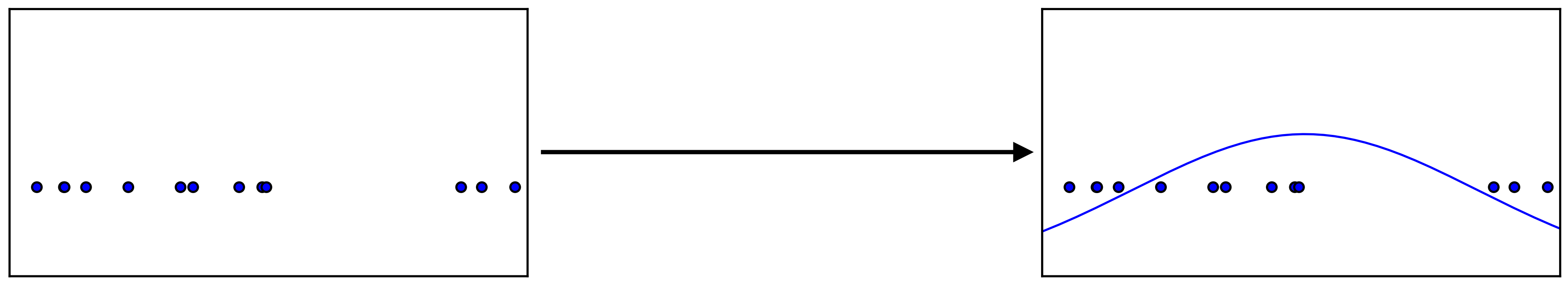
Generative Adversarial Networks (GANs)

Ian Goodfellow, OpenAI Research Scientist

Guest lecture for UC Berkeley COMPSCI 294, 2016-10-3

Generative Modeling

- Density estimation



- Sample generation

Training examples Model samples

(Goodfellow 2016)

Why study generative models?

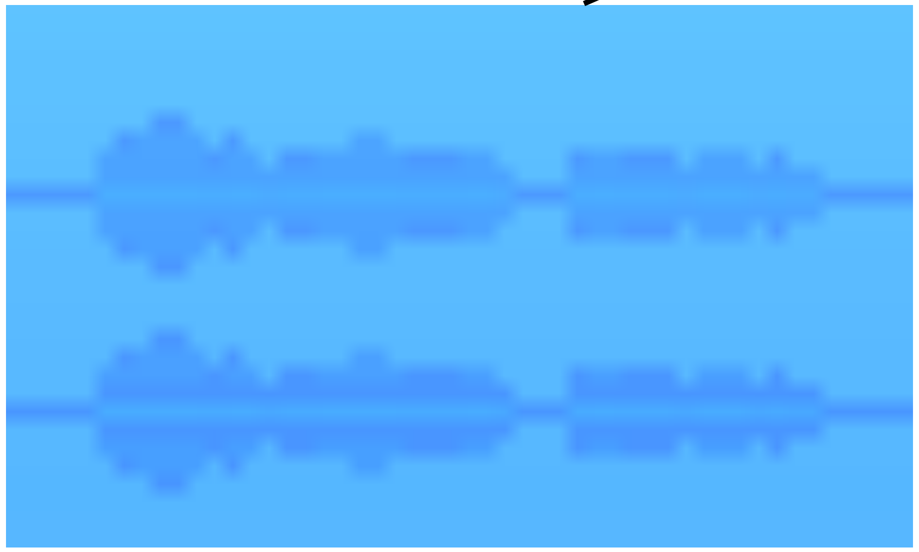
- Forces us to learn to work with high-dimensional, complicated probability distributions
- Simulate possible futures for planning or simulated RL
- Handle missing data (in particular, semi-supervised learning)
- Some applications actually require generation

Conditional Generative Modeling

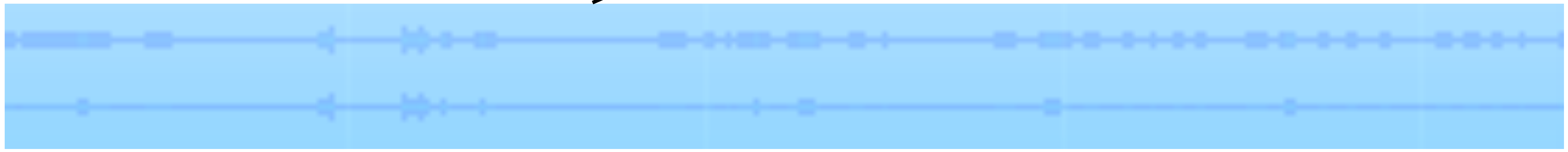
SO, I REMEMBER WHEN THEY CAME HERE

Semi-supervised learning

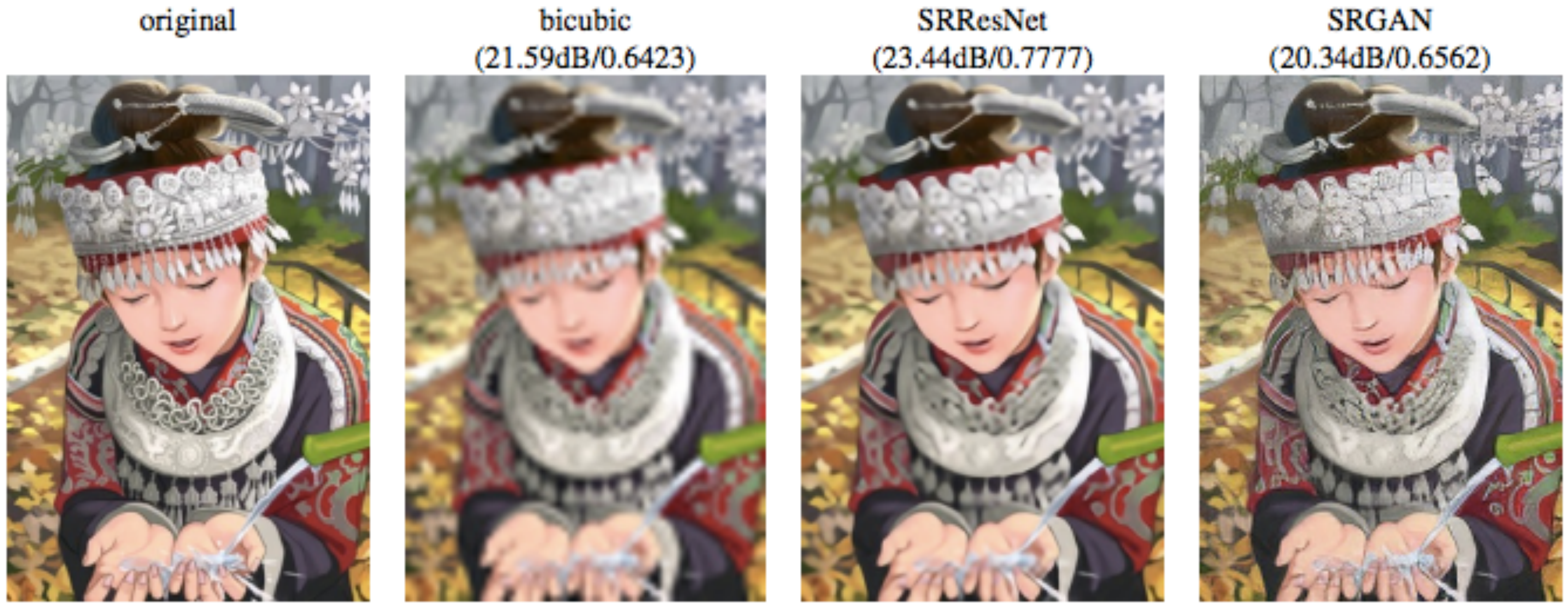
SO, I REMEMBER WHEN THEY CAME HERE



???

