

# Lecture 7: Implicit Generative Models (GANs)

## Efstratios Gavves

# Lecture overview

---

- Gentle intro to generative models
- Generative Adversarial Networks
- Variants of Generative Adversarial Networks

## Generative models



(a) EBGAN (64x64)



(b) Our results (128x128)

# Types of Learning

---

- Generative modelling
  - Learn the joint pdf:  $p(x, y)$
  - Model the world → Perform tasks, e.g. use Bayes rule to classify:  $p(y|x)$
  - Naïve Bayes, Variational Autoencoders, GANs
- Discriminative modelling
  - Learn the conditional pdf:  $p(y|x)$
  - Task-oriented
  - E.g., Logistic Regression, SVM

# Types of Learning

---

- What to pick?
  - V. Vapnik: “One should solve the [classification] problem directly and never solve a more general [and harder] problem as an intermediate step.”
- Typically, discriminative models are selected to do **the job**
- Generative models give us more theoretical guarantees that the model is going to work as intended
  - Better generalization
  - Less overfitting
  - Better modelling of causal relationships

# Applications of generative modeling?

---

# Applications of generative modeling?

---

- Act as a regularizer in discriminative learning
  - Discriminative learning often too goal-oriented
  - Overfitting to the observations
- Semi-supervised learning
  - Missing data
- Simulating “possible futures” for Reinforcement Learning
- Data-driven generation/sampling/simulation

# Applications: Image Generation



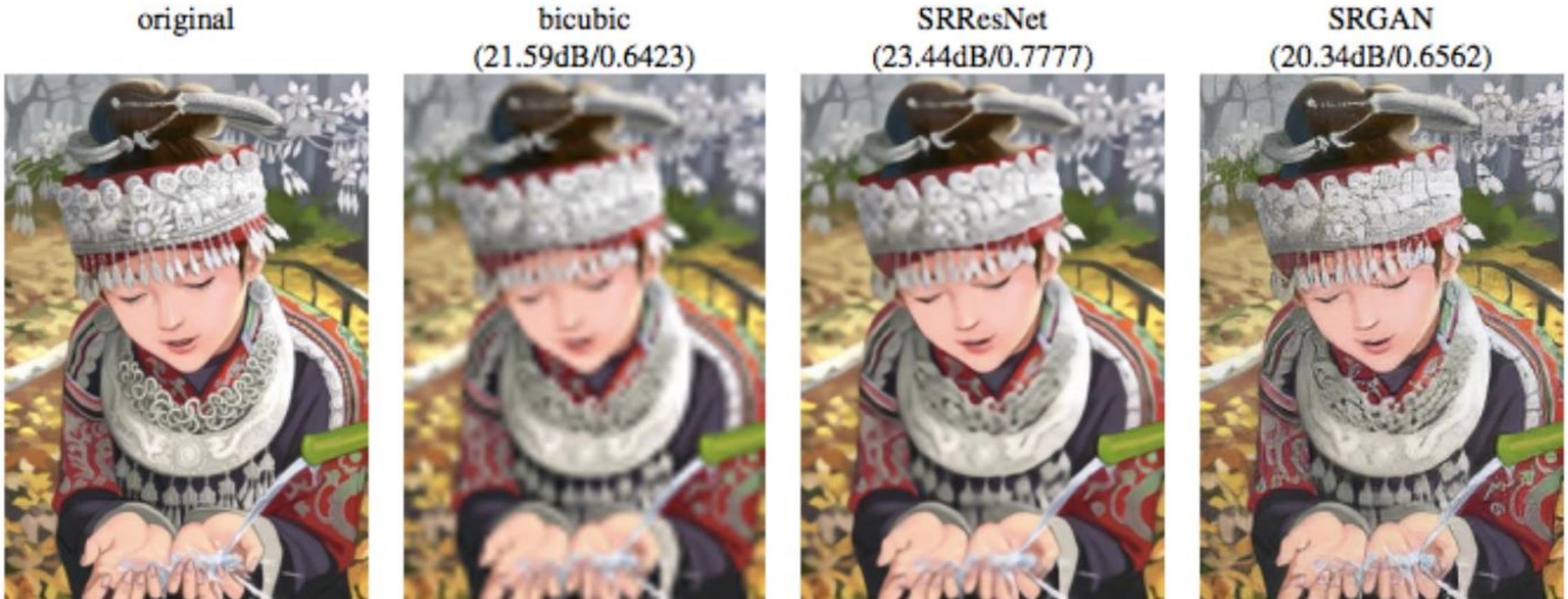
(a) Generated by LSGANs.



(b) Generated by DCGANs (Reported in [13]).



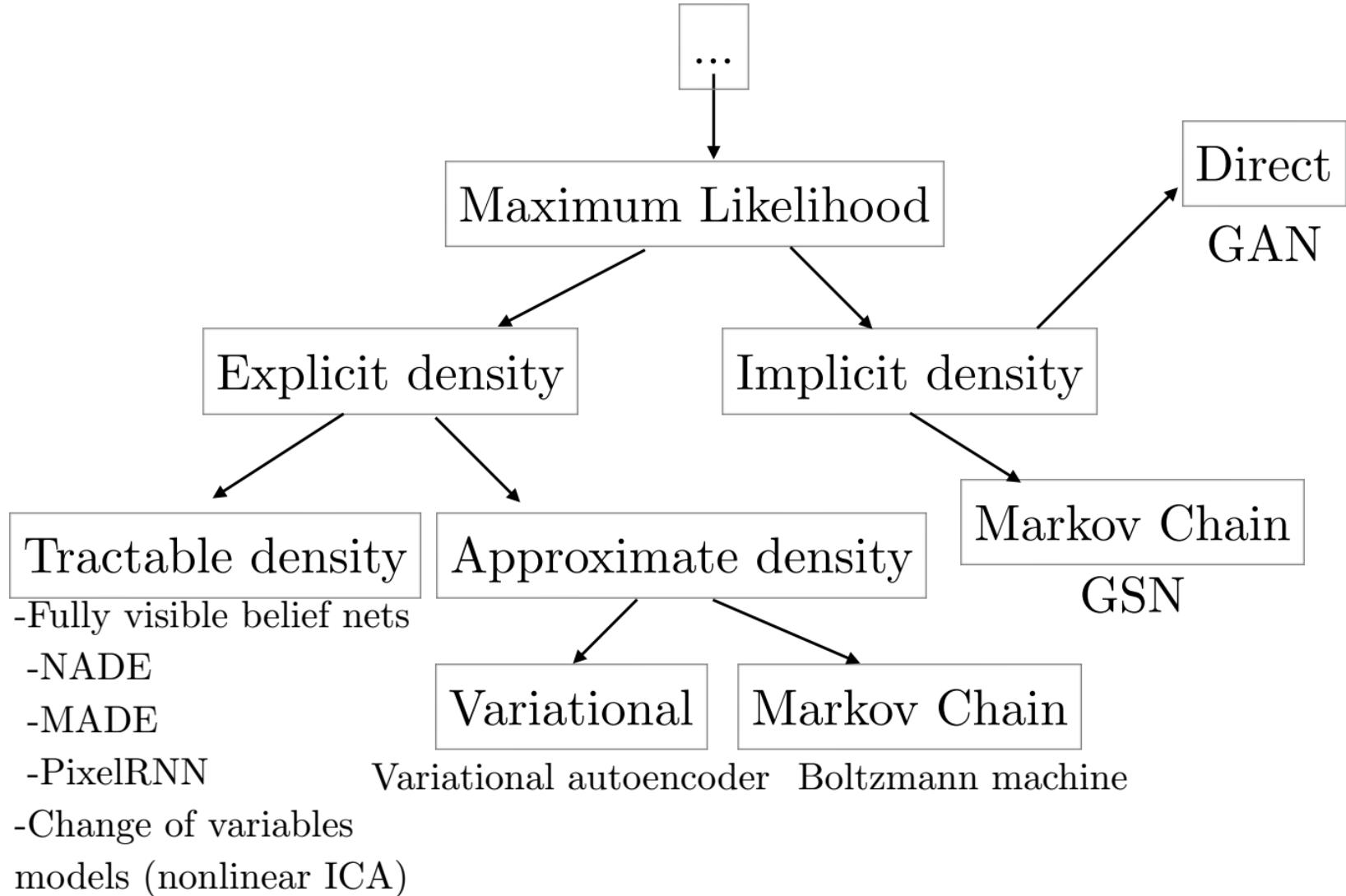
# Applications: Super-resolution



# Applications: Cross-model translation



# A map of generative models

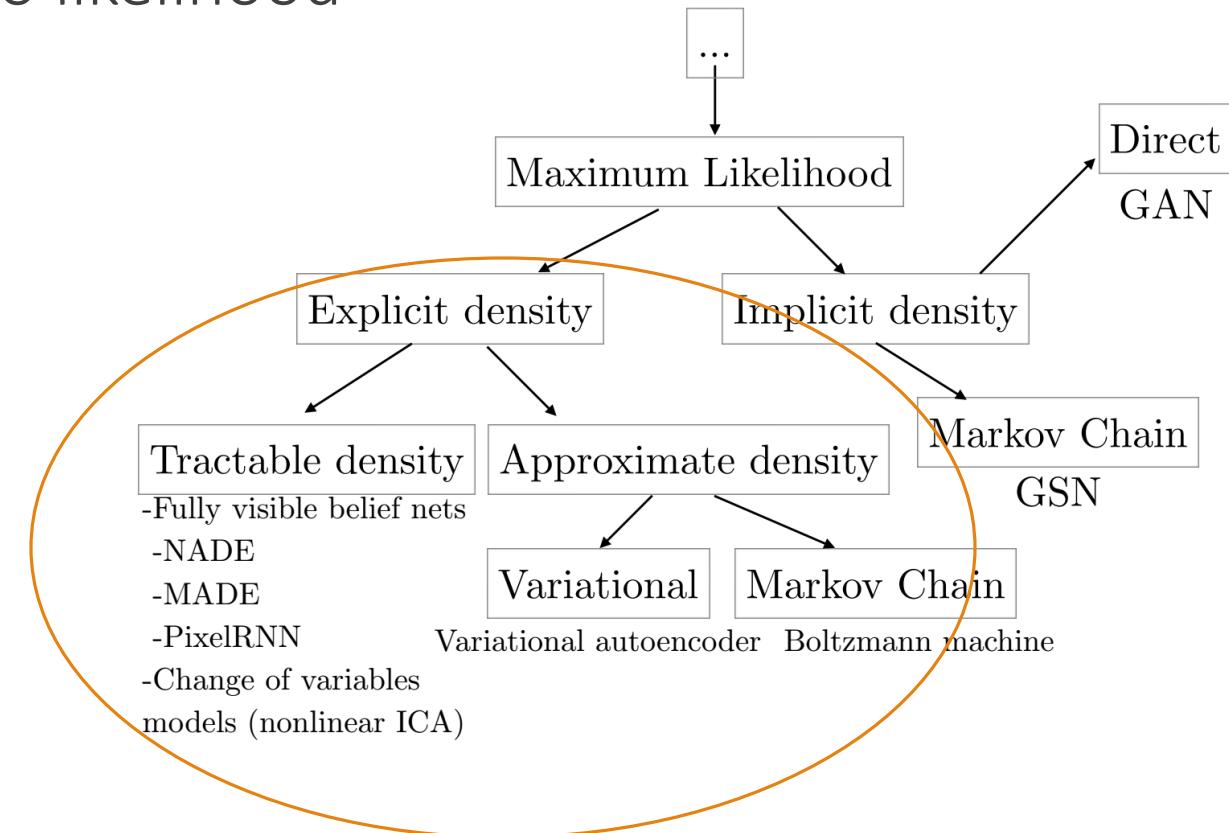


# Explicit density models

- Plug in the model density function to likelihood
- Then maximize the likelihood

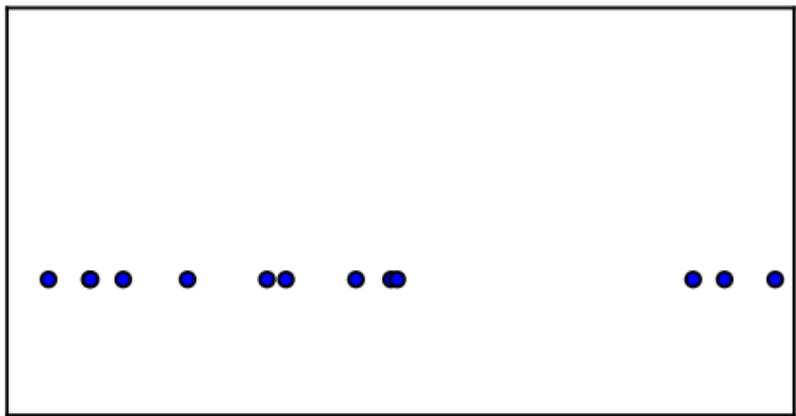
## Problems

- Modes must be complex enough → to match data complexity
- Also, model must be computationally tractable
- More details in the next lectures

