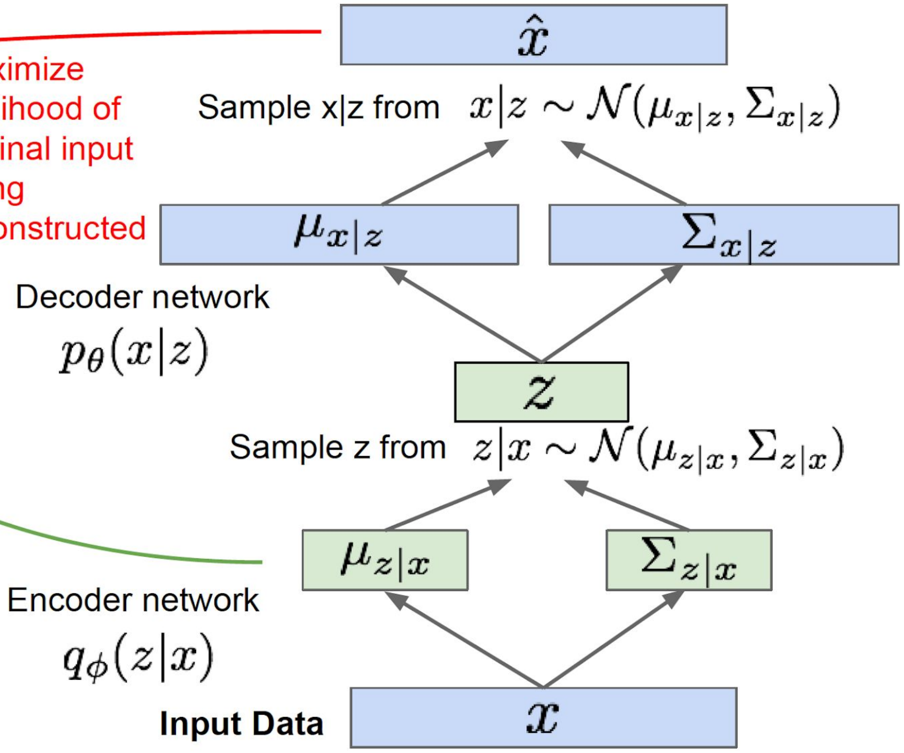
Putting it all together: maximizing the likelihood lower bound

$$\underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)},\theta,\phi)}$$

Maximize likelihood of original input being reconstructed

Make approximate posterior distribution close to prior

For every minibatch of input data: compute this forward pass, and then backprop!



Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

Agenda

- Generative Models
- Variational Autoencoders (VAE)
- Generative Adversarial Networks (GAN)
    - Basic Mechanism: Generator and Discriminator
    - Dual Loss Function
    - Major Issues
        - Mode Collapse
        - Why are GANs so hard to train
- Types of GAN
    - Deep Convolution GAN
    - Cycle GAN
    - Etc.
- Applications of GAN

Basic Mechanism

- Generator Network: try to fool the discriminator by generating real-looking images
- Discriminator Network: try to distinguish between real and fake images