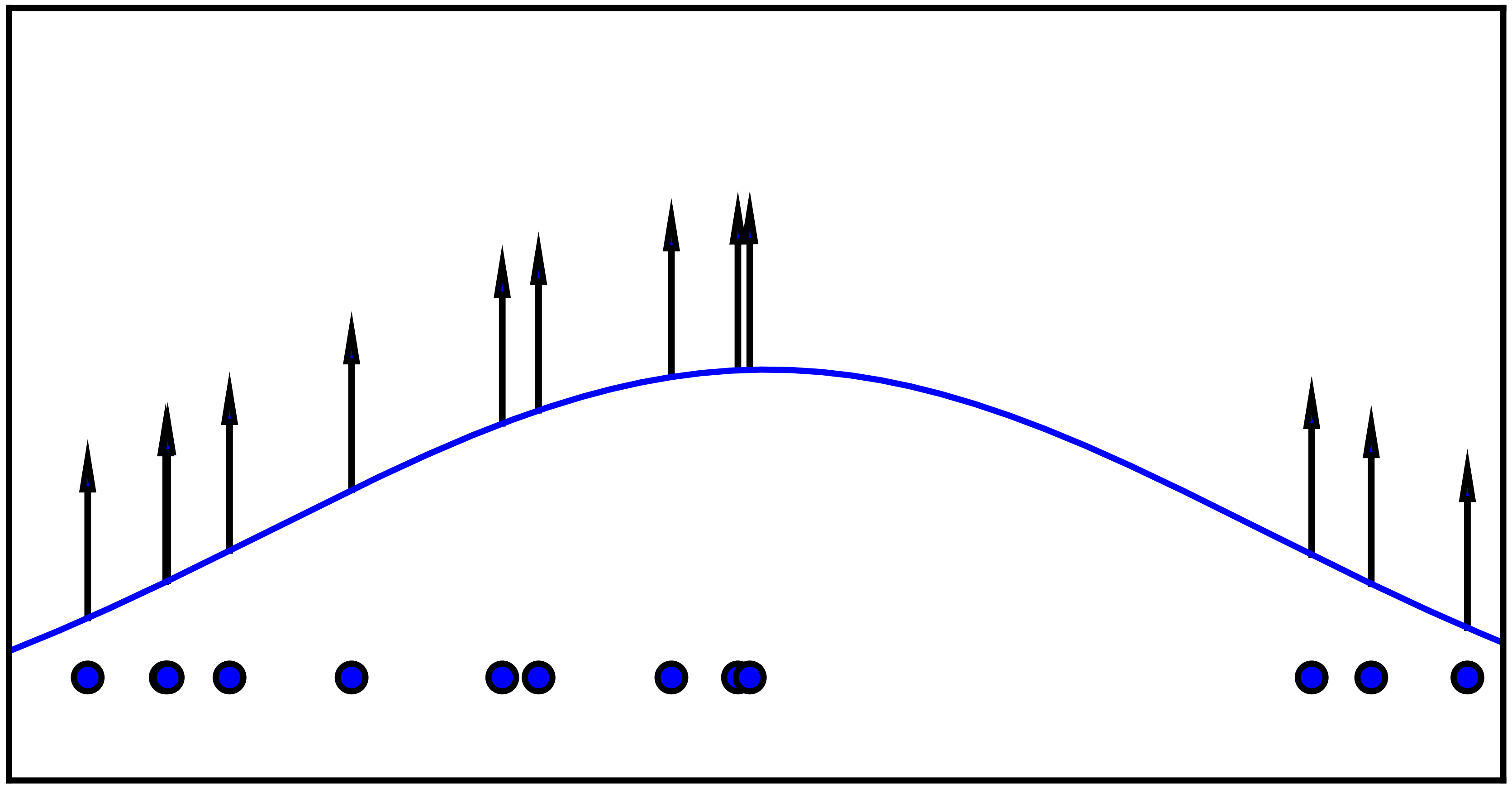


Single Image Super-Resolution



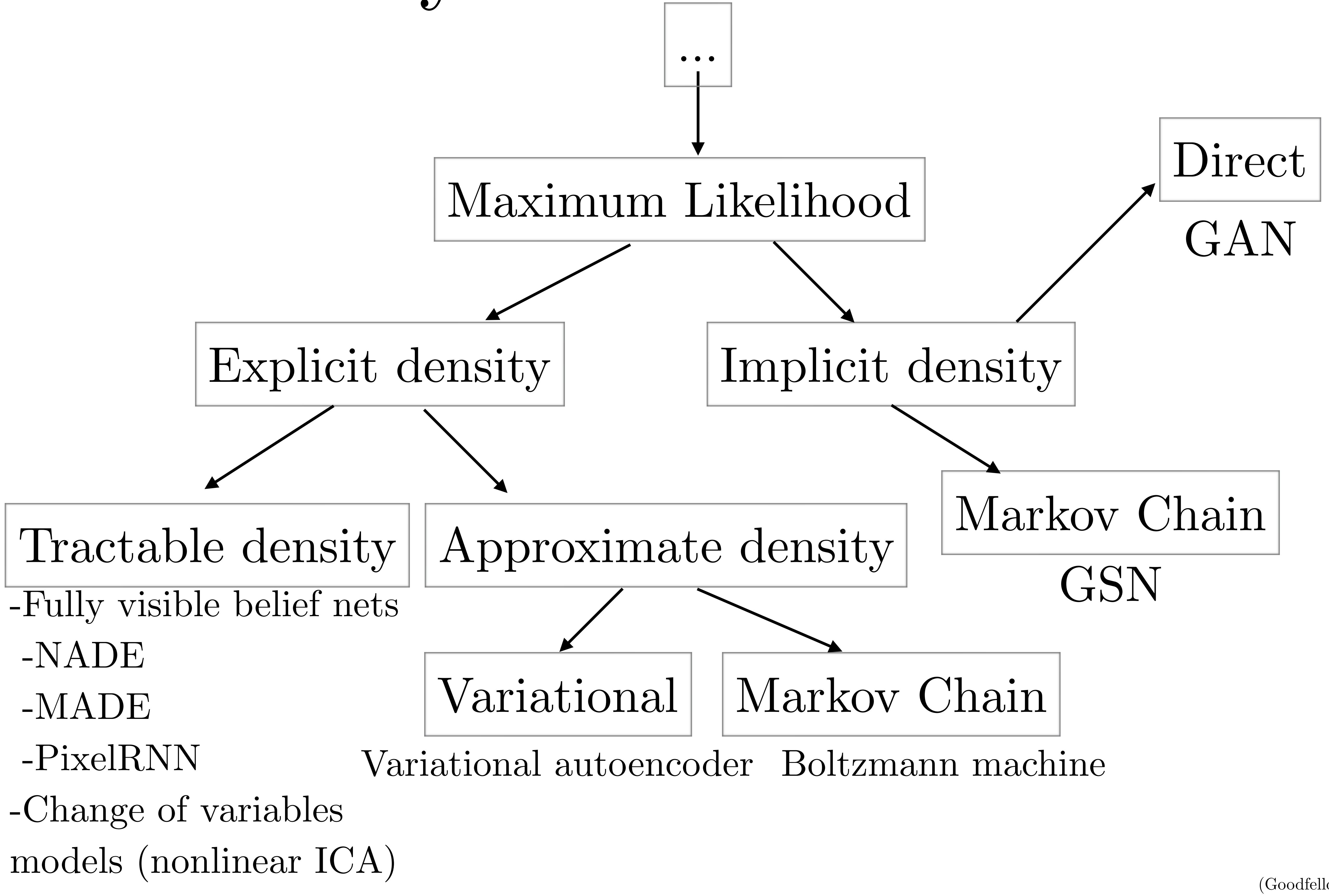
(Ledig et al 2016)

Maximum Likelihood



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

Taxonomy of Generative Models



Fully Visible Belief Nets

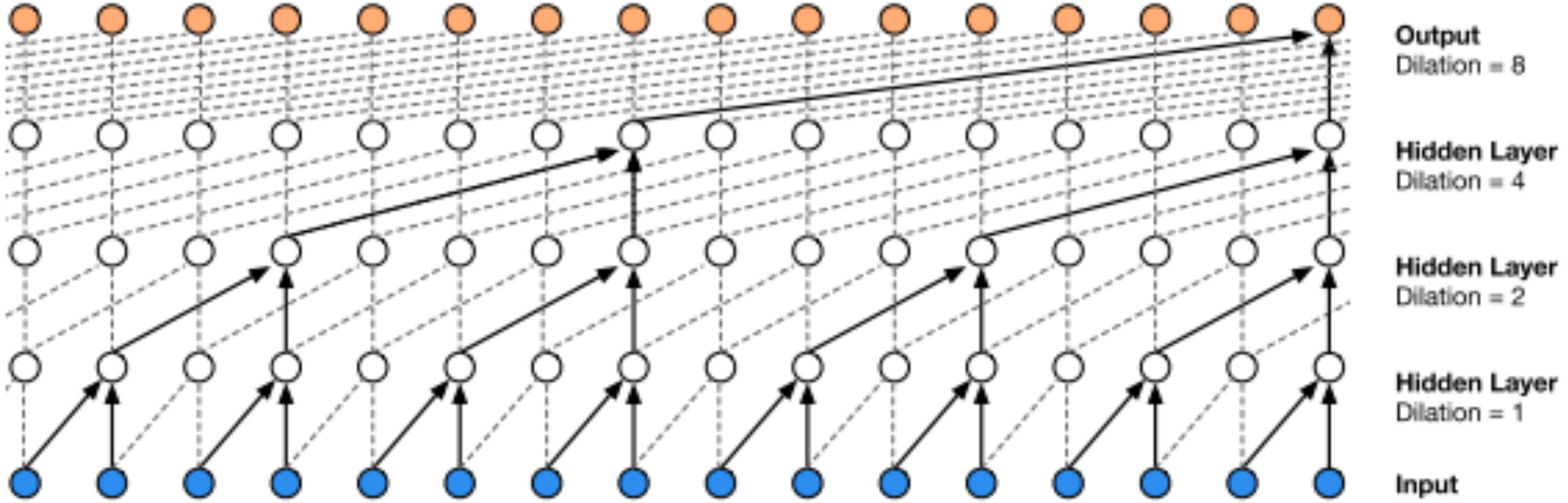
- Explicit formula based on chain rule: (Frey et al, 1996)

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

- Disadvantages:
  - O(n) sample generation cost
  - Currently, do not learn a useful latent representation

PixelCNN elephants
(van den Ord et al 2016)

WaveNet



Amazing quality

Sample generation slow

(Not sure how much is just research code not being optimized and how much is intrinsic)

Two minutes to synthesize one second of audio

Change of Variables

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

e.g. Nonlinear ICA (Hyvärinen 1999)

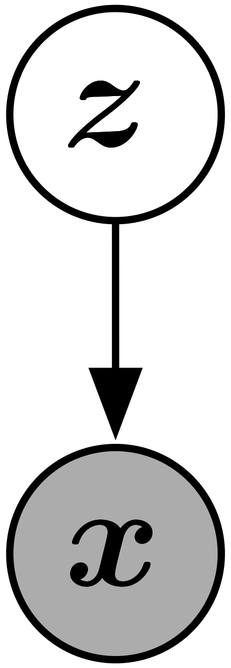
Disadvantages:

- 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)



$$\log p(\boldsymbol{x}) \geq \log p(\boldsymbol{x}) - D_{\mathrm{KL}}\left(q(\boldsymbol{z}) \| p(\boldsymbol{z} \mid \boldsymbol{x})\right)$$
$$= \mathbb{E}_{\boldsymbol{z} \sim q} \log p(\boldsymbol{x}, \boldsymbol{z}) + H(q)$$

CIFAR-10 samples
(Kingma et al 2016)

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

Boltzmann Machines

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

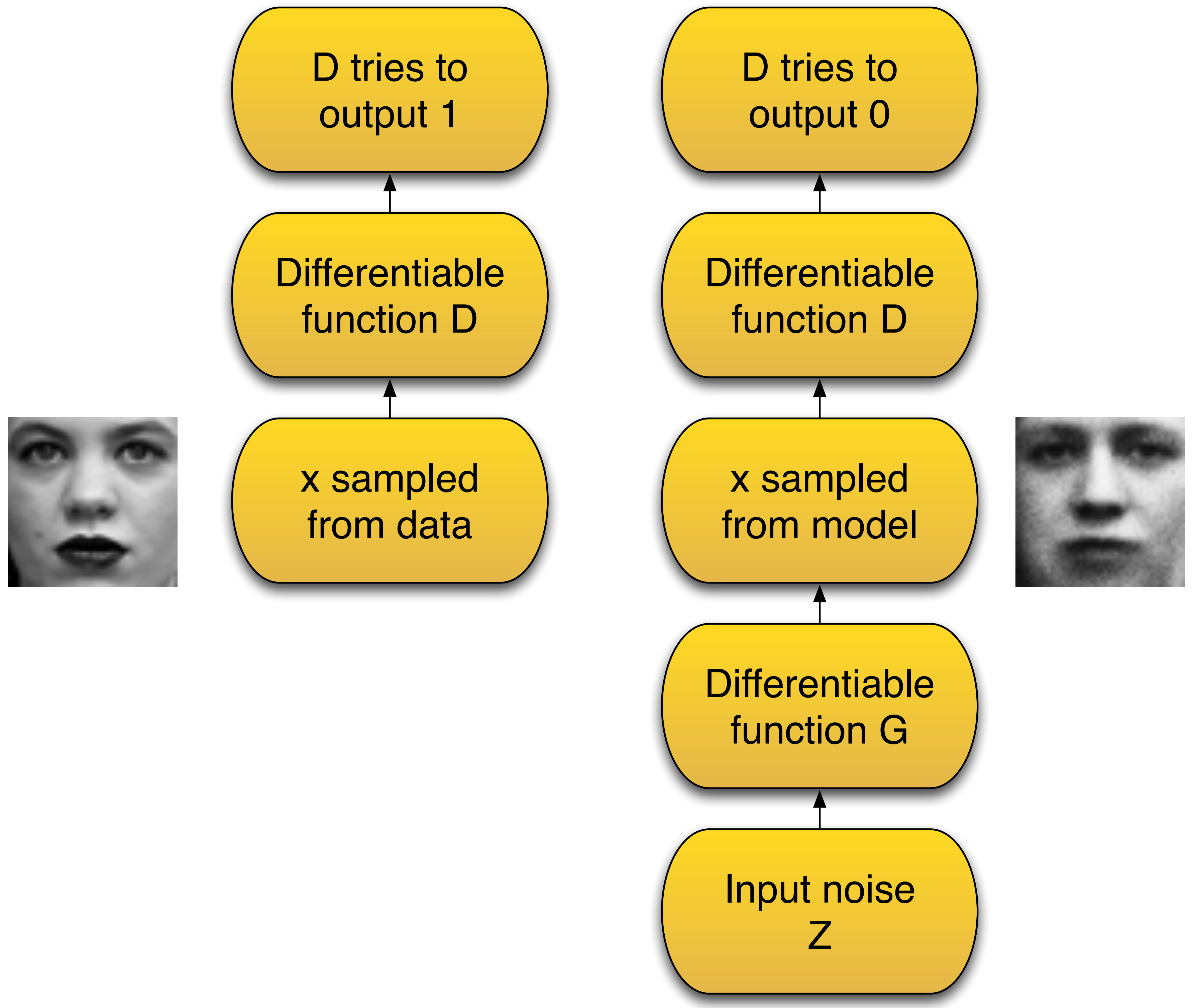
$$Z = \sum_{\boldsymbol{x}} \sum_{\boldsymbol{z}} \exp\left(-E(\boldsymbol{x}, \boldsymbol{z})\right)$$

- 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

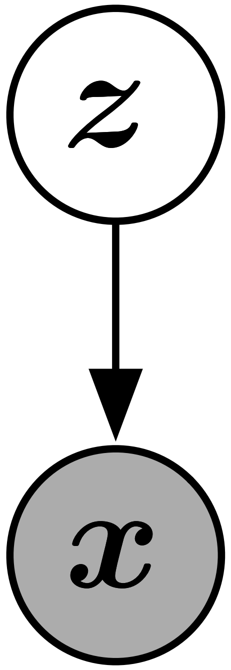
Adversarial Nets Framework



(Goodfellow 2016)

Generator Network

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



-Must be differentiable
 - In theory, could use REINFORCE for discrete variables
- 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

