Generative Adversarial Networks

Note: Much of this material follows from Goodfellow (2016) “Tutorial: Generative Adversarial Networks” https://arxiv.org/abs/1701.00160

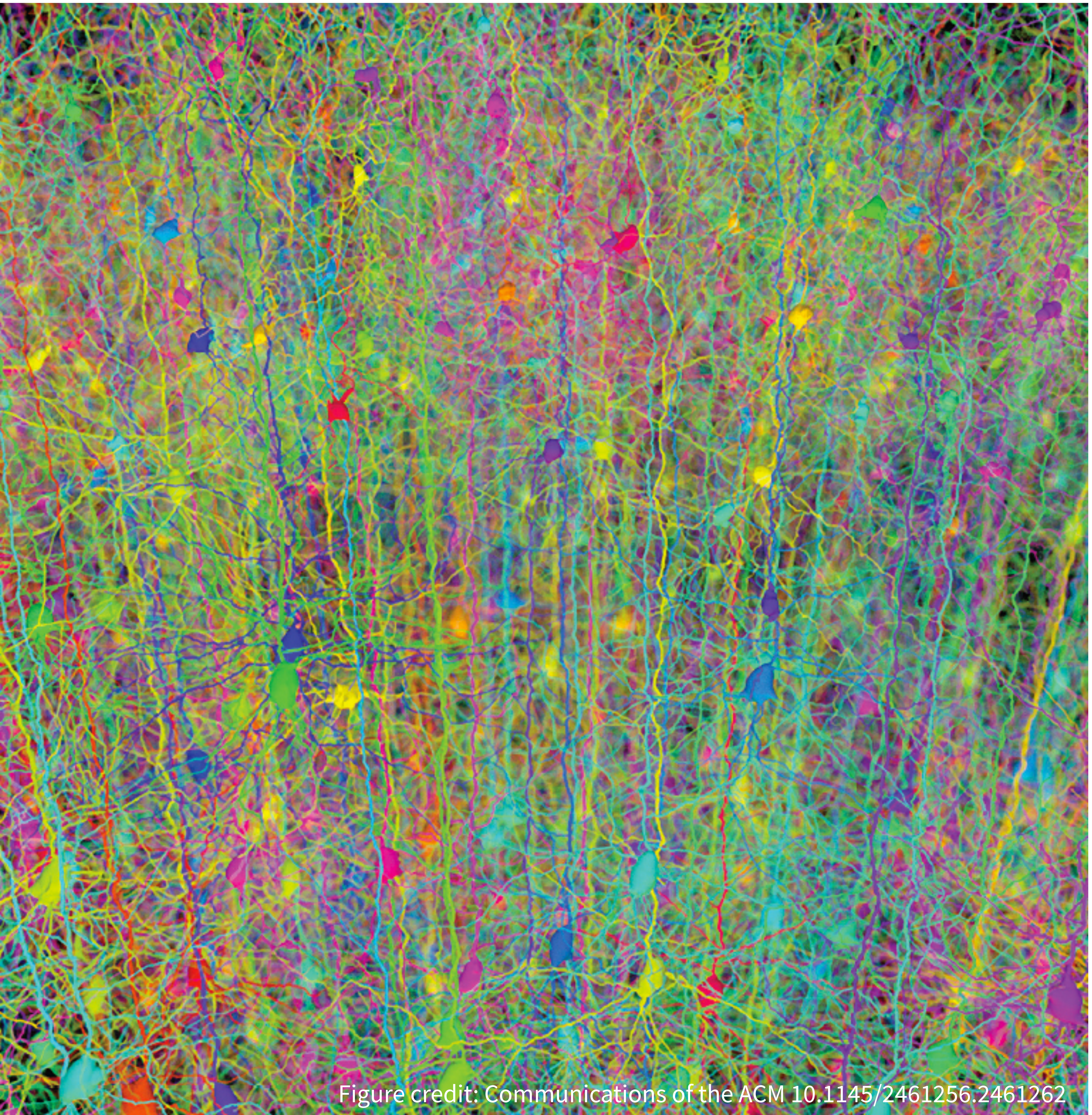


Figure credit: Communications of the ACM 10.1145/2461256.2461262

GRAHAM TAYLOR

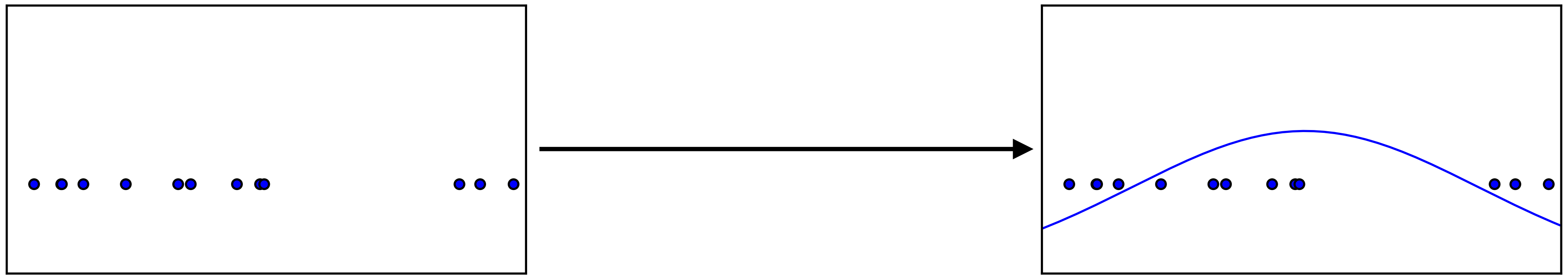
VECTOR INSTITUTE

SCHOOL OF ENGINEERING
UNIVERSITY OF GUELPH

CANADIAN INSTITUTE
FOR ADVANCED RESEARCH

Refresher: Generative Models

- Some models try to estimate the “data generating distribution” explicitly, e.g. density models $p_{\text{model}} \approxeq p_{\text{data}}$