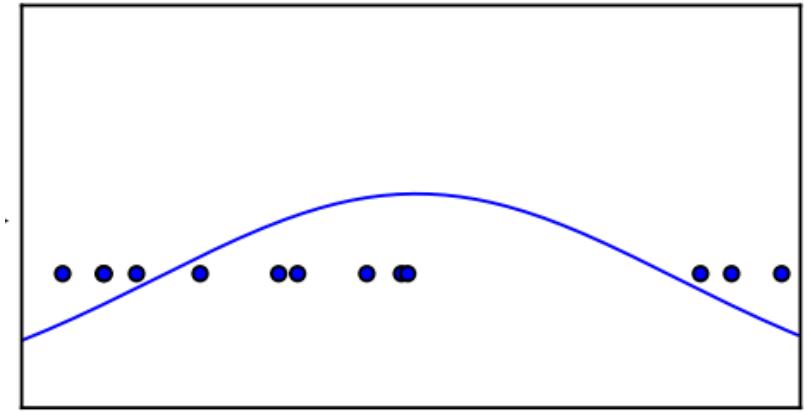


# Generative modeling: Case I

- Density estimation



Train set

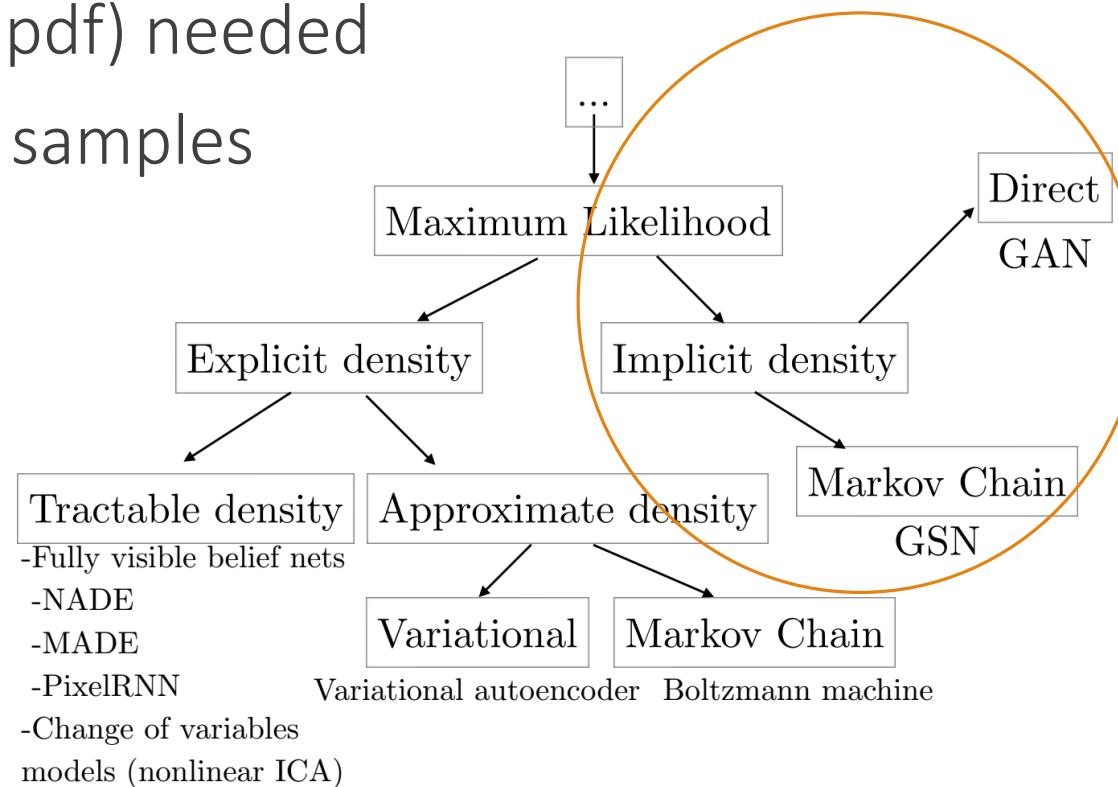


Fitted model



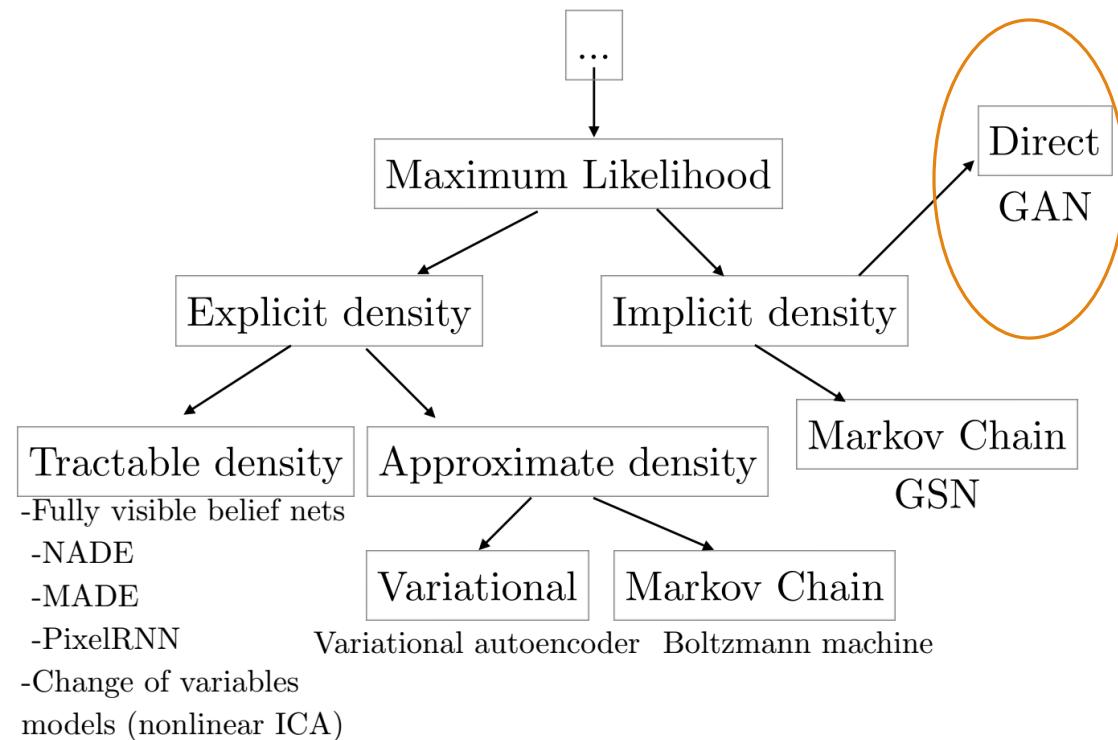
# Implicit density models

- No explicit probability density function (pdf) needed
- Instead, a sampling mechanism to draw samples from the pdf without knowing the pdf



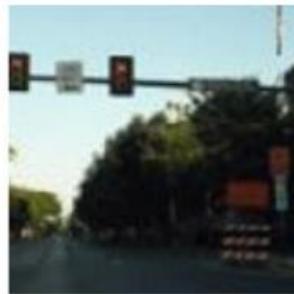
# Implicit density models: GANs

- Sample data in parallel
- Few restrictions on generator model
- No Markov Chains needed
- No variational bounds
- Better qualitative examples
  - Weak but true



# Generative modeling: Case II

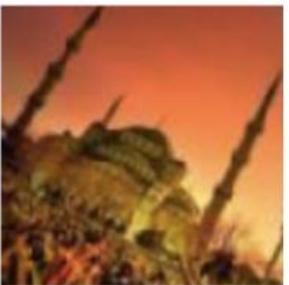
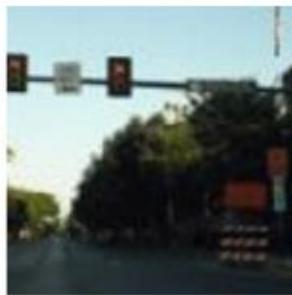
- Sample Generation



Train examples

# Generative modeling: Case II

- Sample Generation



Train examples

New samples (ideally)

# What is a GAN?

---

- **G**enerative

- You can sample novel input samples
- E.g., you can literally “create” images that never existed

- **A**dversarial

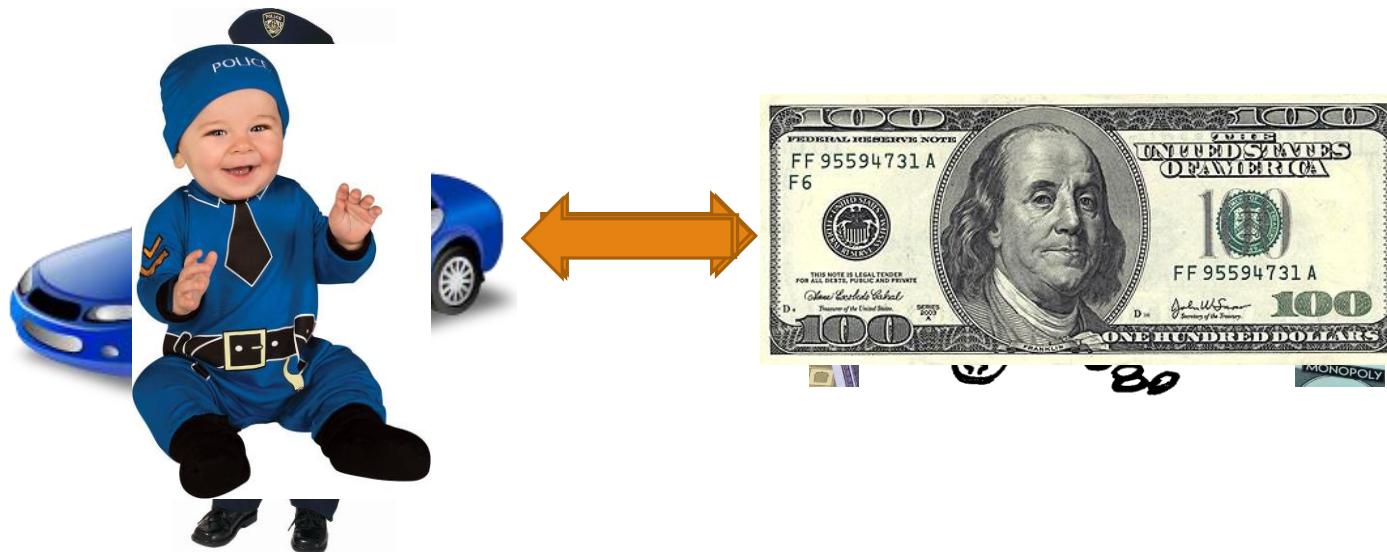
- Our generative model  $G$  learns adversarially, by fooling an discriminative oracle model  $D$

- **N**etwork

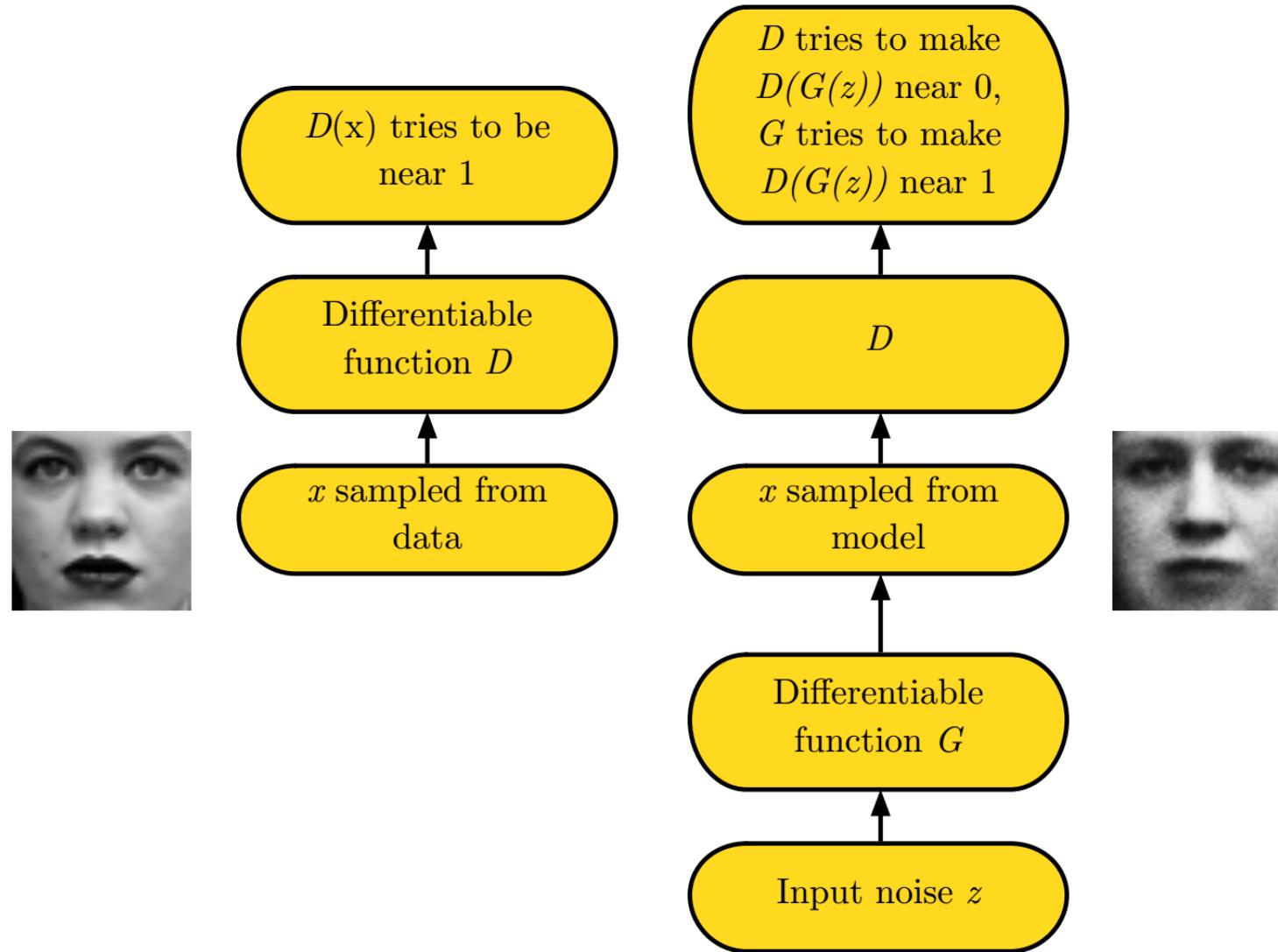
- Implemented typically as a (deep) neural network
- Easy to incorporate new modules
- Easy to learn via backpropagation

