#DLUPC

Day 4 Lecture 3

Generative Adversarial Networks

Verónica Vilaplana
veronica.vilaplana@upc.edu

Associate Professor
Universitat Politecnica de Catalunya
Technical University of Catalonia

UPC

Index

Supervised vs unsupervised learning

Supervised Learning

Data: (x,y)

x data, y label

Goal: learn a function to map x-> y

Examples: classification, regression, object detection, semantic segmentation, image captioning, etc.

Unsupervised learning

Data: x

just data, no labels

Goal: learn some underlying hidden structure

Examples: clustering, dimensionality reduction, feature learning, density estimation, etc.

S. Credit: Stanford cs231_2017

Generative modeling

- Generative models take training samples from some (unknown) data distribution p_{data} and learn a model p_{model} that represents that distribution

- Density estimation
  (explicit)

- Sample generation

(implicit density estimation)

Training samples p_{data}

Generated samples p_{model}

Why are generative models important?

- **Represent and manipulate** very complex and high dimensional distributions
- Enable **inference of latent representations** that can be used as general features
- Used in **reinforcement learning applications**: generative models of time-series data can be used for simulation or planning
- **Generate training data** for discriminative models
- **Manipulate real samples** with the assistance of a generative model (ex image inpainting)
- **Generate realistic synthetic** samples (ex. for artwork, super-resolution, image-to-image translation applications, colorization, etc.)

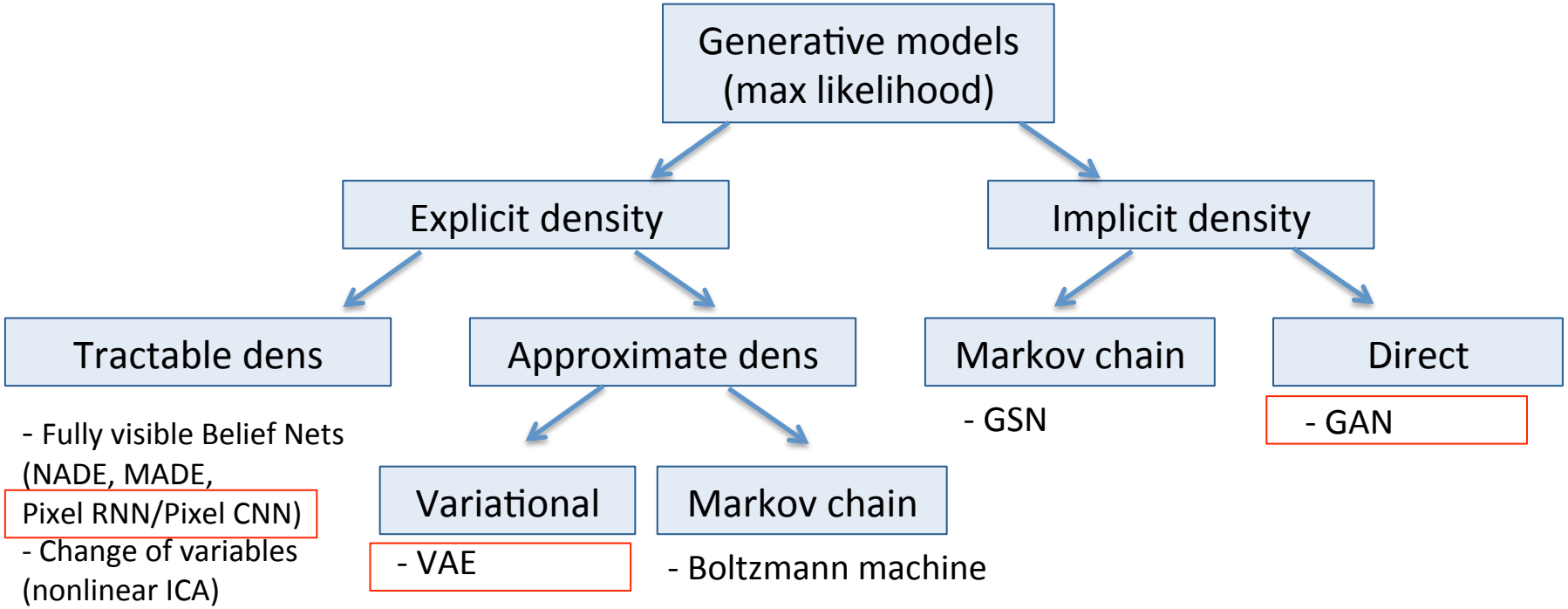
Maximum likelihood

- We will focus on generative models that work via the principle of maximum likelihood:
- Define a model that **provides an estimate of a probability distribution** p_{model}, parameterized by Θ.
- Likelihood of the training data $\{x^{(i)}\}_i$: $p_\theta(x) = p_{\text{model}}(x,\theta) = \prod_{i=1}^{m} p_{\text{model}}(x^{(i)},\theta)$
- Choose the parameters for the model that **maximize the likelihood of the training data**

$$\theta^* = \arg\max_\theta \prod_{i=1}^{m} p_{\text{model}}(x^{(i)},\theta)$$

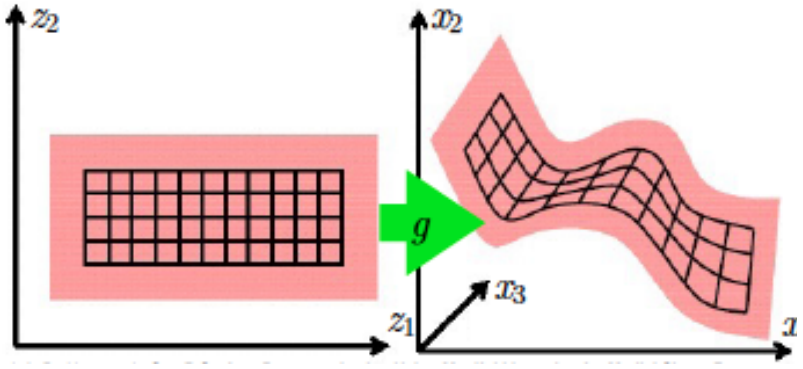
$$= \arg\max_\theta \sum_{i=1}^{m} \log p_{\text{model}}(x^{(i)},\theta)$$