- Other models are only able to generate samples from p_{model}

Training examples   Model samples

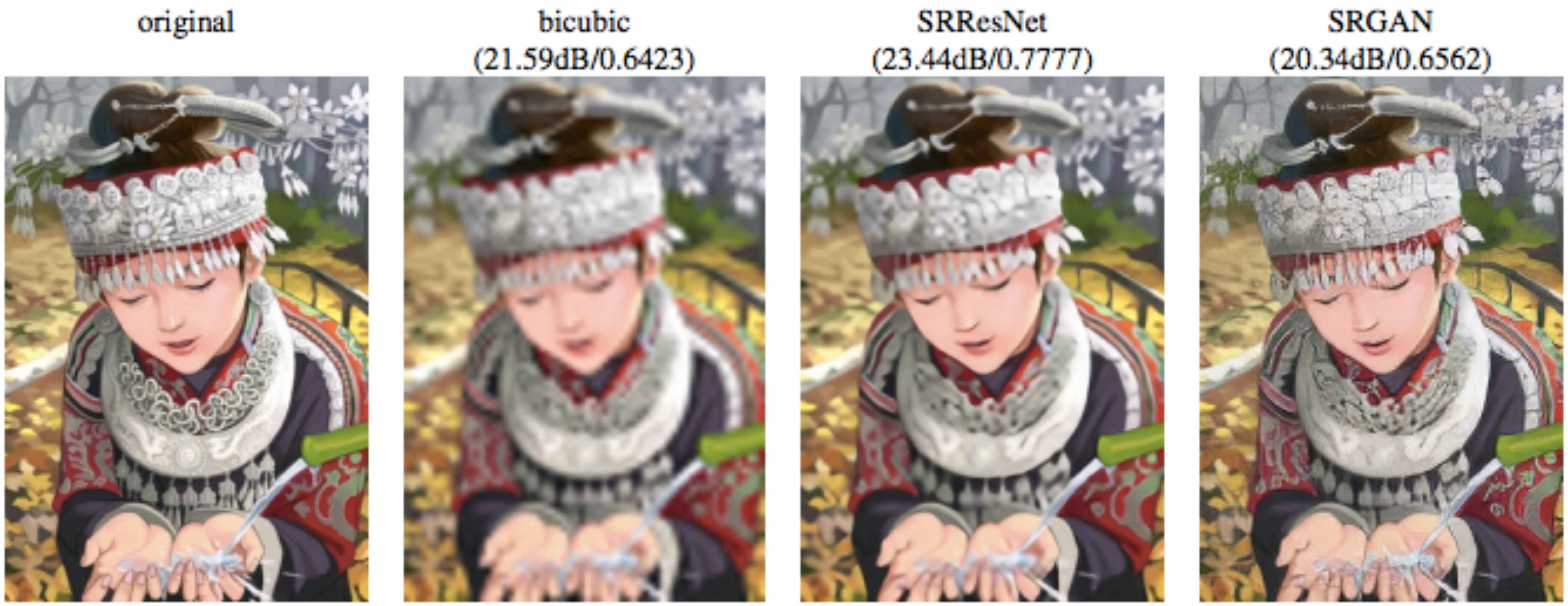
via Goodfellow et al. 2016

Why Study Generative Models?

- Can test our ability to build and use high-dimensional complex probability distributions
- Generative models can be incorporated into planning and reinforcement learning
  - e.g. simulating possible futures
- Multi-modal outputs (one or many-to-many mappings)
- Can provide predictions on inputs that are missing data
- Realistic generation

via Goodfellow et al. 2016

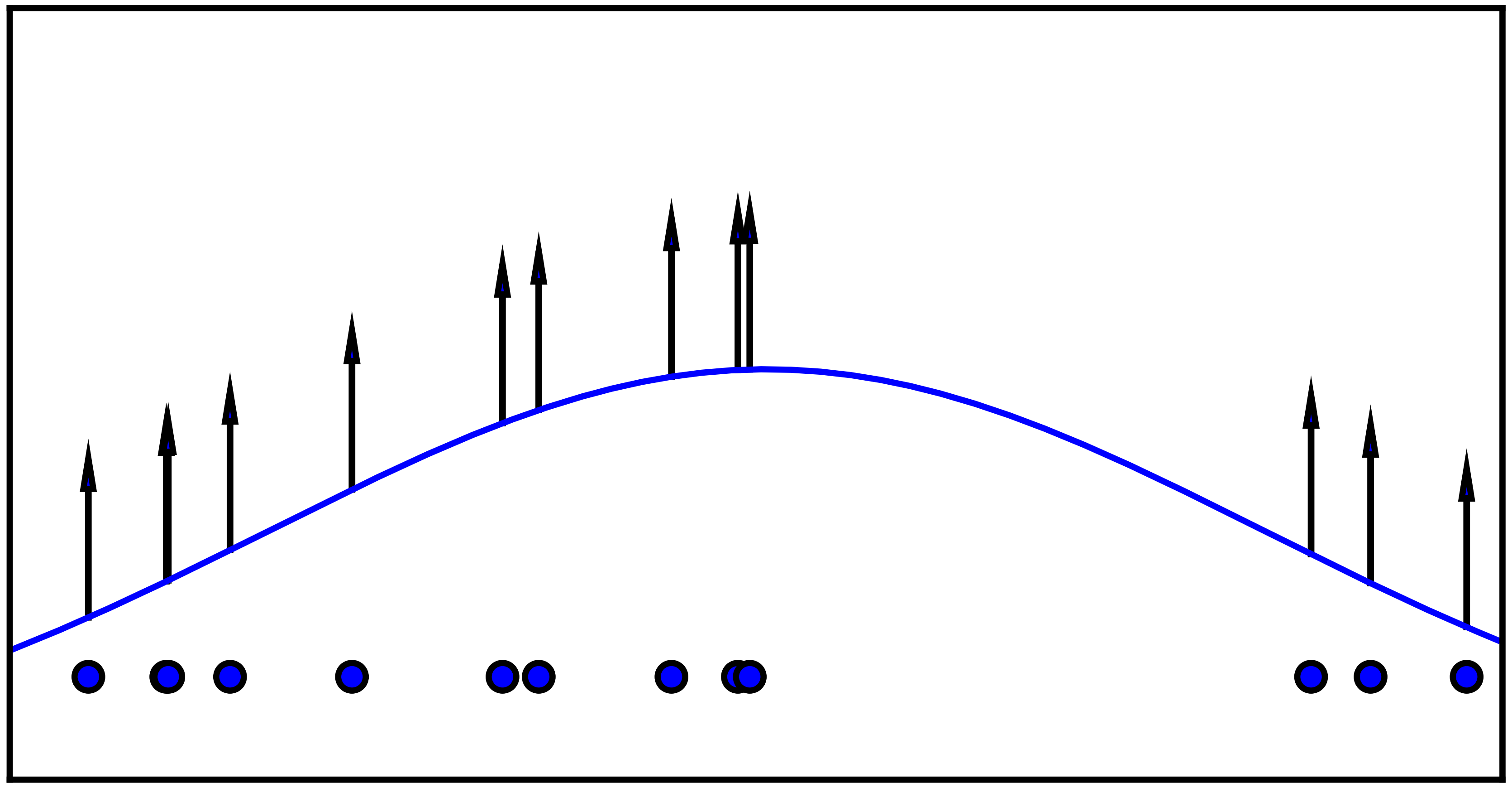
Single Image Super-resolution



Ledig et al. (2016)