# GAN: Intuition

- Assume you have two parties
  - Police: wants to recognize fake money as reliably as possible
  - Counterfeiter: wants to make as realistic fake money as possible
- The police forces the counterfeiter to get better (and vice versa)
- Solution relates to Nash equilibrium



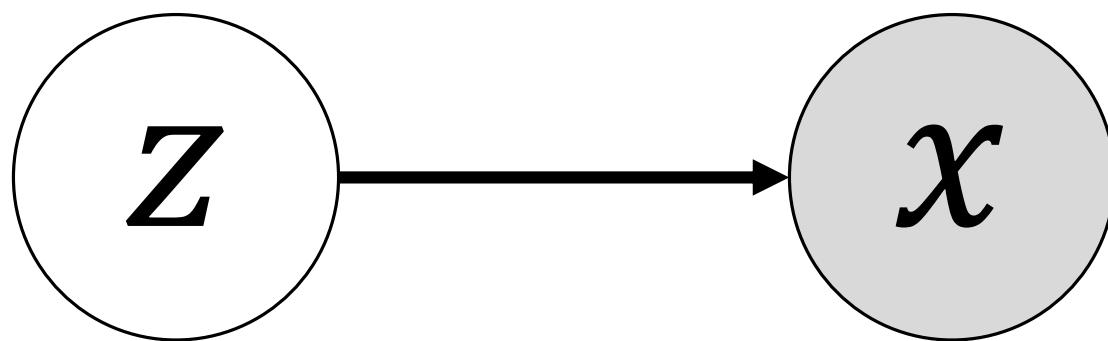
# GAN: Pipeline



# Generator network $x = G(z; \theta^{(G)})$

---

- Must be differentiable
- No invertibility requirement
- Trainable for any size of  $z$
- Can make conditionally Gaussian given  $z$ , but no strict requirement



# Generator & Discriminator: Implementation

- The discriminator is just a standard neural network
- The generator looks like an inverse discriminator

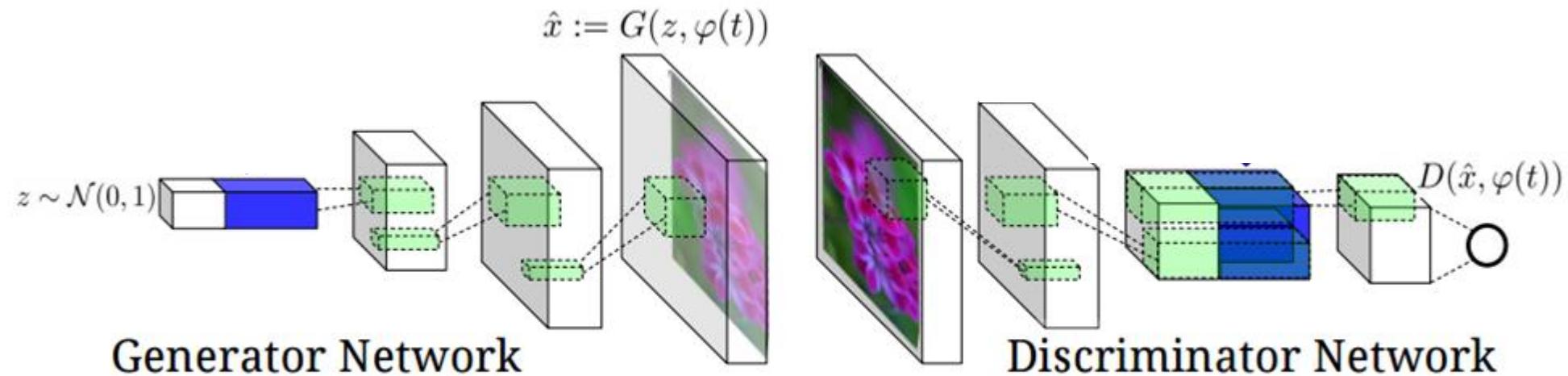


Figure 2. Our text-conditional convolutional GAN architecture. Text encoding  $\varphi(t)$  is used by both generator and discriminator. It is projected to a lower-dimensions and depth concatenated with image feature maps for further stages of convolutional processing.

[Network Architecture](#)

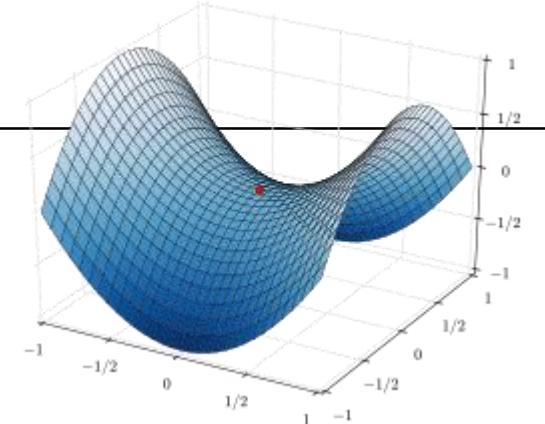
# Training definitions

---

- Minimax
- Maximin
- Heuristic, non-saturating game
- Max likelihood game

# Minimax Game

$$\circ J^{(D)} = -\frac{1}{2} \mathbb{E}_{x \sim p_{data}} \log D(x) - \frac{1}{2} \mathbb{E}_{z \sim p_z} \log(1 - D(G(z)))$$



- $D(x) = 1 \rightarrow$  The discriminator believes that  $x$  is a true image
- $D(G(z)) = 1 \rightarrow$  The discriminator believes that  $G(z)$  is a true image
  
- Equilibrium is a saddle point of the discriminator loss
- Resembles Jensen-Shannon divergence
- Generator minimizes the log-probability of the discriminator being correct

[NIPS 2016 Tutorial: Generative Adversarial Networks](#)

# A reasonable loss for the generator?

---

# Minimax Game

---

- For the simple case of zero-sum game

$$J^{(G)} = -J^{(D)}$$

- So, we can summarize game by

$$V(\theta^{(D)}, \theta^{(G)}) = -J^{(D)}(\theta^{(D)}, \theta^{(G)})$$

- Easier theoretical analysis

- In practice not used → when the discriminator starts to recognize fake samples, then ...

# Minimax Game

---

- For the simple case of zero-sum game

$$J^{(G)} = -J^{(D)}$$

- So, we can summarize game by

$$V(\theta^{(D)}, \theta^{(G)}) = -J^{(D)}(\theta^{(D)}, \theta^{(G)})$$

- Easier theoretical analysis

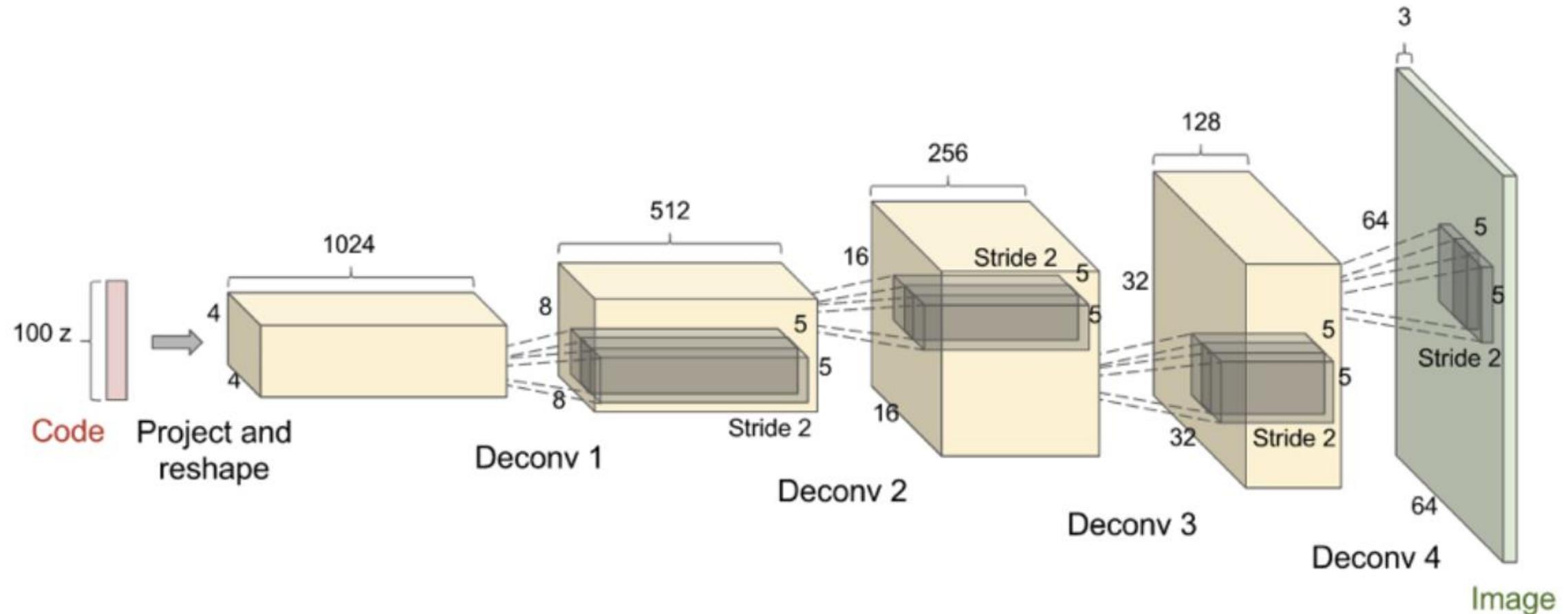
- In practice not used → when the discriminator starts to recognize fake samples, the generator gradients vanish