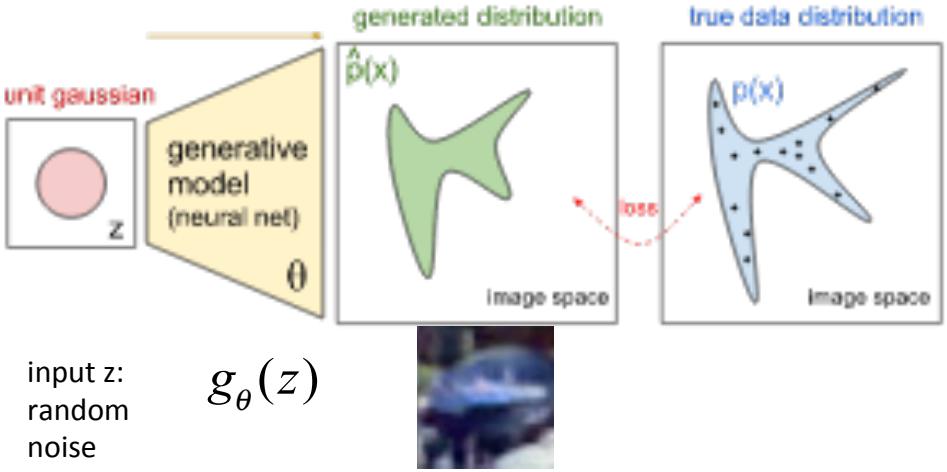
Taxonomy of generative models



Generative Adversarial Networks

- Don’t work with any explicit density function
- Problem: no direct way to sample from complex high-dimensional training distribution
- Solution: sample from a simple distribution (e.g. random noise), learn a transformation to training distribution





input z: random noise

$g_\theta(z)$

Output x: sample from training distribution

Generative Adversarial Networks

- A game-theoretic approach: learn to generate from training distribution through a 2-player game: **generator** (G) and **discriminator** (D)

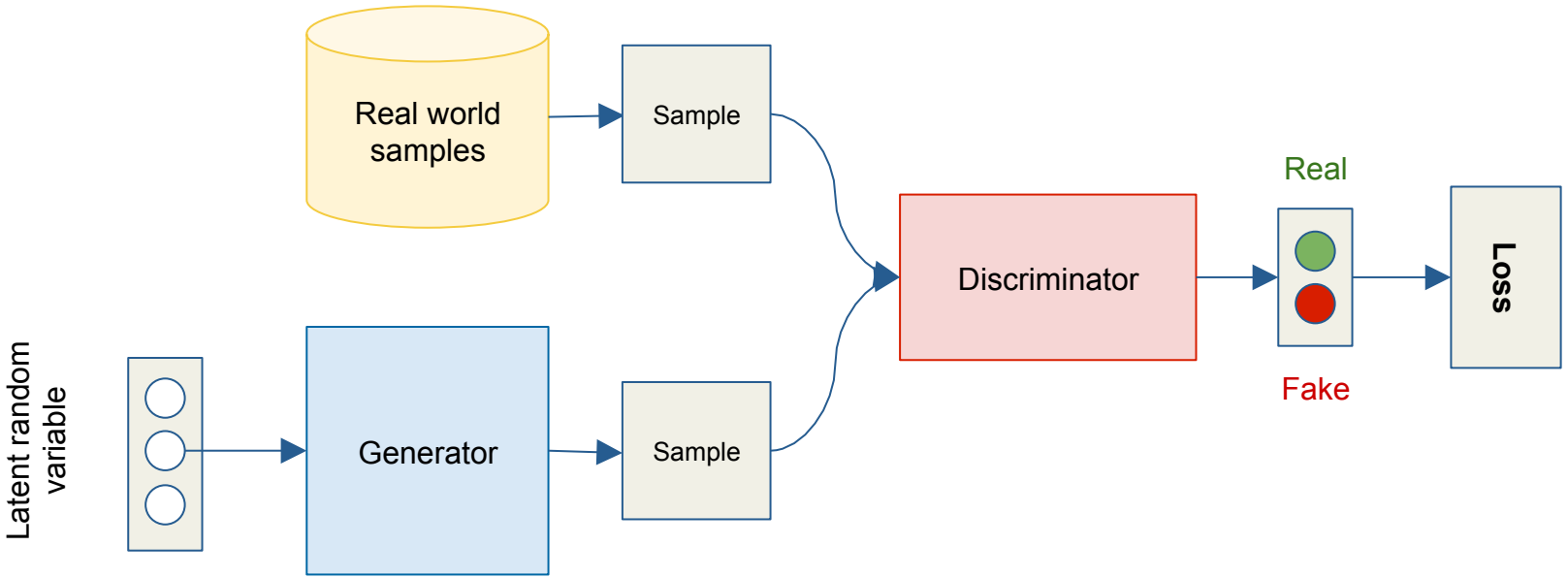
  - They 'fight' against each other during training -> **Adversarial training**
  - G: tries to fool D by generating real-looking samples
  - D: tries to distinguish between real and fake samples generated by G

  Both are deep networks (differentiable functions)

  Can be trained with backpropagation: train discriminator for a while, then train generator, then discriminator…

Adversarial training

- **Generator:** deterministic mapping from a latent random vector z to sample from p_{model}, which should be similar to p_{data} ; usually a deep neural network
- **Discriminator**: parameterized function that tries to distinguish between real and fake samples, usually a deep neural network



S. Credit: K. McGuinness

Adversarial training

- G and D are trained jointly in **minimax game**
- Minimax objective function:

Discriminator outputs likelihood in (0,1) of real image

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

Discriminator output for real data x

Discriminator output for generated fake data G(z)

$$x\sim p_{data} \quad z\sim N(0,1) \quad \hat{x}=G(z)\sim p_{\text{model}} \quad \theta_d:\ \text{D parameters} \quad \theta_g:\ G\ \text{parameters}$$

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

S. Credit: Stanford cs231_2017

Adversarial training

Minimax objective function:

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

Alternate between:

1. **Gradient ascent** on discriminator

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

2. **Gradient descent** on generator

$$\min_{\theta_g} \left[E_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

In practice, optimizing this generator objective does not work well

S. Credit: Stanford cs231_2017

Adversarial training

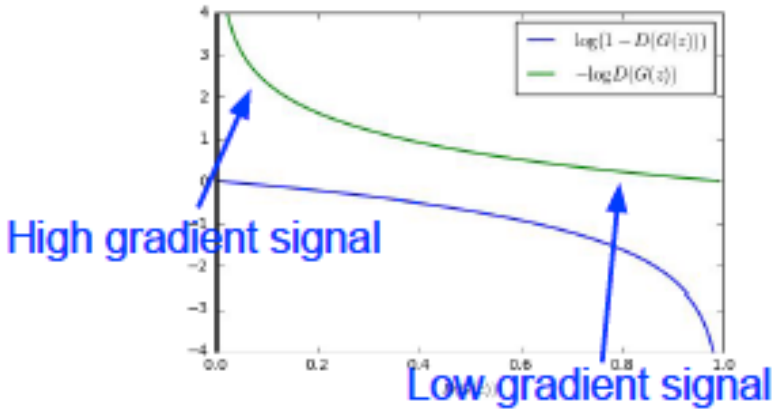
Alternate between

1. **Gradient ascent** on discriminator

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

2. **Instead: Gradient <u>ascent</u>** on generator, different objective

$$\max_{\theta_g}\left[E_{z\sim p(z)} \log(D_{\theta_d}(G_{\theta_g}(z))) \right]$$

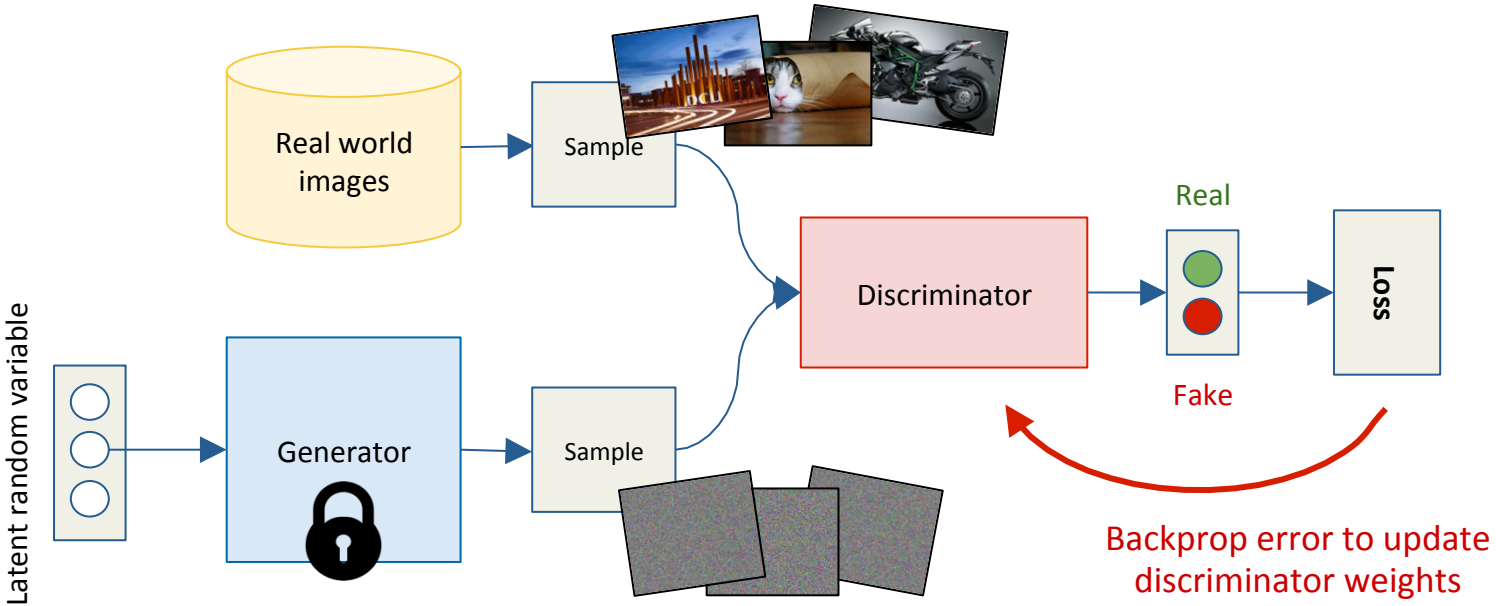


Maximizing likelihood of discriminator being wrong;
works much better in practice
(higher gradient signal for bad samples)

S. Credit: Stanford cs231_2017

Training GANs

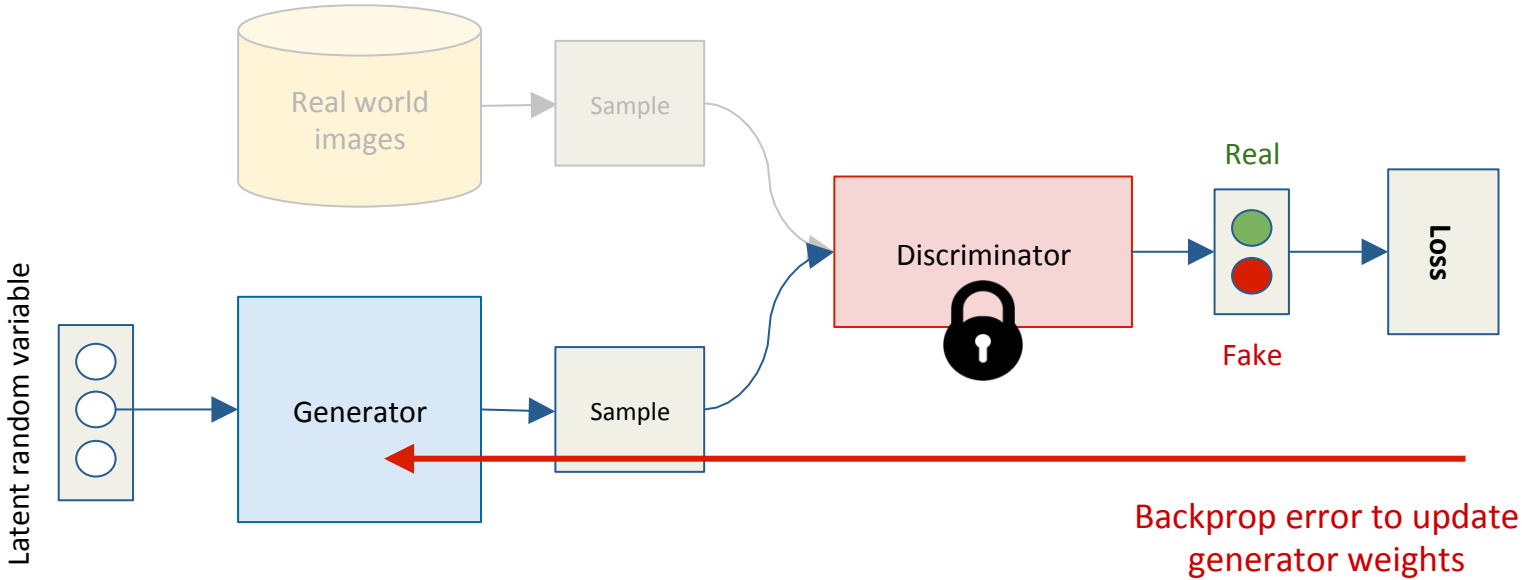
1. Fix generator weights, draw samples from both real world and generated samples
2. Train discriminator to distinguish between real world and generated samples