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

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

Non-Saturating Game

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

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

-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

Maximum Likelihood Game

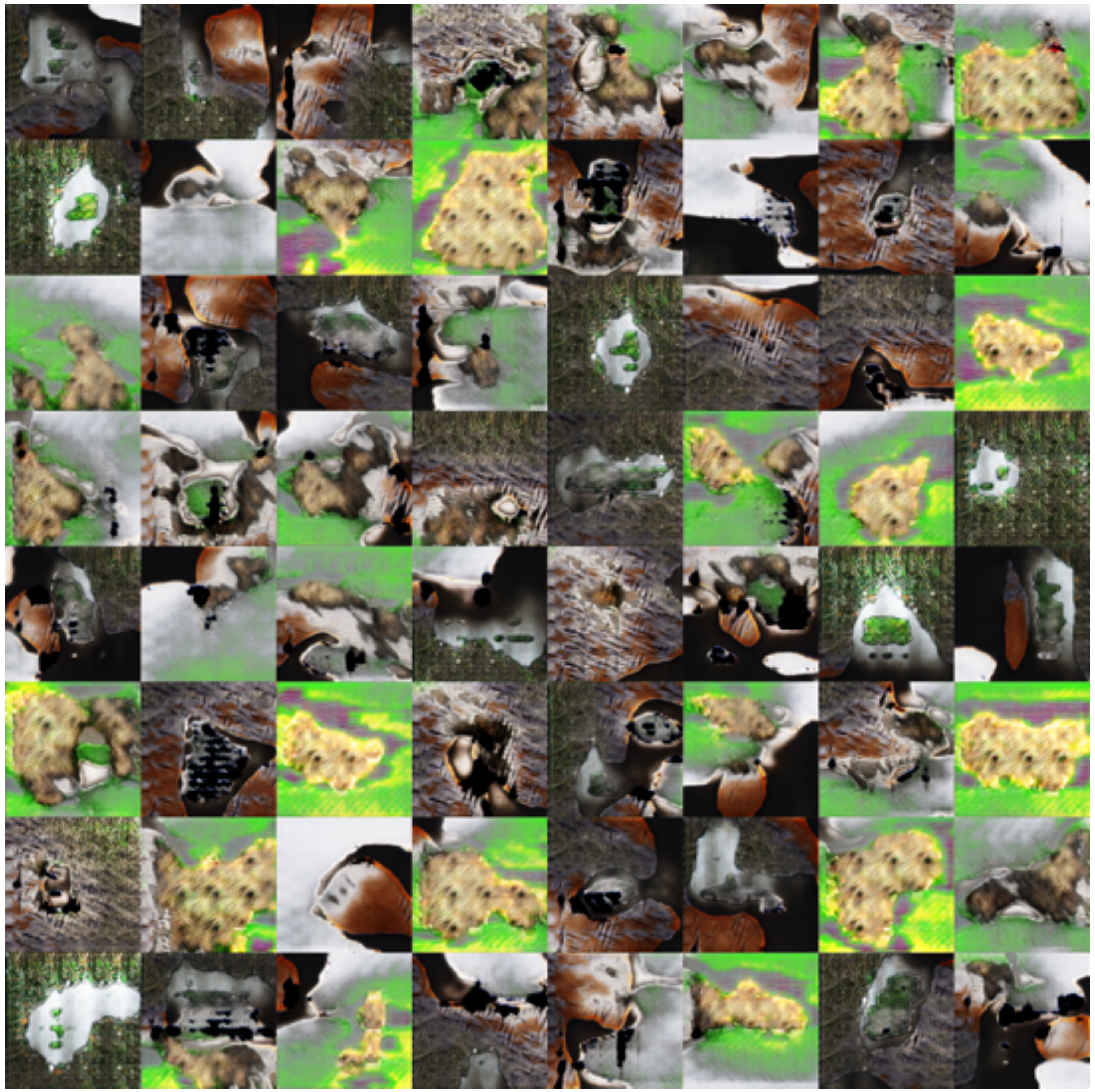
$$J^{(D)} = -\frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}} \log D(\boldsymbol{x}) - \frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log\left(1 - D\left(G(\boldsymbol{z})\right)\right)$$

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

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

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

Maximum Likelihood Samples

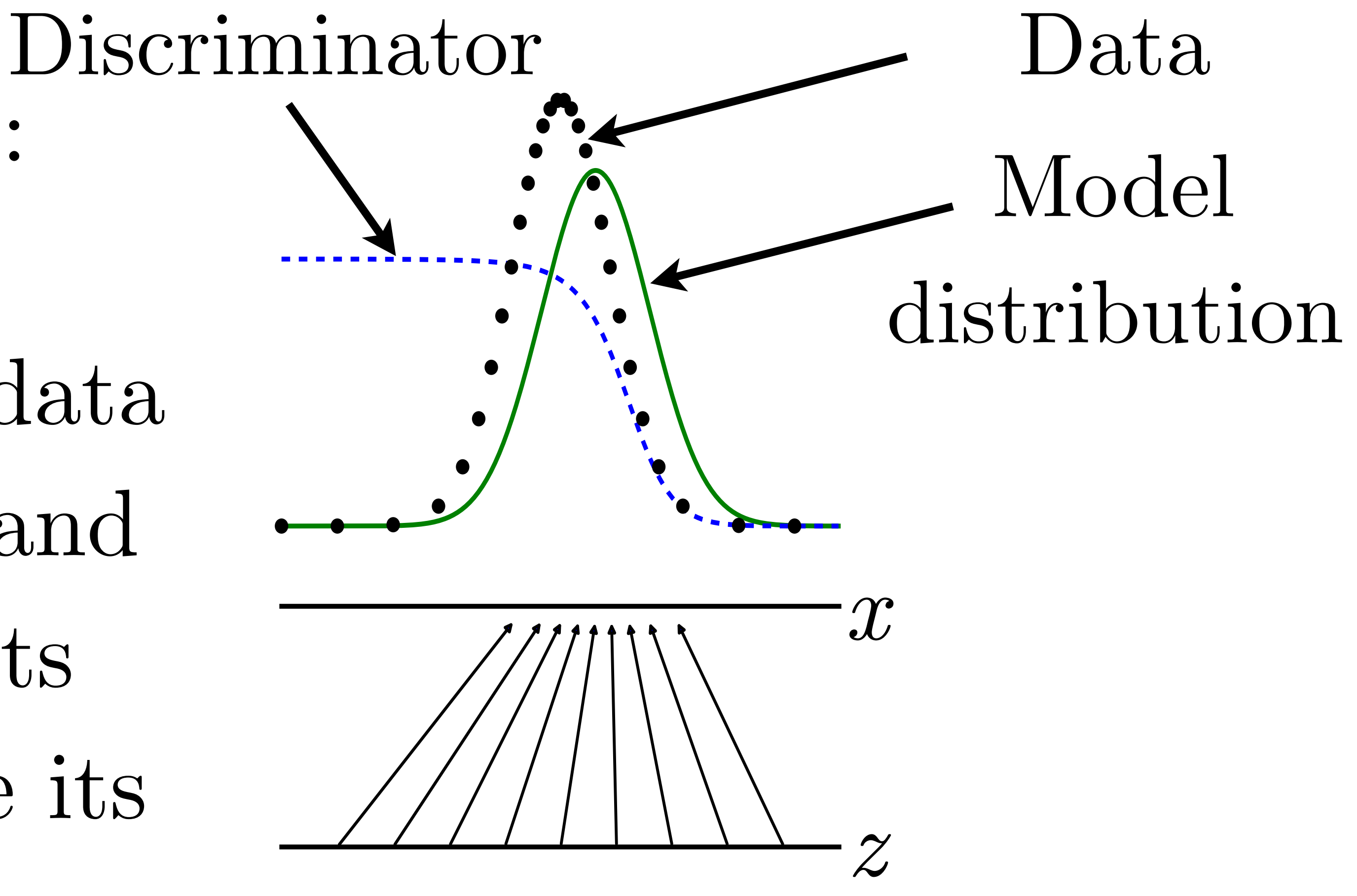


Discriminator Strategy

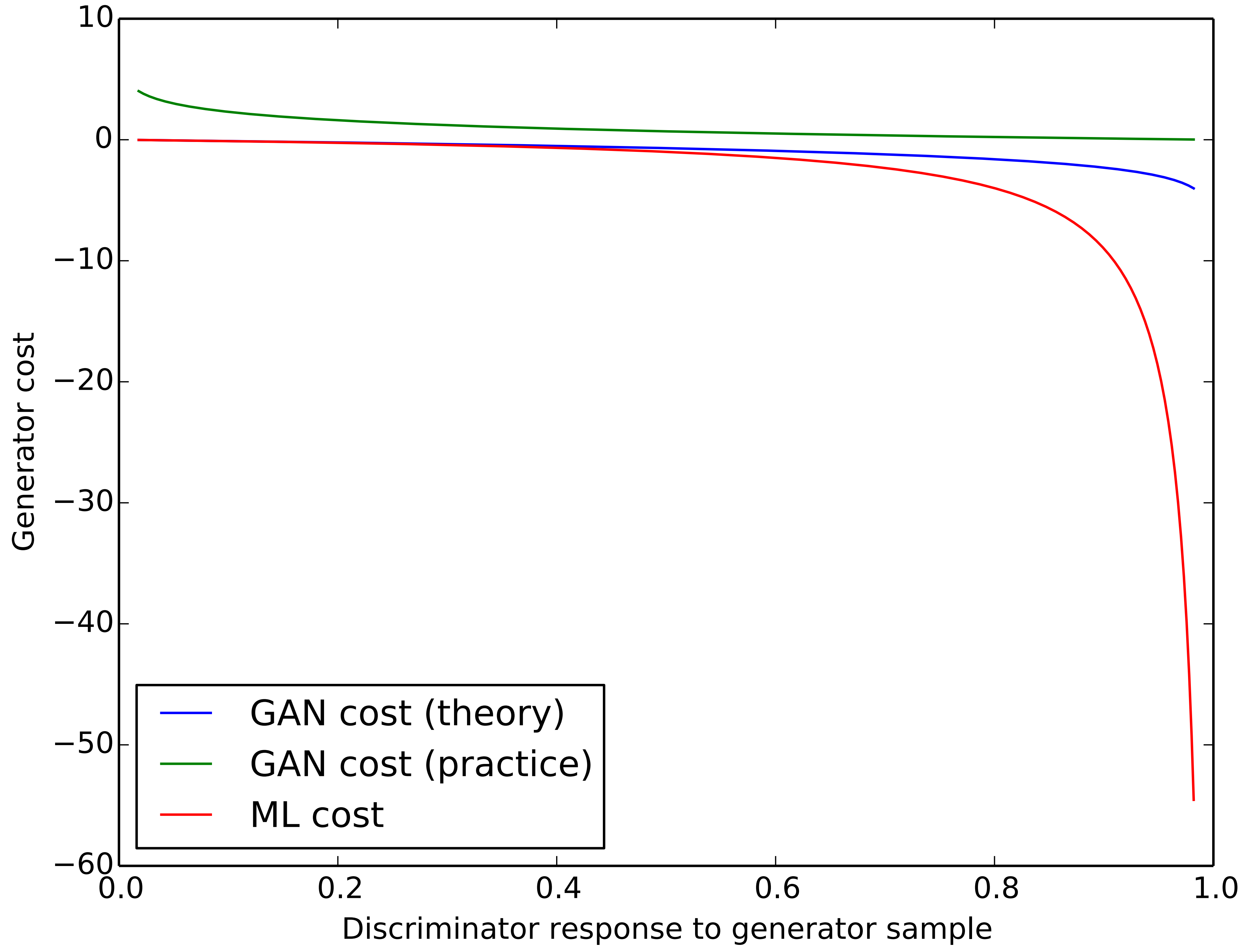
Optimal $D(\boldsymbol{x})$ for any $p_{\text{data}}(\boldsymbol{x})$ and $p_{\text{model}}(\boldsymbol{x})$ is always

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

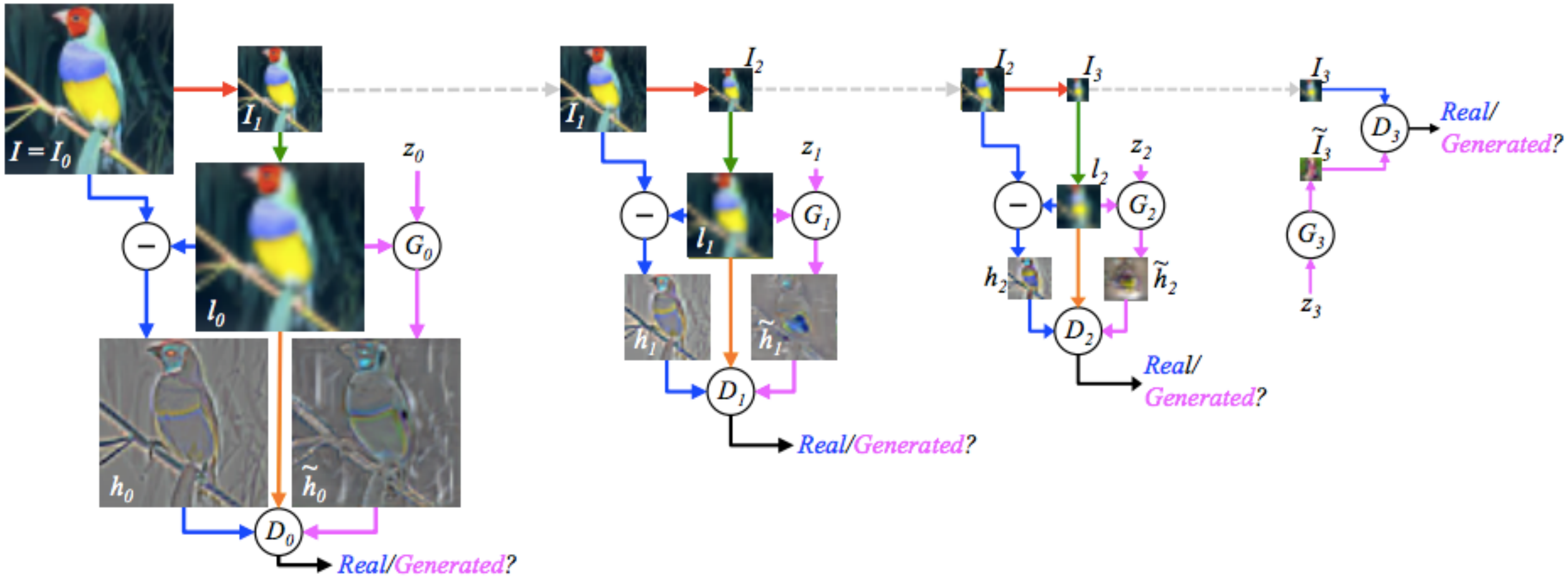
A *cooperative* rather than adversarial view of GANs: the discriminator tries to estimate the ratio of the data and model distributions, and informs the generator of its estimate in order to guide its improvements.