# Heuristic non-saturating game

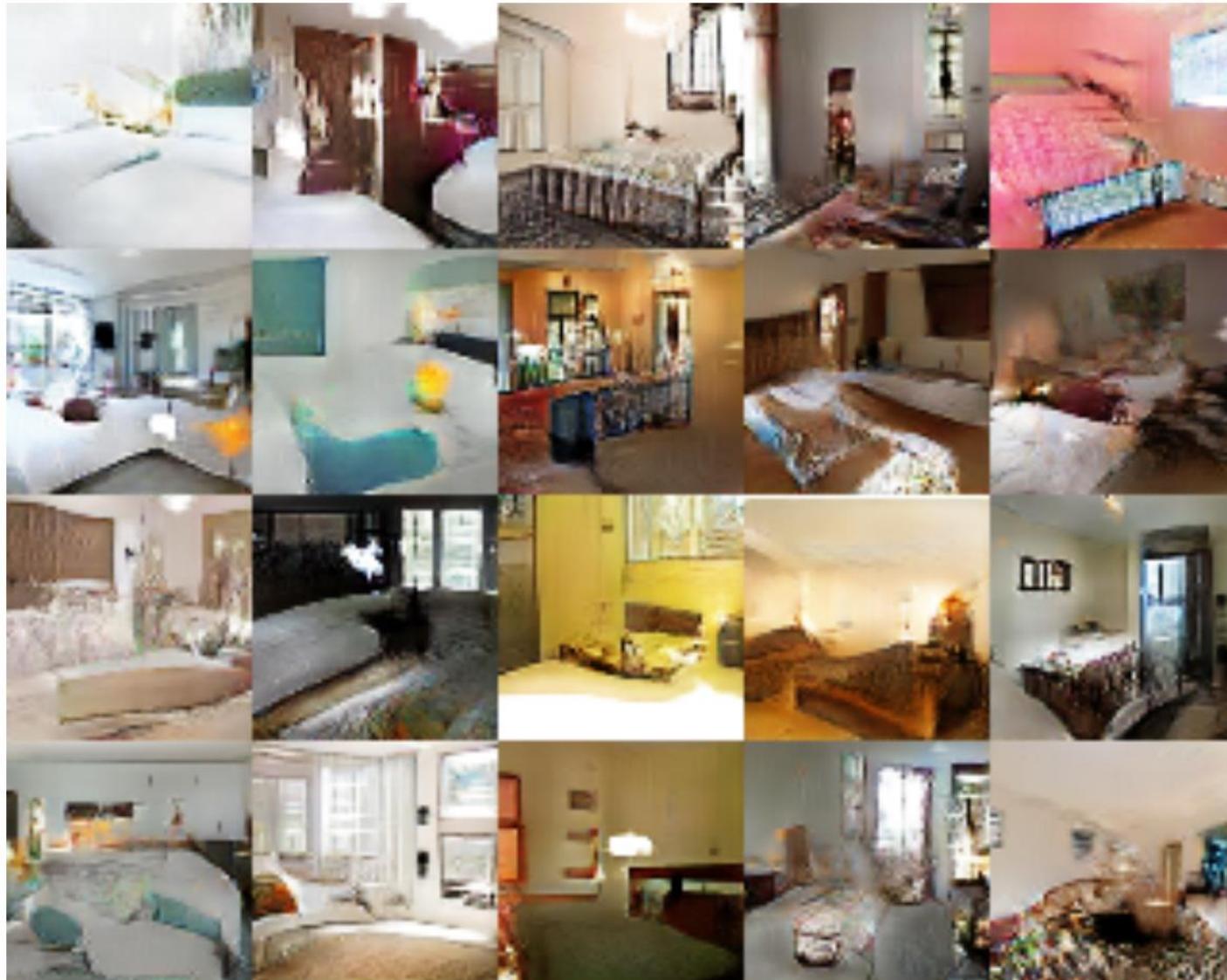
---

- $J^{(D)} = -\frac{1}{2} \mathbb{E}_{x \sim p_{data}} \log D(x) - \frac{1}{2} \mathbb{E}_{z \sim p_z} \log(1 - D(G(z)))$
- $J^{(G)} = -\frac{1}{2} \mathbb{E}_{z \sim p_z} \log(D(G(z)))$
  
- Equilibrium not any more describable by single loss
- Generator maximizes the log-probability of the discriminator being mistaken
  - Good  $G(z)$      $\rightarrow$      $D(G(z)) = 1$      $\rightarrow$      $J^{(G)}$  is maximized
- Heuristically motivated; generator can still learn even when discriminator successfully rejects all generator samples

# DCGAN Architecture



# Examples



# Even vector space arithmetics ...



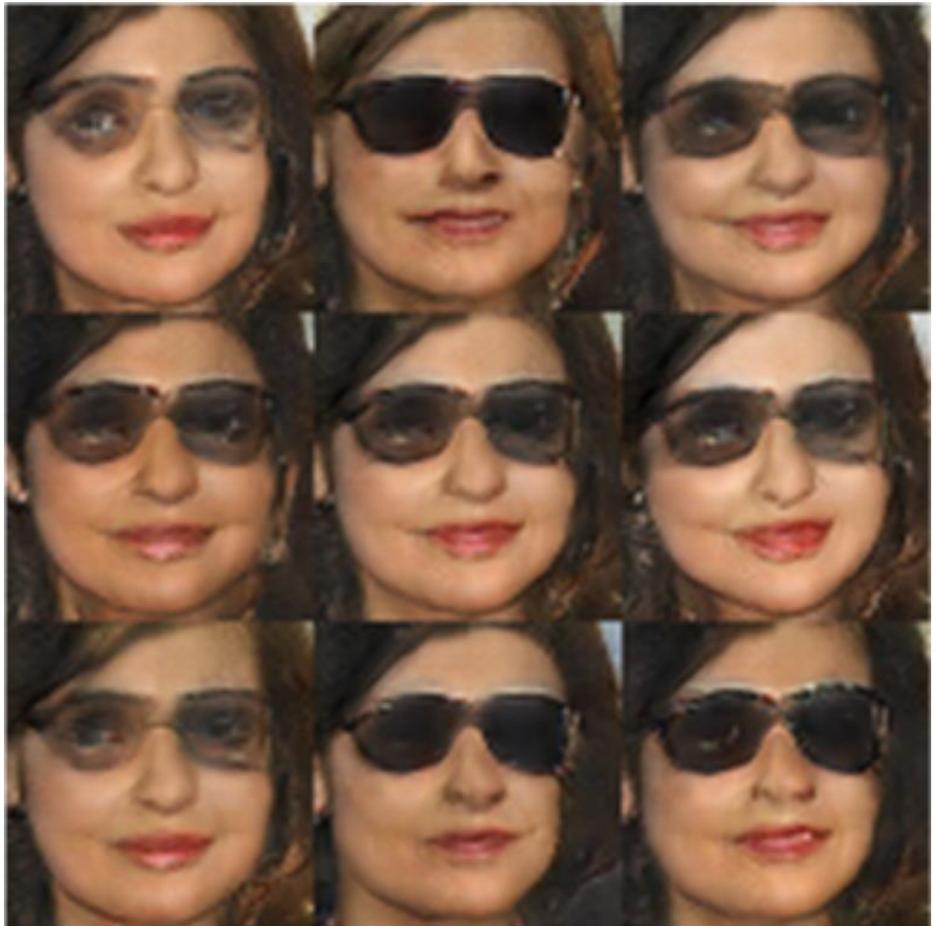
Man  
with  
glasses

Man

+



=



Woman with  
glasses

# Modifying GANs for Max-Likelihood

---

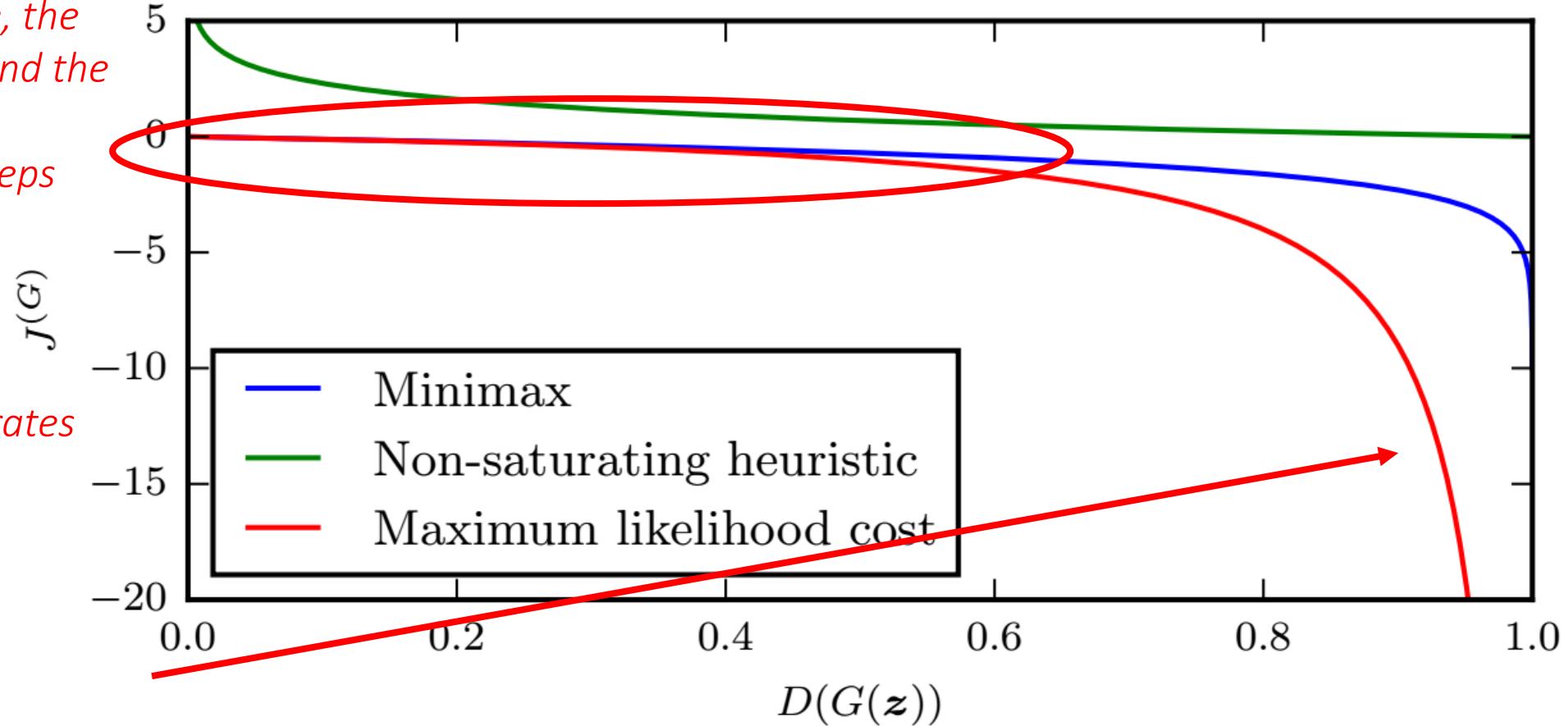
- $J^{(D)} = -\frac{1}{2} \mathbb{E}_{x \sim p_{data}} \log D(x) - \frac{1}{2} \mathbb{E}_z \log(1 - D(G(z)))$
- $J^{(G)} = -\frac{1}{2} \mathbb{E}_z \log(\sigma^{-1}(D(G(z))))$
- When discriminator is optimal, the generator gradient matches that of maximum likelihood

[On distinguishability criteria for estimating generative models](#)

# Comparison of Generator Losses

*When sample is likely fake, the non-saturating heuristic and the ML cost are flat  
→ no gradients in early steps*

*The ML cost variant generates gradients mostly from the “good generations”  
→ all gradients from few samples  
→ high variance  
→ Variance reduction?*

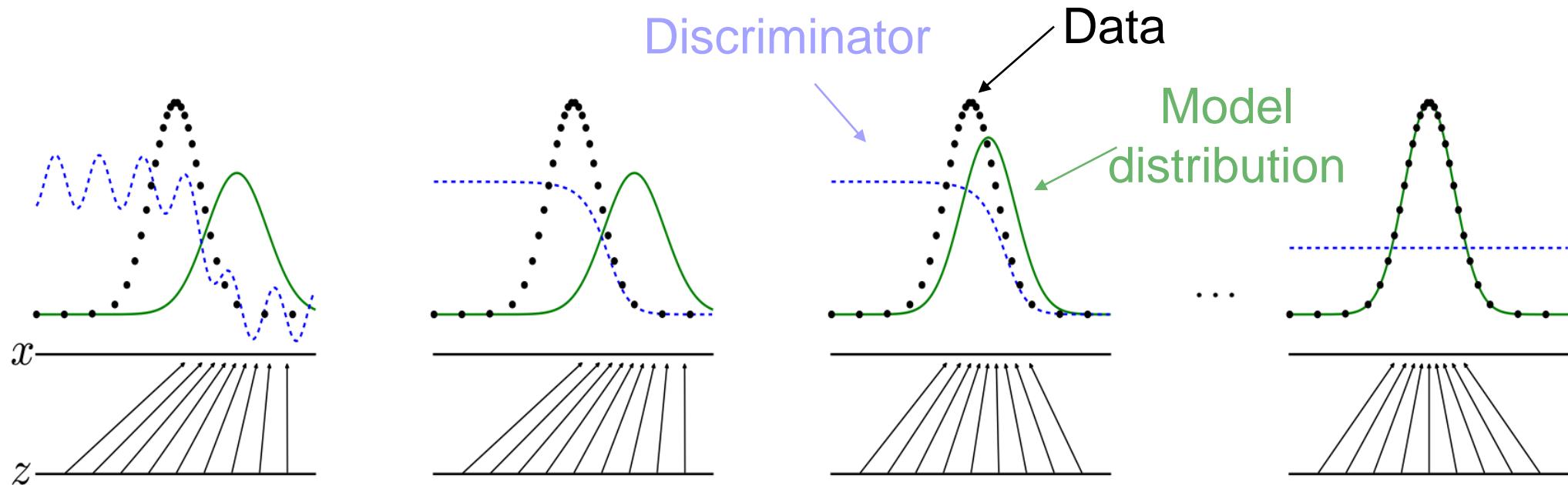


# Optimal discriminator

- Optimal  $D(x)$  for any  $p_{data}(x)$  and  $p_{model}(x)$  is always

$$D(x) = \frac{p_{data}(x)}{p_{data}(x) + p_{model}(x)}$$

- Estimating this ratio with supervised learning (discriminator) is the key



# Why is this the optimal discriminator?

---

- $L(D, G) = \int_x p_r(x) \log D(x) + p_g(x) \log(1 - D(x)) dx$ 
  - Minimize  $L(D, G)$  w.r.t.  $D \rightarrow \frac{dL}{dD} = 0$  and ignore the integral (we sample over all  $x$ )
  - The function  $x \rightarrow a \log x + b \log(1 - x)$  attains max in  $[0, 1]$  at  $\frac{a}{a+b}$
- The optimal discriminator

$$D^*(x) = \frac{p_r(x)}{p_r(x) + p_g(x)}$$

- And at optimality  $p_g(x) \rightarrow p_r(x)$ , thus

$$\begin{aligned} D^*(x) &= \frac{1}{2} \\ L(G^*, D^*) &= -2 \log 2 \end{aligned}$$

# GANs and Jensen-Shannon divergence

---

- By expanding the Jensen-Shannon divergence, we have

$$\begin{aligned} D_{JS}(p_r || p_g) &= \frac{1}{2} D_{KL}(p_r || \frac{p_r + p_g}{2}) + \frac{1}{2} D_{KL}(p_g || \frac{p_r + p_g}{2}) \\ &= \frac{1}{2} \left( \log 2 + \int_x p_r(x) \log \frac{p_r(x)}{p_r(x) + p_g(x)} dx + \log 2 \right) \end{aligned}$$

# GANs and Jensen-Shannon divergence

---

- By expanding the Jensen-Shannon divergence, we have

$$\begin{aligned} D_{JS}(p_r \parallel p_g) &= \frac{1}{2} D_{KL}(p_r \parallel \frac{p_r + p_g}{2}) + \frac{1}{2} D_{KL}(p_g \parallel \frac{p_r + p_g}{2}) \\ &= \frac{1}{2} \left( \log 2 + \int_x p_r(x) \log \frac{p_r(x)}{p_r(x) + p_g(x)} dx + \log 2 \right) \end{aligned}$$

<https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html>

# Is the divergence important?

---

- Does the divergence make a difference?
- Is there a difference between KL-divergence, Jensen-Shannon divergence, ...