via Goodfellow et al. 2016

Maximum Likelihood

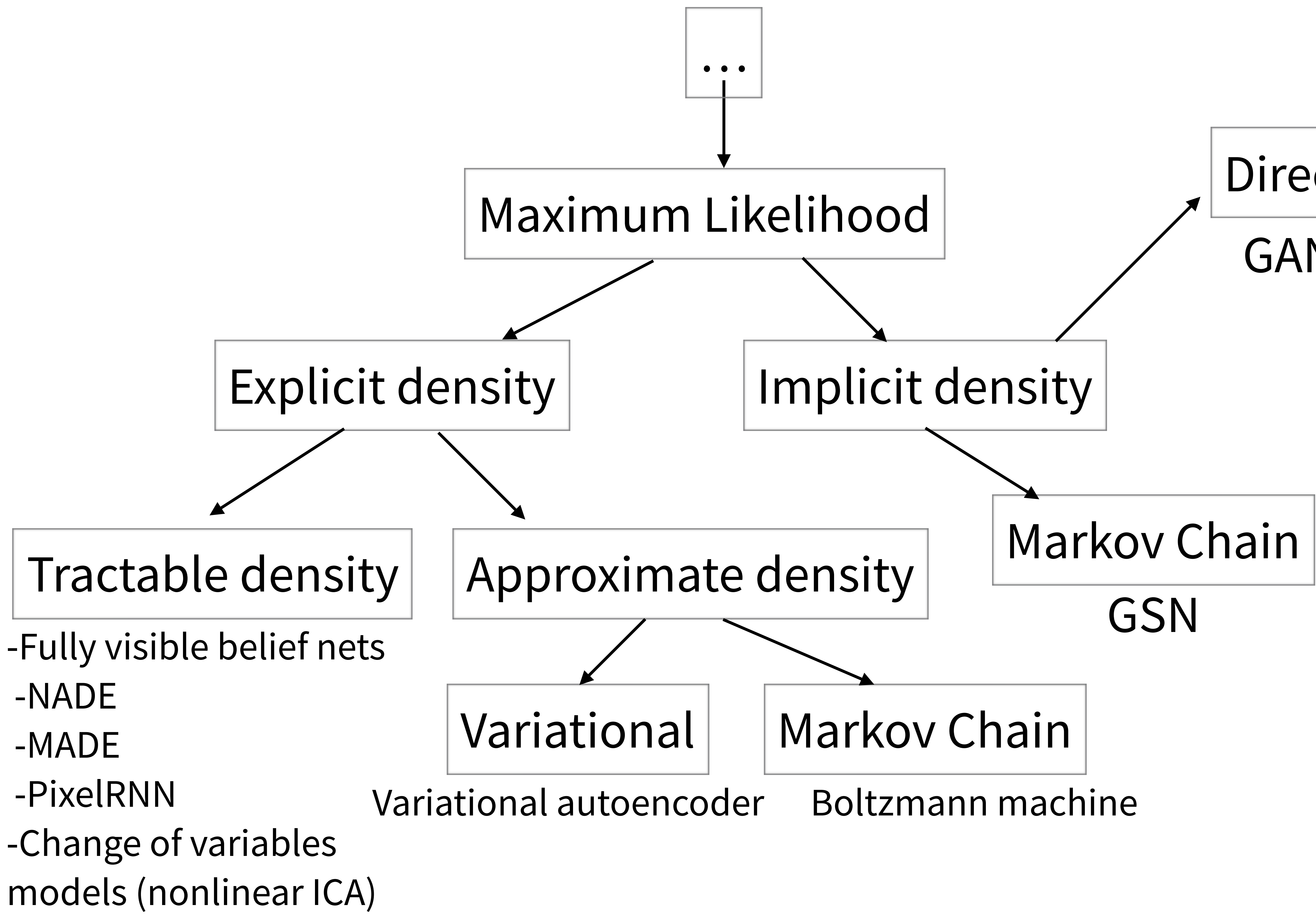


$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}} \log p_{\text{model}}(\boldsymbol{x};\boldsymbol{\theta})$$

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} D_{\text{KL}}\left(p_{\text{data}}\left(\boldsymbol{x}\right) || p_{\text{model}}\left(\boldsymbol{x};\boldsymbol{\theta}\right)\right)$$