Comparison of Generator Losses



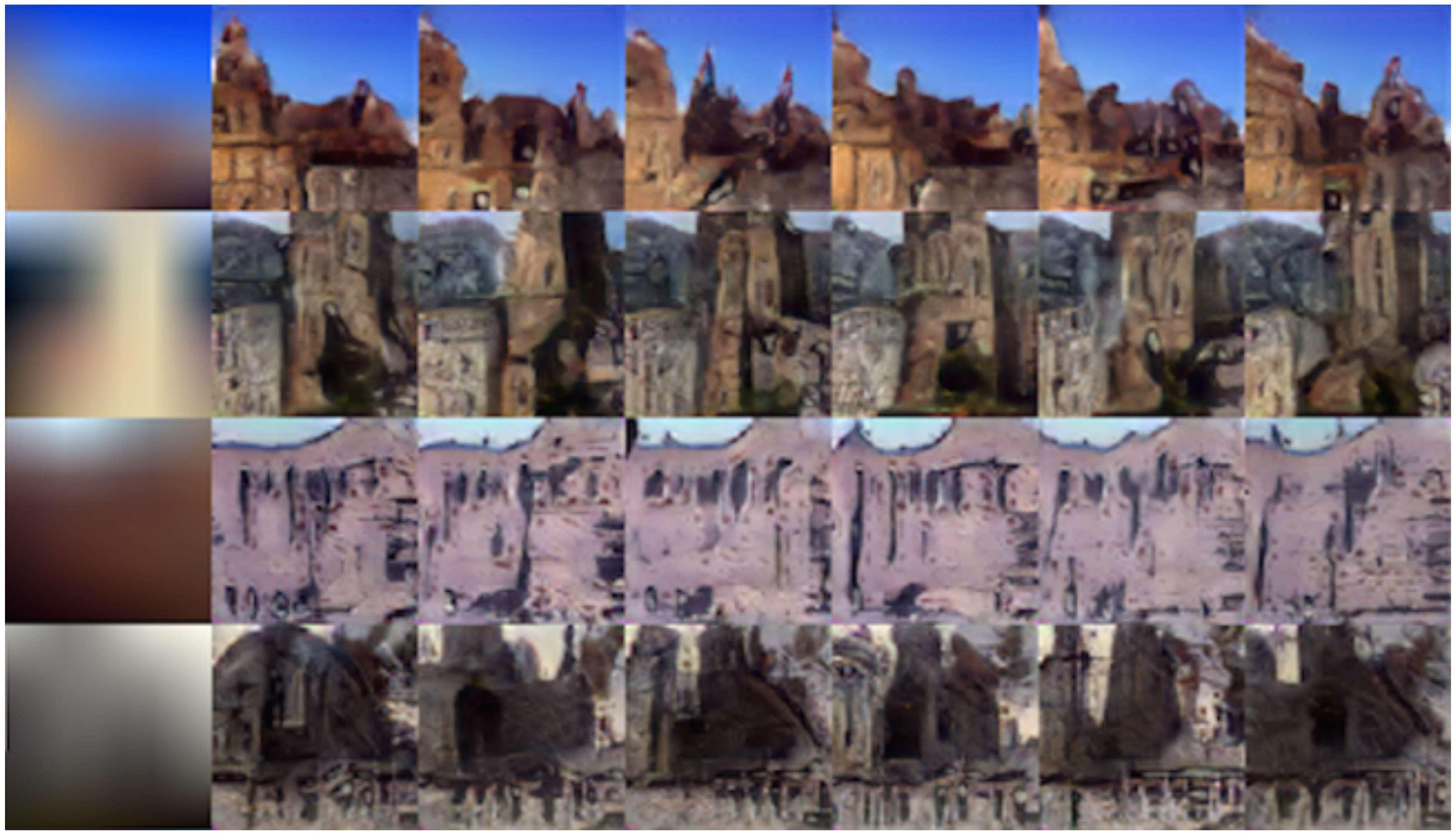
Laplacian Pyramid



(Denton+Chintala et al, 2015)

LAPGAN results

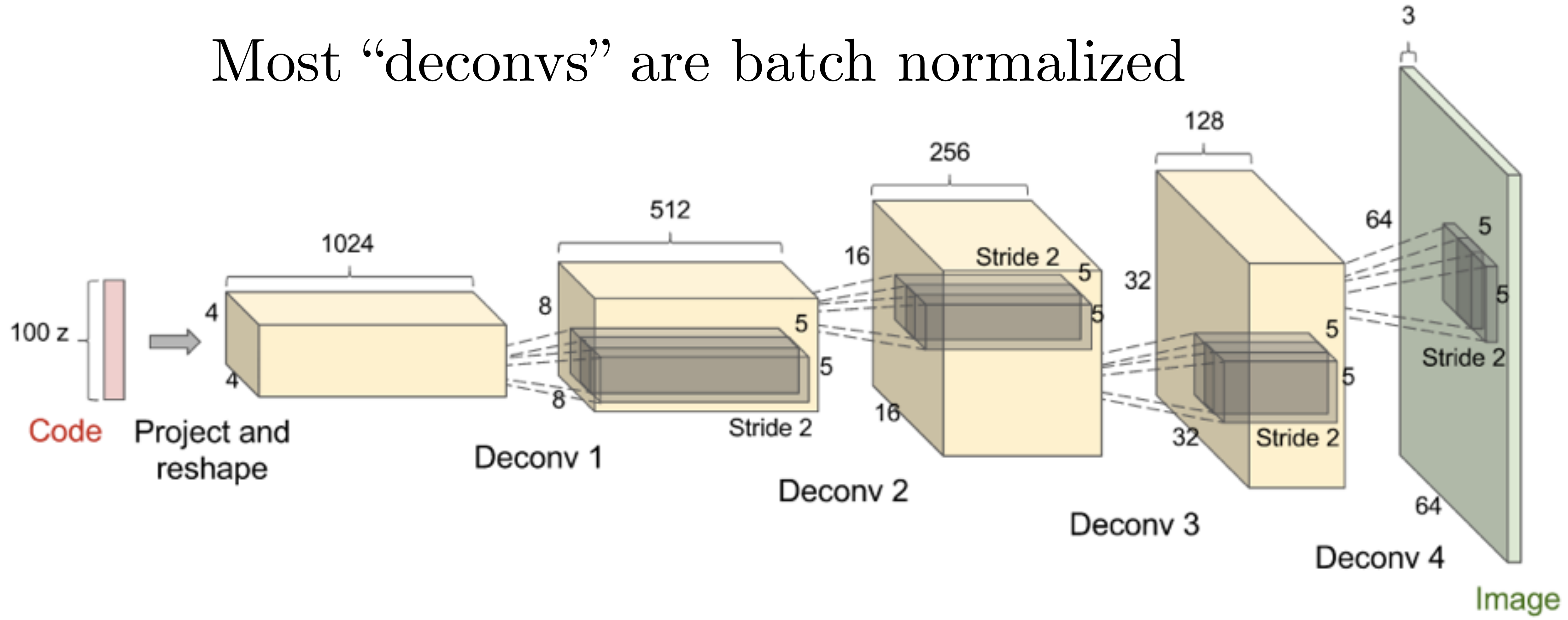
40% of samples mistaken *by humans* for real photos



(Denton+Chintala et al, 2015)

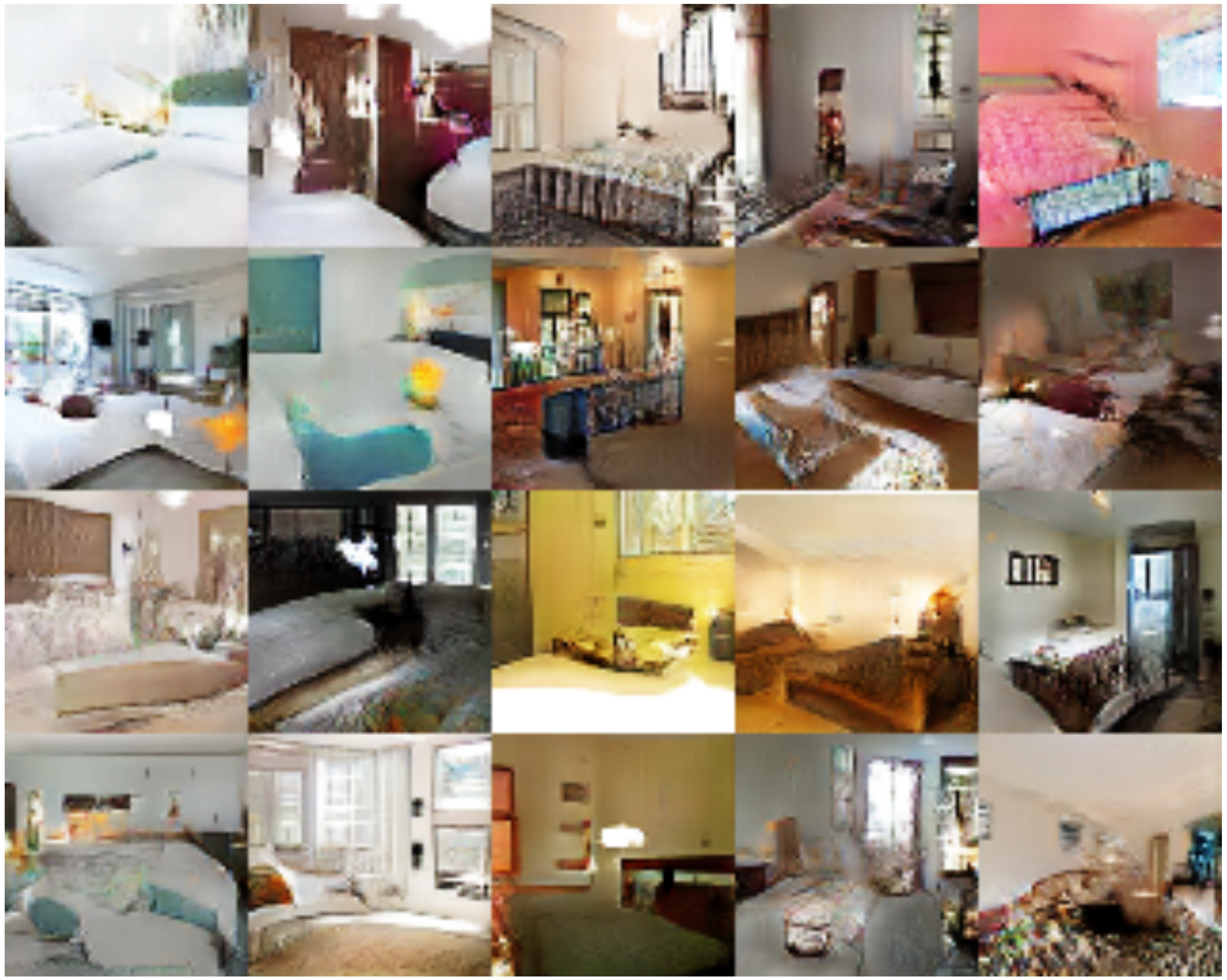
DCGAN Architecture

Most "deconvs" are batch normalized



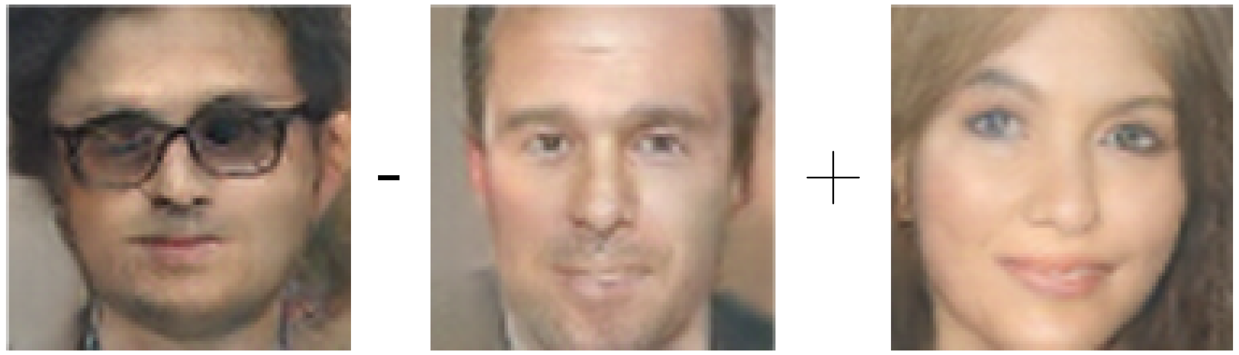
(Radford et al 2015)