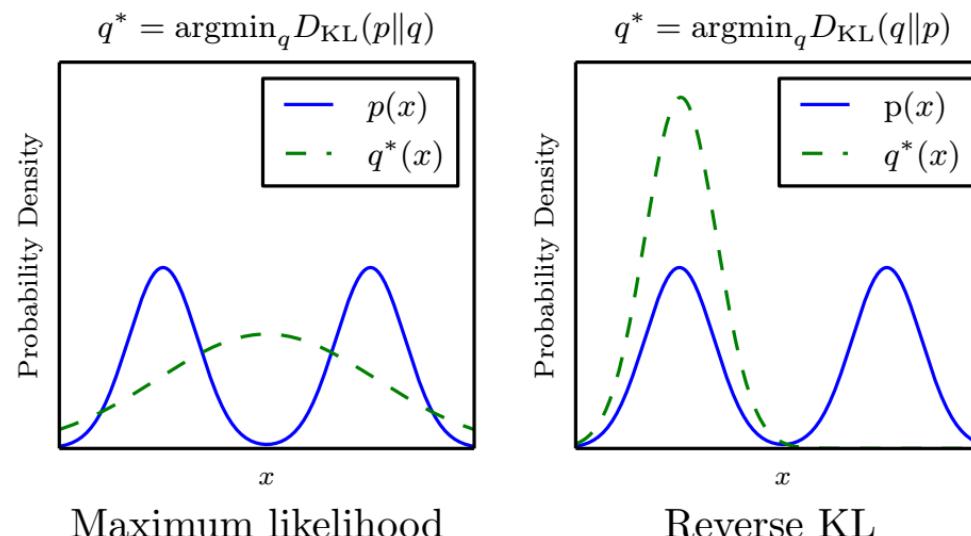
$$D_{KL}(p_r || p_g) = \int_x p_r \log \frac{p_r}{p_g} dx$$

$$D_{JS}(p_r || p_g) = \frac{1}{2} D_{KL}(p_r || \frac{p_r + p_g}{2}) + \frac{1}{2} D_{KL}(p_g || \frac{p_r + p_g}{2})$$

- Let's check the KL-divergence

# Is the divergence important?

- Forward KL divergence:  $D_{KL}(p(x)||q^*(x)) \rightarrow$  high probability everywhere that the data occurs
- Backward KL divergence:  $D_{KL}(q^*(x)||p(x)) \rightarrow$  low probability wherever the data does not occur
- Which version makes the model “conservative”?

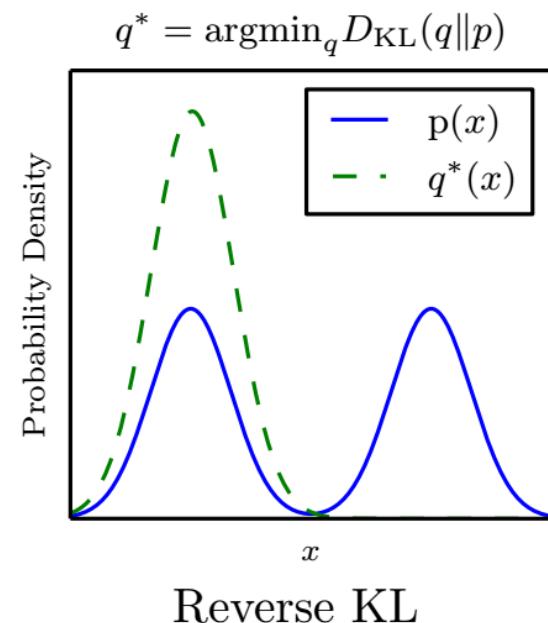
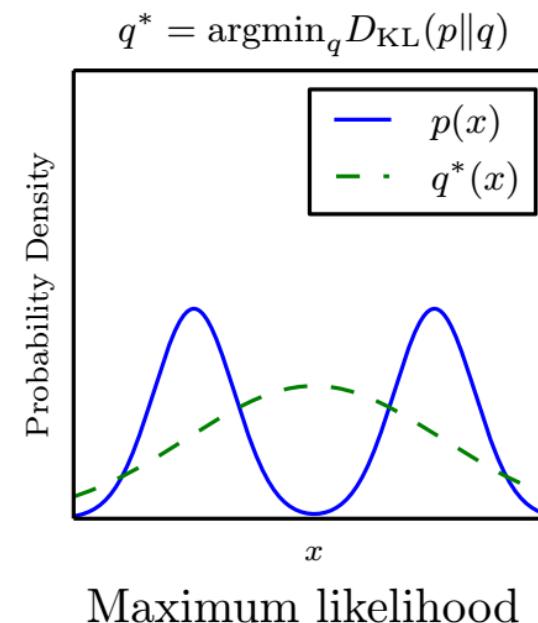


$p_r$  is what we get and cannot change  
 $p_g$  is what we make through our model and (through training) change

$$D_{KL}(p_r||p_g) = \int_x p_r \log \frac{p_r}{p_g} dx$$

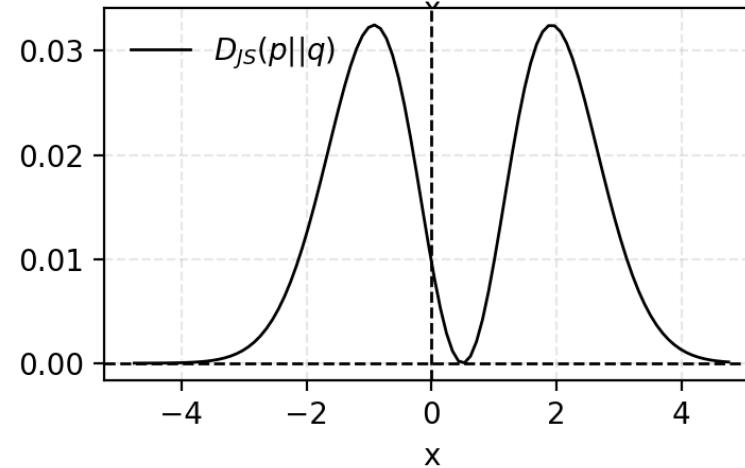
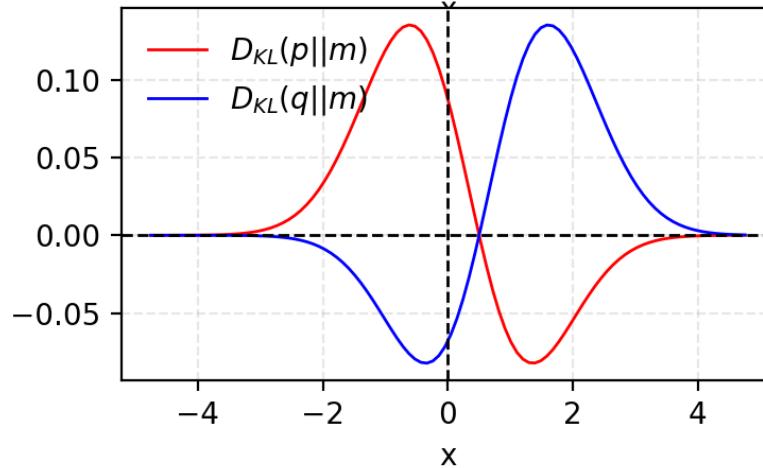
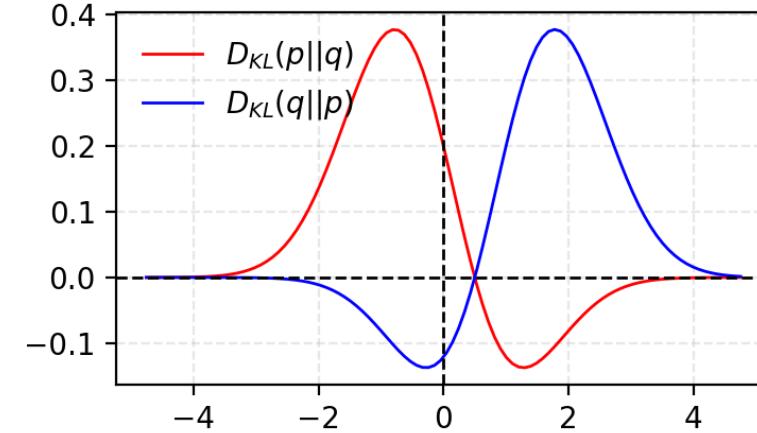
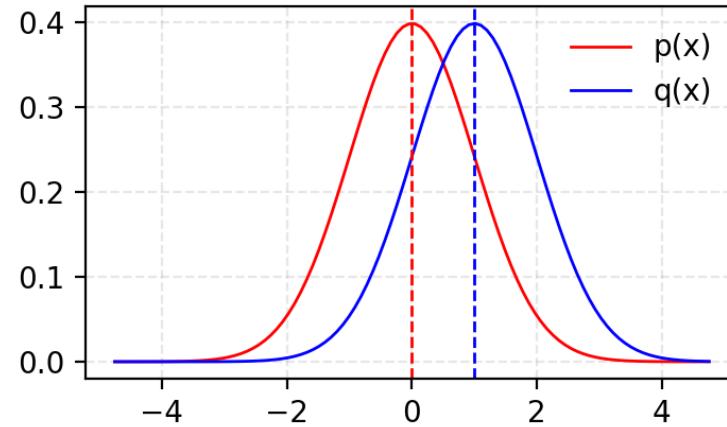
# Is the divergence important?

- $D_{KL}(p(x)||q^*(x)) \rightarrow$  high probability everywhere that the data occurs
- $D_{KL}(q^*(x)||p(x)) \rightarrow$  low probability wherever the data does not occur
- Which version makes the model “conservative”?
- $D_{KL}(q^*(x)||p(x)) = \int q^*(x) \log \frac{q^*(x)}{p(x)}$ 
  - Avoid areas where  $p(x) \rightarrow 0$
- Zero-forcing
  - $q^*(x) \rightarrow 0$  in areas when approximation  $\frac{q^*(x)}{p(x)}$  cannot be good



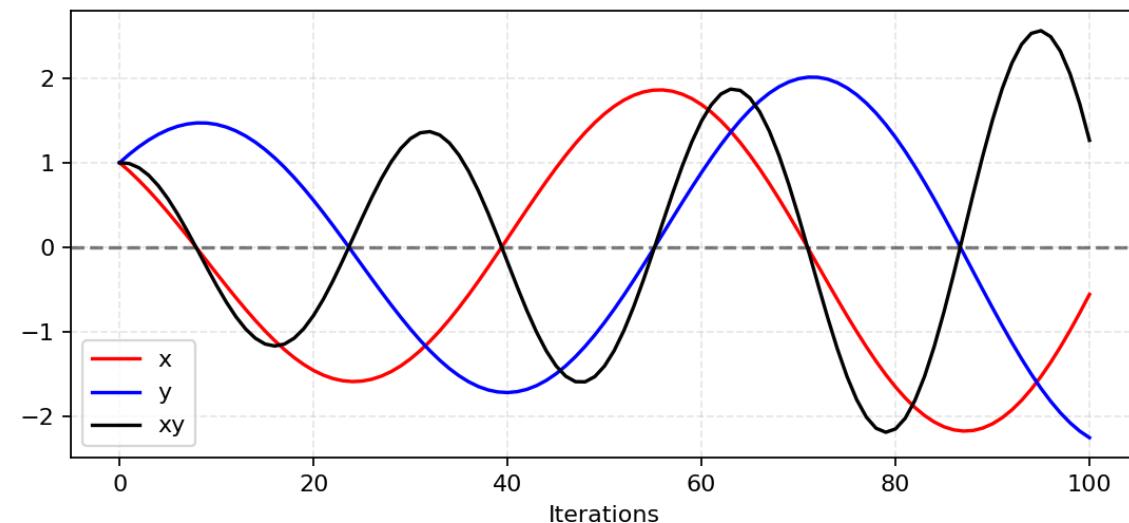
# KL vs JS

○ JS is symmetric, KL is not



# GAN Problems: Reaching Nash equilibrium causes instabilities

- GANs is a mini-max optimization
  - Non-cooperative game with a tied objective
- Training is not always easy
  - When optimizing one player/network, we might hurt the other one
  - oscillations
- Assume two players  $f(x) = xy$   
We optimize one step at a time
  - Player 1 minimizes:  $\min_x f_1(x) = xy \Rightarrow \frac{df_1}{dx} = y$   
 $\Rightarrow x_{t+1} = x_t - \eta \cdot y$
  - Player 2 minimizes:  $\min_y f_2(x) = -xy \Rightarrow \frac{df_2}{dx} = -x$   
 $\Rightarrow y_{t+1} = y_t + \eta \cdot x$