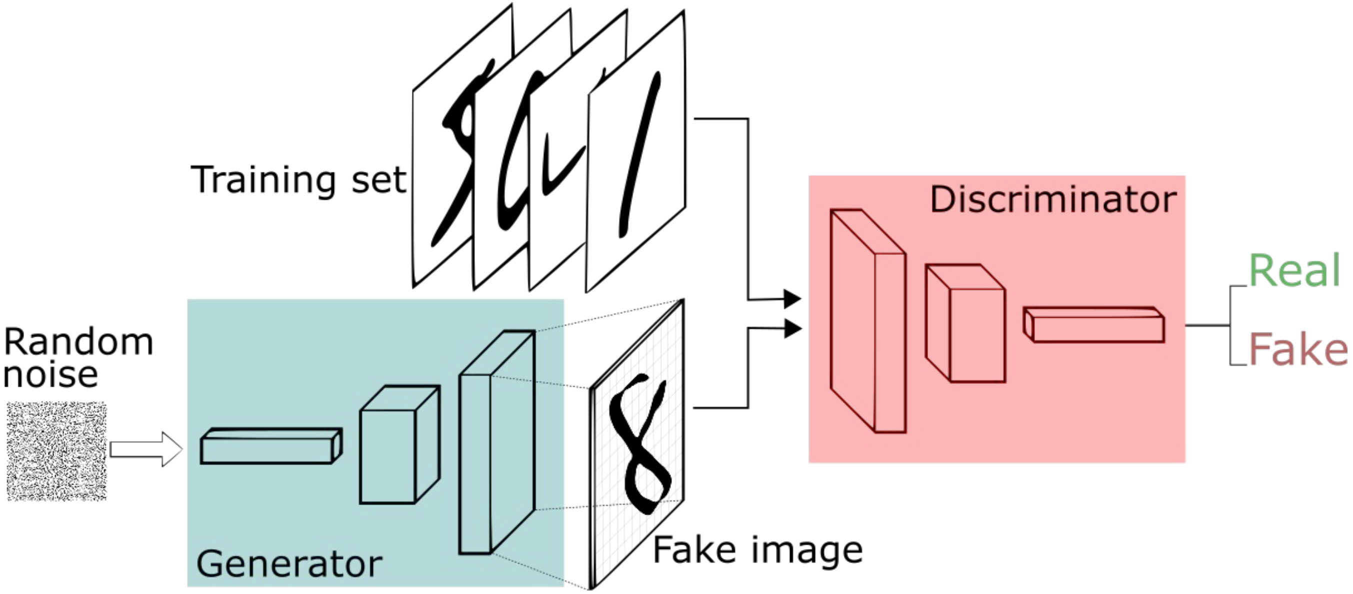


Image from: https://medium.freecodecamp.org/an-intuitive-introduction-to-generative-adversarial-networks-gans-7a2264a81394

Minmax Loss Function

$$\min_{\theta_g} \max_{\theta_d} \left[\boldsymbol{E}_{x\sim p_{data}} log \underbrace{D_{\theta_d}(x)}_{\substack{\text{Discriminator output} \\ \text{for real data x}}} + \boldsymbol{E}_{z\sim p(z)} \log(1 - \underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\substack{\text{Discriminator output for} \\ \text{generated fake data G(z)}}}) \right]$$

- Discriminator (θ_d) wants to **maximize objective** such that $D(x)$ is close to *1* (real) and $D(G(z))$ is close to *0* (fake)
- Generator (θ_g) wants to **minimize objective** such that $D(G(z))$ is close to 1 (discriminator is fooled into thinking generated $G(z)$ is real)

Ian Goodfellow et al. “General Adversarial Nets”, NIPS 2014

Training

Minmax loss function:

$$\min_{\theta_g} \max_{\theta_d} \left[\boldsymbol{E}_{x \sim p_{data}} log D_{\theta_d}(x) + \boldsymbol{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between

- **Gradient ascent** on discriminator

$$\max_{\theta_d} \left[\boldsymbol{E}_{x \sim p_{data}} log D_{\theta_d}(x) + \boldsymbol{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

- **Gradient descent** on generator

$$\boxed{\min_{\theta_g} \boldsymbol{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))}$$

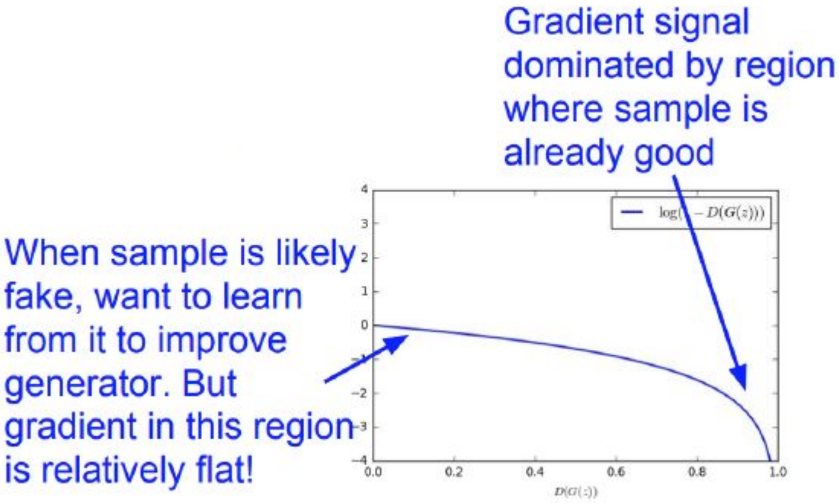
However, this doesn't work well in practice

