

# Generative Adversarial Networks (GANs)

Ian Goodfellow, OpenAI Research Scientist

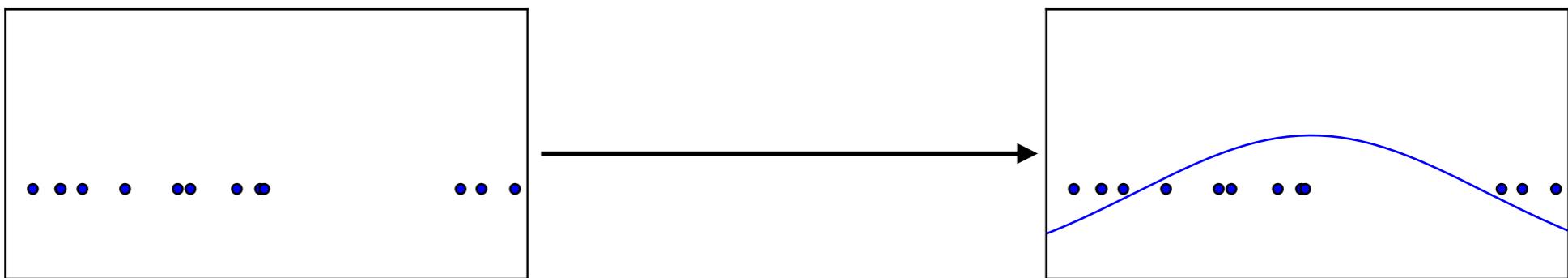
NIPS 2016 tutorial

Barcelona, 2016-12-4

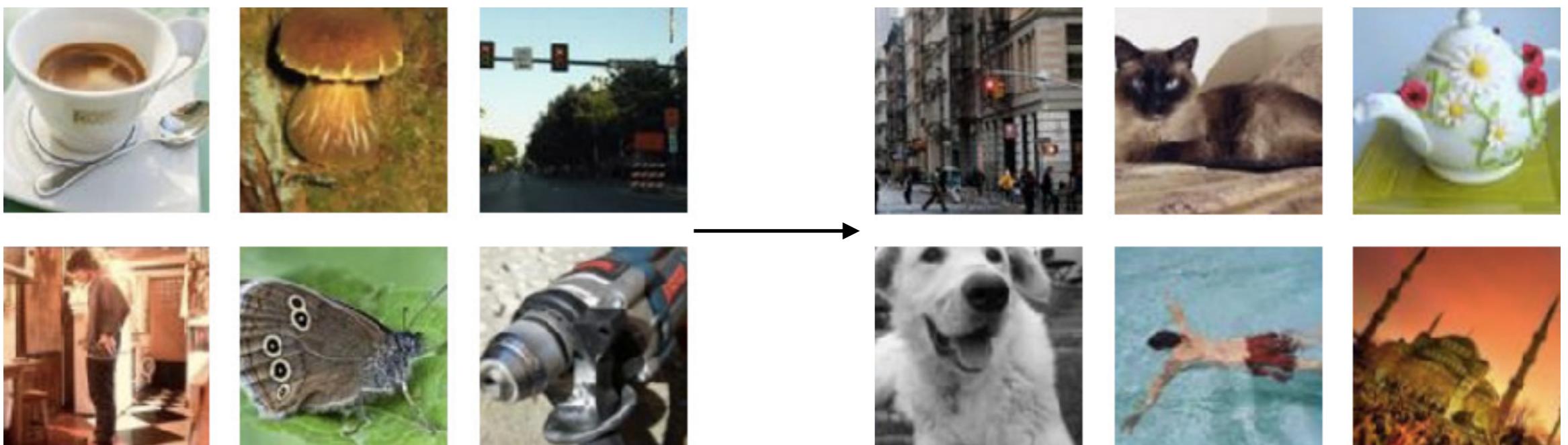
OpenAI

# Generative Modeling

- Density estimation



- Sample generation



Training examples

Model samples

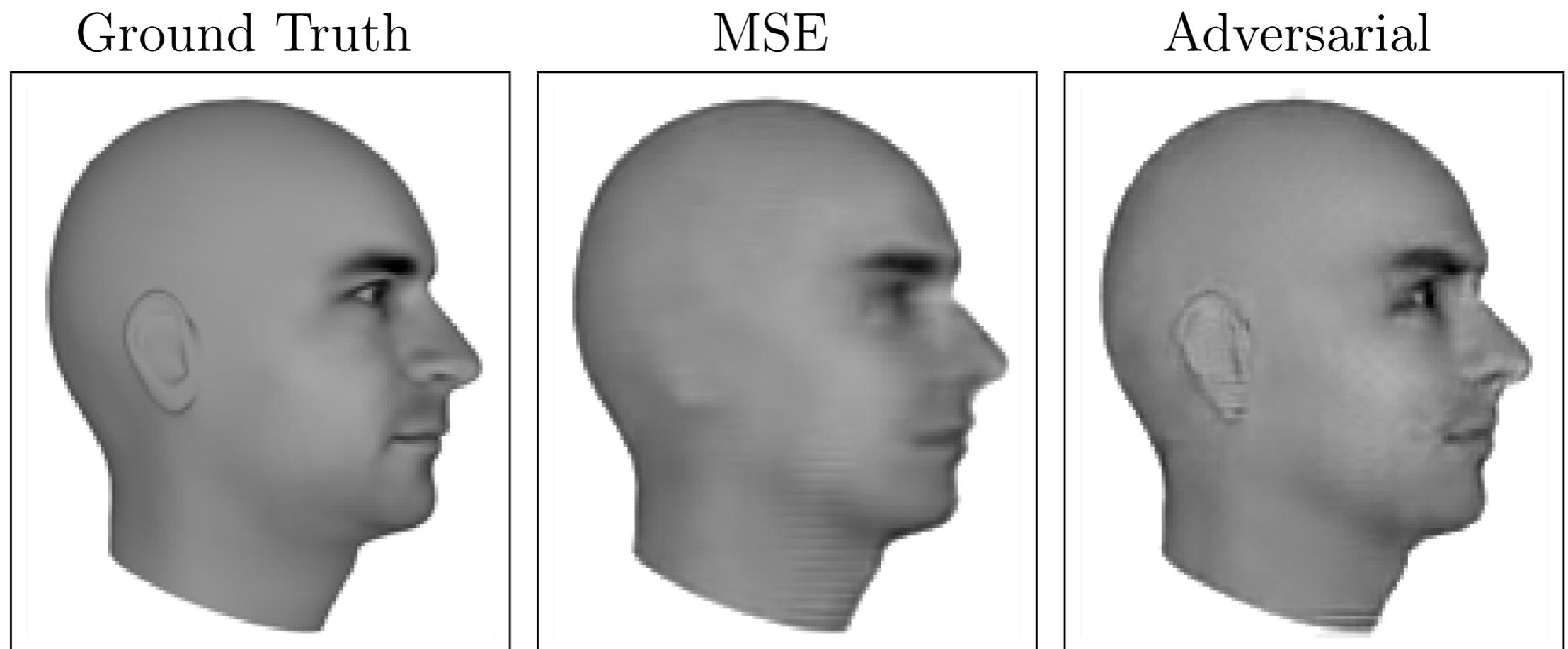
# Roadmap

- Why study generative modeling?
- How do generative models work? How do GANs compare to others?
- How do GANs work?
- Tips and tricks
- Research frontiers
- Combining GANs with other methods

# Why study generative models?

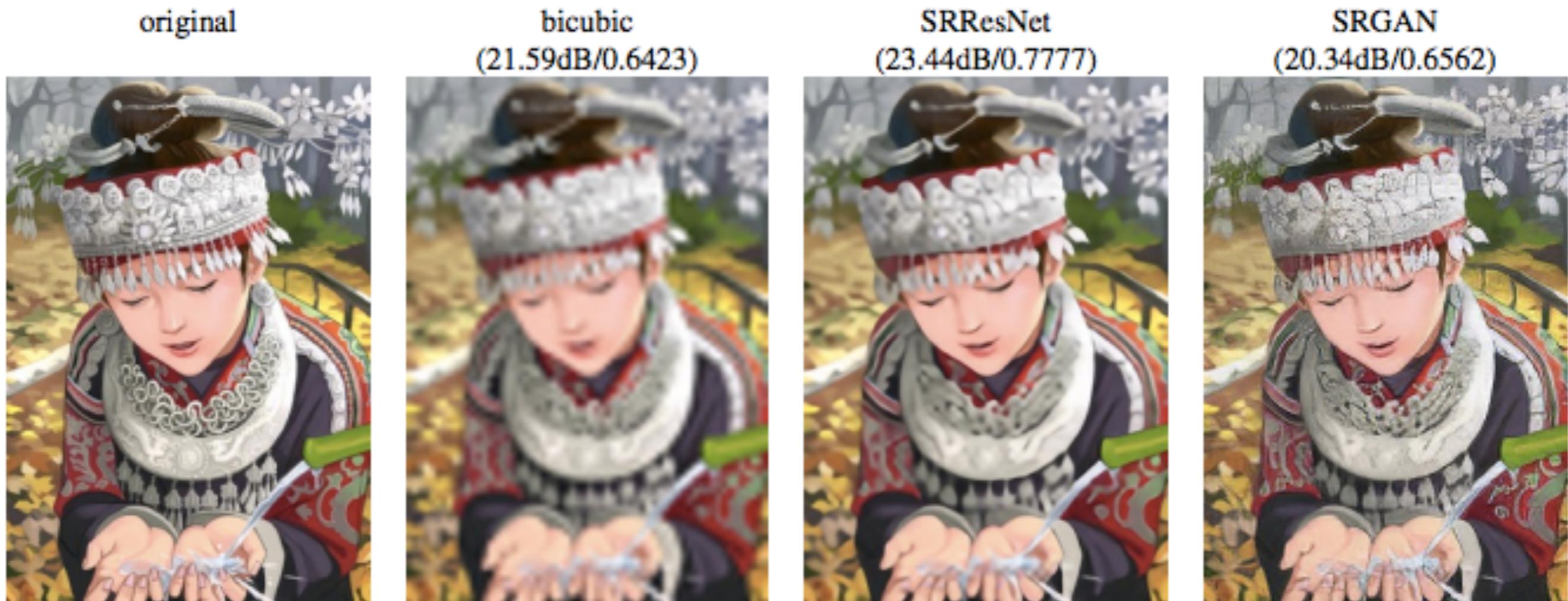
- Excellent test of our ability to use high-dimensional, complicated probability distributions
- Simulate possible futures for planning or simulated RL
- Missing data
  - Semi-supervised learning
- Multi-modal outputs
- Realistic generation tasks

# Next Video Frame Prediction



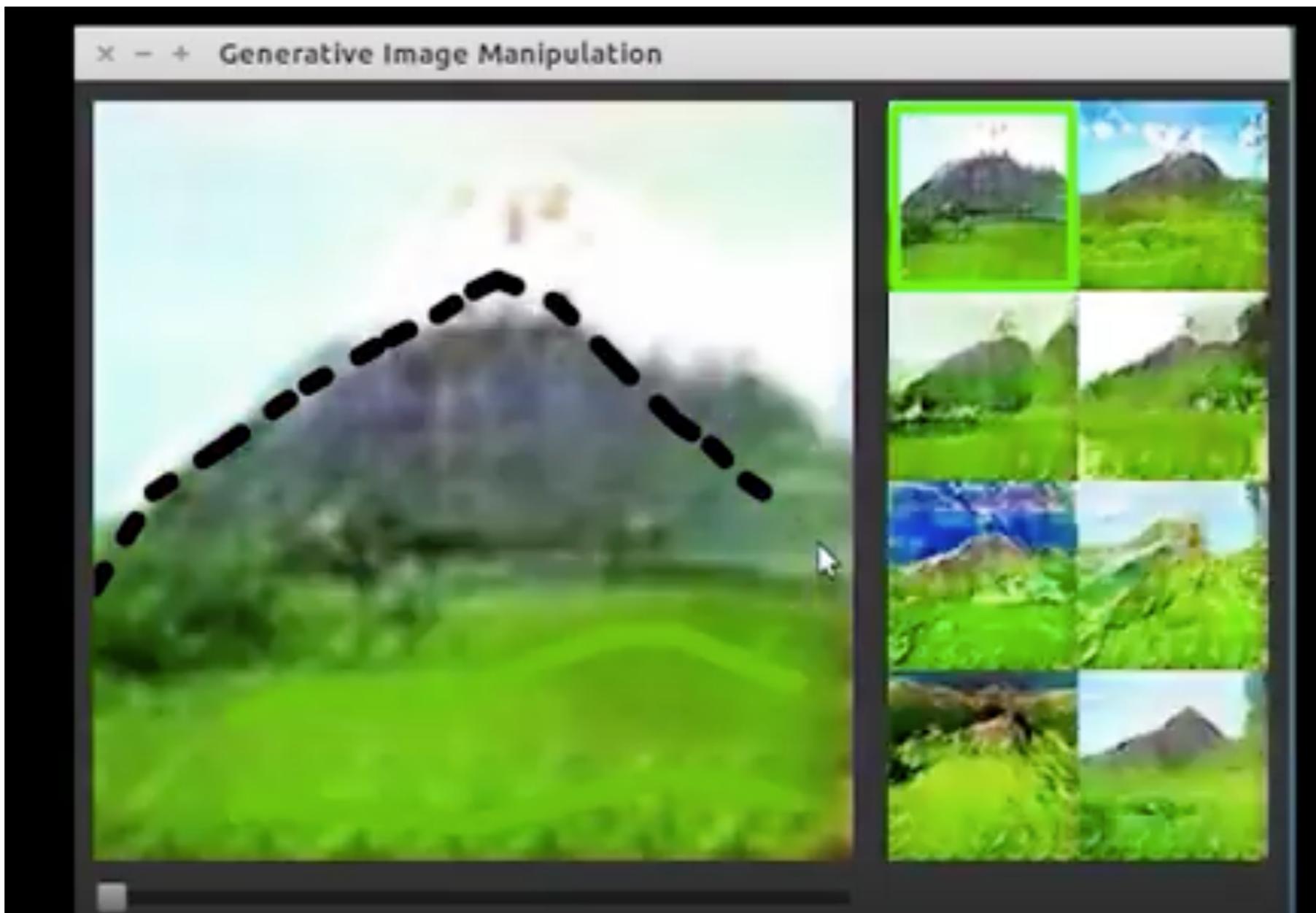
(Lotter et al 2016)

# Single Image Super-Resolution



(Ledig et al 2016)