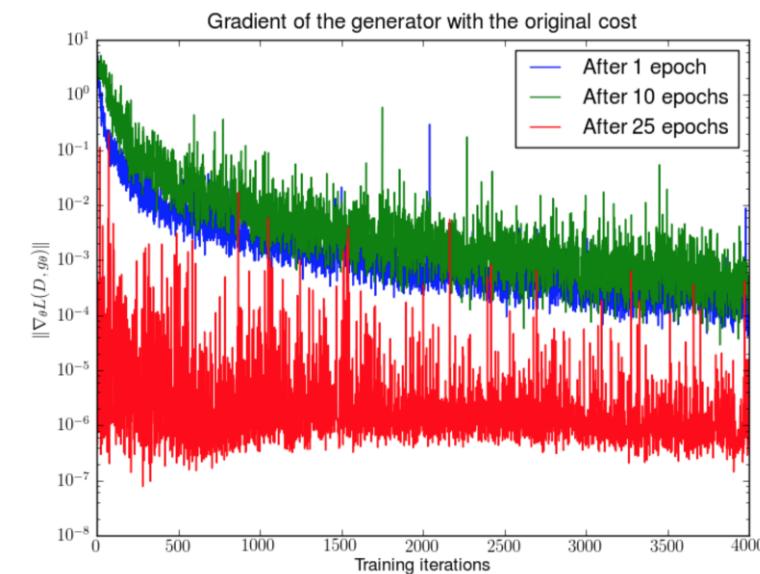


<https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html>

# GAN Problems: Vanishing Gradients

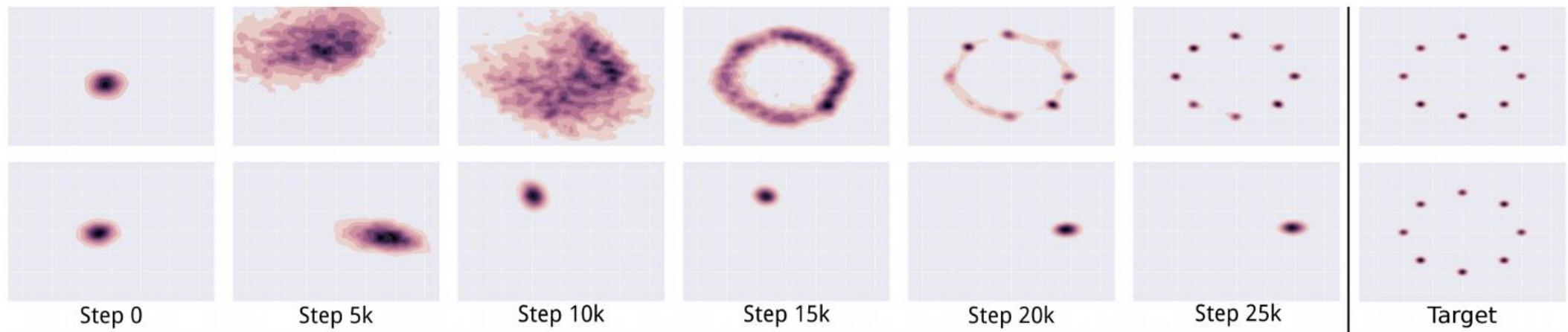
$$J^{(D)} = -\frac{1}{2} \mathbb{E}_{x \sim p_{data}} \log D(x) - \frac{1}{2} \mathbb{E}_z \log(1 - D(G(z)))$$
$$J^{(G)} = -\frac{1}{2} \mathbb{E}_z \log(D(G(z)))$$

- If the discriminator is quite bad
  - no accurate feedback for generator
  - no reasonable generator gradients
- But, if the discriminator is perfect,  $D(x) = D^*(x)$ 
  - gradients go to 0
  - no learning anymore
- Bad when this happens early in the training
  - Easier to train the discriminator than the generator



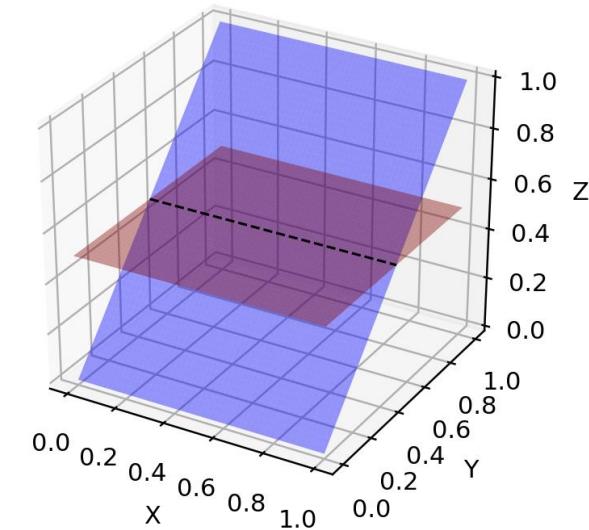
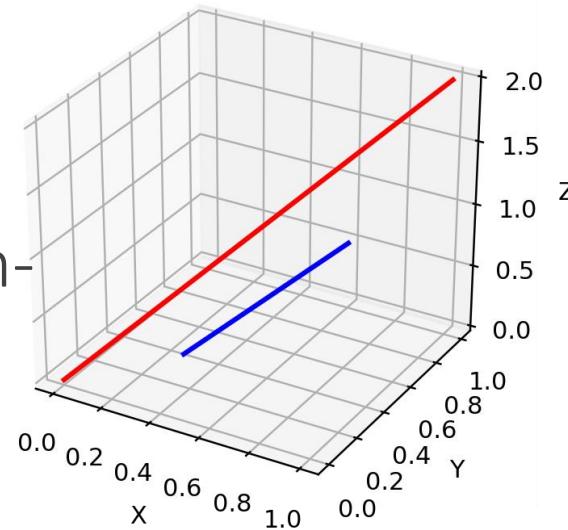
# GAN Problems: Mode collapse

- Very low variability
- It is safer for the generator to produce samples from the mode it knows it approximates well



# GAN Problems: Low dimensional supports

- Data lie in low-dim manifolds
- However, the manifold is not known
- During training  $p_g$  is not perfect either, especially in the start
- So, the support of  $p_r$  and  $p_g$  is non-overlapping and disjoint  
→ not good for KL/JSD divergences
- Easy to find a discriminating line

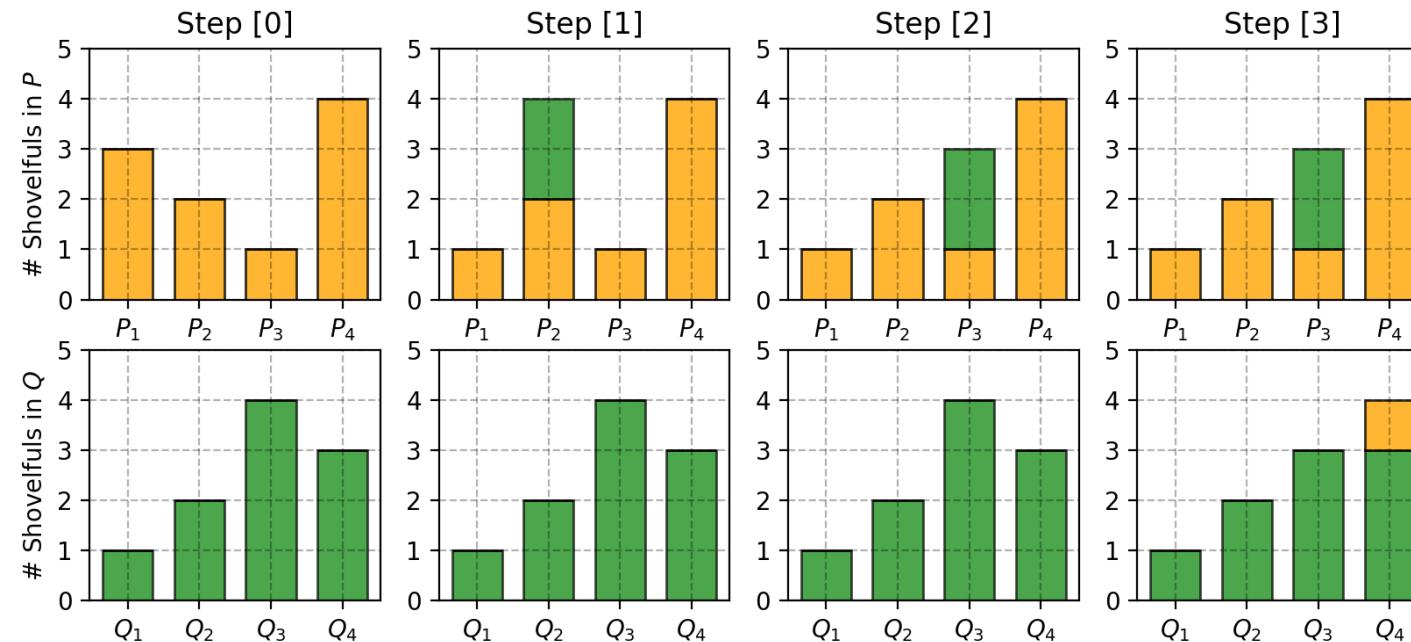


# Wasserstein GAN

- Instead of KL/JSD, use Wasserstein (Earth Mover's) Distance

$$W(p_r, p_g) = \inf_{\gamma \sim \Pi(p_r, p_g)} \mathbb{E}_{(x,y) \sim \gamma} |x - y|$$

- Even for non-overlapping supports, the distance is meaningful



# Feature matching

---

- Instead of matching image statistics, match feature statistics

$$J^{(D)} = \left\| \mathbb{E}_{x \sim p_r} f(x) - \mathbb{E}_{z \sim p_z} f(G(z)) \right\|_2^2$$

- $f$  can be any statistic of the data, like the mean or the median

# Training procedure

---

- Use SGD-like algorithm of choice
  - Adam Optimizer is a good choice
- Use two mini-batches simultaneously
  - The first mini-batch contains real examples from the training set
  - The second mini-batch contains fake generated examples from the generator
- Optional: run k-steps of one player (e.g. discriminator) for every step of the other player (e.g. generator)

# Use labels if possible

---

- Learning a conditional model  $p(y|x)$  is often generates better samples
  - Denton et al., 2015
- Even learning  $p(x,y)$  makes samples look more realistic
  - Salimans et al., 2016
- Conditional GANs are a great addition for learning with labels

# One-sided label smoothing

---

- Default discriminator cost:

```
cross_entropy(1., discriminator(data))  
+ cross_entropy(0., discriminator(samples))
```

- One-sided label smoothing:

```
cross_entropy(0.9, discriminator(data))  
+ cross_entropy(0., discriminator(samples))
```

- Do not smooth negative labels:

```
cross_entropy(1.-alpha, discriminator(data))  
+ cross_entropy(beta, discriminator(samples))
```