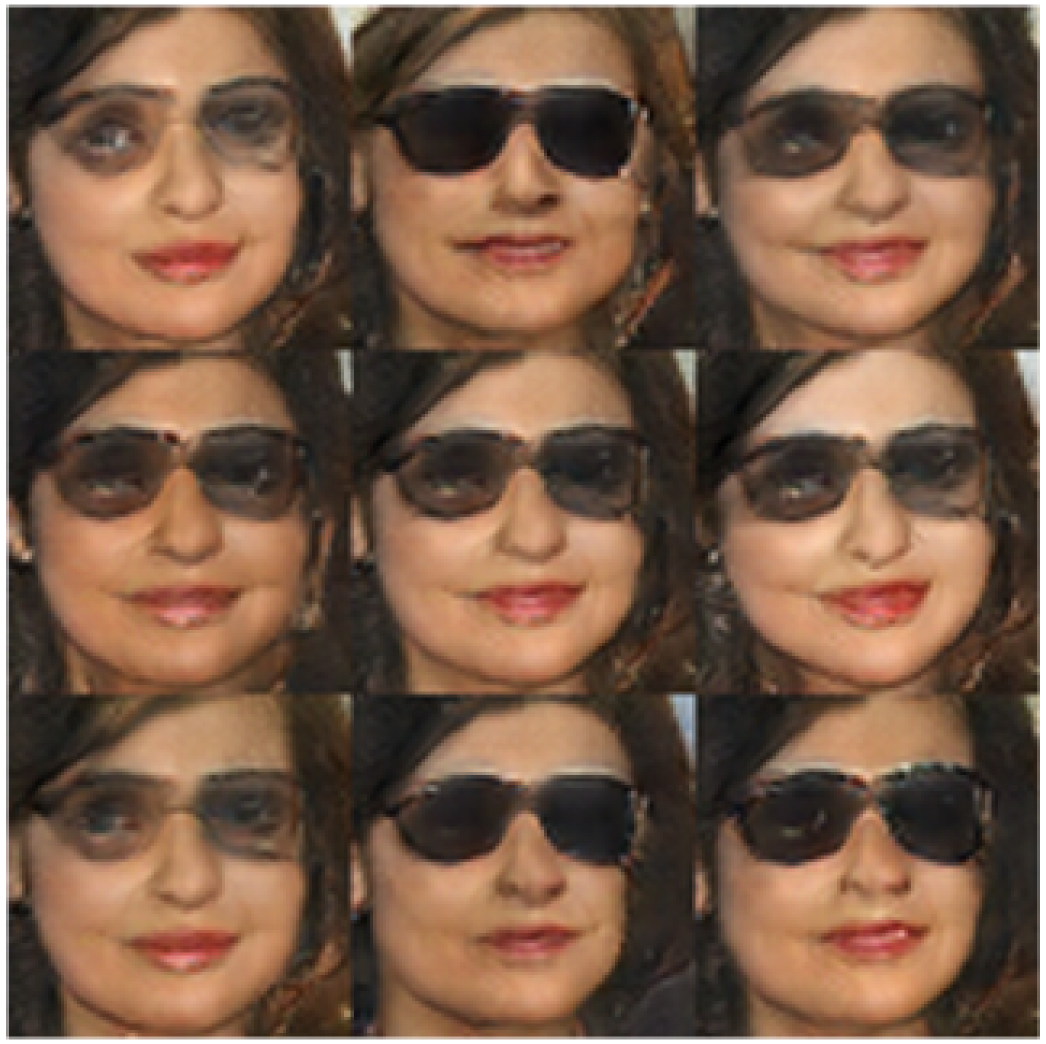
DCGANs for LSUN Bedrooms



(Radford et al 2015)

Vector Space Arithmetic





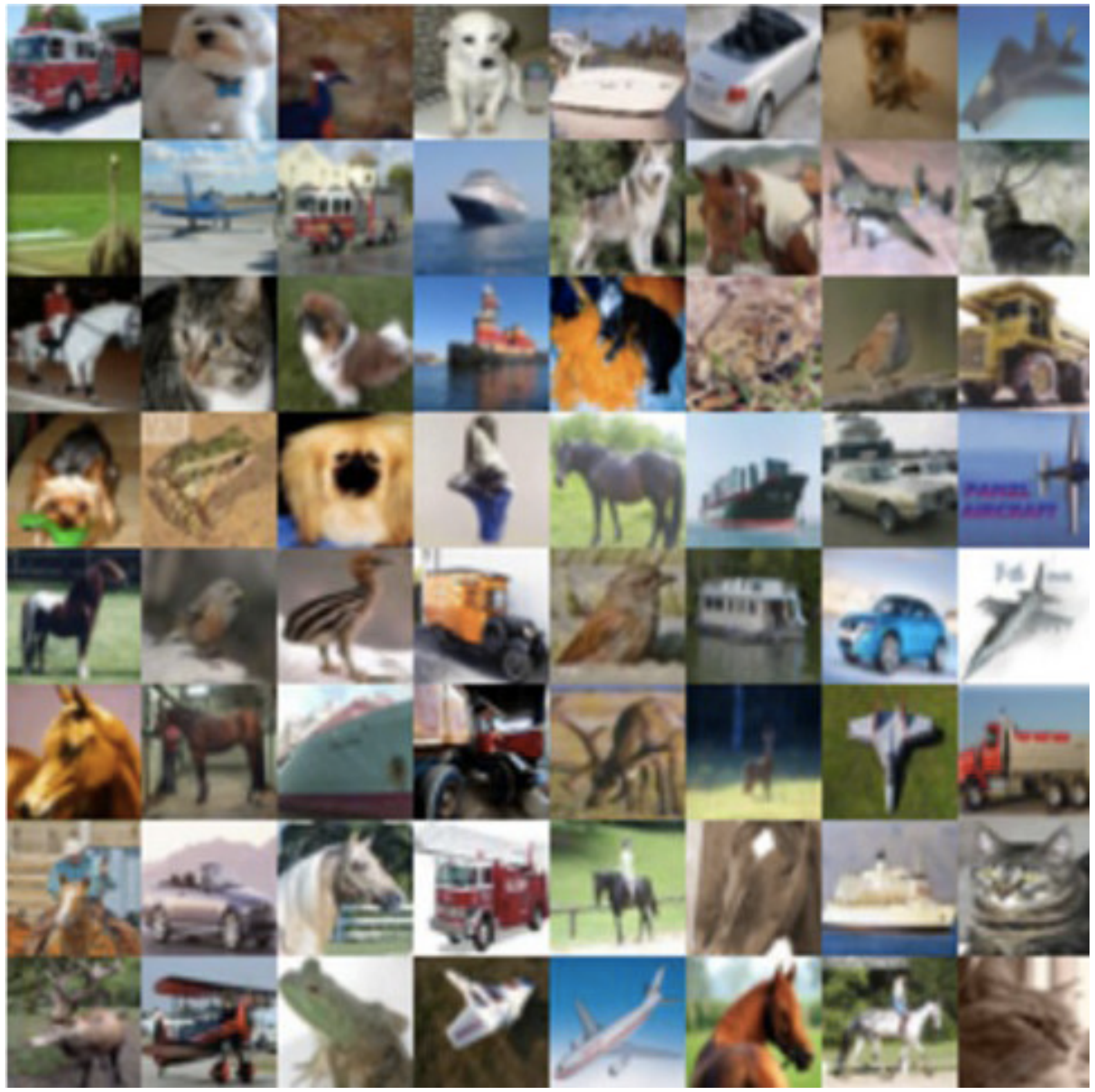
(Goodfellow 2016)

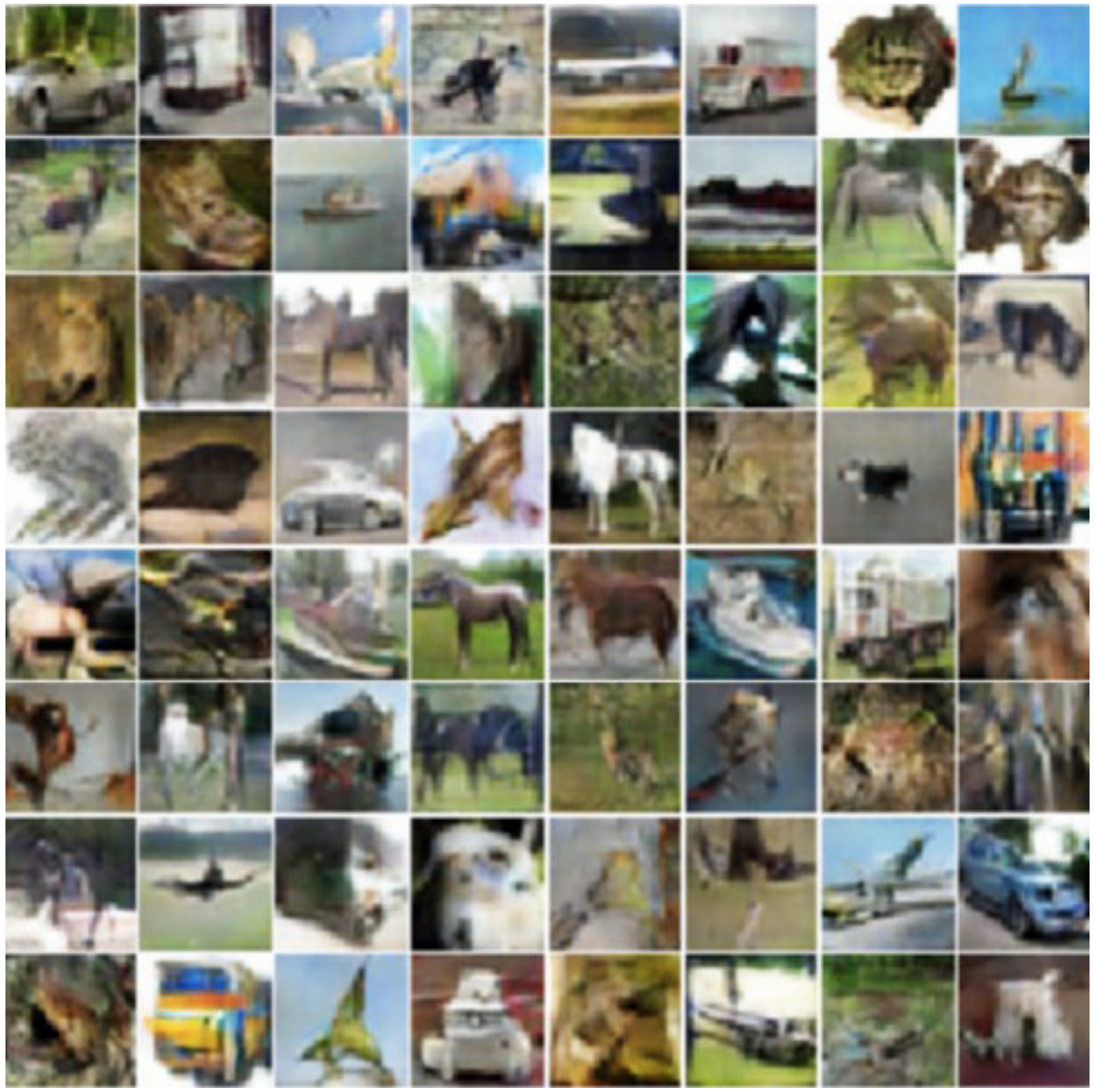
Mode Collapse

- Fully optimizing the discriminator with the generator held constant is safe
- Fully optimizing the generator with the discriminator held constant results in mapping all points to the argmax of the discriminator
- Can partially fix this by adding nearest-neighbor features constructed from the current minibatch to the discriminator (“minibatch GAN”)