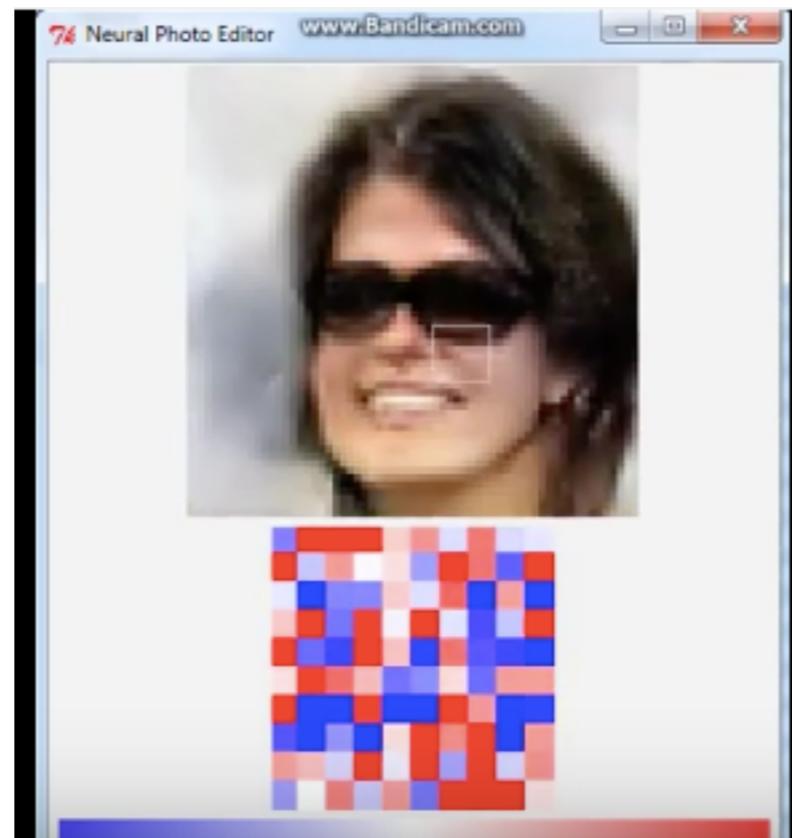
# iGAN



youtube

(Zhu et al 2016)

# Introspective Adversarial Networks



youtube

(Brock et al 2016)

# Image to Image Translation

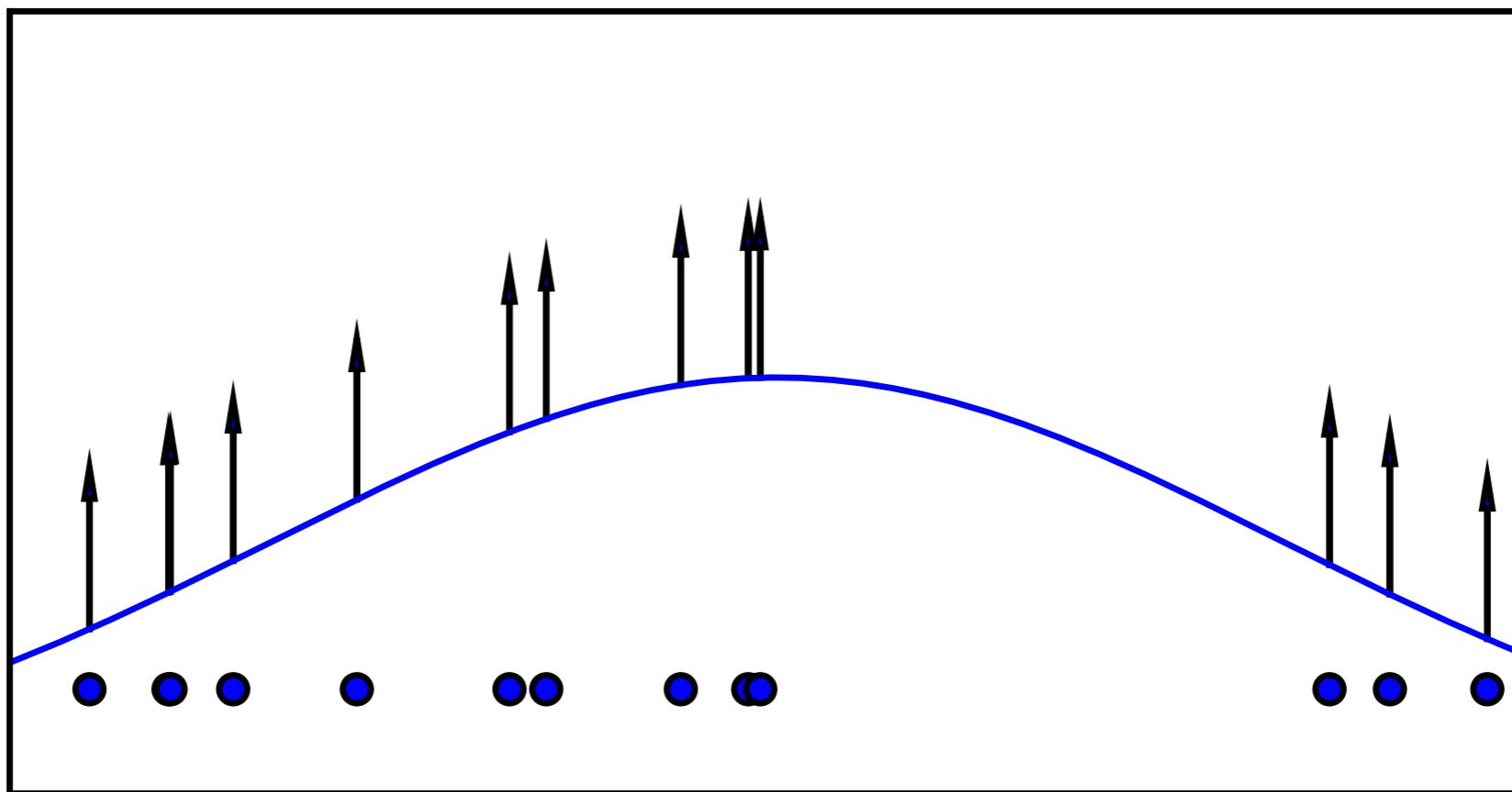


(Isola et al 2016)

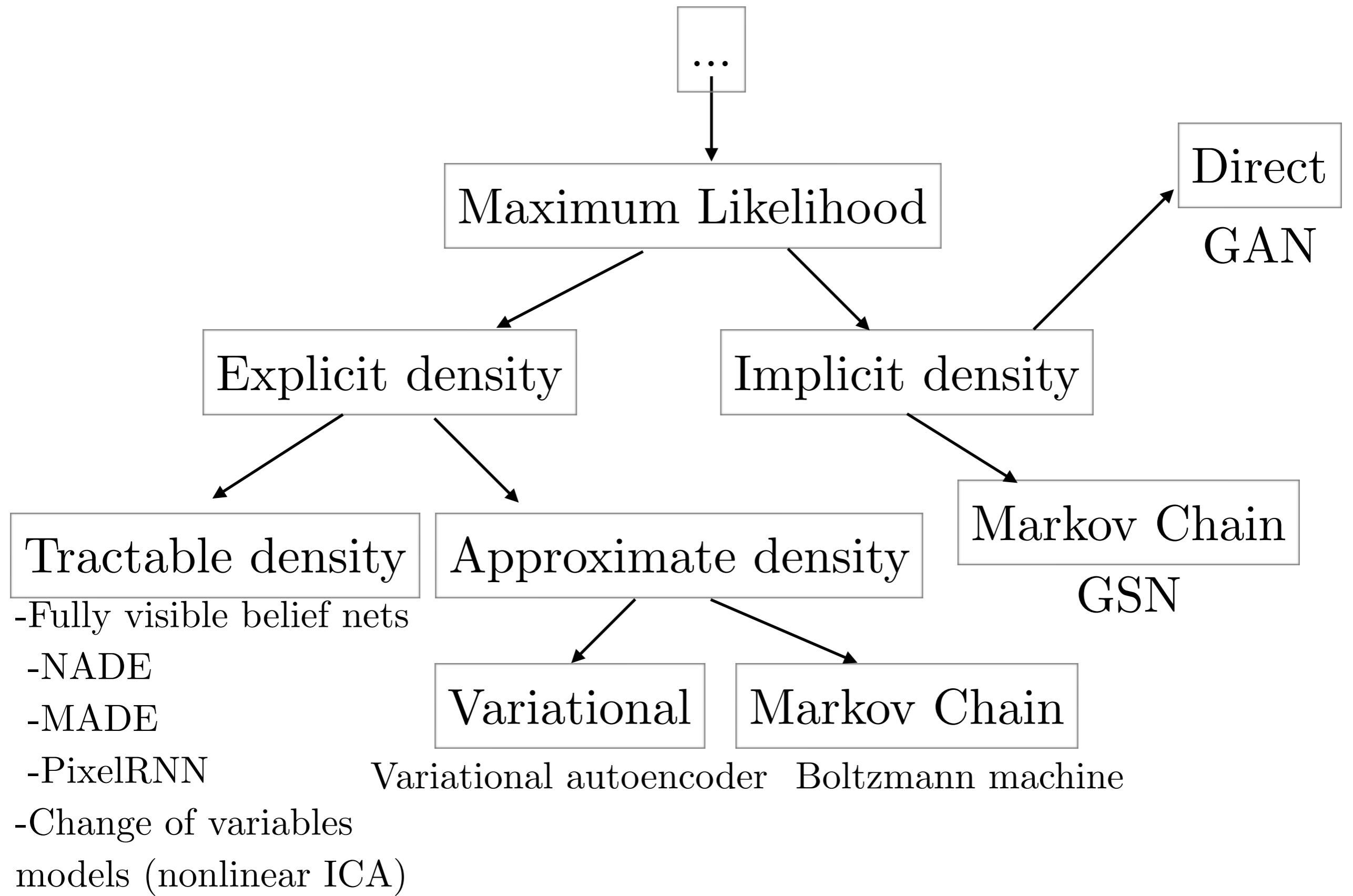
# Roadmap

- Why study generative modeling?
- How do generative models work? How do GANs compare to others?
- How do GANs work?
- Tips and tricks
- Research frontiers
- Combining GANs with other methods

# Maximum Likelihood



$$\theta^* = \arg \max_{\theta} \mathbb{E}_{x \sim p_{\text{data}}} \log p_{\text{model}}(x \mid \theta)$$



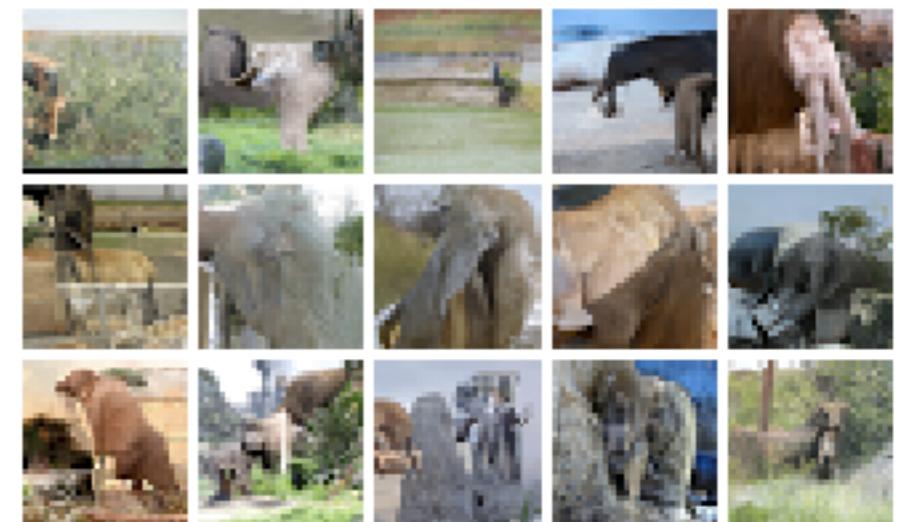
# Fully Visible Belief Nets

- Explicit formula based on chain (Frey et al, 1996)  
rule:

$$p_{\text{model}}(\mathbf{x}) = p_{\text{model}}(x_1) \prod_{i=2}^n p_{\text{model}}(x_i \mid x_1, \dots, x_{i-1})$$

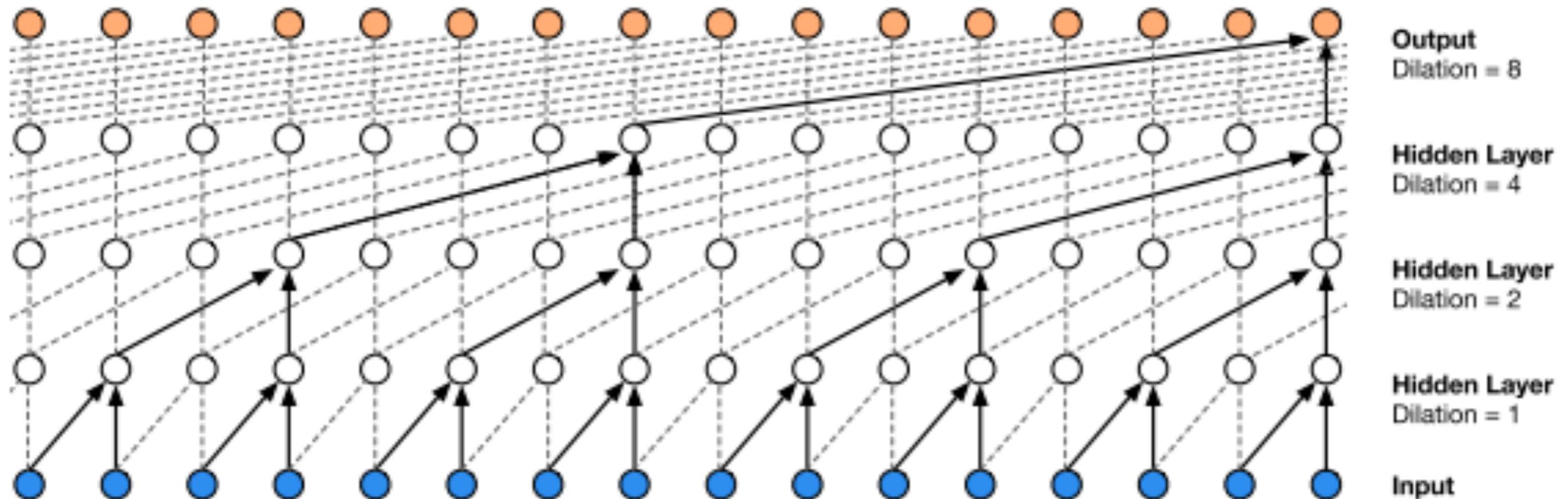
- Disadvantages:

- $O(n)$  sample generation cost
- Generation not controlled by a latent code



PixelCNN elephants  
(van den Ord et al 2016)

# WaveNet



Amazing quality  
Sample generation slow

Two minutes to synthesize  
one second of audio

# Change of Variables

$$y = g(x) \Rightarrow p_x(\mathbf{x}) = p_y(g(\mathbf{x})) \left| \det \left( \frac{\partial g(\mathbf{x})}{\partial \mathbf{x}} \right) \right|$$

e.g. Nonlinear ICA (Hyvärinen 1999)



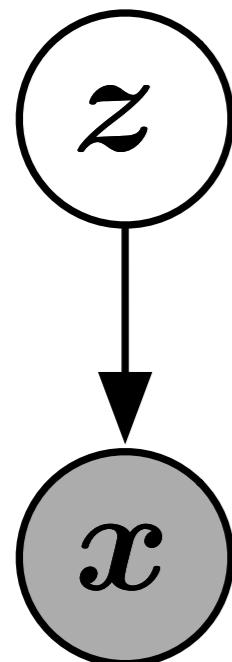
Disadvantages:

- Transformation must be invertible
- Latent dimension must match visible dimension

64x64 ImageNet Samples  
Real NVP (Dinh et al 2016)

# Variational Autoencoder

(Kingma and Welling 2013, Rezende et al 2014)



CIFAR-10 samples

(Kingma et al 2016)

$$\begin{aligned}\log p(\mathbf{x}) &\geq \log p(\mathbf{x}) - D_{\text{KL}}(q(z) \| p(z | \mathbf{x})) \\ &= \mathbb{E}_{z \sim q} \log p(\mathbf{x}, z) + H(q)\end{aligned}$$