Ian Goodfellow et al. "General Adversarial Nets", NIPS 2014

Training

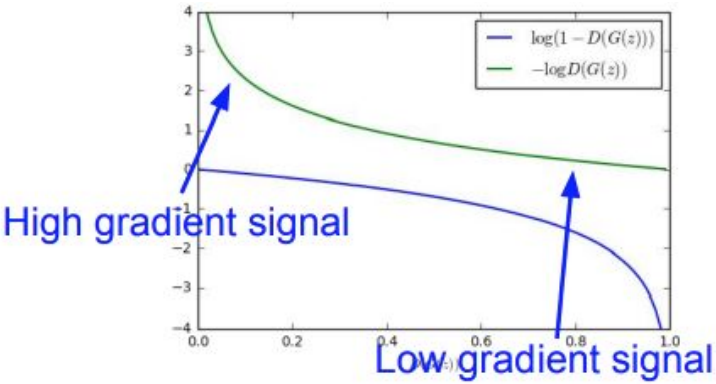
Gradient descent on generator:

$$\min_{\theta_g} \boldsymbol{E}_{z\sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$



We thus choose **gradient ascent on generator**:

$$\max_{\theta_g} \boldsymbol{E}_{z\sim p(x)} \log(D_{\theta_d}(G_{\theta_g}(z)))$$



Training

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

for number of training iterations **do**

  for k steps **do**

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

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

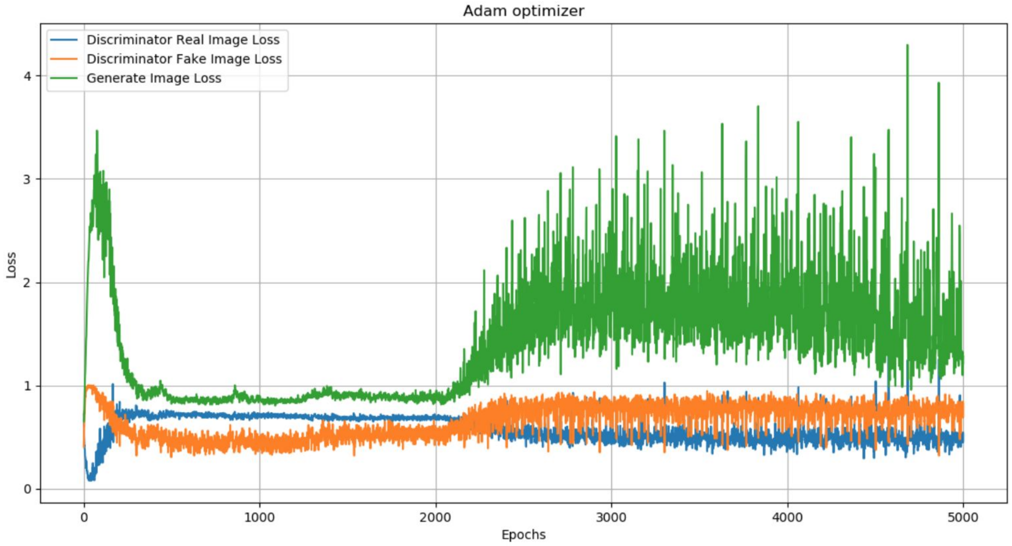
  end for

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

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

end for

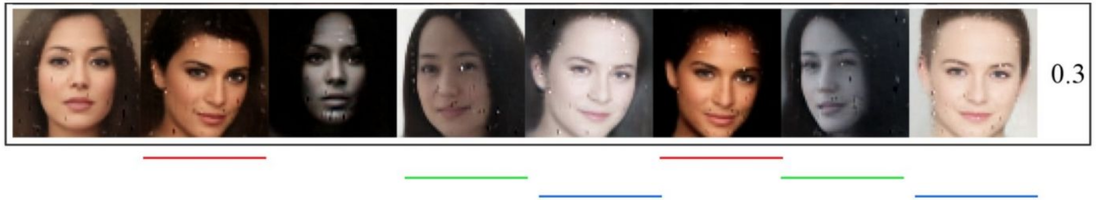
The gradient-based updates can use any standard gradient-based learning rule. We used momen tum in our experiments.