  (Salimans et al 2016)

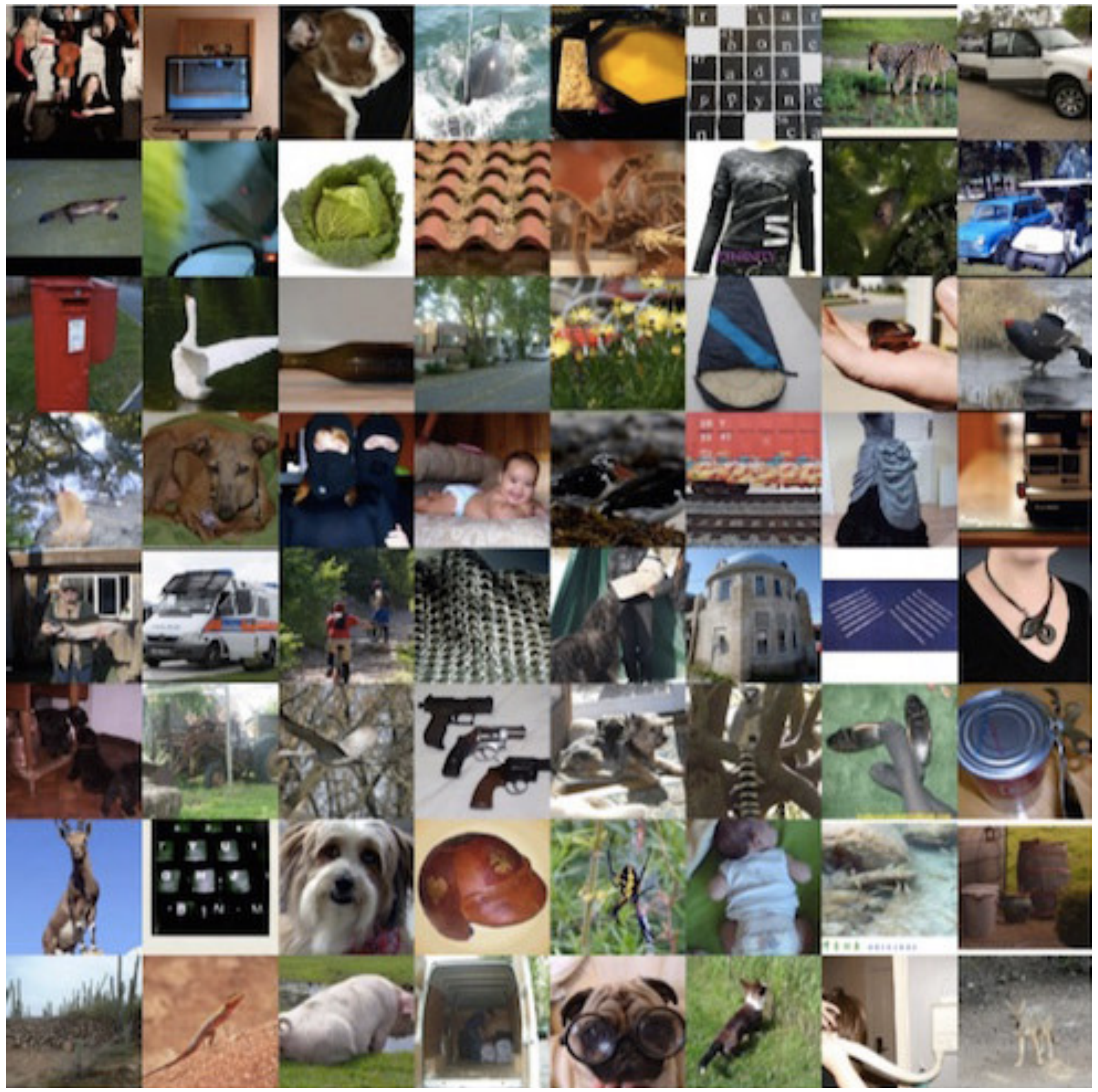
Minibatch GAN on CIFAR

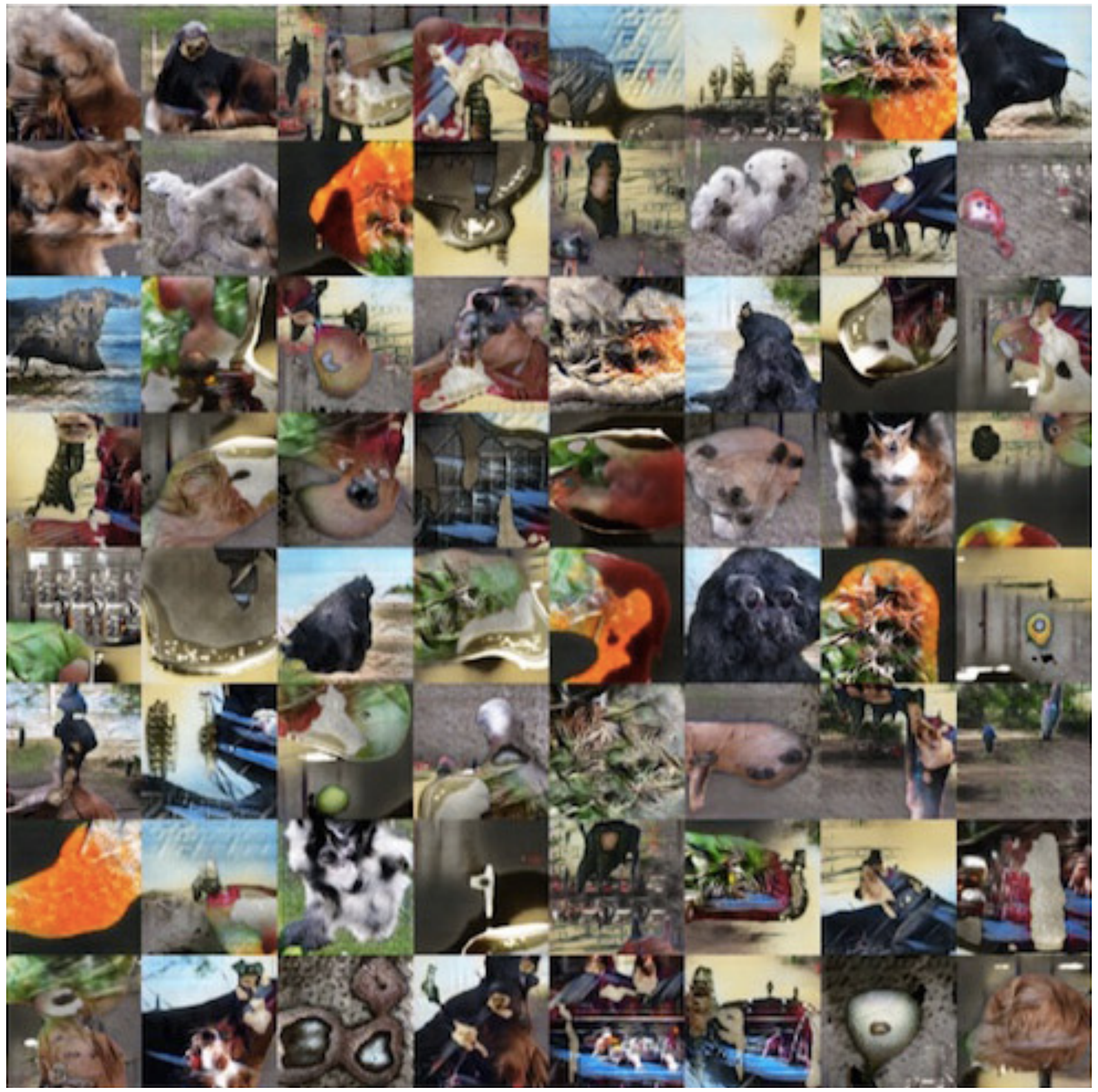

Training Data


Samples

(Salimans et al 2016)

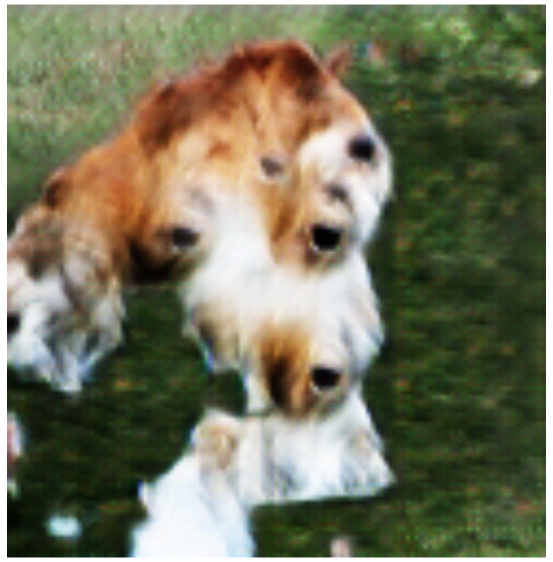
Minibatch GAN on ImageNet





(Salimans et al 2016)

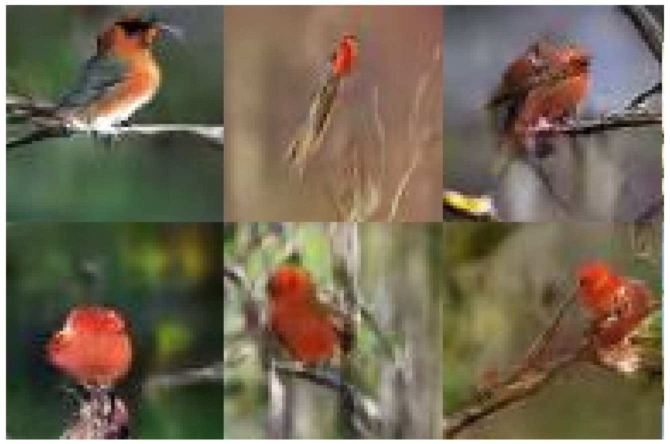
Cherry-Picked Results



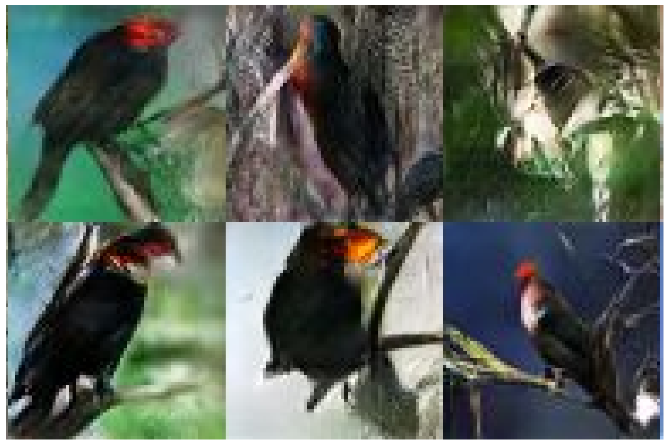
(Goodfellow 2016)

GANs Work Best When Output Entropy is Low

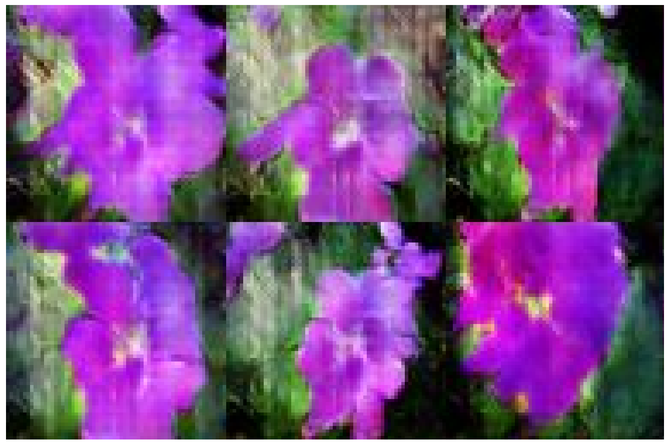
this small bird has a pink breast and crown, and black primaries and secondaries.