Disadvantages:

- Not asymptotically consistent unless  $q$  is perfect
- Samples tend to have lower quality

# Boltzmann Machines

$$p(\mathbf{x}) = \frac{1}{Z} \exp(-E(\mathbf{x}, \mathbf{z}))$$

$$Z = \sum_{\mathbf{x}} \sum_{\mathbf{z}} \exp(-E(\mathbf{x}, \mathbf{z}))$$

- Partition function is intractable
- May be estimated with Markov chain methods
- Generating samples requires Markov chains too

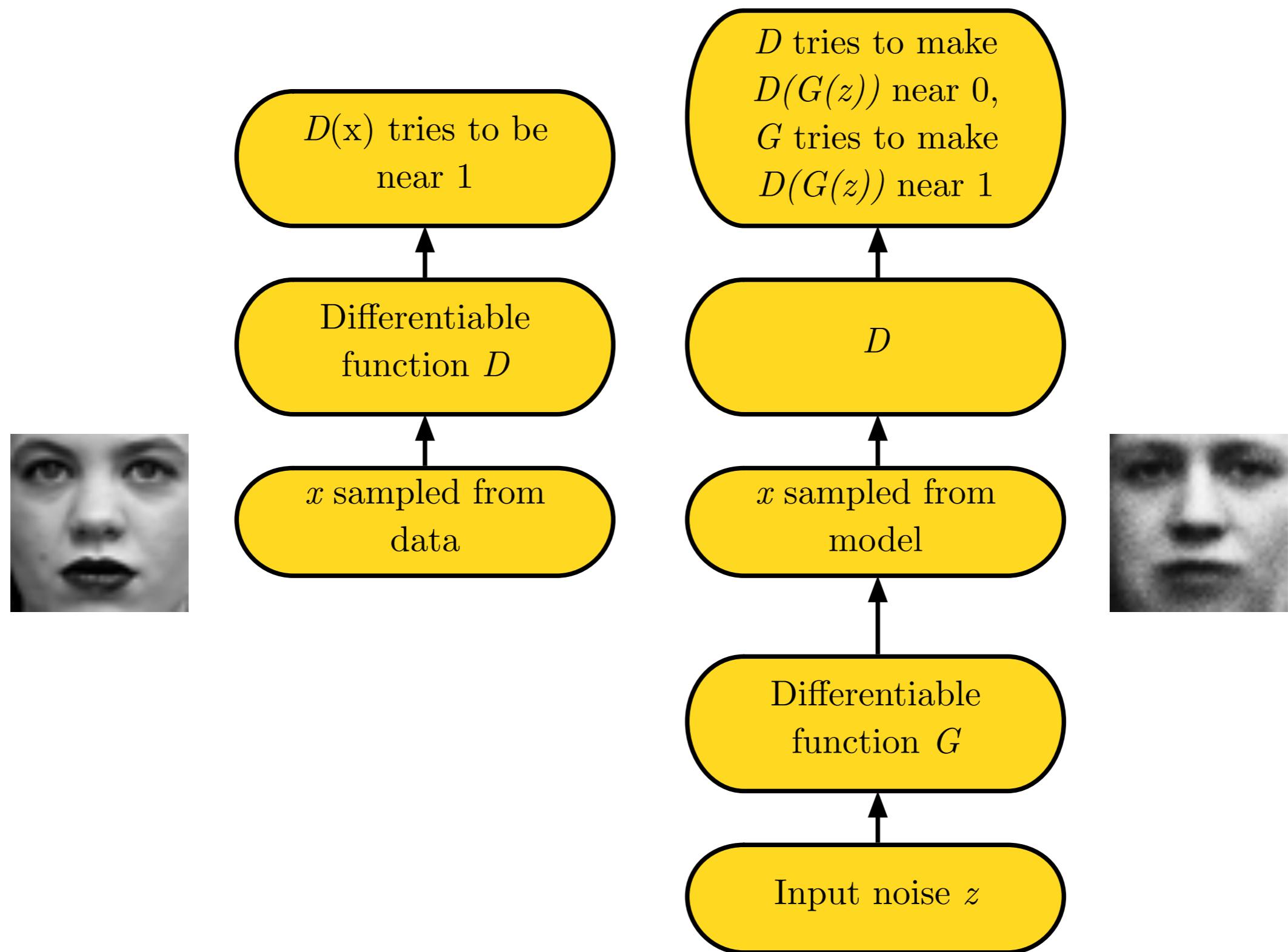
# GANs

- Use a latent code
- Asymptotically consistent (unlike variational methods)
- No Markov chains needed
- Often regarded as producing the best samples
  - No good way to quantify this

# Roadmap

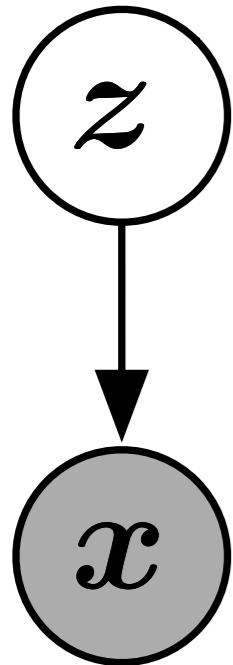
- Why study generative modeling?
- How do generative models work? How do GANs compare to others?
- How do GANs work?
- Tips and tricks
- Research frontiers
- Combining GANs with other methods

# Adversarial Nets Framework



# Generator Network

$$\mathbf{x} = G(\mathbf{z}; \theta^{(G)})$$



- Must be differentiable
  - No invertibility requirement
  - Trainable for any size of  $z$
  - Some guarantees require  $z$  to have higher dimension than  $x$
  - Can make  $x$  conditionally Gaussian given  $z$  but need not do so

# Training Procedure

- Use SGD-like algorithm of choice (Adam) on two minibatches simultaneously:
  - A minibatch of training examples
  - A minibatch of generated samples
- Optional: run  $k$  steps of one player for every step of the other player.

# Minimax Game

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

- Equilibrium is a saddle point of the discriminator loss
- Resembles Jensen-Shannon divergence
- Generator minimizes the log-probability of the discriminator being correct

# Exercise 1

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

- What is the solution to  $D(x)$  in terms of  $p_{\text{data}}$  and  $p_{\text{generator}}$ ?
- What assumptions are needed to obtain this solution?

# Solution

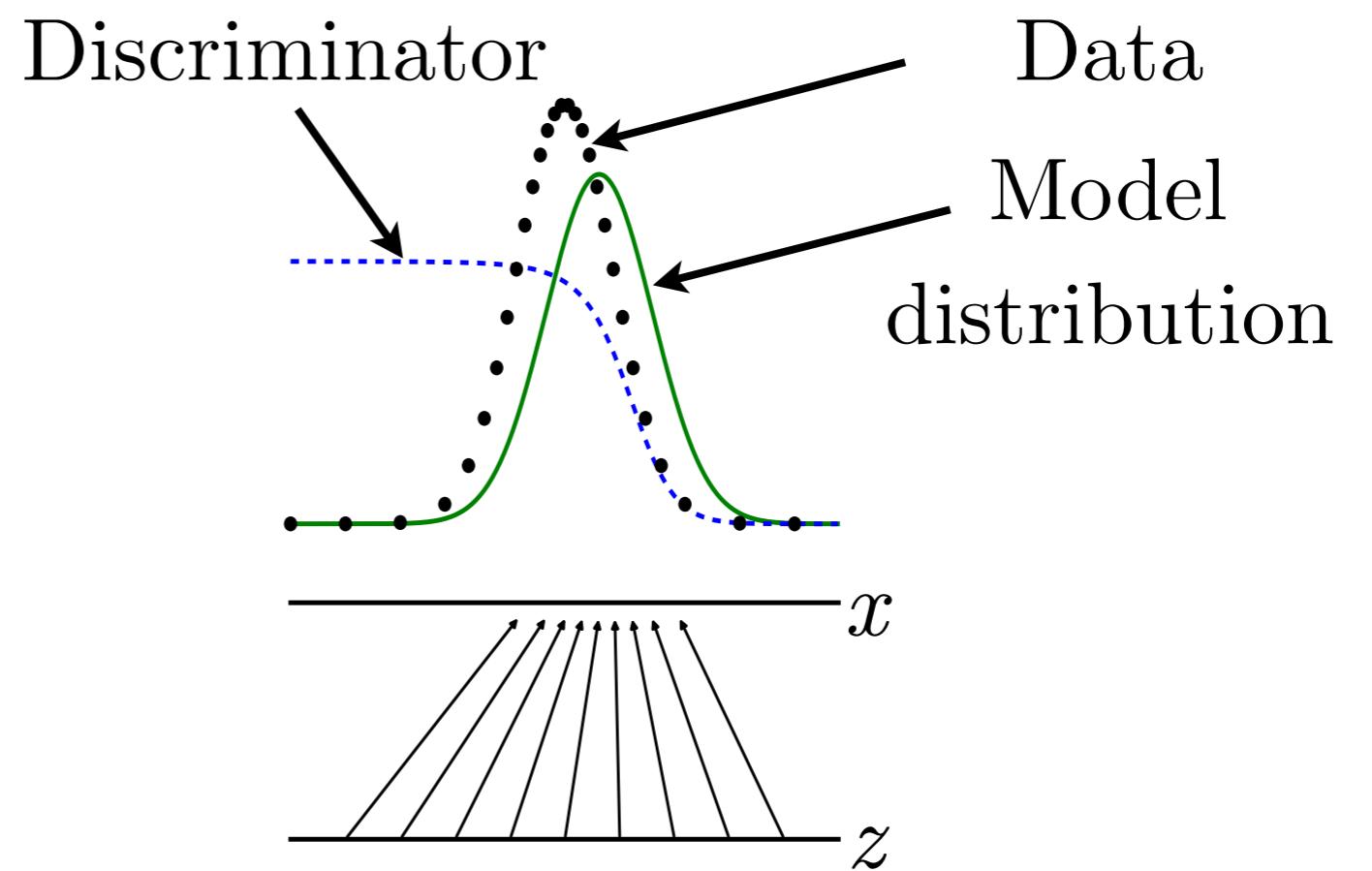
- Assume both densities are nonzero everywhere
  - If not, some input values  $x$  are never trained, so some values of  $D(x)$  have undetermined behavior.
- Solve for where the functional derivatives are zero:

$$\frac{\delta}{\delta D(\mathbf{x})} J^{(D)} = 0$$

# Discriminator Strategy

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

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



# Non-Saturating Game

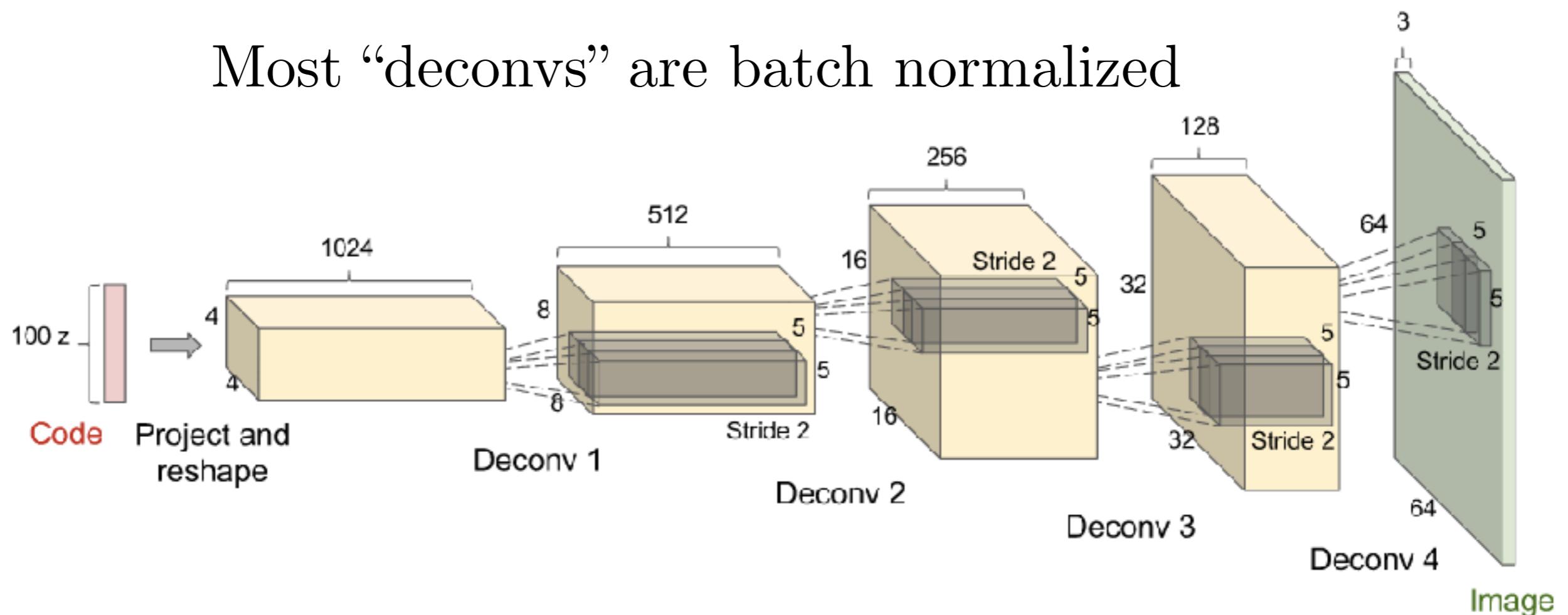
$$J^{(D)} = -\frac{1}{2} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \log D(\mathbf{x}) - \frac{1}{2} \mathbb{E}_{\mathbf{z}} \log (1 - D(G(\mathbf{z})))$$

$$J^{(G)} = -\frac{1}{2} \mathbb{E}_{\mathbf{z}} \log D(G(\mathbf{z}))$$

- Equilibrium no longer describable with a single loss
- Generator maximizes the log-probability of the discriminator being mistaken
- Heuristically motivated; generator can still learn even when discriminator successfully rejects all generator samples

# DCGAN Architecture

Most “deconvs” are batch normalized



(Radford et al 2015)