Ian Goodfellow et al. “General Adversarial Nets”, NIPS 2014

Major Issues - mode collapse

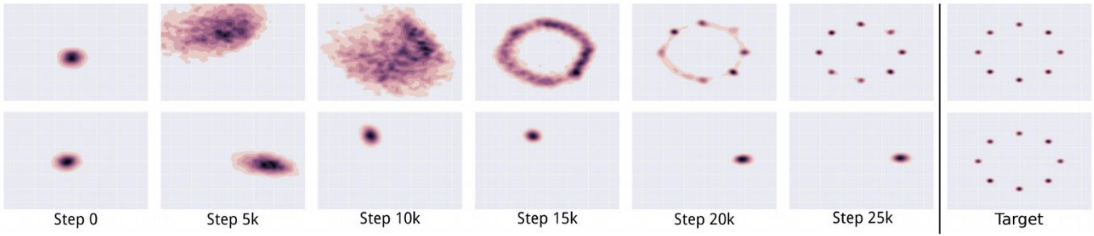
- Mode collapse happens when the model only generates few modes of data



- Why does it happen? Consider the most extreme case where G is trained so extensively without updates to D that it finds the most optimal generated image that fools D the most, a.k.a the mode collapses to a single point.

$$x^* = argmax_x D(x) \qquad \frac{\partial J}{\partial z} \approx 0$$

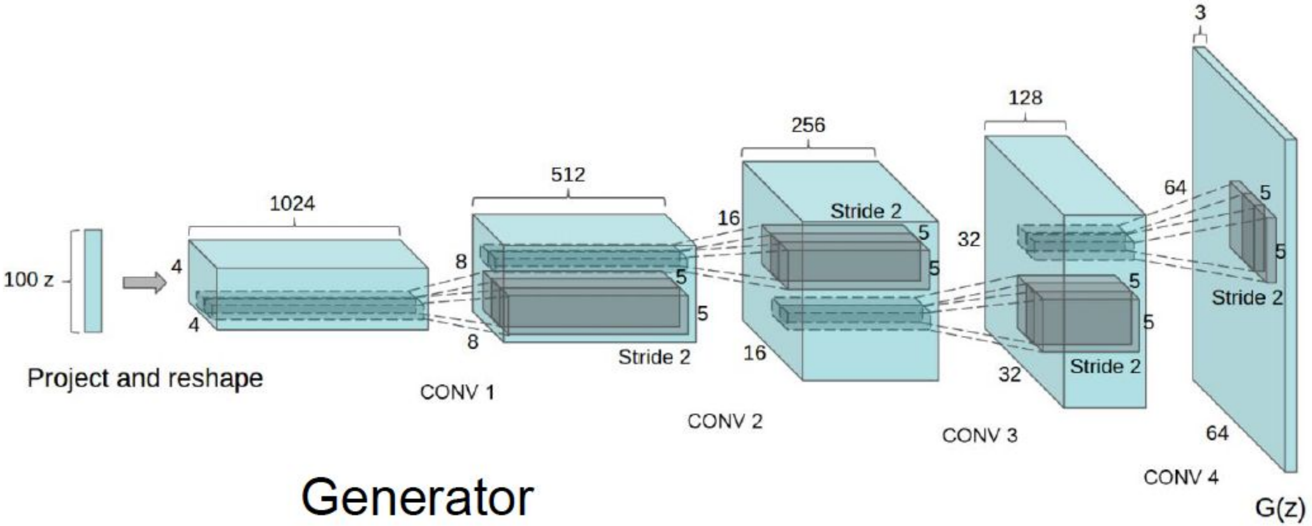
- When D starts to train again, it can easily detect the single mode, which leads D to overfit to short-term opponent's weaknesses.