via Goodfellow et al. 2016

Taxonomy of Generative Models

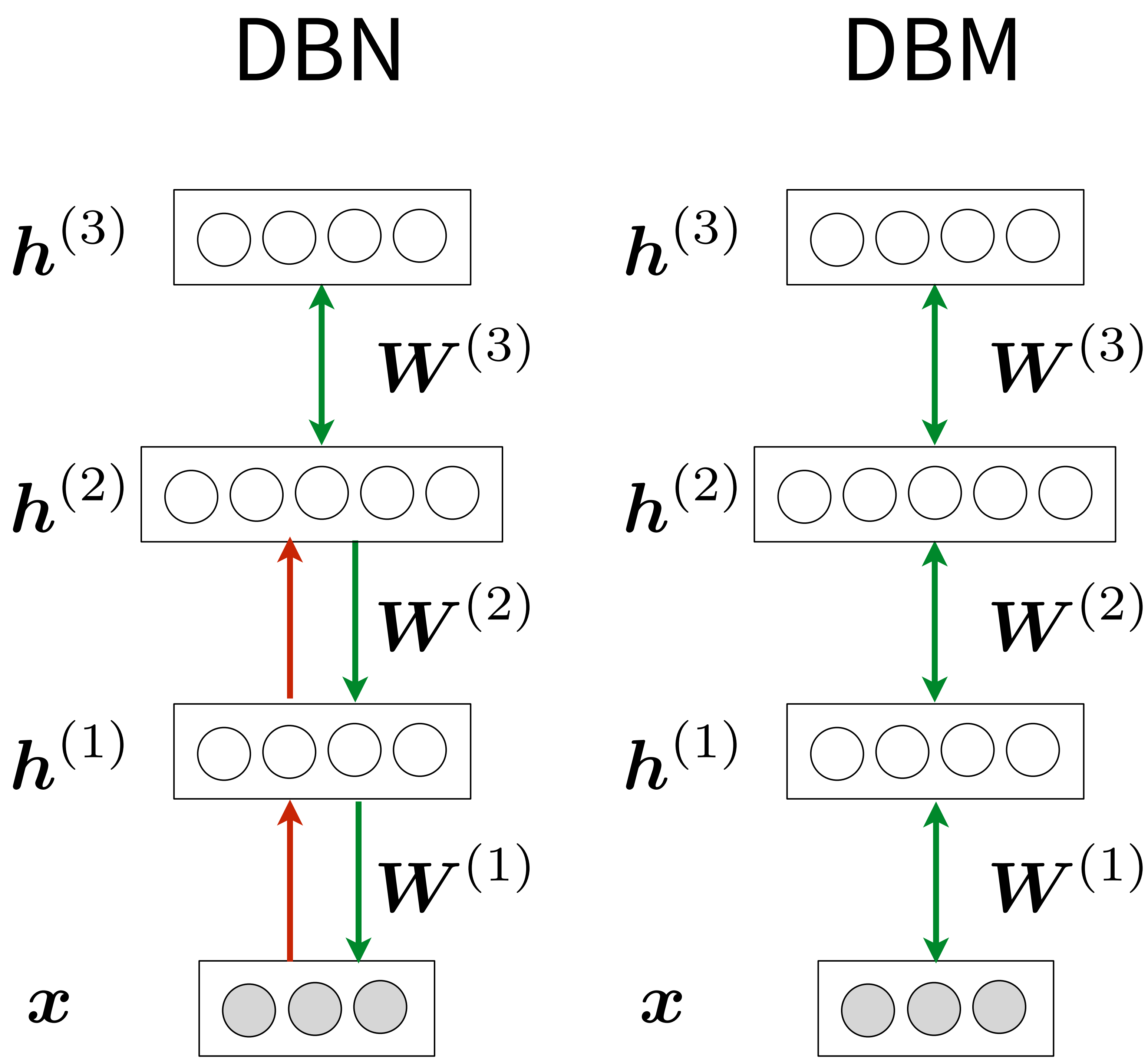


via Goodfellow et al. 2016

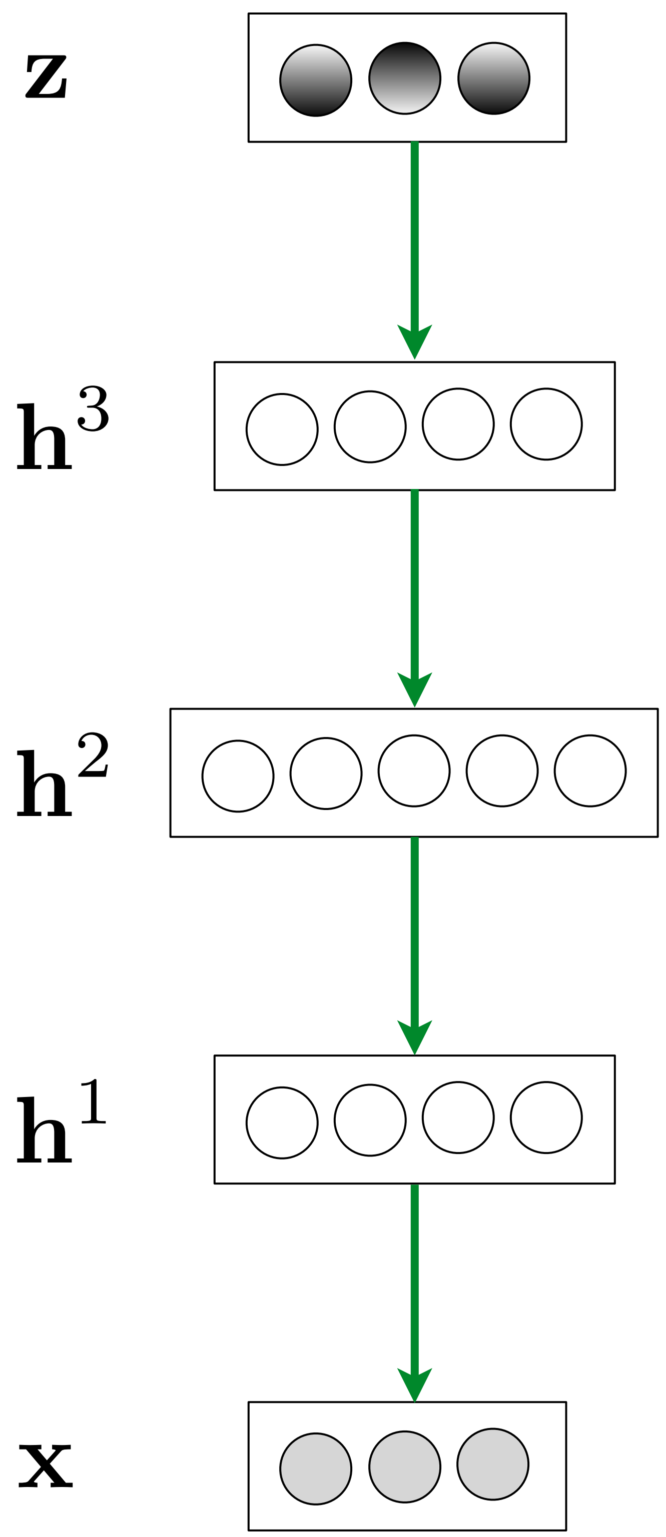
Classical Generative Models

<u>Downsides</u>:

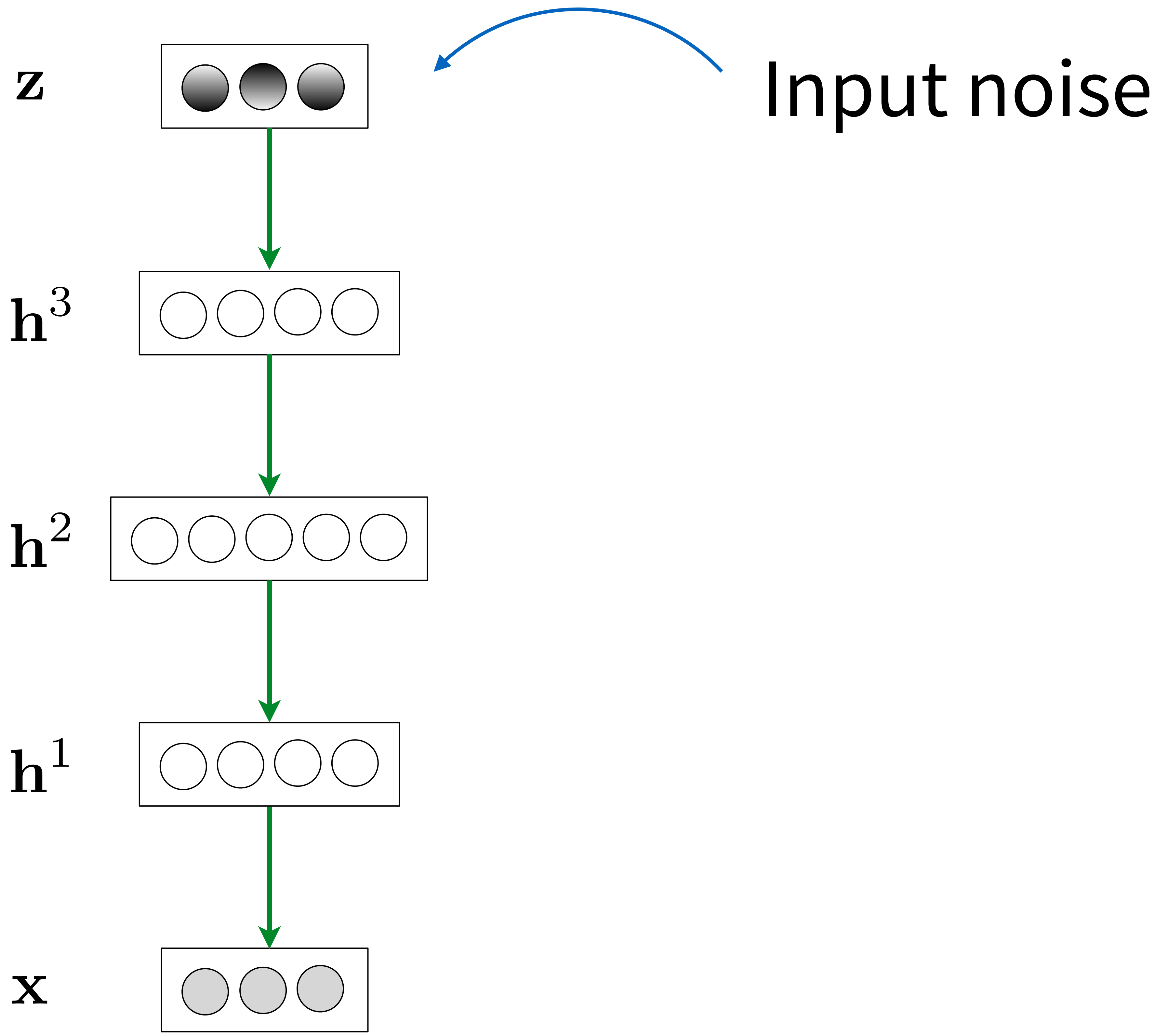
- sampling using MCMC
  - slow and does not mix well
- noise injection at every layer
  - high frequency spatial and temporal noise