# DCGANs for LSUN Bedrooms

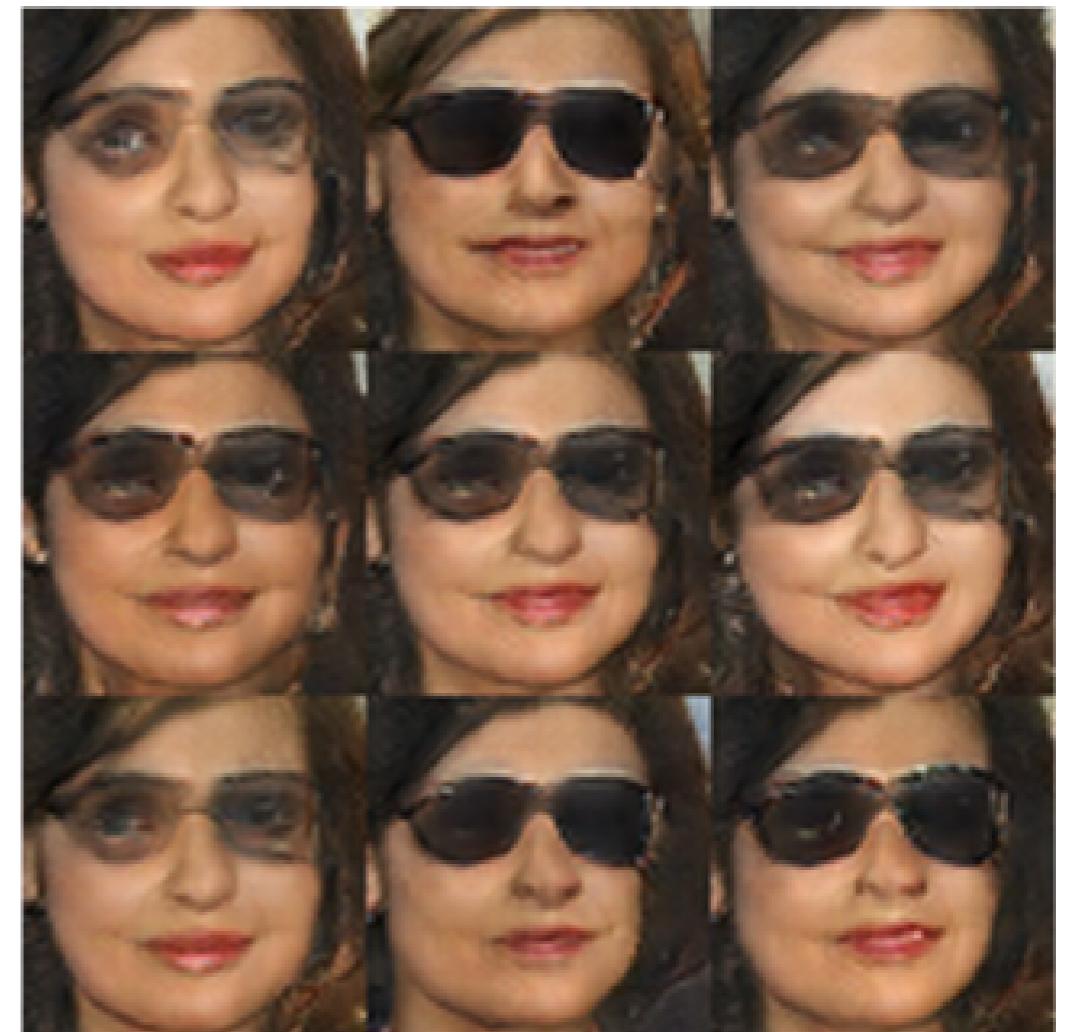


(Radford et al 2015)

# Vector Space Arithmetic



Man      Man      Woman  
with glasses

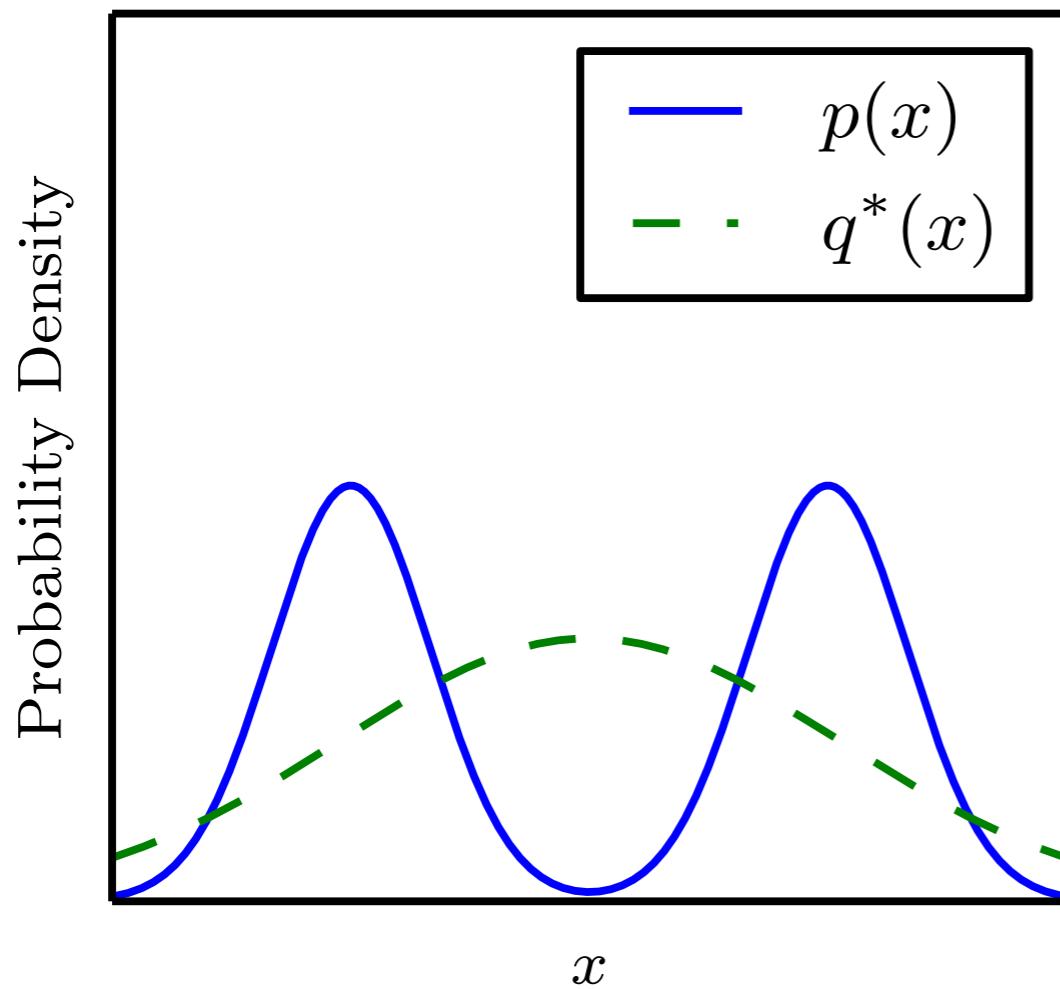


Woman with Glasses

(Radford et al, 2015)

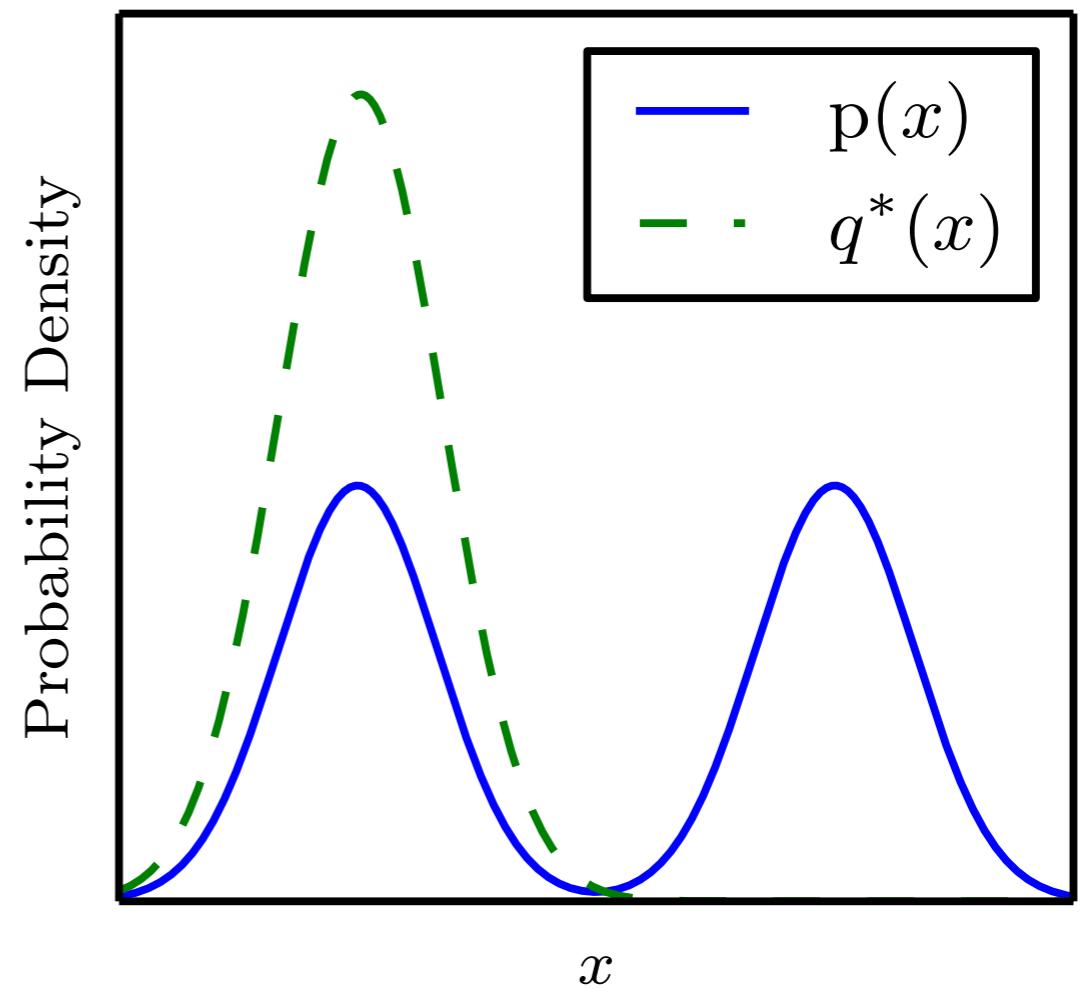
# Is the divergence important?

$$q^* = \operatorname{argmin}_q D_{\text{KL}}(p\|q)$$



Maximum likelihood

$$q^* = \operatorname{argmin}_q D_{\text{KL}}(q\|p)$$



Reverse KL

(Goodfellow et al 2016)

# Modifying GANs to do Maximum Likelihood

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

$$J^{(G)} = -\frac{1}{2} \mathbb{E}_{\mathbf{z}} \exp (\sigma^{-1} (D(G(\mathbf{z}))))$$

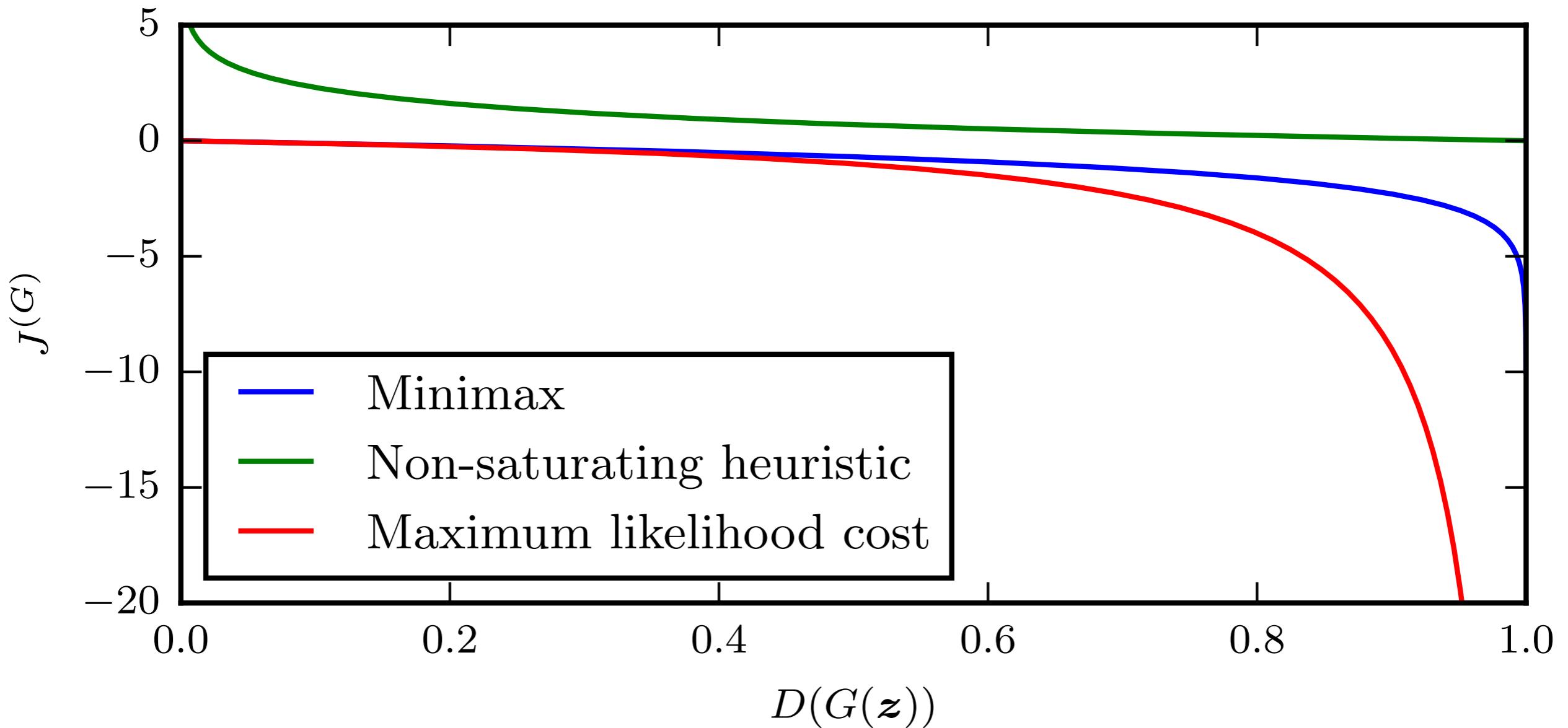
When discriminator is optimal, the generator gradient matches that of maximum likelihood

(“On Distinguishability Criteria for Estimating Generative Models”, Goodfellow 2014, pg 5)

# Reducing GANs to RL

- Generator makes a sample
- Discriminator evaluates a sample
- Generator's cost (negative reward) is a function of  $D(G(z))$
- Note that generator's cost does not include the data,  $x$
- Generator's cost is always monotonically decreasing in  $D(G(z))$
- Different divergences change the location of the cost's fastest decrease

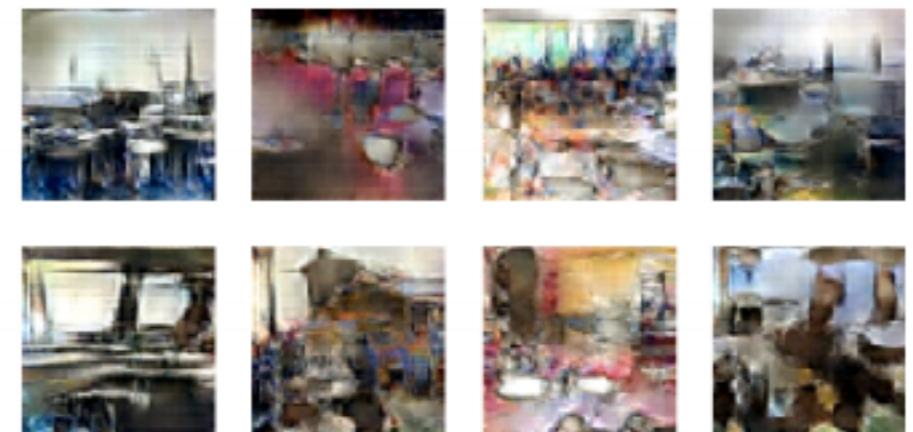
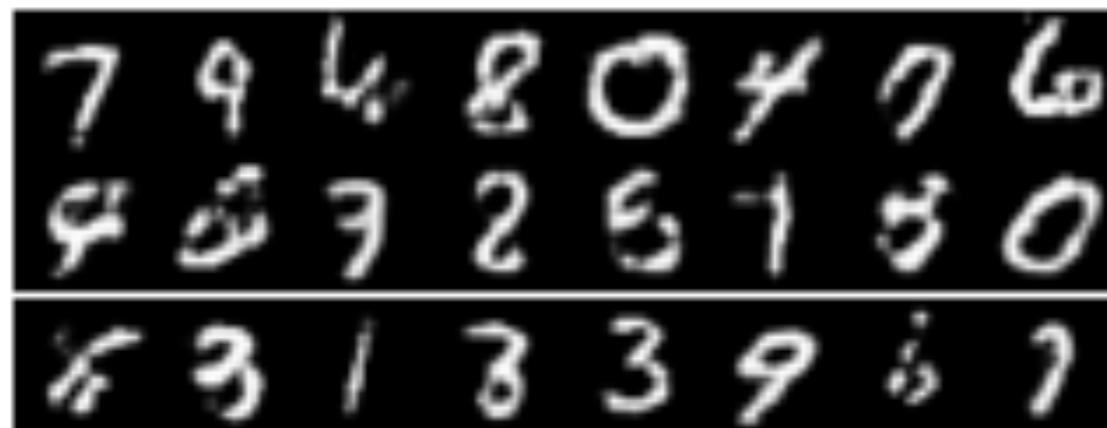
# Comparison of Generator Losses