Agenda

- Generative Models
- Variational Autoencoders (VAE)
- Generative Adversarial Networks (GAN)
    - Basic Mechanism: Generator and Discriminator
    - Dual Loss Function
    - Major Issues
        - Mode Collapse
        - Why are GANs so hard to train
- Types of GAN
    - Deep Convolution GAN
    - Cycle GAN
    - etc.
- Applications of GAN

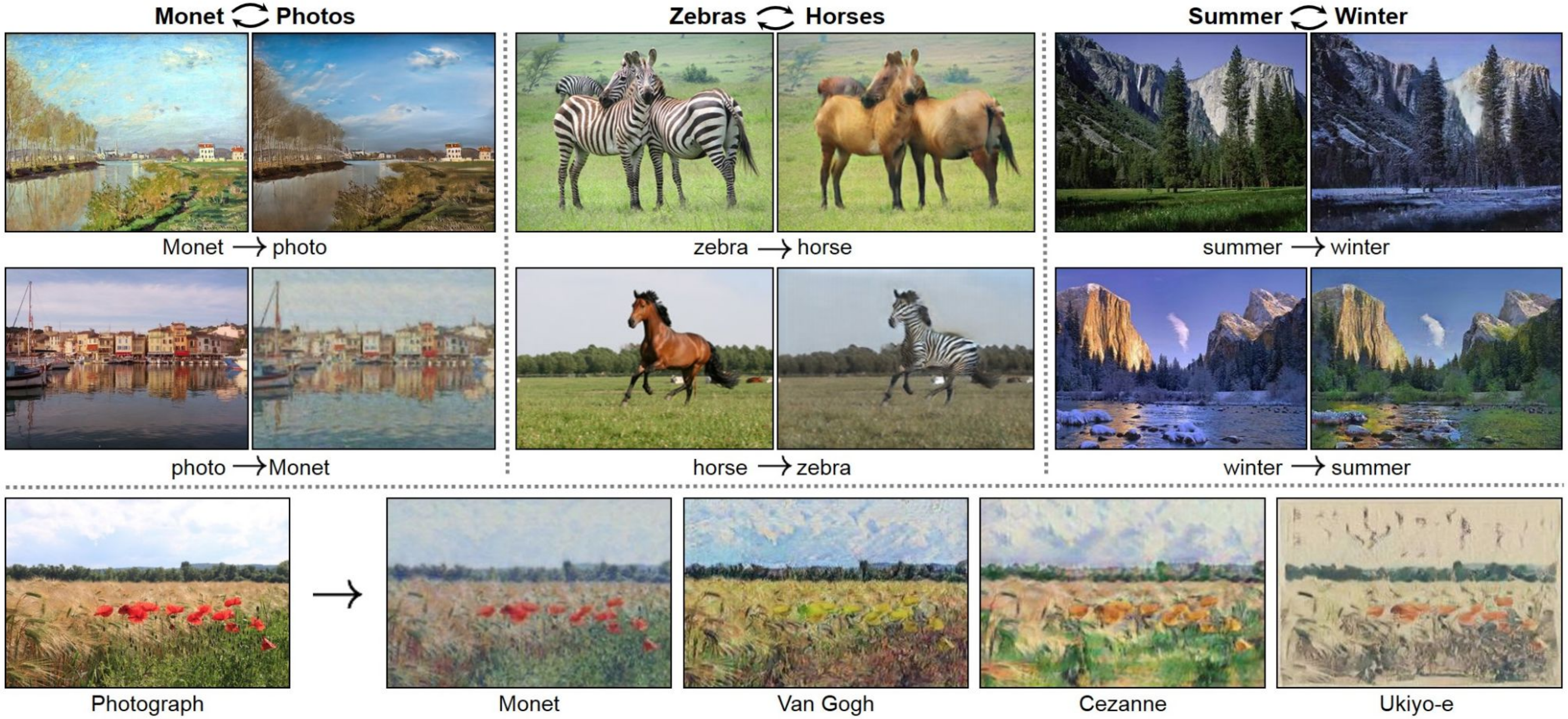
Deep Convolution GAN



- Generator is an upsampling network with fractionally-strided convolutions
- Discriminator is a convolutional network