S. Credit: K. McGuinness

Training GANs

1. Fix discriminator weights
2. Sample from generator
3. Backprop error through discriminator to update generator weights

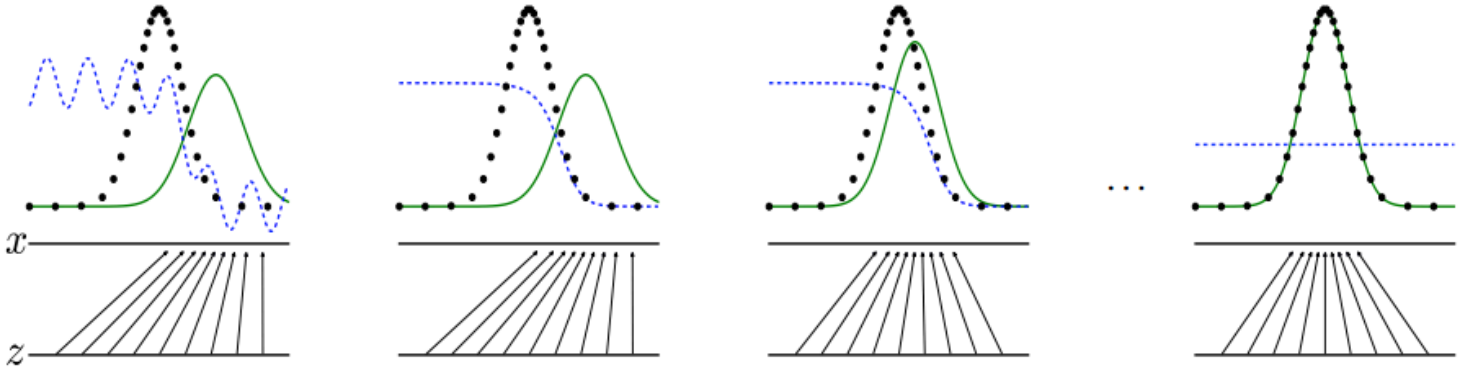


Iterate these two steps until convergence (which may not happen)

S. Credit: K. McGuinness

Training GANs

- Iterate these two steps until convergence (which may not happen)
  - Updating the discriminator should make it better at discriminating between real images and generated ones (**discriminator improves**)
  - Updating the generator makes it better at fooling the current discriminator (**generator improves**)
- Eventually (we hope) that the generator gets so good that it is impossible for the discriminator to tell the difference between real and generated images. Discriminator accuracy = 0.5



S. Credit: K. McGuinness

GAN training algorithm

for number of training iterations **do**

Discriminator training

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:

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

k >1 discriminator updates may be more stable

end for

Generator training

- 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

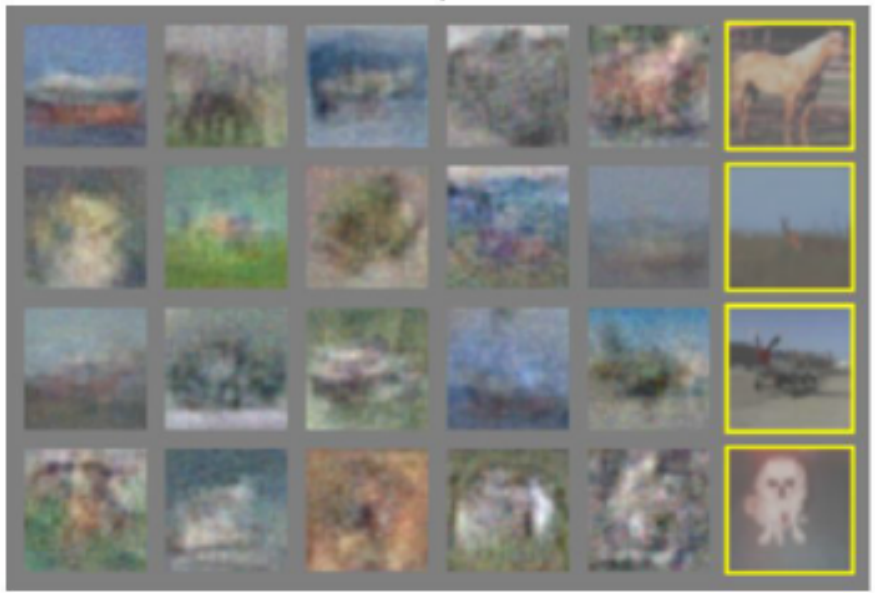
GAN training example

- MNIST digits

Generated samples

- After training we use generator network to generate new samples

CIFAR dataset



Nearest neighbor from the training set

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

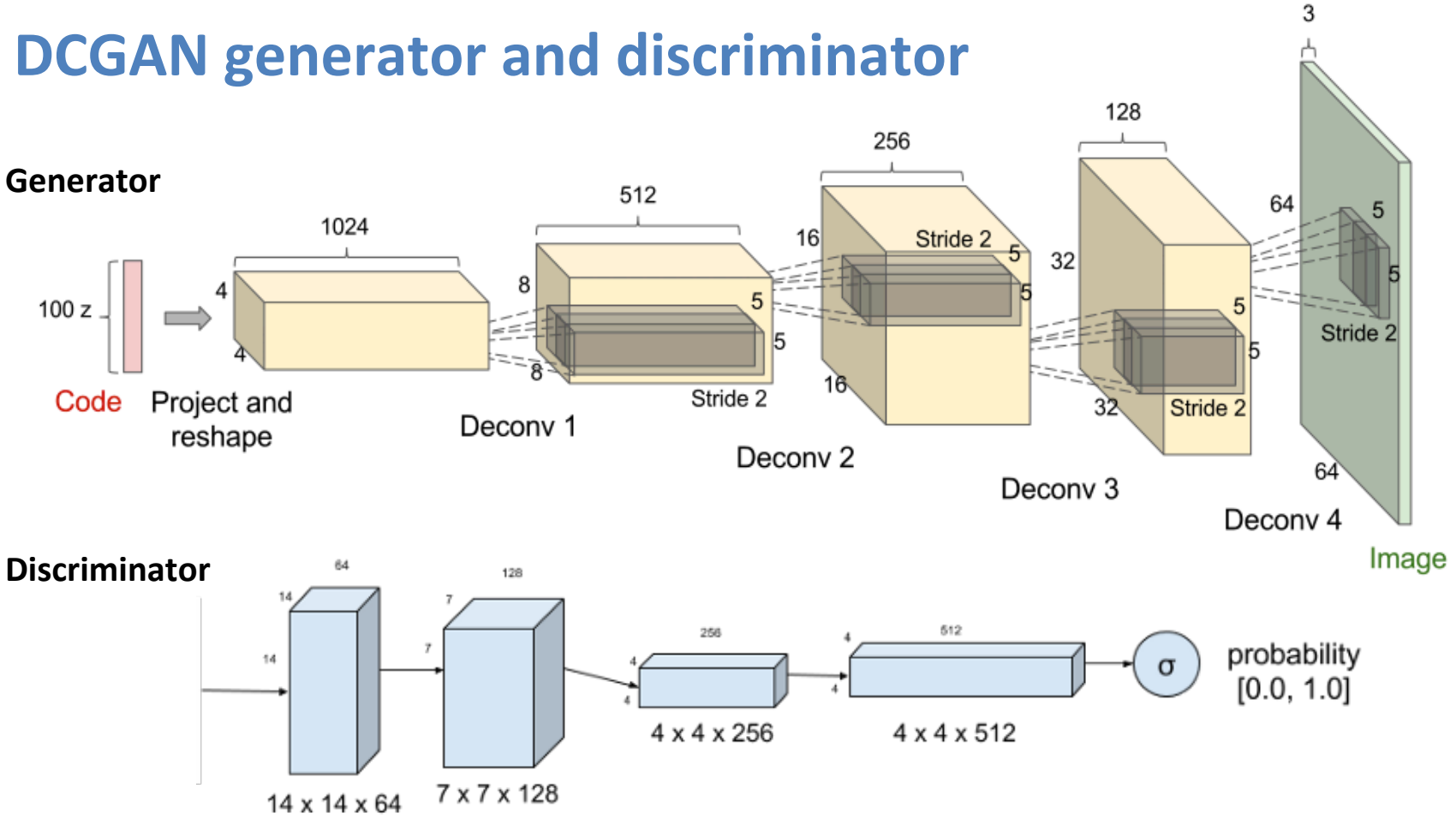
GANs: convolutional architectures

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

- Architecture guidelines for stable Deep Convolutional GANs
  - replace pooling layer with strided convolutions (D) and fractionally strided convolutions (G)
  - Use batch normalization in both G and D (except G output layer, D input layer)
  - Remove fully connected hidden layers for deeper architectures
  - Use ReLu activation in G for all layers except the output (tanh)
  - Use LeakyRelu activation in D for all layers