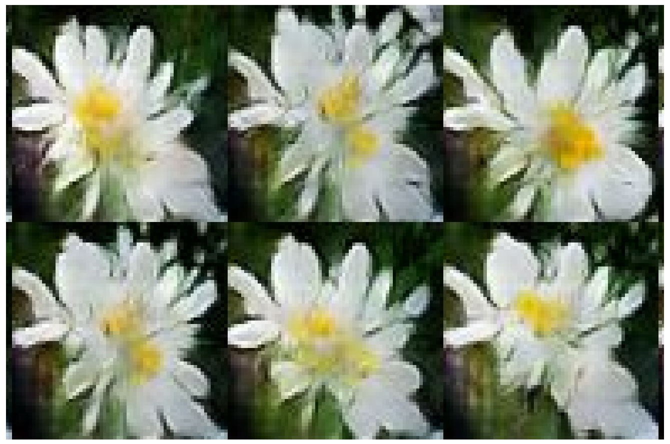
this magnificent fellow is almost all black with a red crest, and white cheek patch.



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

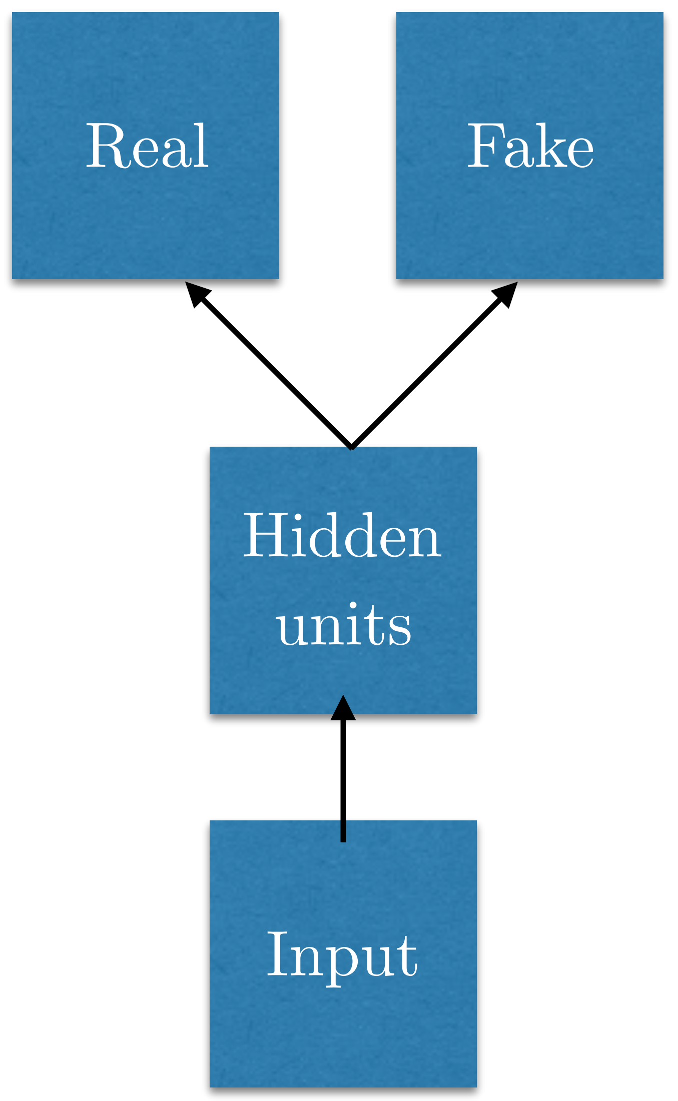


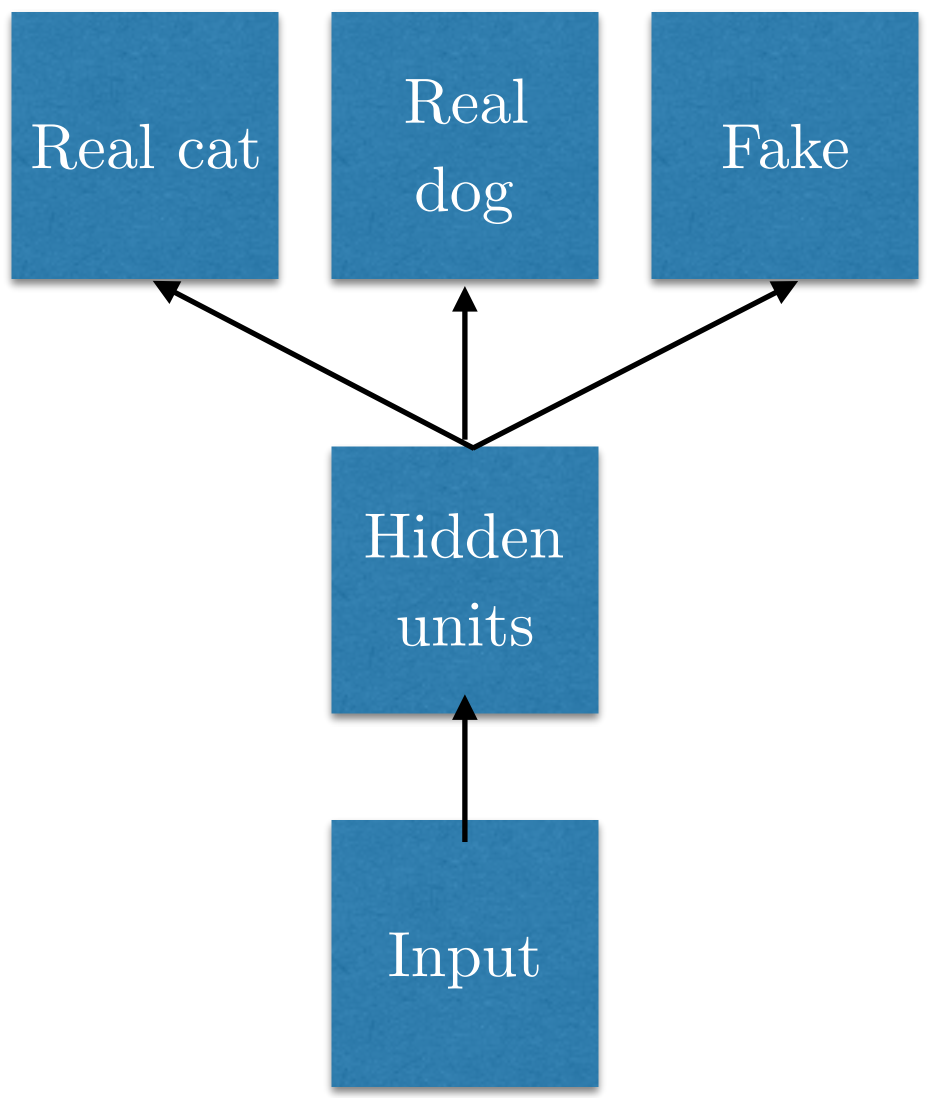
this white and yellow flower have thin white petals and a round yellow stamen



(Reed et al 2016)

Supervised Discriminator





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)

Optimization and Games

Optimization: find a minimum:

$$\boldsymbol{\theta}^* = \operatorname{argmin}_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

Game:

Player 1 controls $\boldsymbol{\theta}^{(1)}$

Player 2 controls $\boldsymbol{\theta}^{(2)}$

Player 1 wants to minimize $J^{(1)}(\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)})$

Player 2 wants to minimize $J^{(2)}(\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)})$

Depending on J functions, they may compete or cooperate.

Games $\supseteq$ optimization

Example:

$\boldsymbol{\theta}^{(1)} = \boldsymbol{\theta}$

$\boldsymbol{\theta}^{(2)} = \{\}$

$J^{(1)}(\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}) = J(\boldsymbol{\theta}^{(1)})$

$J^{(2)}(\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}) = 0$

Nash Equilibrium

- No player can reduce their cost by changing their own strategy:

$$\forall \boldsymbol{\theta}^{(1)}, J^{(1)}(\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)*}) \geq J^{(1)}(\boldsymbol{\theta}^{(1)*}, \boldsymbol{\theta}^{(2)*})$$
$$\forall \boldsymbol{\theta}^{(2)}, J^{(2)}(\boldsymbol{\theta}^{(1)*}, \boldsymbol{\theta}^{(2)}) \geq J^{(2)}(\boldsymbol{\theta}^{(1)*}, \boldsymbol{\theta}^{(2)*})$$

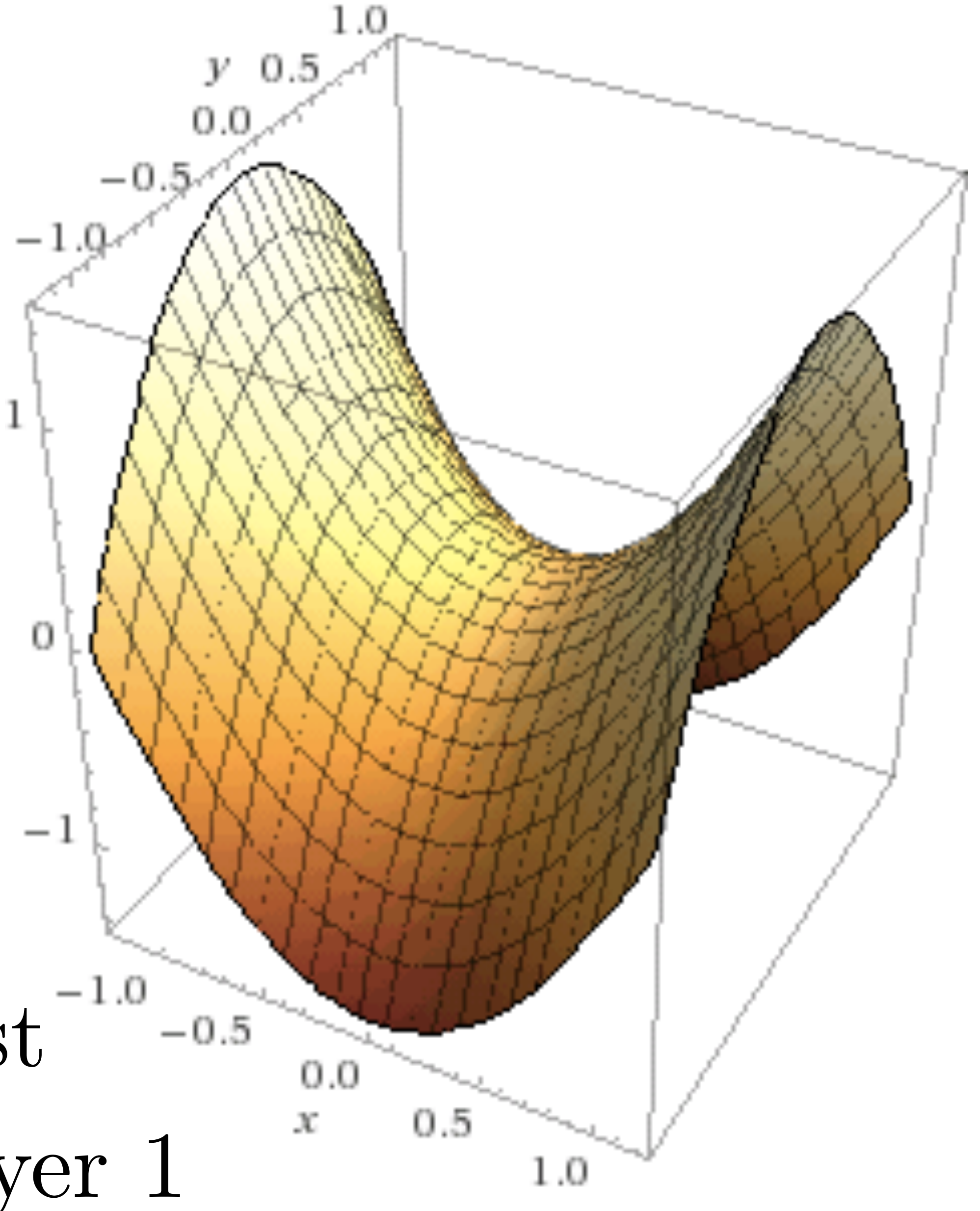
- In other words, *each player's cost is minimal with respect to that player's strategy*

- Finding Nash equilibria $\subseteq$ optimization (but not clearly useful)

Well-Studied Cases

- Finite minimax (zero-sum games)
- Finite mixed strategy games
- Continuous, convex games
- Differential games (lion chases gladiator)

Continuous Minimax Game



Solution is a *saddle point* of V.