Radford et al, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”, ICLR 2016

Cycle GAN



Zhu et al. “Unpaired Image-to-image Translation using Cycle-consistent Adversarial Networks”, ICCV 2017

Cycle GAN

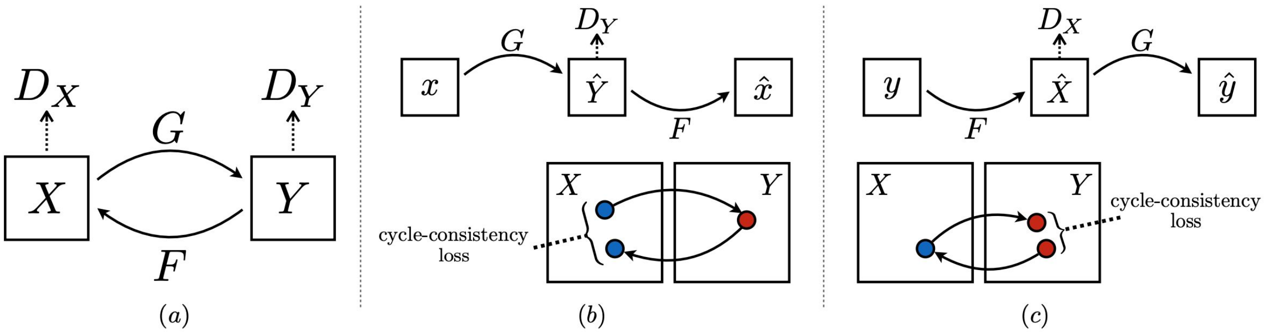


Figure 3: (a) Our model contains two mapping functions $G : X \rightarrow Y$ and $F : Y \rightarrow X$, and associated adversarial discriminators D_Y and D_X. D_Y encourages G to translate X into outputs indistinguishable from domain Y, and vice versa for D_X and F. To further regularize the mappings, we introduce two *cycle consistency losses* that capture the intuition that if we translate from one domain to the other and back again we should arrive at where we started: (b) forward cycle-consistency loss: $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$, and (c) backward cycle-consistency loss: $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$

$$\begin{aligned}\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) &= \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] \\ &+ \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))]\end{aligned}$$

Adversarial Loss

$$\begin{aligned}\mathcal{L}_{\text{cyc}}(G, F) &= \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] \\ &+ \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1].\end{aligned}$$

Cycle Consistency Loss

Zhu et al. “Unpaired Image-to-image Translation using Cycle-consistent Adversarial Networks”, ICCV 2017

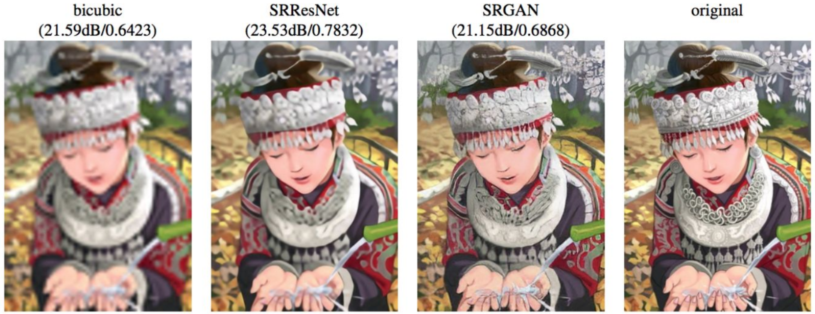
Agenda

- Generative Models
- Variational Autoencoders (VAE)
- Generative Adversarial Networks (GAN)
  - Adversarial Examples
  - Basic Mechanism: Generator and Discriminator
  - Dual Loss Function
  - Major Issues
    - Mode Collapse
    - Why are GANs so hard to train
- Types of GAN
  - Deep Convolution GAN
  - Cycle GAN
  - etc.
- Applications of GAN

Application

Improved resolution (SRGAN)

Ledig et al, "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network"



Text to Image (StackGAN)

Zhang et al, "StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks"

Image Inpainting

Pathak et al, "Context encoding: feature learning by inpainting", CVPR 2016