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

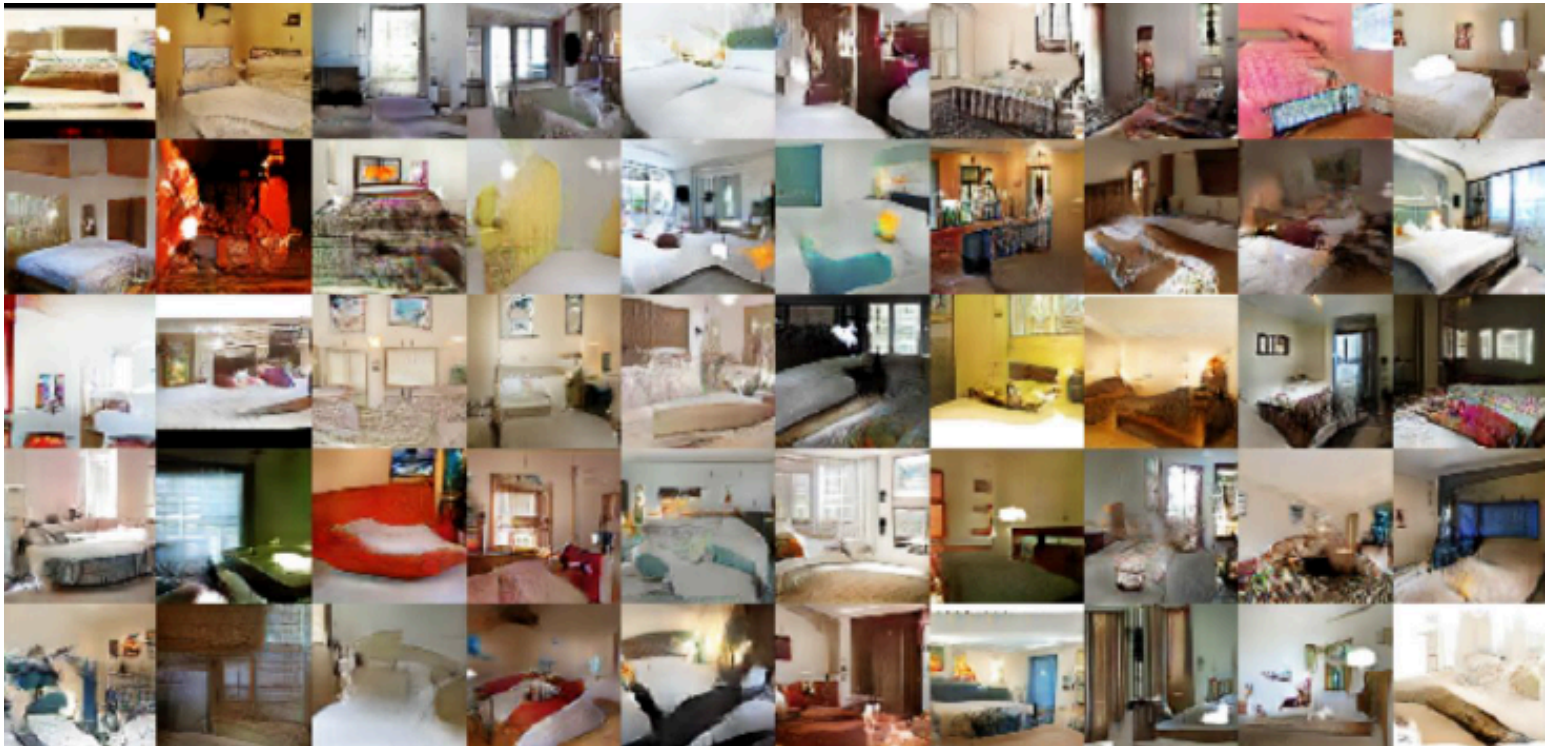
DCGAN generator and discriminator



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

Examples DCGAN

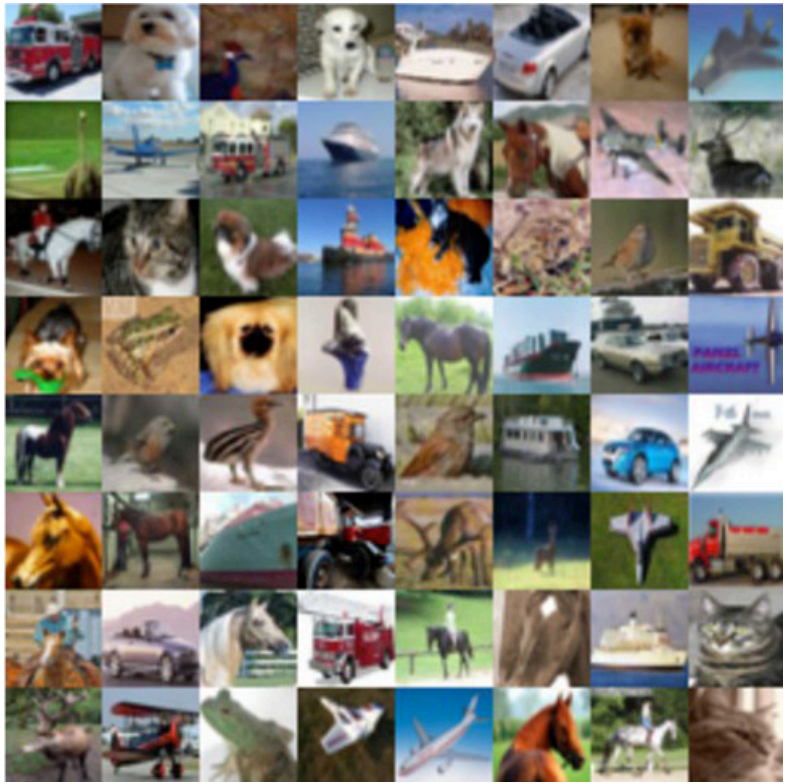
- Generated bedrooms after 5 epochs of training (LSUN bedrooms dataset, 64x64)

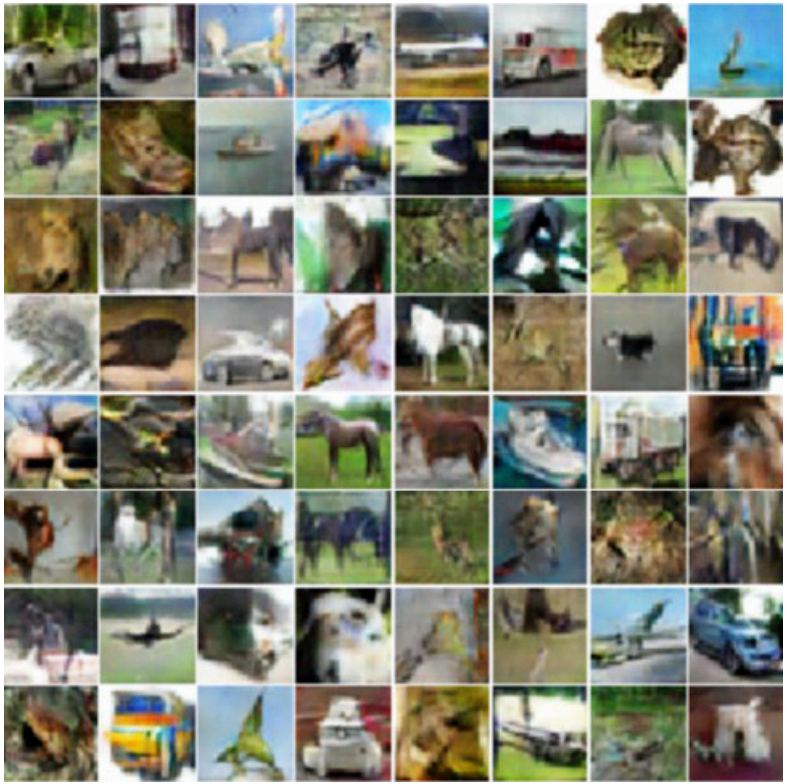


Radford et al, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks", ICLR 2016

Examples DCGAN

- CIFAR dataset





Source: https://openai.com/blog/generative-models/

Examples DCGAN

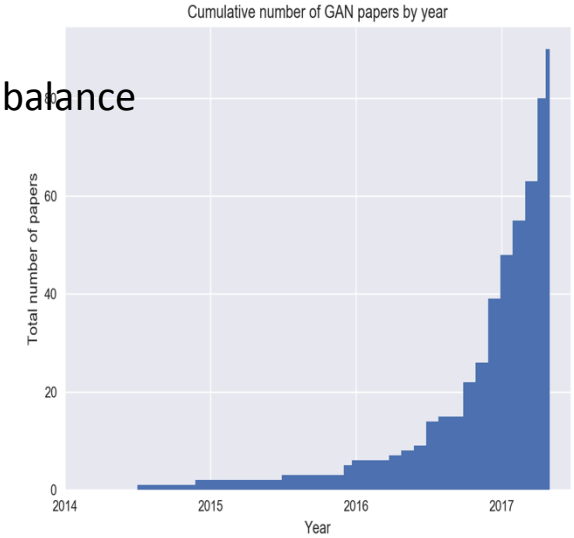
- Interpolating between random points in latent space show that the space learned has smooth transitions, with every image in the space plausibly looking like a bedroom