# Benefits of label smoothing

---

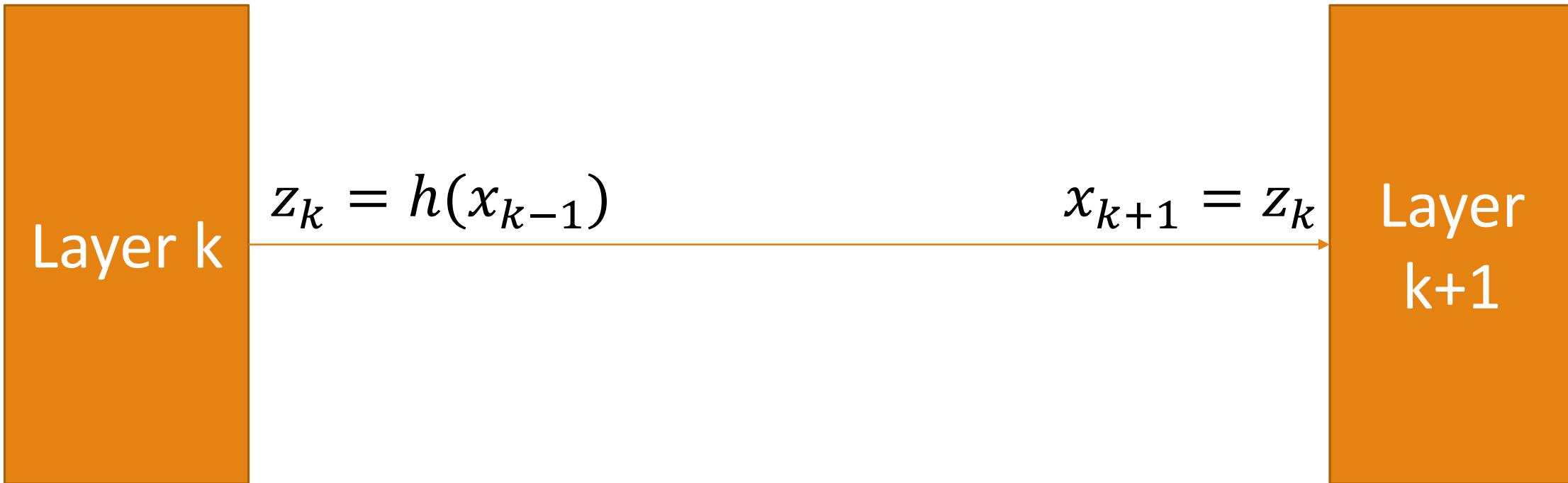
- Max likelihood often is overconfident
  - Might return accurate prediction, but too high probabilities
- Good regularizer
  - Szegedy et al., 2015
- Does not reduce classification accuracy, only confidence
- Specifically for GANs
  - Prevents discriminator from giving very large gradient signals to generator
  - Prevents extrapolating to encourage extreme samples

# Batch normalization

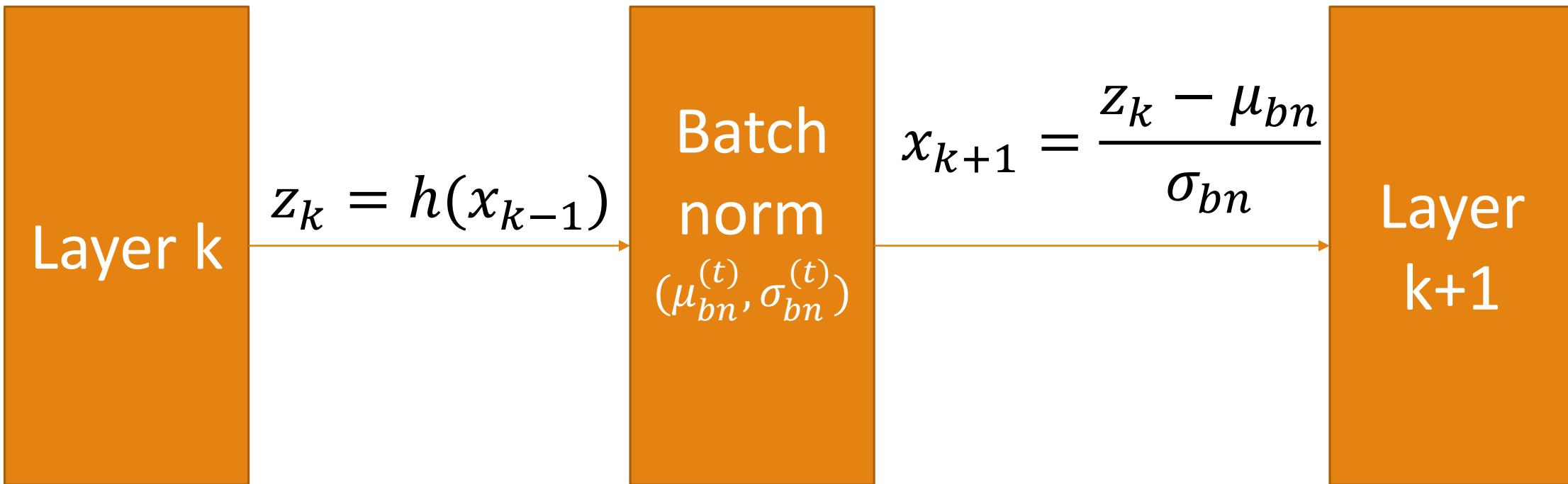
---

- Generally, good practice for neural networks
- Given inputs  $X = \{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$
- Compute mean and standard deviation of features of  $X$ :  $\mu_{bn}, \sigma_{bn}$
- Normalize features
  - Subtract mean, divide by standard deviation

# Batch normalization: Graphically



# Batch normalization: Graphically



# But, can cause strong intra-batch correlation



# Reference batch normalization

- Training with two mini-batches
- One fixed reference mini-batch for computing mean and standard deviation
- The other for doing the training as usual
- Proceed as normal, only use the mean and standard deviation for the batch norm from the fixed reference mini-batch
- Problem: Overfitting to the reference mini-batch

	Standard mini-batch	Reference mini-batch
Iteration 1	$\frac{dJ^{(1)}}{d\theta}$	$\mu_{bn}, \sigma_{bn}$
Iteration 2	$\frac{dJ^{(2)}}{d\theta}$	$\mu_{bn}, \sigma_{bn}$
Iteration 3	$\frac{dJ^{(3)}}{d\theta}$	$\mu_{bn}, \sigma_{bn}$

# Solution: Virtual batch normalization

- Mini-batch= standard mini-batch + reference/fixed mini-batch

	Standard mini-batch	Reference mini-batch
Iteration 1	$\frac{dJ^{(1)}}{d\theta}$	$\mu_{bn}^{(R)}, \sigma_{bn}^{(R)}$
Iteration 2	$\frac{dJ^{(2)}}{d\theta}$	$\mu_{bn}^{(R)}, \sigma_{bn}^{(R)}$
Iteration 3	$\frac{dJ^{(3)}}{d\theta}$	$\mu_{bn}^{(R)}, \sigma_{bn}^{(R)}$

# Balancing Generator & Discriminator

---

- Usually the discriminator wins
  - That's good, in that the theoretical justification assume a perfect discriminator
- Usually the discriminator network is bigger than the generator
- Sometimes running discriminator more often than generator works better
  - However, no real consensus
- Do not limit the discriminator to avoid making it too smart
  - Better use non-saturating cost
  - Better use label smoothing

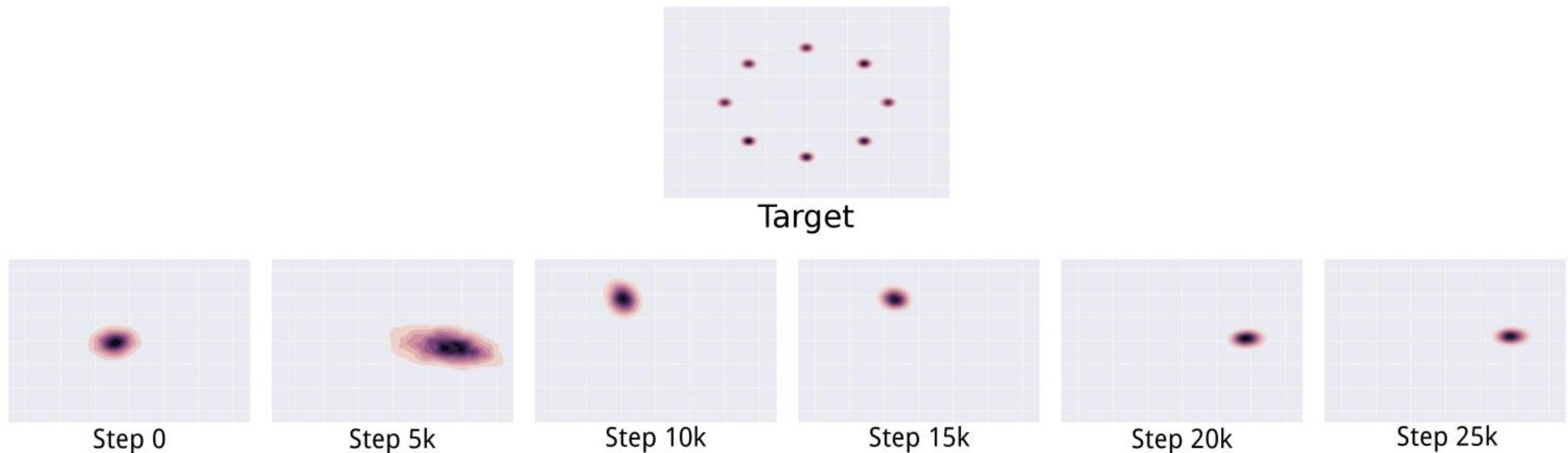
# Open Question: Non-convergence

---

- Optimization is tricky and unstable
  - finding a saddle point does not imply a global minimum
- An equilibrium might not even be reached
- Mode-collapse is the most severe form of non-convergence

# Open Question: Mode collapse

- Discriminator converges to the correct distribution
- Generator however places all mass in the most likely point



# Open Question: Mode collapse

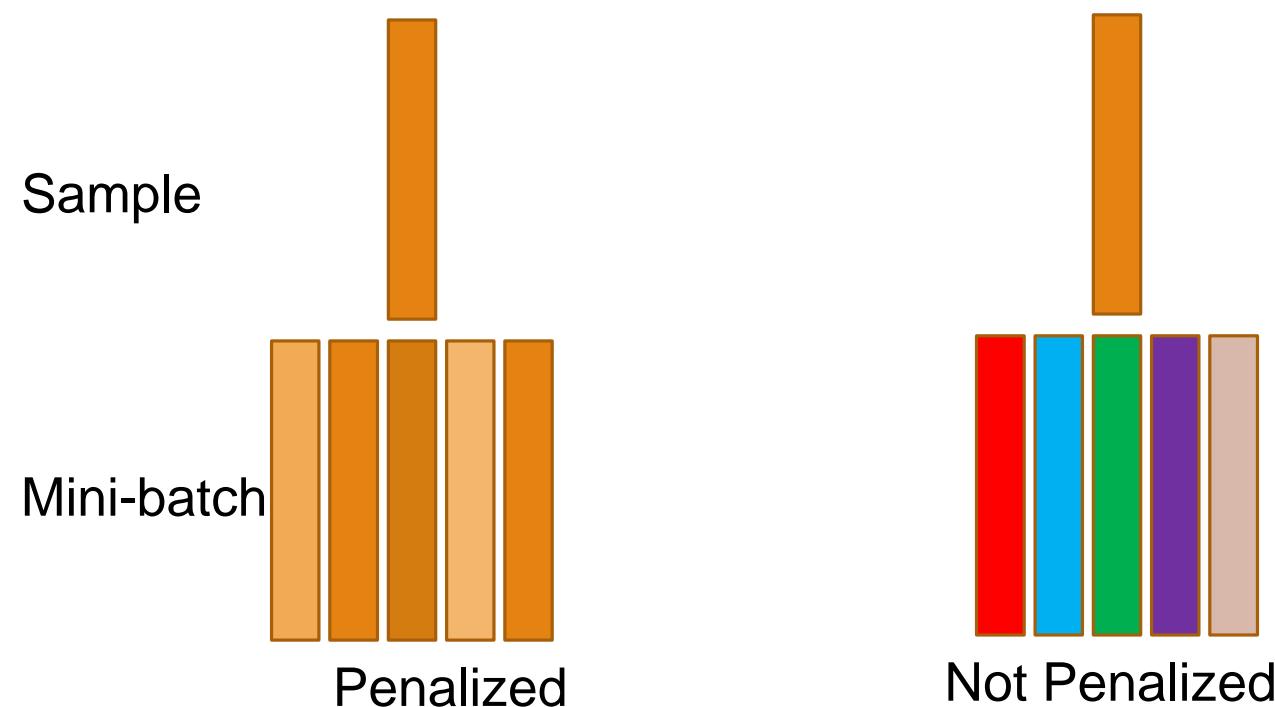
---

- Discriminator converges to the correct distribution
- Generator however places all mass in the most likely point
- Problem: low sample diversity



# Minibatch features

- Classify each sample by comparing to other examples in the mini-batch
- If samples are too similar, the model is penalized