(Goodfellow 2014)

# Loss does not seem to explain why GAN samples are sharp

KL



Reverse  
KL



(Nowozin et al 2016)

KL samples from LSUN

Takeaway: the approximation strategy  
matters more than the loss

# Comparison to NCE, MLE

$$V(G, D) = \mathbb{E}_{p_{\text{data}}} \log D(\mathbf{x}) + \mathbb{E}_{p_{\text{generator}}} (\log (1 - D(\mathbf{x})))$$

	NCE (Gutmann and Hyvärinen 2010)	MLE	GAN
$D$	$D(x) = \frac{p_{\text{model}}(\mathbf{x})}{p_{\text{model}}(\mathbf{x}) + p_{\text{generator}}(\mathbf{x})}$		Neural network
Goal	Learn $p_{\text{model}}$	Learn $p_{\text{generator}}$	
$G$ update rule	None ( $G$ is fixed)	Copy $p_{\text{model}}$ parameters	Gradient descent on $V$
$D$ update rule	Gradient ascent on $V$		

(“On Distinguishability Criteria...”, Goodfellow 2014)

# Roadmap

- Why study generative modeling?
- How do generative models work? How do GANs compare to others?
- How do GANs work?
- Tips and tricks
- Research frontiers
- Combining GANs with other methods

# Labels improve subjective sample quality

- Learning a conditional model  $p(y|x)$  often gives much better samples from all classes than learning  $p(x)$  does (Denton et al 2015)
- Even just learning  $p(x,y)$  makes samples from  $p(x)$  look much better to a human observer (Salimans et al 2016)
- Note: this defines three categories of models (no labels, trained with labels, generating condition on labels) that should not be compared directly to each other

# One-sided label smoothing

- Default discriminator cost:  
$$\text{cross\_entropy}(1., \text{discriminator}(\text{data})) + \text{cross\_entropy}(0., \text{discriminator}(\text{samples}))$$
- One-sided label smoothed cost (Salimans et al 2016):  
$$\text{cross\_entropy}(.9, \text{discriminator}(\text{data})) + \text{cross\_entropy}(0., \text{discriminator}(\text{samples}))$$

# Do not smooth negative labels

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

Reinforces current generator behavior

$$D(\mathbf{x}) = \frac{(1 - \alpha)p_{\text{data}}(\mathbf{x}) + \beta p_{\text{model}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_{\text{model}}(\mathbf{x})}$$

# Benefits of label smoothing

- Good regularizer (Szegedy et al 2015)
- Does not reduce classification accuracy, only confidence
- Benefits specific to GANs:
  - Prevents discriminator from giving very large gradient signal to generator
  - Prevents extrapolating to encourage extreme samples

# Batch Norm

- Given inputs  $X = \{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$
- Compute mean and standard deviation of features of  $X$
- Normalize features (subtract mean, divide by standard deviation)
- Normalization operation is part of the graph
  - Backpropagation computes the gradient through the normalization
  - This avoids wasting time repeatedly learning to undo the normalization

Batch norm in  $G$  can cause  
strong intra-batch correlation



# Reference Batch Norm

- Fix a *reference batch*  $R = \{r^{(1)}, r^{(2)}, \dots, r^{(m)}\}$
- Given new inputs  $X = \{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$
- Compute mean and standard deviation of features of  $R$ 
  - Note that though  $R$  does not change, the feature values change when the parameters change
- Normalize the features of  $X$  using the mean and standard deviation from  $R$
- Every  $x^{(i)}$  is always treated the same, regardless of which other examples appear in the minibatch

# Virtual Batch Norm

- Reference batch norm can overfit to the reference batch. A partial solution is *virtual batch norm*
- Fix a *reference batch*  $R = \{r^{(1)}, r^{(2)}, \dots, r^{(m)}\}$
- Given new inputs  $X = \{x^{(1)}, x^{(2)}, \dots, x^{(n)}\}$
- For each  $x^{(i)}$  in  $X$ :
  - Construct a *virtual batch*  $V$  containing both  $x^{(i)}$  and all of  $R$
  - Compute mean and standard deviation of features of  $V$
  - Normalize the features of  $x^{(i)}$  using the mean and standard deviation from  $V$

# Balancing $G$ and $D$

- Usually the discriminator “wins”
- This is a good thing—the theoretical justifications are based on assuming  $D$  is perfect
- Usually  $D$  is bigger and deeper than  $G$
- Sometimes run  $D$  more often than  $G$ . Mixed results.
- Do not try to limit  $D$  to avoid making it “too smart”
  - Use non-saturating cost
  - Use label smoothing

# Roadmap

- Why study generative modeling?
- How do generative models work? How do GANs compare to others?
- How do GANs work?
- Tips and tricks
- Research frontiers
- Combining GANs with other methods

# Non-convergence

- Optimization algorithms often approach a saddle point or local minimum rather than a global minimum
- Game solving algorithms may not approach an equilibrium at all

# Exercise 2

- For scalar  $x$  and  $y$ , consider the value function:
$$V(x, y) = xy$$
- Does this game have an equilibrium? Where is it?
- Consider the learning dynamics of simultaneous gradient descent with infinitesimal learning rate (continuous time). Solve for the trajectory followed by these dynamics.

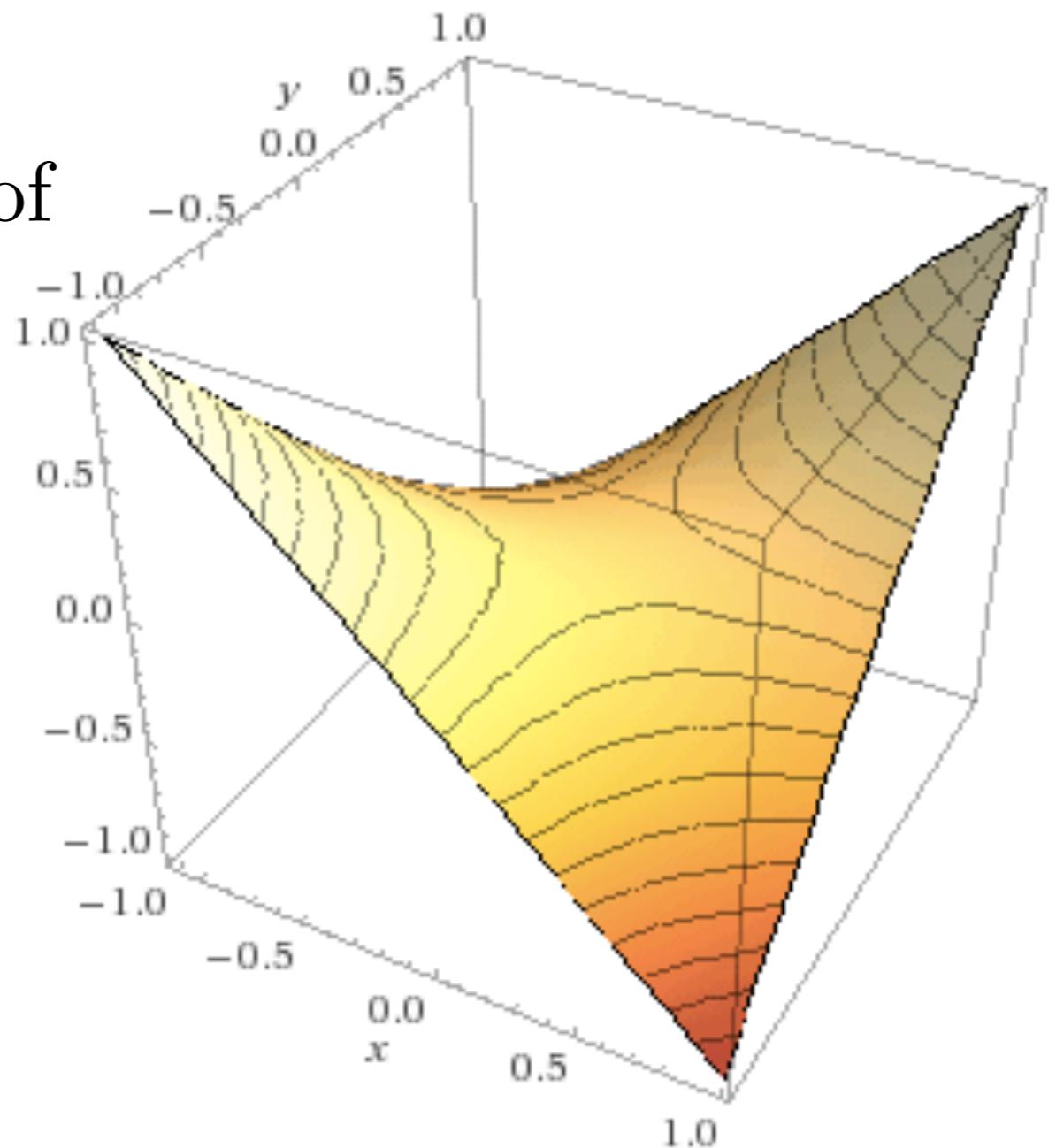
$$\frac{\partial x}{\partial t} = -\frac{\partial}{\partial x} V(x(t), y(t))$$

$$\frac{\partial y}{\partial t} = \frac{\partial}{\partial y} V(x(t), y(t))$$

# Solution

This is the canonical example of a saddle point.

There is an equilibrium, at  $x = 0, y = 0$ .



# Solution

- The gradient dynamics are:

$$\frac{\partial x}{\partial t} = -y(t)$$

$$\frac{\partial y}{\partial t} = x(t)$$

- Differentiating the second equation, we obtain:

$$\frac{\partial^2 y}{\partial t^2} = \frac{\partial x}{\partial t} = -y(t)$$

- We recognize that  $y(t)$  must be a sinusoid

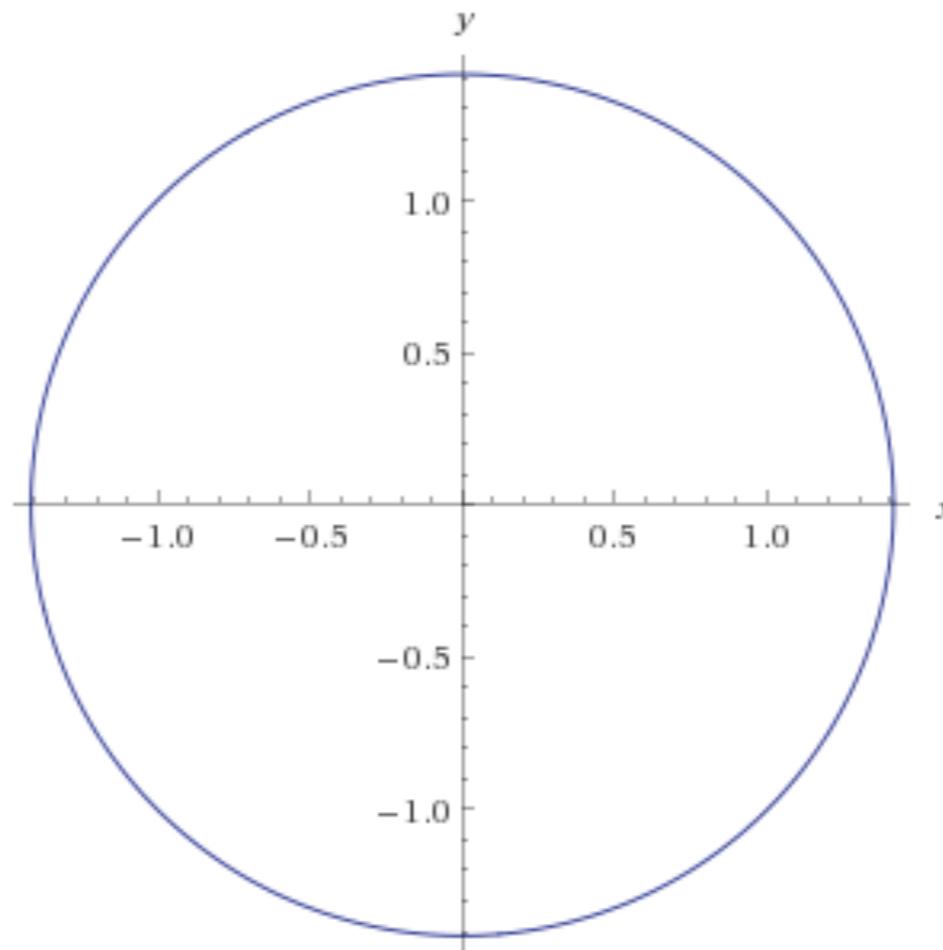
# Solution

- The dynamics are a circular orbit:

$$x(t) = x(0) \cos(t) - y(0) \sin(t)$$

$$y(t) = x(0) \sin(t) + y(0) \cos(t)$$

Discrete time  
gradient descent  
can spiral  
outward for large  
step sizes



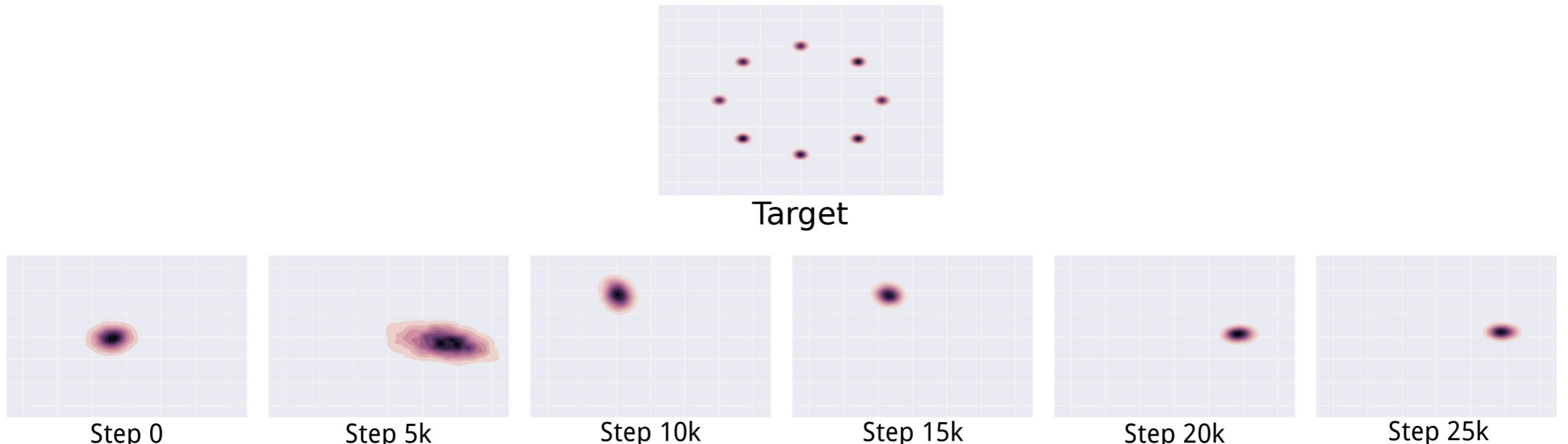
# Non-convergence in GANs

- Exploiting convexity in function space, GAN training is theoretically guaranteed to converge if we can modify the density functions directly, but:
  - Instead, we modify  $G$  (sample generation function) and  $D$  (density ratio), not densities
  - We represent  $G$  and  $D$  as highly non-convex parametric functions
- “Oscillation”: can train for a very long time, generating very many different categories of samples, without clearly generating better samples
- Mode collapse: most severe form of non-convergence