Not just any saddle point: must specifically be a maximum for player 1 and a minimum for player 2

Local Differential Nash Equilibria

$\nabla_{\boldsymbol{\theta}^{(i)}} J^{(i)}(\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}) = \mathbf{0}$

Necessary:

$\nabla^2_{\boldsymbol{\theta}^{(i)}} J^{(i)}(\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)})$ is positive semi-definite

Sufficient:

$\nabla^2_{\boldsymbol{\theta}^{(i)}} J^{(i)}(\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)})$ is positive definite

(Ratliff et al 2013)

Sufficient Condition for Simultaneous Gradient Descent to Converge

$$\boldsymbol{\omega} = \left[\begin{array}{c} \nabla_{\boldsymbol{\theta}^{(1)}} J^{(1)}(\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}) \\ \nabla_{\boldsymbol{\theta}^{(2)}} J^{(2)}(\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}) \end{array} \right]$$

The eigenvalues of $\nabla_{\boldsymbol{\theta}} \boldsymbol{\omega}$ must have positive real part:

$$\left[\begin{array}{cc} \nabla^2_{\boldsymbol{\theta}^{(1)}} J^{(1)} & \nabla_{\boldsymbol{\theta}^{(1)}} \nabla_{\boldsymbol{\theta}^{(2)}} J^{(2)} \\ \nabla_{\boldsymbol{\theta}^{(2)}} \nabla_{\boldsymbol{\theta}^{(1)}} J^{(1)} & \nabla^2_{\boldsymbol{\theta}^{(2)}} J^{(2)} \end{array} \right]$$

(I call this the "generalized Hessian")

(Ratliff et al 2013)

Interpretation

- Each player's Hessian should have large, positive eigenvalues, expressing a strong preference to keep doing their current strategy

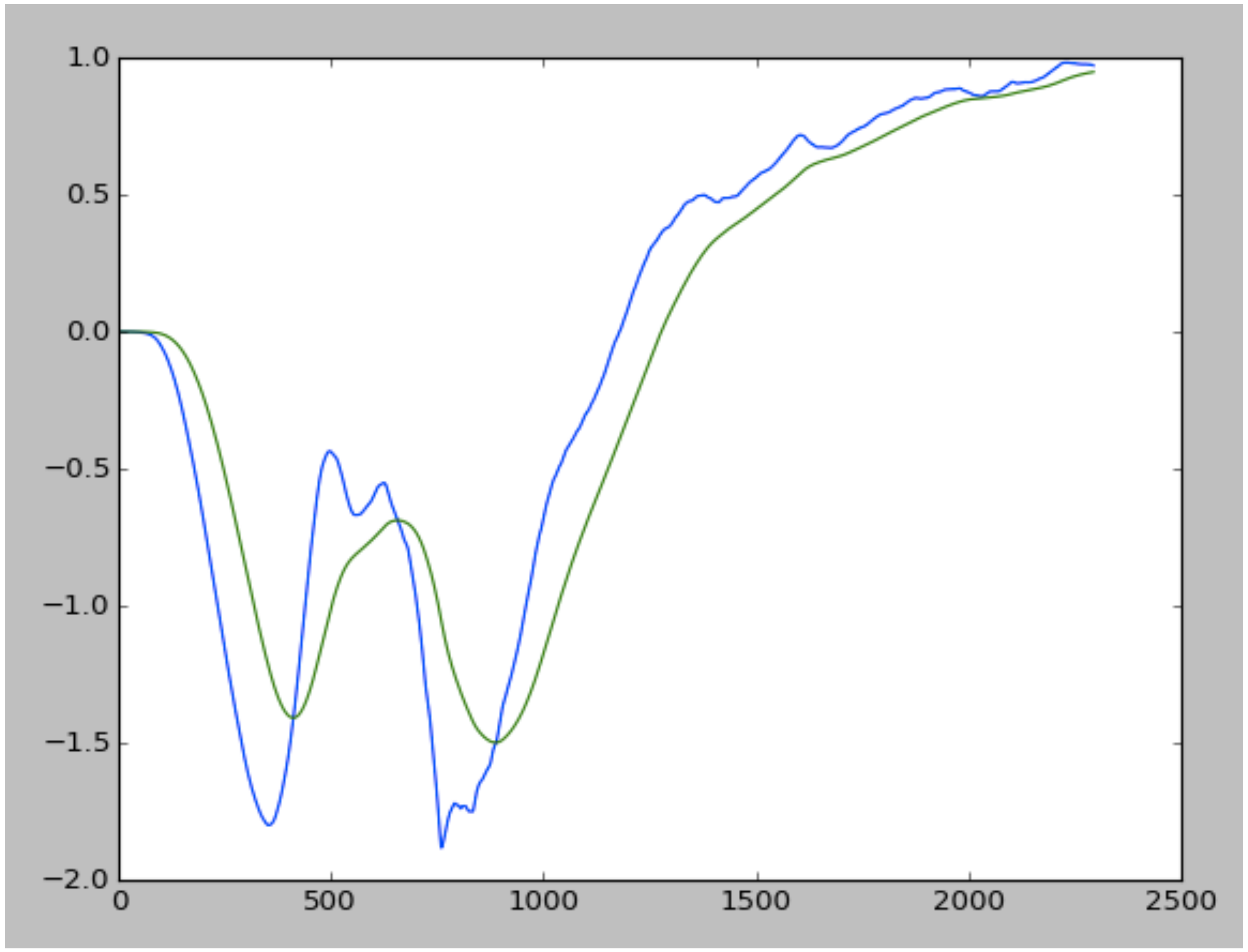
- The Jacobian of one player's gradient with respect to the other player's parameters should have smaller contributions to the eigenvalues, meaning each player has limited ability to change the other player's behavior at convergence

- Does not apply to GANs, so their convergence remains an open question

Equilibrium Finding Heuristics

- Keep parameters near their running average
    - Periodically assign running average value to parameters
    - Constrain parameters to lie near running average
    - Add loss for deviation from running average

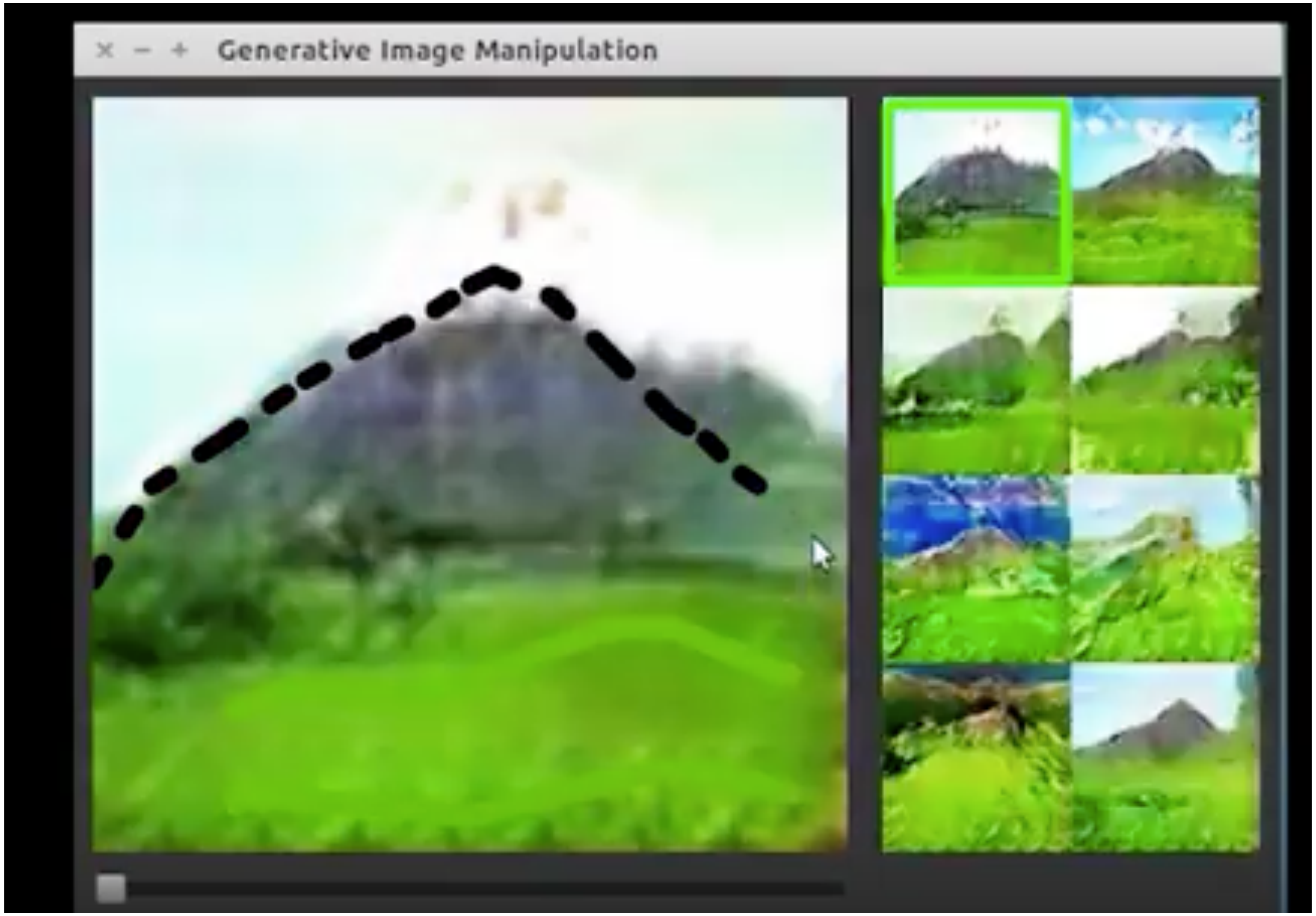
Stabilized Training



Other Games in AI

- Robust optimization / robust control
  - for security/safety, e.g. resisting adversarial examples
- Domain-adversarial learning for domain adaptation
- Adversarial privacy
- Guided cost learning
- Predictability minimization
- ...

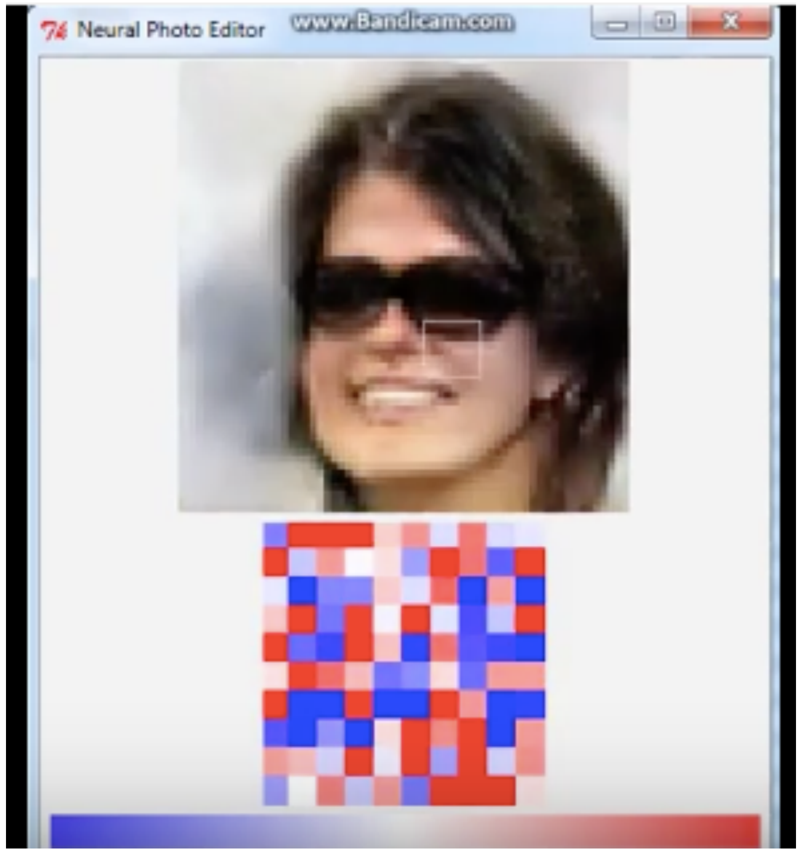
iGAN



youtube

(Zhu et al 2016)

Introspective Adversarial Networks



youtube

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