# Open Question: Evaluation of GANs

---

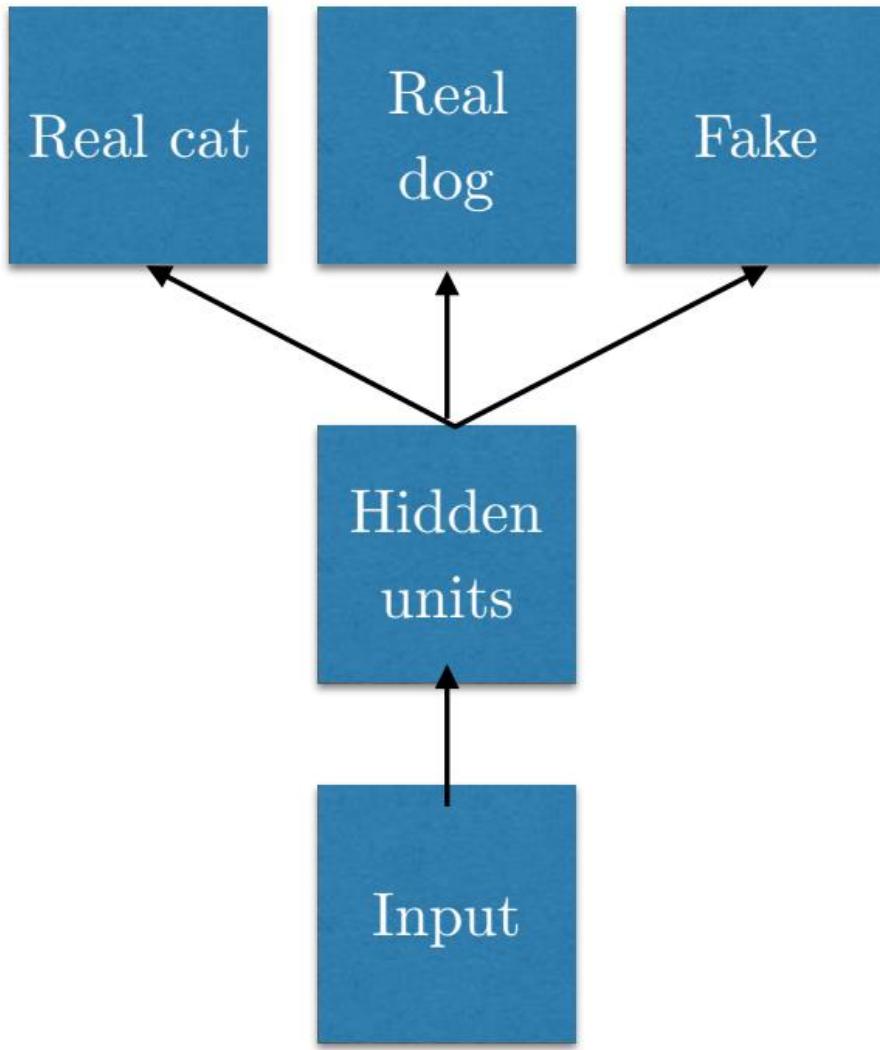
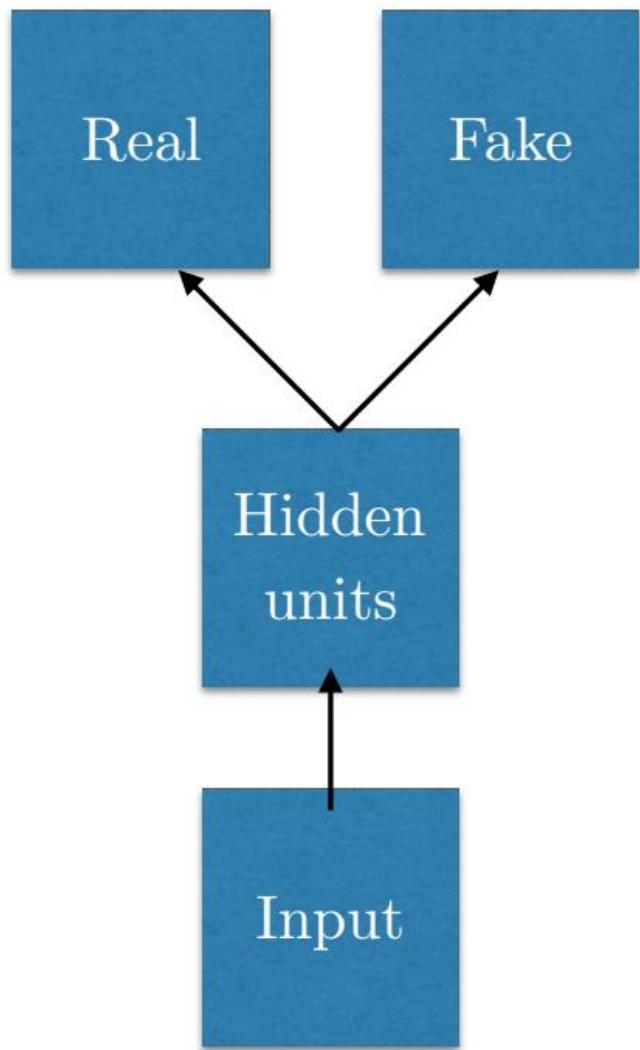
- Despite the nice images, who cares?
- It would be nice to quantitatively evaluate the model
- For GANs it is even hard to estimate the likelihood

# Open Question: Discrete outputs

---

- The generator must be differentiable
- It cannot be differentiable if outputs are discrete
- E.g., harder to make it work for text
- Possible workarounds
  - REINFORCE [Williams, 1992]
  - Concrete distribution [Maddison et al., 2016]
  - Gumbel softmax [Jang et al., 2016]
  - Train GAN to generate continuous embeddings

# Open Question: Semi-supervised classification



# Interpretable latent codes

- InfoGAN [Chen et al., 2016]



(a) Azimuth (pose)

(b) Elevation

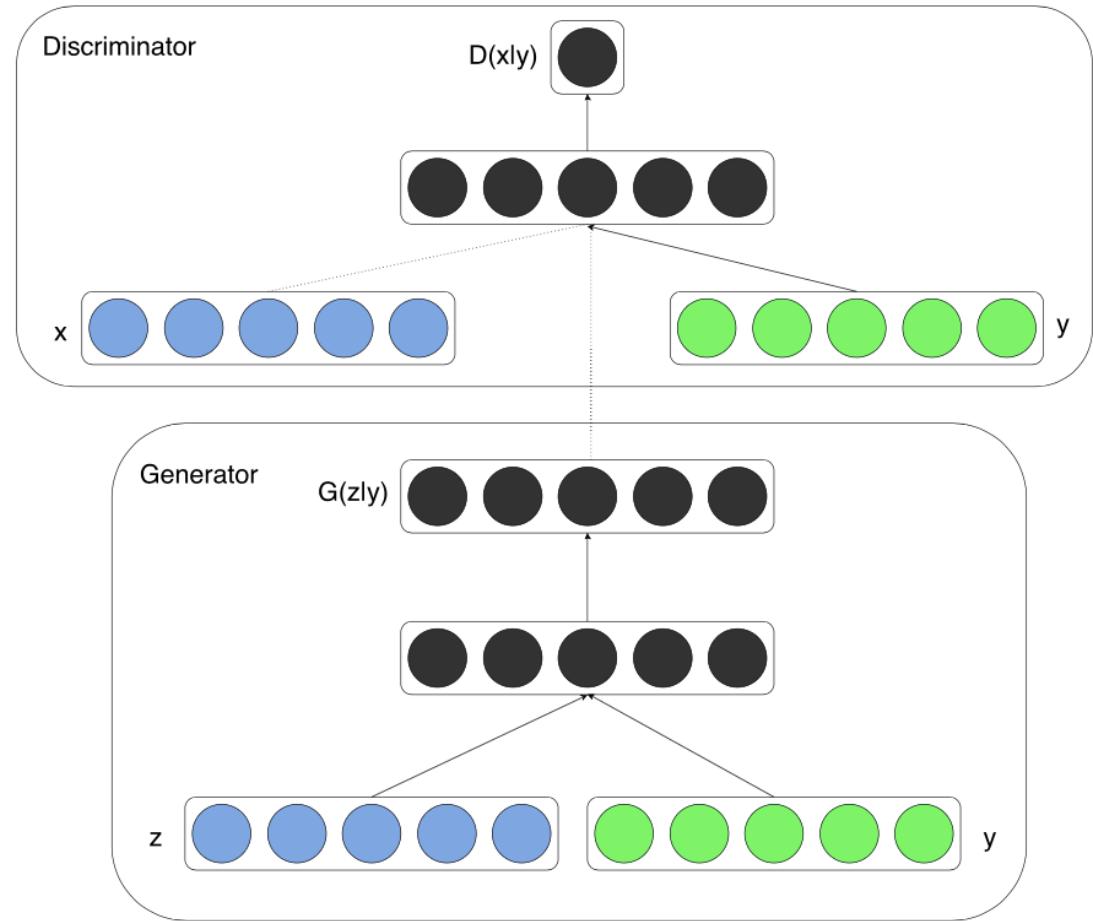


(c) Lighting

(d) Wide or Narrow

# GAN spinoffs

- Conditional GANs
  - Standard GANs have no encoder!
- Actor-Critic
  - Related to Reinforcement Learning



# Connections to Reinforcement Learning

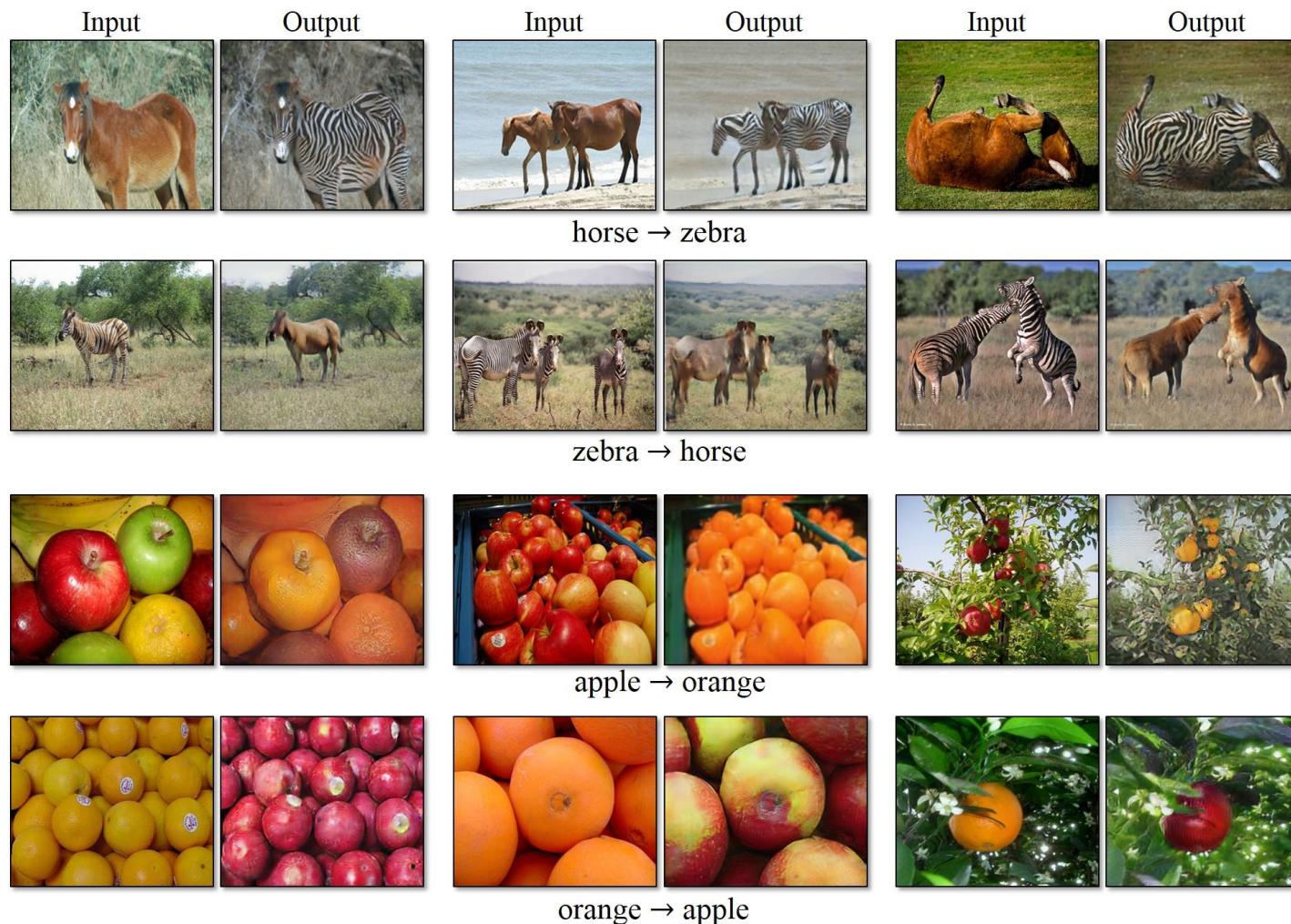
---

- GANs interpreted as actor-critic [Pfau and Vinyals, 2016]
- GANs as inverse reinforcement learning [Finn et al., 2016]
- GANs for imitation learning [Ho and Ermin 2016]

# Application: Image to Image translation



# Application: Style transfer



# Application: Face generation

---

- <https://www.youtube.com/watch?v=XOxxPcy5Gr4>

## Summary

- GANs are generative models using supervised learning to approximate an intractable cost function
- GANs can simulate many cost functions, including max likelihood
- Finding Nash equilibria in high-dimensional, continuous, non-convex games is an important open research problem
- GAN research is in its infancy, most works published only in 2016. Not mature enough yet, but very compelling results