# Mode Collapse

$$\min_G \max_D V(G, D) \neq \max_D \min_G V(G, D)$$

- $D$  in inner loop: convergence to correct distribution
- $G$  in inner loop: place all mass on most likely point



(Metz et al 2016)

# Reverse KL loss does not explain mode collapse

- Other GAN losses also yield mode collapse
- Reverse KL loss prefers to fit as many modes as the model can represent and no more; it does not prefer fewer modes in general
- GANs often seem to collapse to far fewer modes than the model can represent

# Mode collapse causes low output diversity

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



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



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



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



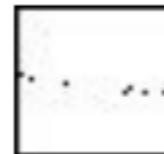
## Key-points



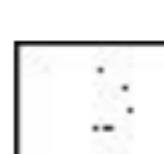
## GAN (Reed 2016b)



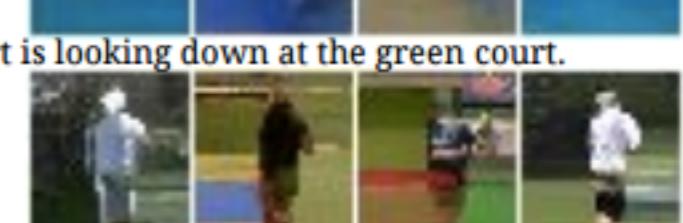
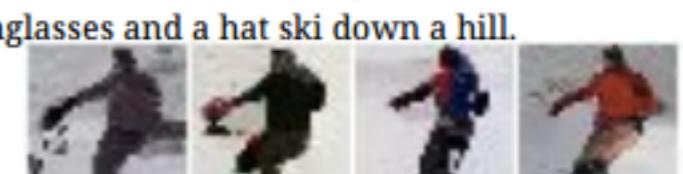
This guy is in black trunks and swimming underwater.



A tennis player in a blue polo shirt is looking down at the green court.



## This work



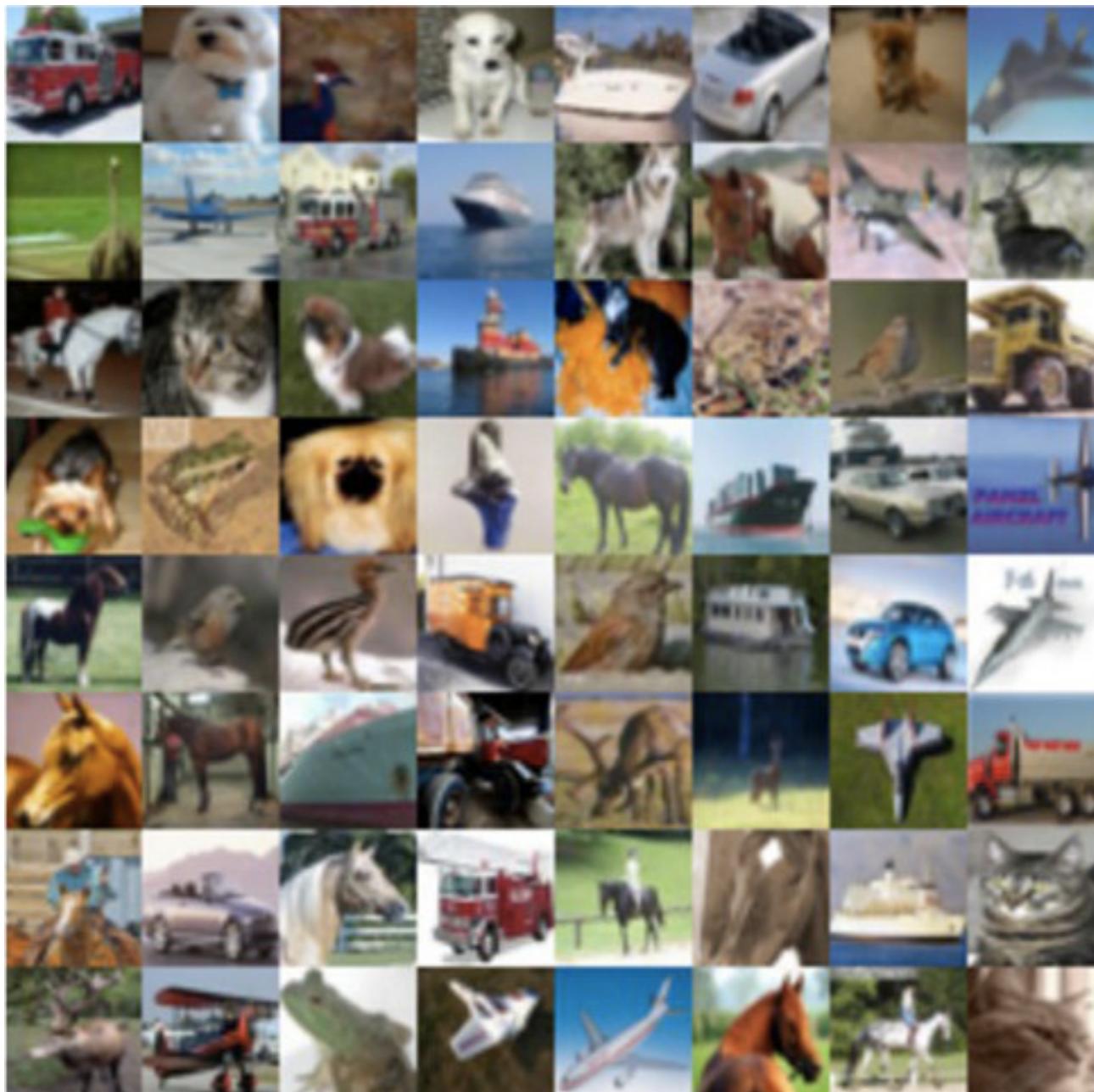
(Reed et al, submitted to  
ICLR 2017)

(Reed et al 2016)

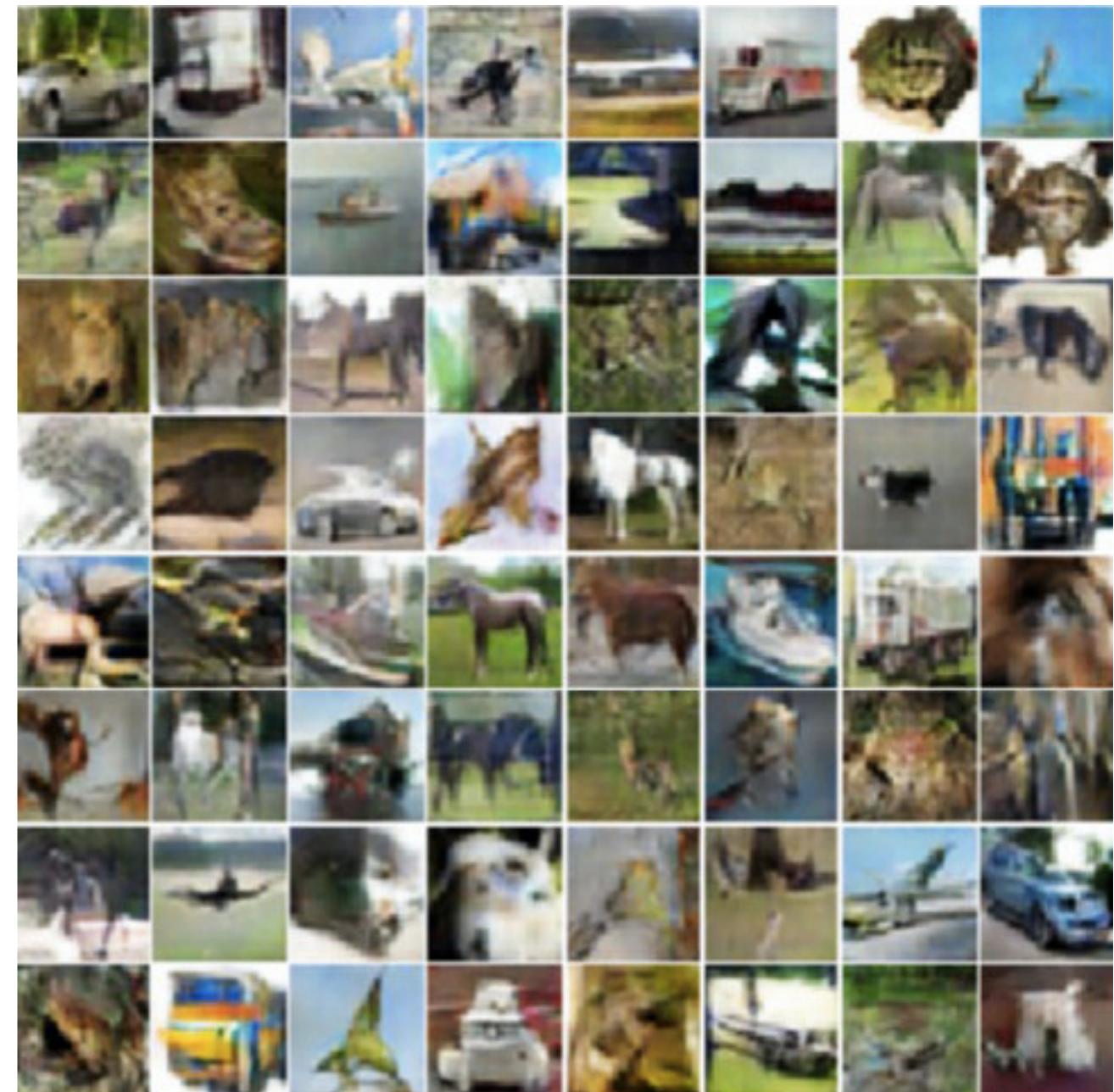
# Minibatch Features

- Add minibatch features that classify each example by comparing it to other members of the minibatch (Salimans et al 2016)
- Nearest-neighbor style features detect if a minibatch contains samples that are too similar to each other