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

Issues

GANs are known to be very difficult to train: we are not minimizing a cost function, but we want both networks to reach Nash equilibrium (saddle point)

- Formulated as a ‘game’ between two networks
- Unstable dynamics: hard to keep generator and discriminator in balance
- Optimization can oscillate between solutions
- Generator can collapse



GAN papers by year

Possible to use supervised labels to help prevent this:

Improved Techniques for Training GANs (Salimans et al. 2016)

Wasserstein GAN can improve dynamics: Wasserstein GAN (Arjovsky, 2017)

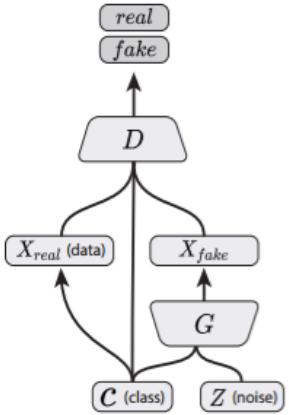
Tricks and tips: https://github.com/soumith/ganhacks

Application

- Generating images conditioned on captions: Text->Image synthesis

GANs can be conditioned on other information in addition to z: text, labels, speech, etc.

- z might capture random characteristics of data (variabilities of plausible features)
- c would condition the deterministic parts

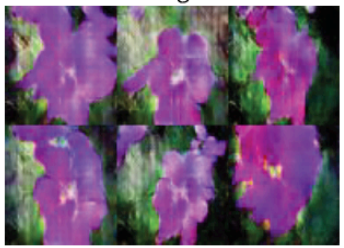


Conditional GAN
(Mirza & Osindero, 2014)

this small bird has a pink breast and crown, and black primaries and secondaries.

this magnificent fellow is almost all black with a red crest, and white cheek patch.

the flower has petals that are bright pinkish purple with white stigma



this white and yellow flower have thin white petals and a round yellow stamen

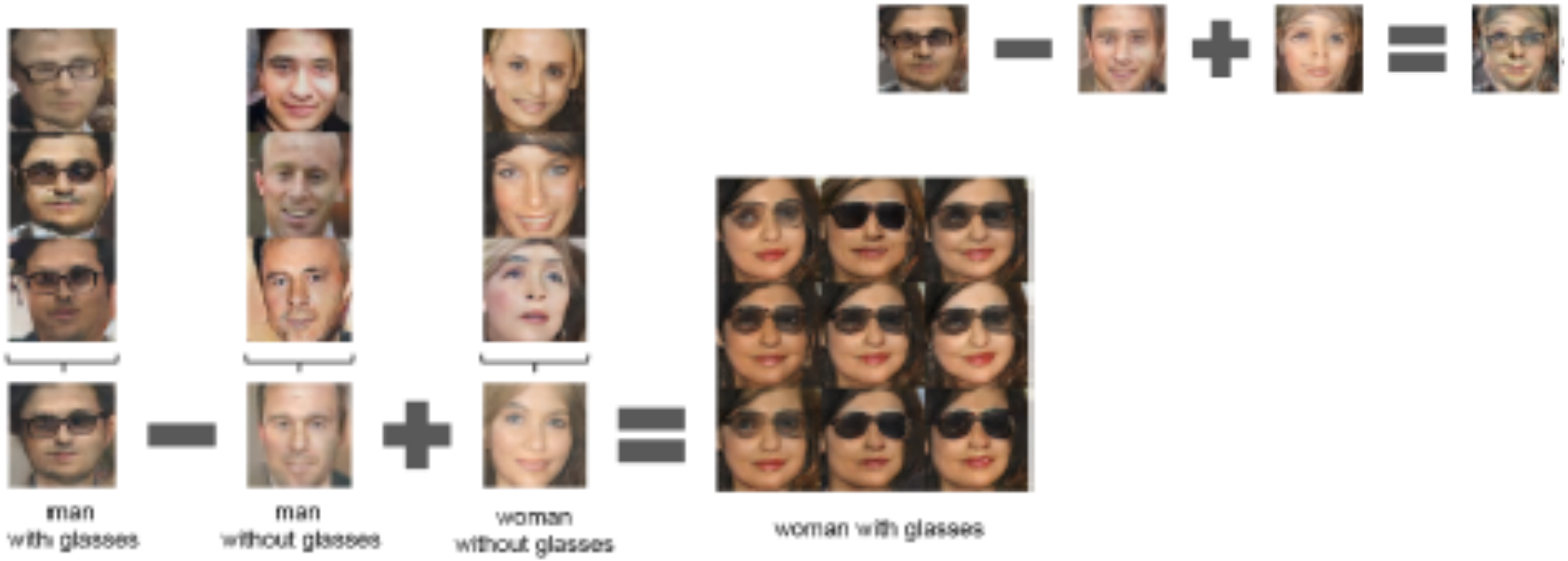
Ways of conditioning Generatie Adversarial Networks (Kwak, 2016)