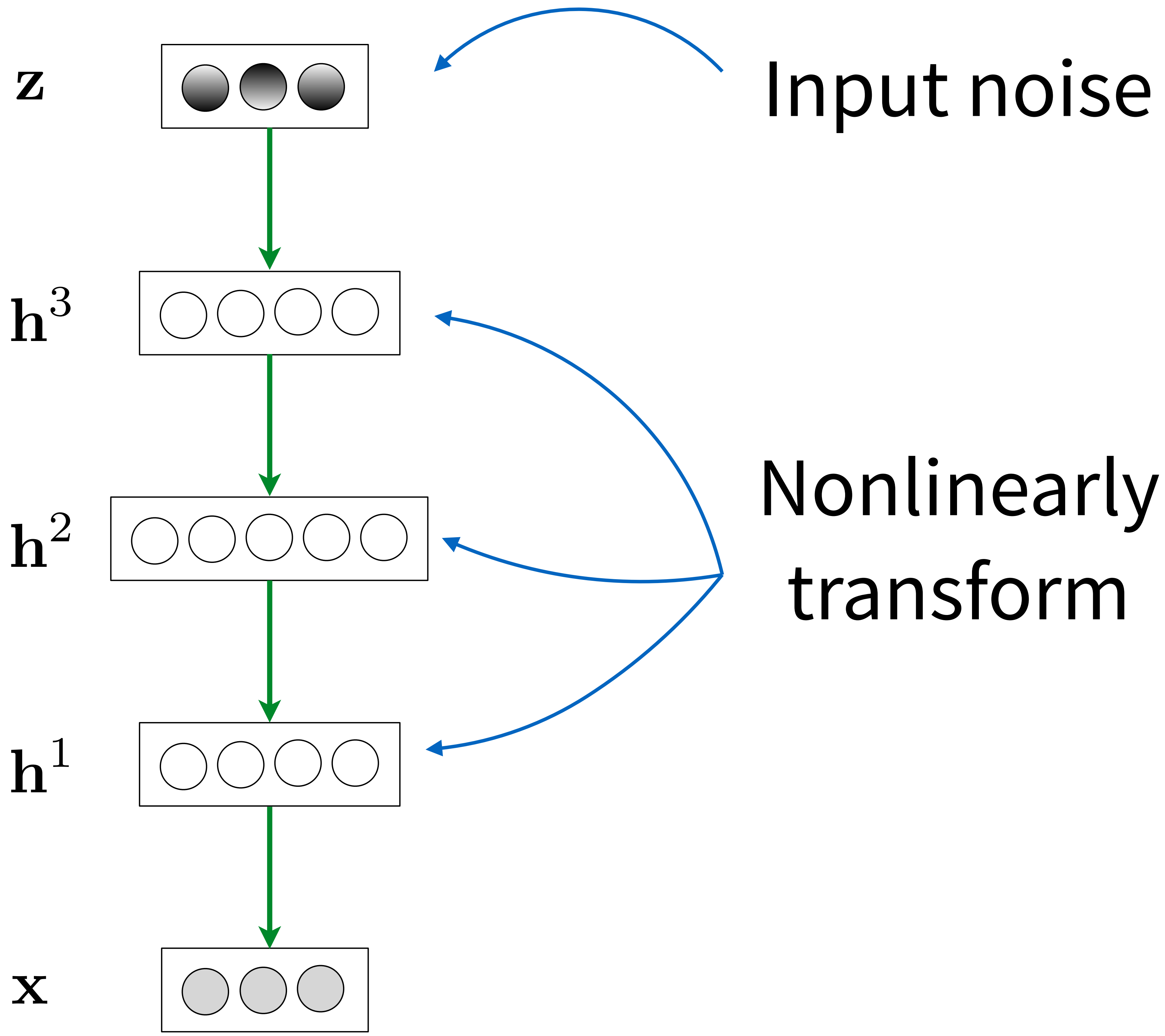
Deep Directed Generative Nets



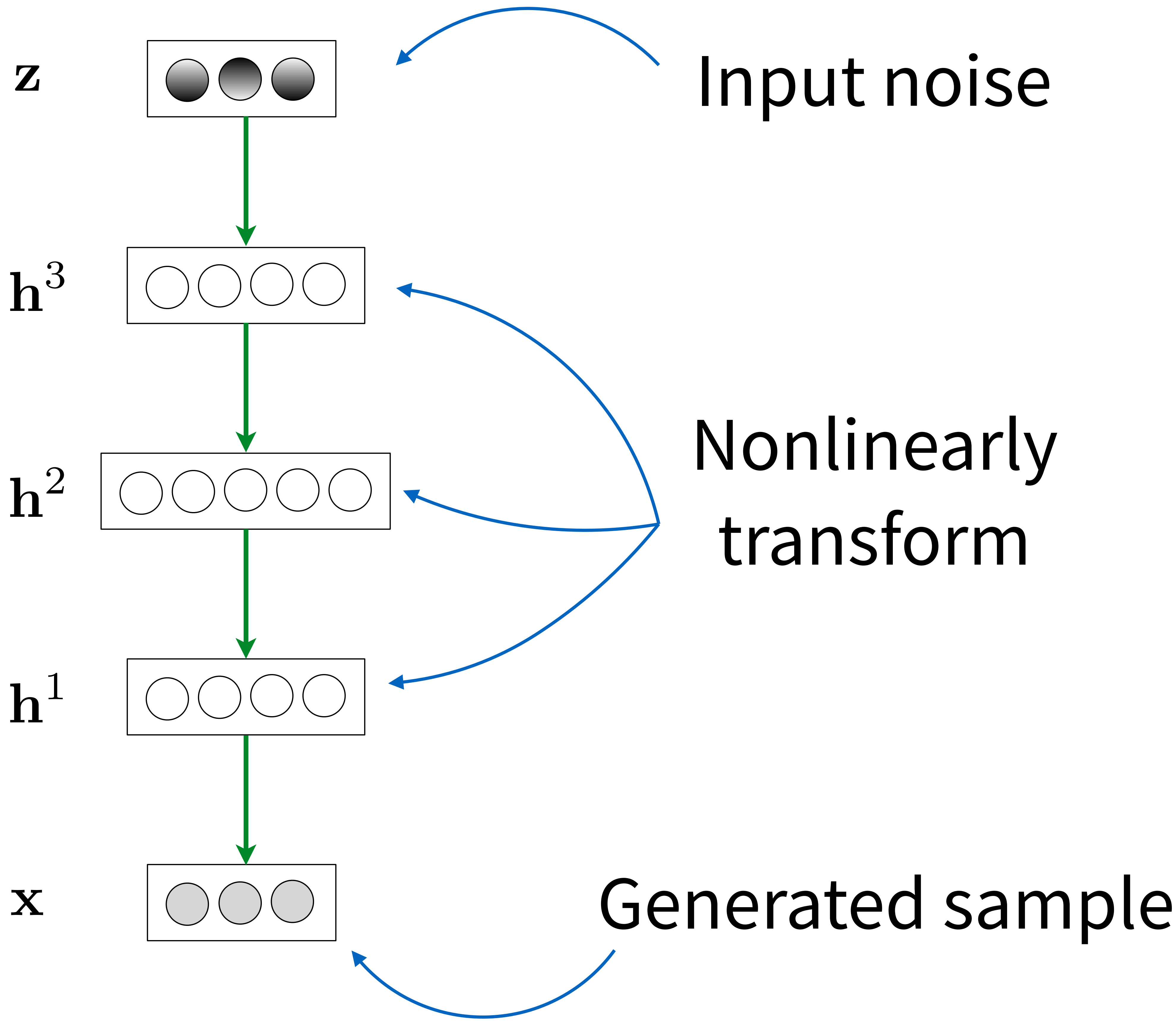
Deep Directed Generative Nets



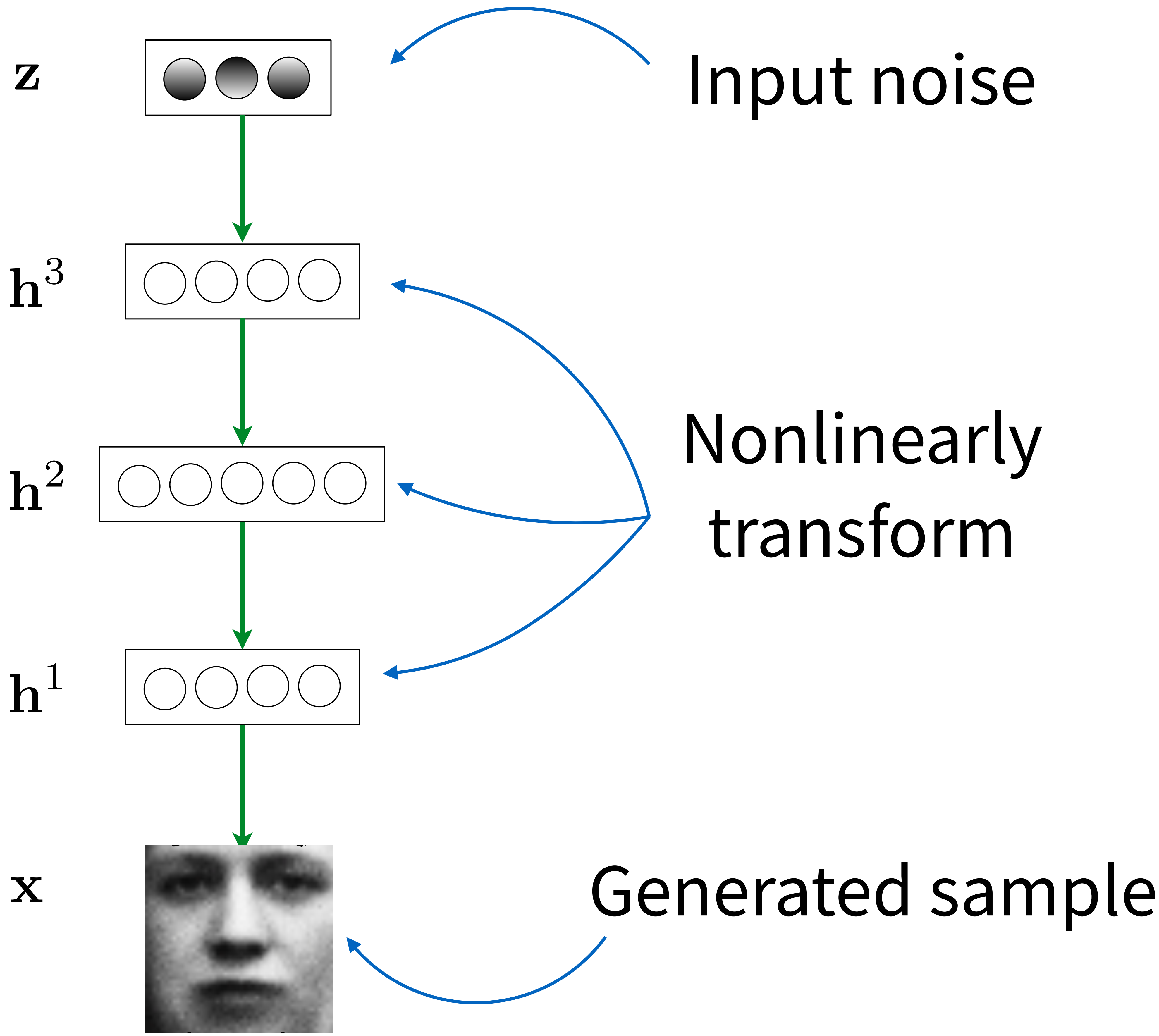
Deep Directed Generative Nets



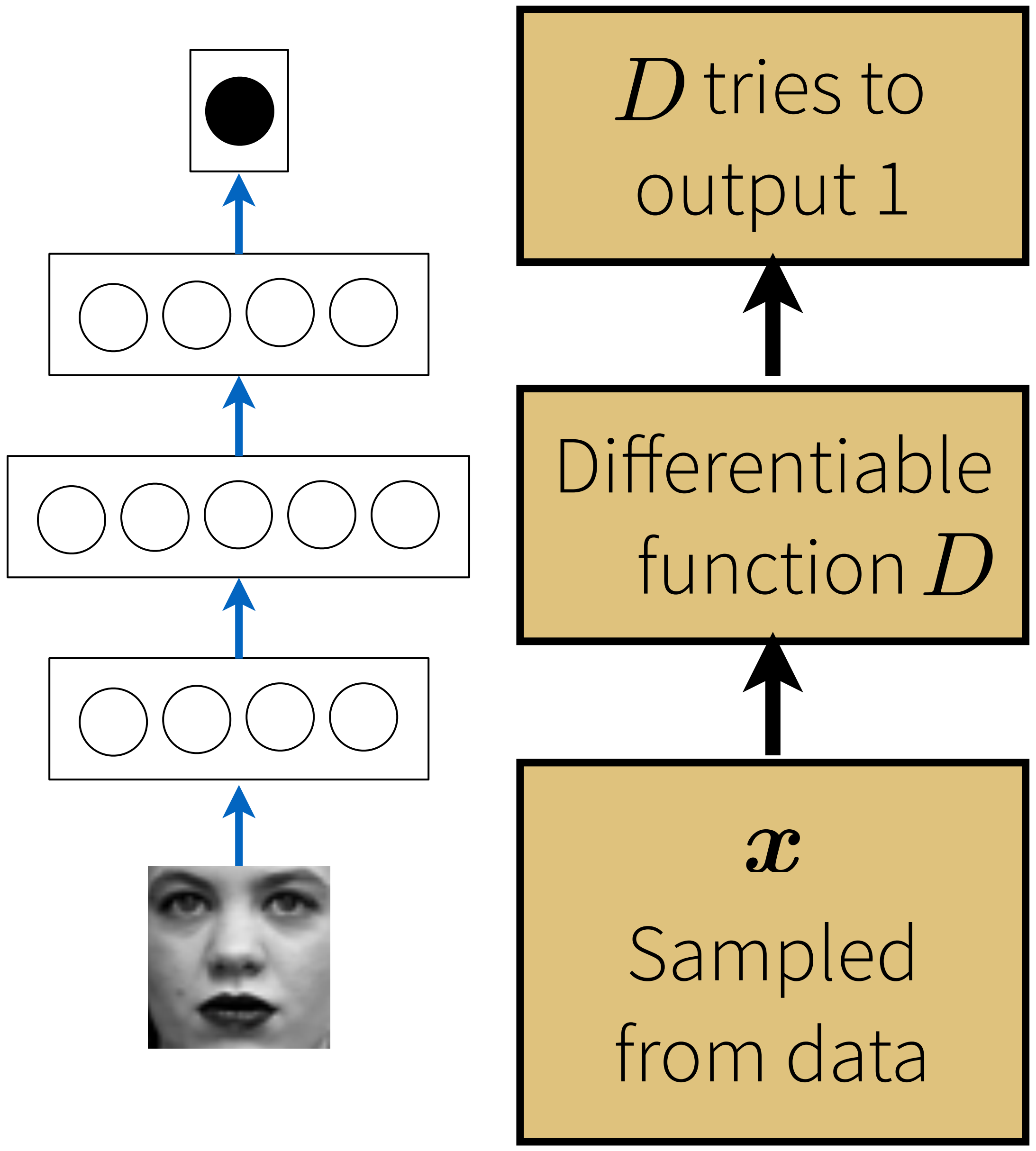
Deep Directed Generative Nets



Deep Directed Generative Nets

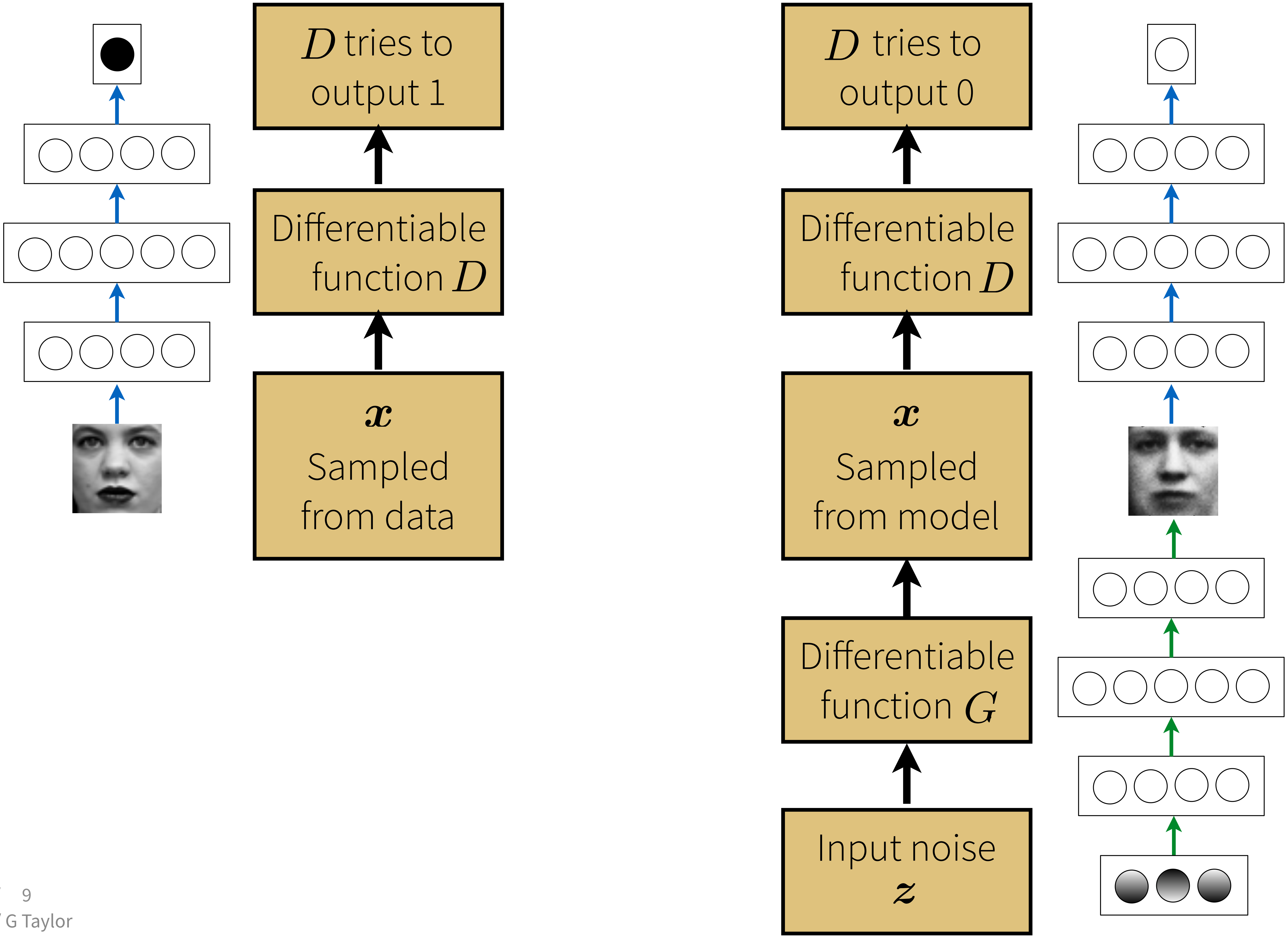


Generative Adversarial Nets



via Goodfellow et al. 2014

Generative Adversarial Nets

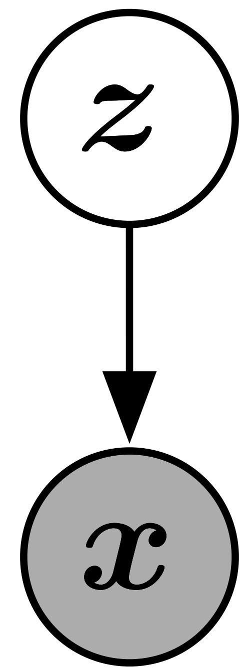


via Goodfellow et al. 2014

Generator Network

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

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



via Goodfellow et al. 2016

Training Procedure