# Minibatch GAN on CIFAR



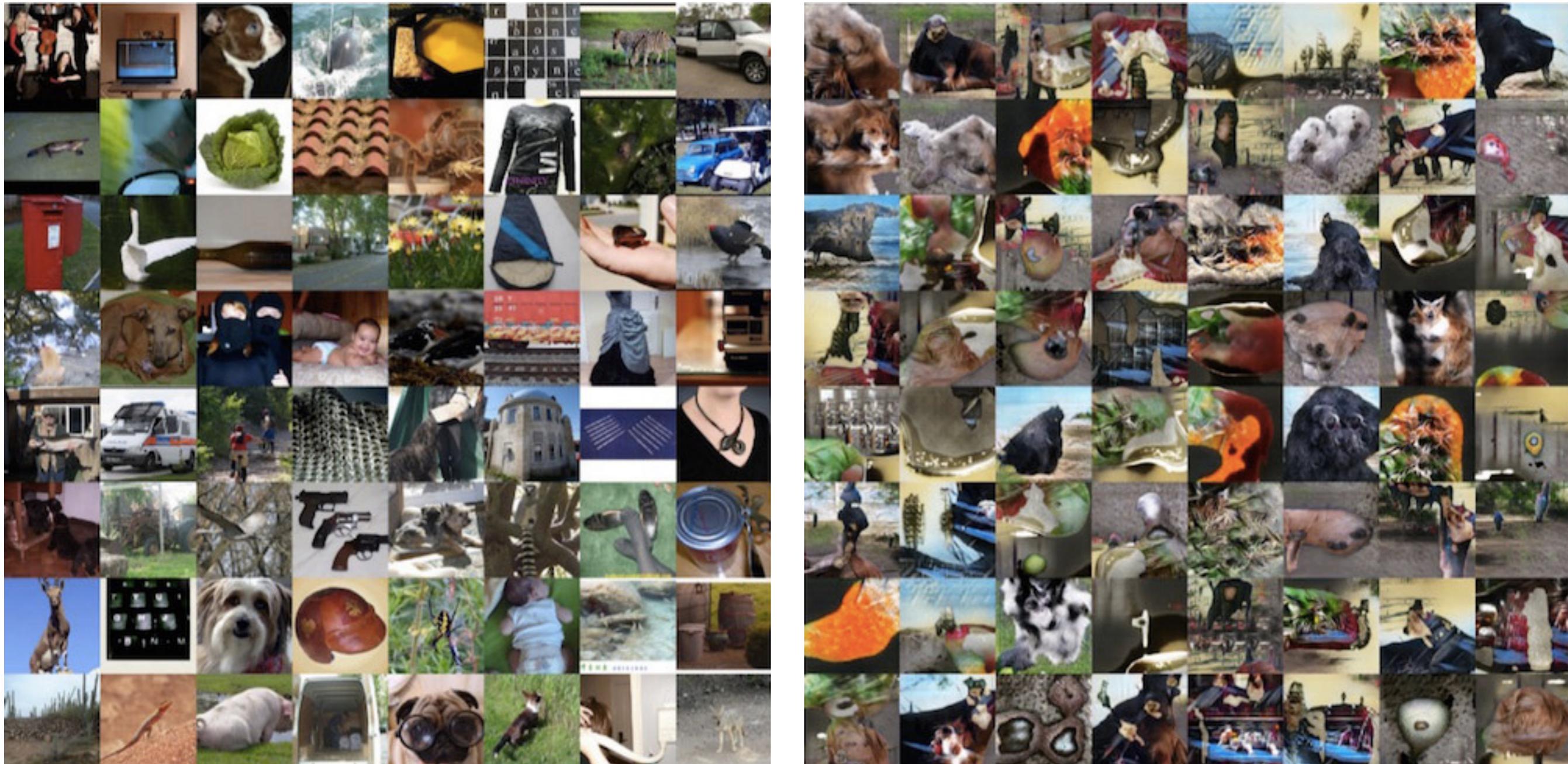
# Training Data



## Samples

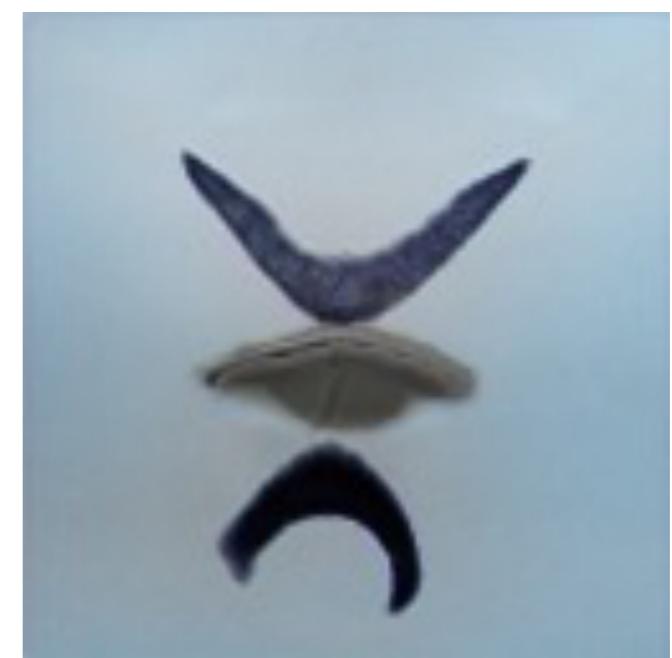
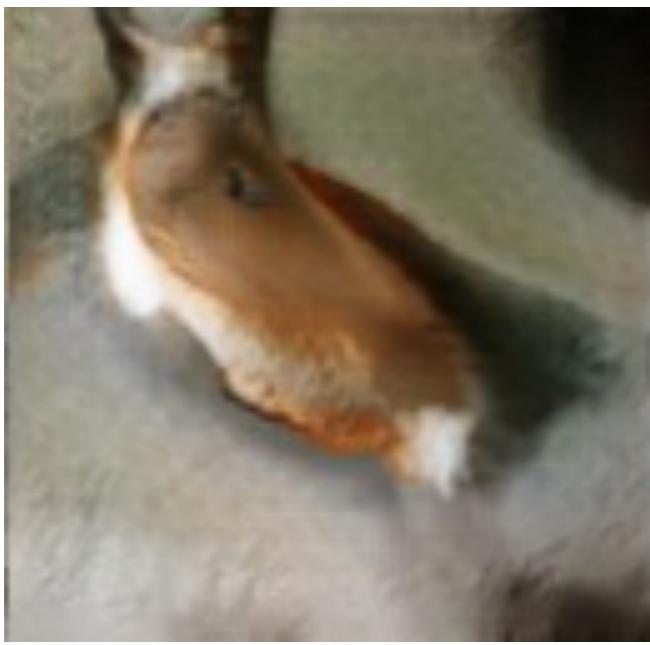
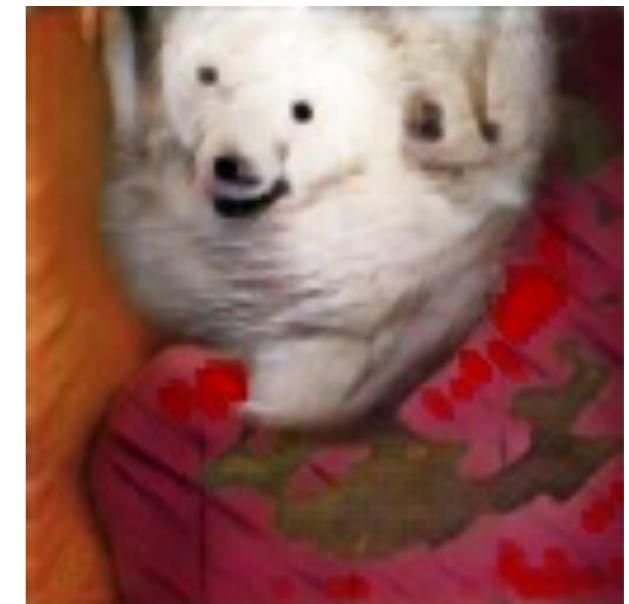
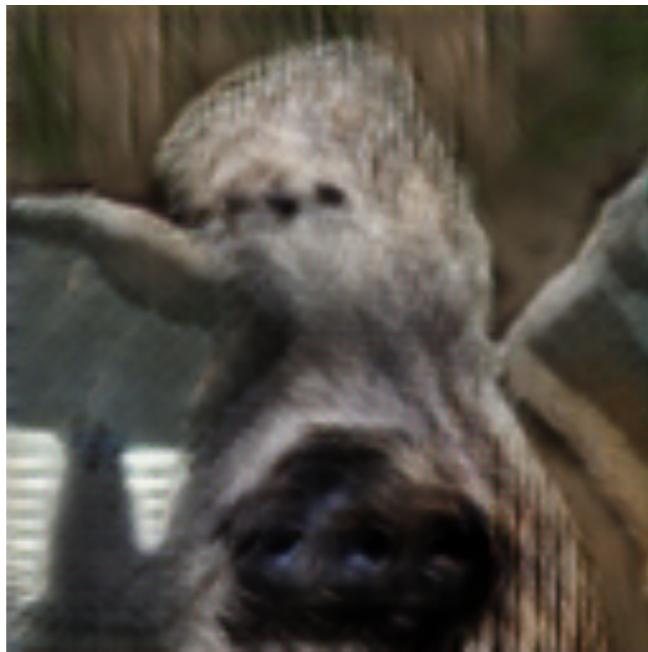
(Salimans et al 2016)

# Minibatch GAN on ImageNet

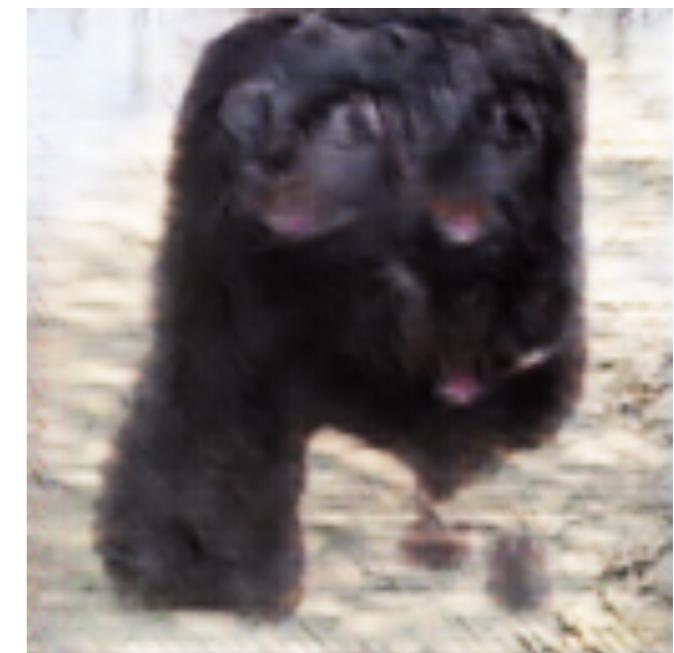
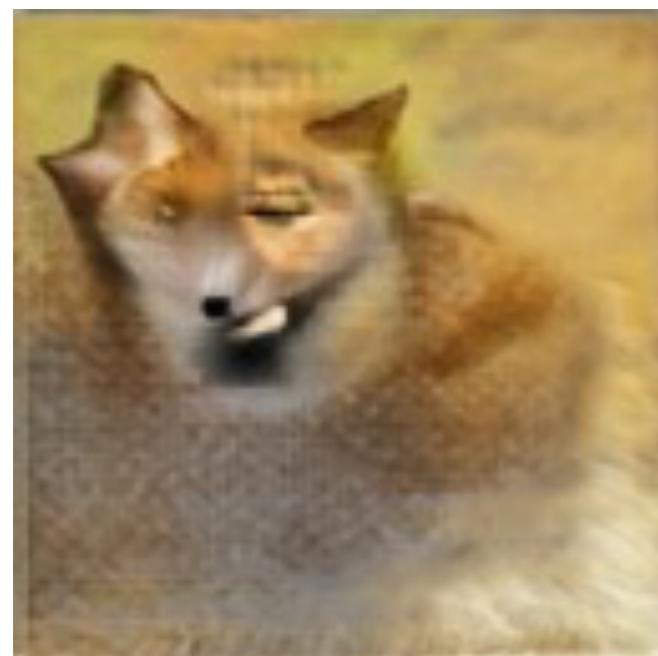


(Salimans et al 2016)

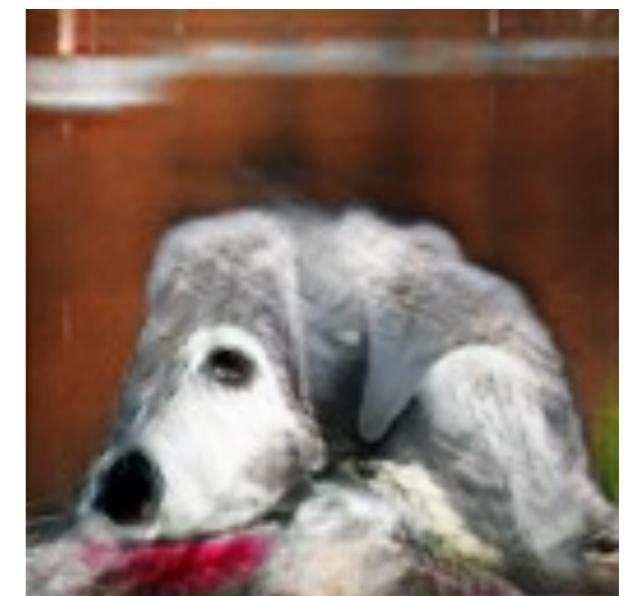
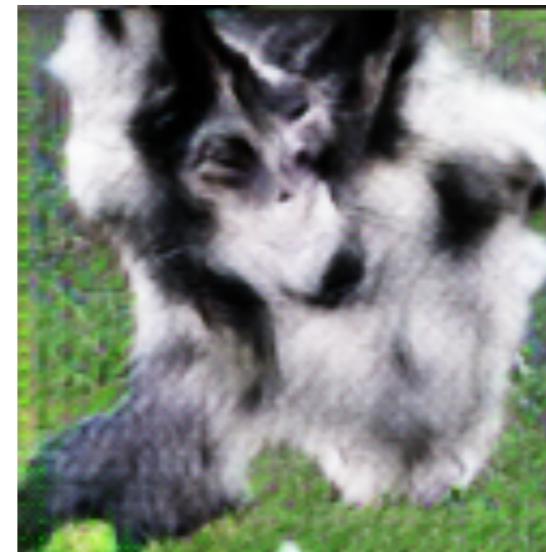
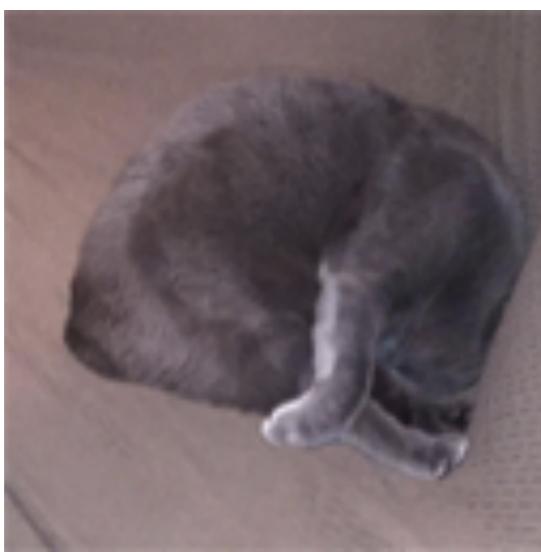
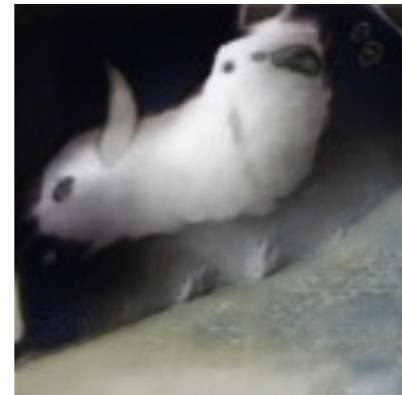
# Cherry-Picked Results



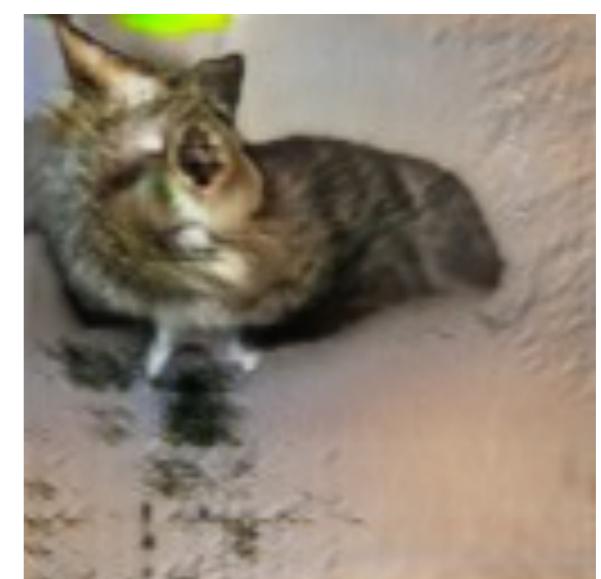
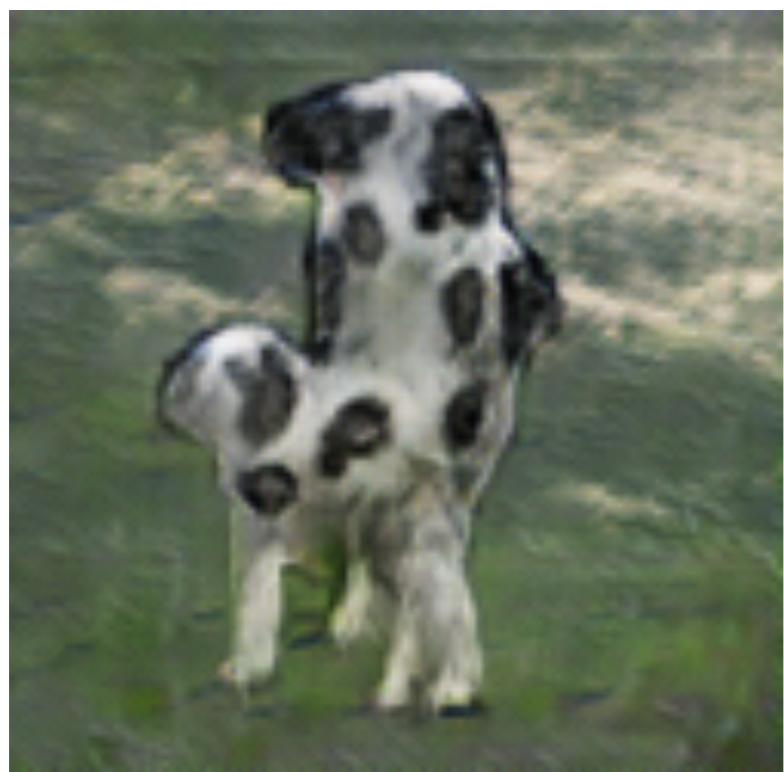
# Problems with Counting