Reed, Generative Adversarial Text to Image Synthesis, 2016

Application

- Unsupervised feature extraction / learning representation

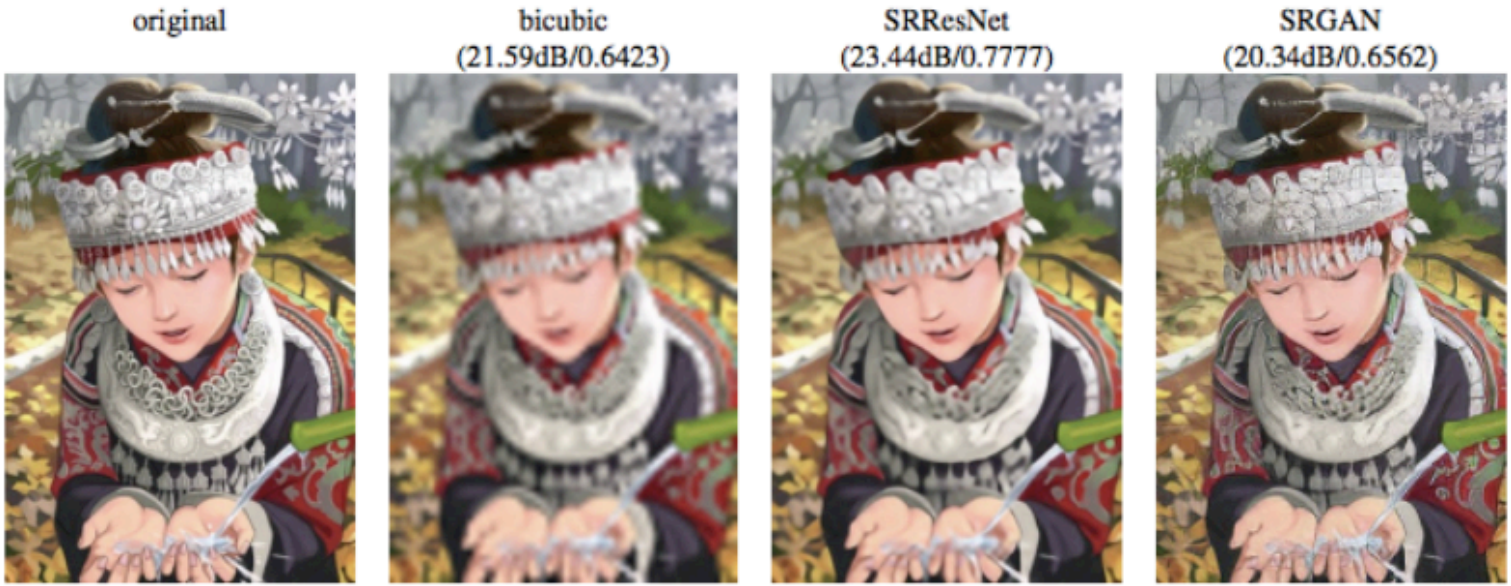
Vector arithmetic on Z vectors of sets of samples for visual concept: arithmetic on mean z vectors of samples,



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

Application

- Image super-resolution: SRGAN (4x upscaling)

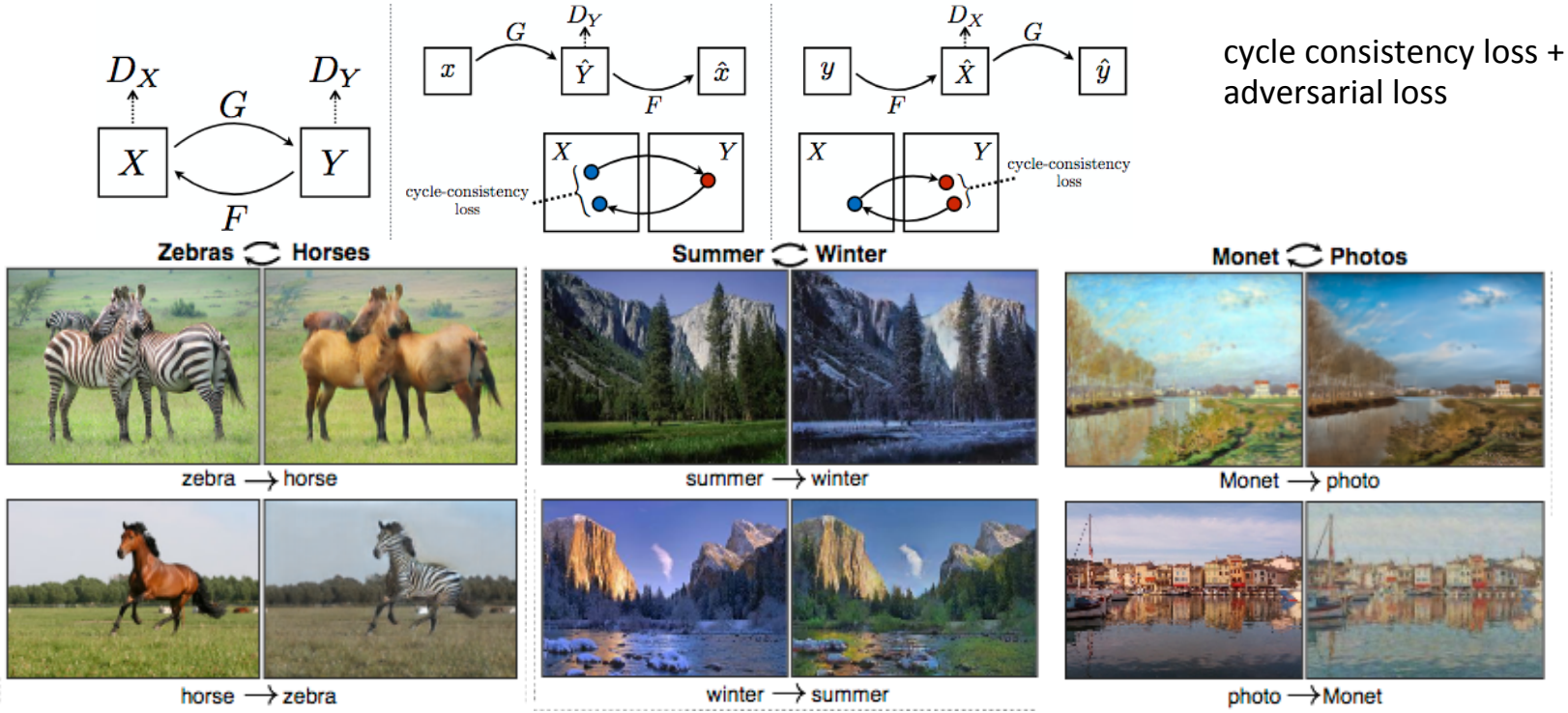


- Uses a perceptual loss function (adversarial loss + content loss). SSResNet uses MSE.

Ledig, Photo realistic single image super-resolution using a GAN, 2017

Applications

- Source -> target domain transfer: CycleGAN (without paired examples)



cycle consistency loss + adversarial loss

Zhu, Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Network, 2017

Summary GANs

- Do not model an explicit density function
- Use a game-theoretic approach: learn to generate from training distribution through 2-player game

- **Pros**
  - Produce state-of-the-art samples

- **Cons**
  - Hard / unstable to train
  - Can't solve inference queries such as p(x), p(z|x)

- **Active area of research:**
  - Better loss functions, more stable training
  - Conditional GANs
  - GANs for many different applications

Bibliograpy

- Goodfellow, NIPS 2016 Tutorial: Generative Adversarial Networks
- Improved Techniques for Training GANs (Salimans et al. 2016)
- Generative Adversarial Networks: An Overview (Creswell et al. 2017)
- Generative Adversarial Networks (Goodfellow et al. 2014)
- S. Yeung, Stanford course CS231n Convolutional Neural Networks for Visual Recognition, Lecture 13