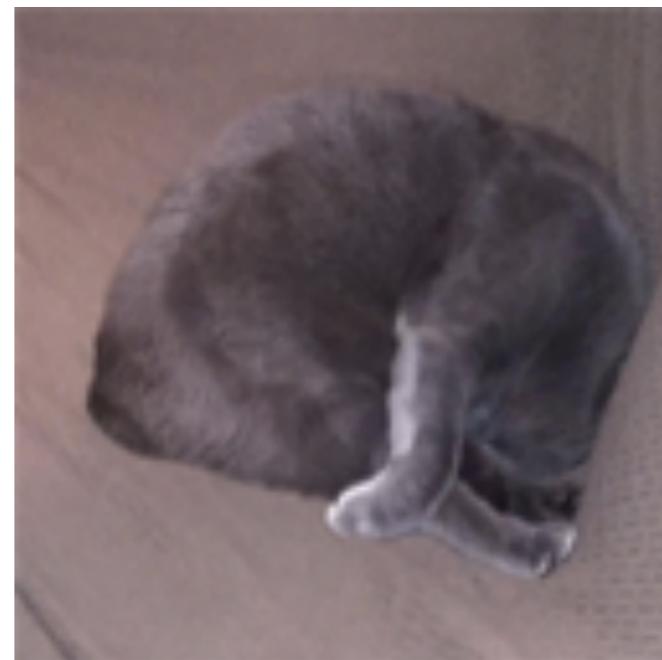
# Problems with Perspective



# Problems with Global Structure

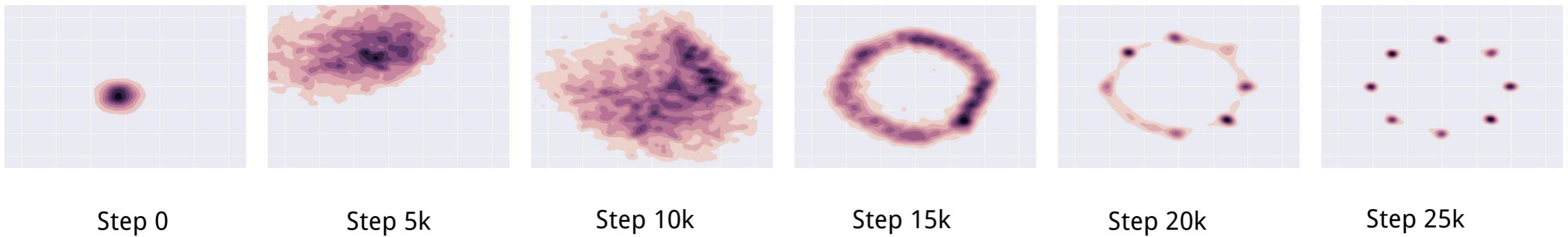


This one is real



# Unrolled GANs

- Backprop through  $k$  updates of the discriminator to prevent mode collapse:



(Metz et al 2016)

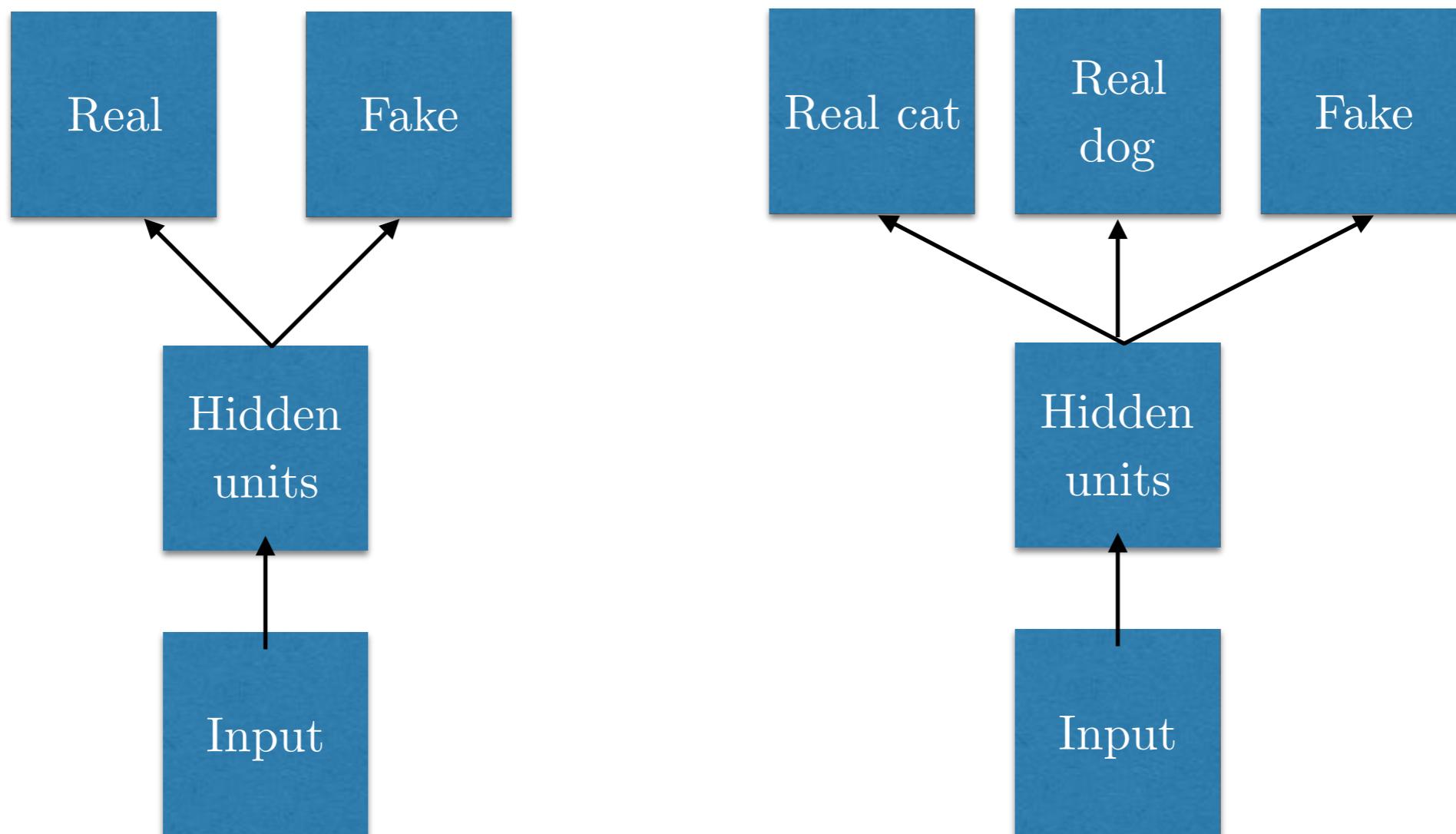
# Evaluation

- There is not any single compelling way to evaluate a generative model
  - Models with good likelihood can produce bad samples
  - Models with good samples can have bad likelihood
  - There is not a good way to quantify how good samples are
- For GANs, it is also hard to even estimate the likelihood
- See “A note on the evaluation of generative models,” Theis et al 2015, for a good overview

# Discrete outputs

- $G$  must be differentiable
- Cannot be differentiable if output is discrete
- Possible workarounds:
  - REINFORCE (Williams 1992)
  - Concrete distribution (Maddison et al 2016) or Gumbel-softmax (Jang et al 2016)
  - Learn distribution over continuous embeddings, decode to discrete

# Supervised Discriminator



(Odena 2016, Salimans et al 2016)

# Semi-Supervised Classification

## MNIST (Permutation Invariant)

Model	Number of incorrectly predicted test examples for a given number of labeled samples			
	20	50	100	200
DGN [21]			333 ± 14	
Virtual Adversarial [22]			212	
CatGAN [14]			191 ± 10	
Skip Deep Generative Model [23]			132 ± 7	
Ladder network [24]			106 ± 37	
Auxiliary Deep Generative Model [23]			96 ± 2	
Our model	1677 ± 452	221 ± 136	93 ± 6.5	90 ± 4.2
Ensemble of 10 of our models	1134 ± 445	142 ± 96	86 ± 5.6	81 ± 4.3

(Salimans et al 2016)

# Semi-Supervised Classification

## CIFAR-10

Model	Test error rate for a given number of labeled samples			
	1000	2000	4000	8000
Ladder network [24]			20.40±0.47	
CatGAN [14]			19.58±0.46	
Our model	21.83±2.01	19.61±2.09	18.63±2.32	17.72±1.82
Ensemble of 10 of our models	19.22±0.54	17.25±0.66	15.59±0.47	14.87±0.89

## SVHN

Model	Percentage of incorrectly predicted test examples for a given number of labeled samples		
	500	1000	2000
DGN [21]		36.02±0.10	
Virtual Adversarial [22]			24.63
Auxiliary Deep Generative Model [23]			22.86
Skip Deep Generative Model [23]			16.61±0.24
Our model	18.44 ± 4.8	8.11 ± 1.3	6.16 ± 0.58
Ensemble of 10 of our models		5.88 ± 1.0	

(Salimans et al 2016)

# Learning interpretable latent codes / controlling the generation process



(a) Azimuth (pose)

(b) Elevation



(c) Lighting

(d) Wide or Narrow

InfoGAN (Chen et al 2016)

# RL connections

- GANs interpreted as actor-critic (Pfau and Vinyals 2016)
- GANs as inverse reinforcement learning (Finn et al 2016)
- GANs for imitation learning (Ho and Ermon 2016)

# Finding equilibria in games

- Simultaneous SGD on two players costs may not converge to a Nash equilibrium
- In finite spaces, fictitious play provides a better algorithm
- What to do in continuous spaces?
  - Unrolling is an expensive solution; is there a cheap one?

# Other Games in AI

- Board games (checkers, chess, Go, etc.)
- Robust optimization / robust control
  - for security/safety, e.g. resisting adversarial examples
- Domain-adversarial learning for domain adaptation
- Adversarial privacy
- Guided cost learning
- ...

# Exercise 3

- In this exercise, we will derive the maximum likelihood cost for GANs.
- We want to solve for  $f(x)$ , a cost function to be applied to every sample from the generator:

$$J^{(G)} = \mathbb{E}_{\mathbf{x} \sim p_g} f(\mathbf{x})$$

- Show the following:

$$\frac{\partial}{\partial \theta} J^{(G)} = \mathbb{E}_{x \sim p_g} f(x) \frac{\partial}{\partial \theta} \log p_g(x)$$

- What should  $f(x)$  be?

# Solution

- To show that  $\frac{\partial}{\partial \theta} J^{(G)} = \mathbb{E}_{x \sim p_g} f(x) \frac{\partial}{\partial \theta} \log p_g(x)$ 
  - Expand the expectation to an integral

$$\frac{\partial}{\partial \theta} \mathbb{E}_{x \sim p_g} f(x) = \frac{\partial}{\partial \theta} \int p_g(x) f(x) dx$$

- Assume that Leibniz's rule may be used

$$\int f(x) \frac{\partial}{\partial \theta} p_g(x) dx$$

- Use the identity

$$\frac{\partial}{\partial \theta} p_g(x) = p_g(x) \frac{\partial}{\partial \theta} \log p_g(x)$$

# Solution

- We now know  $\frac{\partial}{\partial \theta} J^{(G)} = \mathbb{E}_{x \sim p_g} f(x) \frac{\partial}{\partial \theta} \log p_g(x)$
- The KL gradient is  $-\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \frac{\partial}{\partial \theta} \log p_g(\mathbf{x})$
- We can do an importance sampling trick
$$f(\mathbf{x}) = -\frac{p_{\text{data}}(\mathbf{x})}{p_g(\mathbf{x})}$$
- Note that we must *copy* the density  $p_g$  or the derivatives will double-count

# Solution

- We want  $f(\mathbf{x}) = -\frac{p_{\text{data}}(\mathbf{x})}{p_g(\mathbf{x})}$
- We know that  $D(\mathbf{x}) = \sigma(a(\mathbf{x})) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_g(\mathbf{x})}$
- By algebra  $f(x) = -\exp(a(\mathbf{x}))$

# Roadmap

- Why study generative modeling?
- How do generative models work? How do GANs compare to others?
- How do GANs work?
- Tips and tricks
- Combining GANs with other methods

# Plug and Play Generative Models

- New state of the art generative model (Nguyen et al 2016) released days before NIPS
- Generates 227x227 realistic images from all ImageNet classes
- Combines adversarial training, moment matching, denoising autoencoders, and Langevin sampling

# PPGN Samples



redshank

ant

monastery



volcano

(Nguyen et al 2016)

# PPGN for caption to image



oranges on a table next to a liquor bottle

(Nguyen et al 2016)

# Basic idea

- Langevin sampling repeatedly adds noise and gradient of  $\log p(x,y)$  to generate samples (Markov chain)
- Denoising autoencoders estimate the required gradient
- Use a special denoising autoencoder that has been trained with multiple losses, including a GAN loss, to obtain best results

# Sampling without class gradient



$\text{epsilon1} = 0, \text{epsilon2} = 1e-5$   
(Nguyen et al 2016)

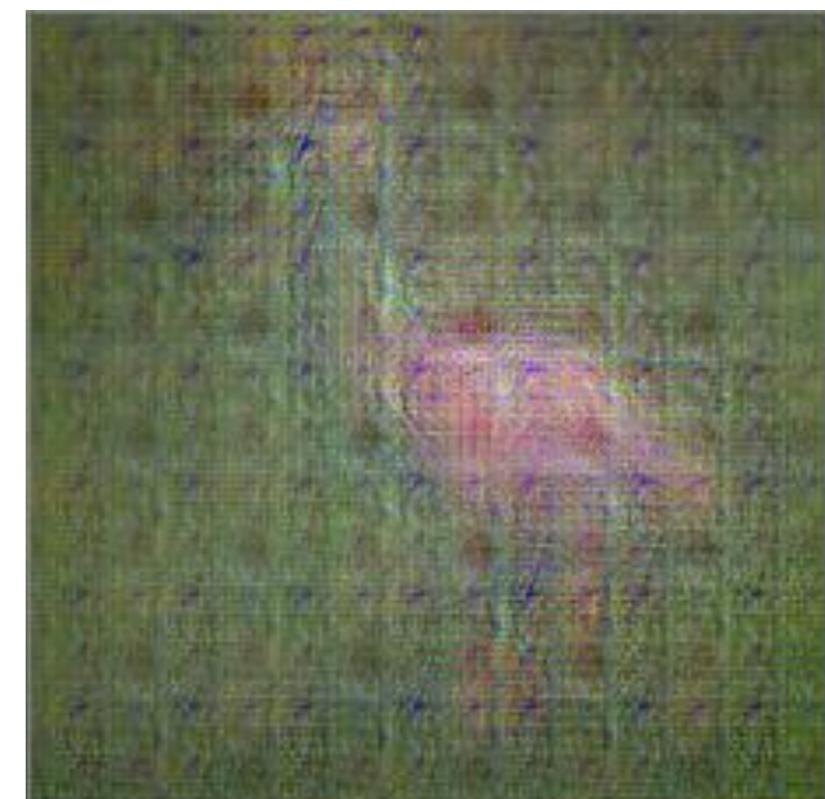
# GAN loss is a key ingredient



Raw data



Reconstruction  
by PPGN



Reconstruction  
by PPGN  
without GAN

Images from Nguyen et al 2016

First observed by Dosovitskiy et al 2016

# Conclusion

- GANs are generative models that use supervised learning to approximate an intractable cost function
- GANs can simulate many cost functions, including the one used for maximum likelihood
- Finding Nash equilibria in high-dimensional, continuous, non-convex games is an important open research problem
- GANs are a key ingredient of PPGNs, which are able to generate compelling high resolution samples from diverse image classes