Use an SGD algorithm of choice (e.g. Adam) on two mini-batches simultaneously:

- A mini-batch of training examples
- A mini-batch of generated samples

via Goodfellow et al. 2016

GAN Objective (Discriminator)

First, consider the discriminator's cost:

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

This is just the standard binary cross-entropy cost.

But, the discriminator is trained on two mini-batches:

- One coming from the dataset (label 1 for all)
- One coming from the generator (label 0 for all)

Adversarial or Co-operative?

- Because the GAN framework can be analyzed with the tools of game theory, we call GANs “adversarial”
- But we can also think of them as co-operative
  - The discriminator is more like a teacher than an adversary, showing the generator how to improve
- Amazingly, GANs have an adaptive objective:
  - The discriminator defines the objective for the generator by discovering holes that the generator needs to fix

GAN Objective (Generator)

There are several ways to specify the cost function for the generator:

Minimax

$$J^{(G)} = -J^{(D)} \qquad V\left(\boldsymbol{\theta}^{(D)}, \boldsymbol{\theta}^{(G)}\right) = -J^{(D)}\left(\boldsymbol{\theta}^{(D)}, \boldsymbol{\theta}^{(G)}\right)$$

$$\boldsymbol{\theta}^{(G)*} = \arg\min_{\boldsymbol{\theta}^{(G)}} \max_{\boldsymbol{\theta}^{(D)}} V\left(\boldsymbol{\theta}^{(D)}, \boldsymbol{\theta}^{(G)}\right)$$

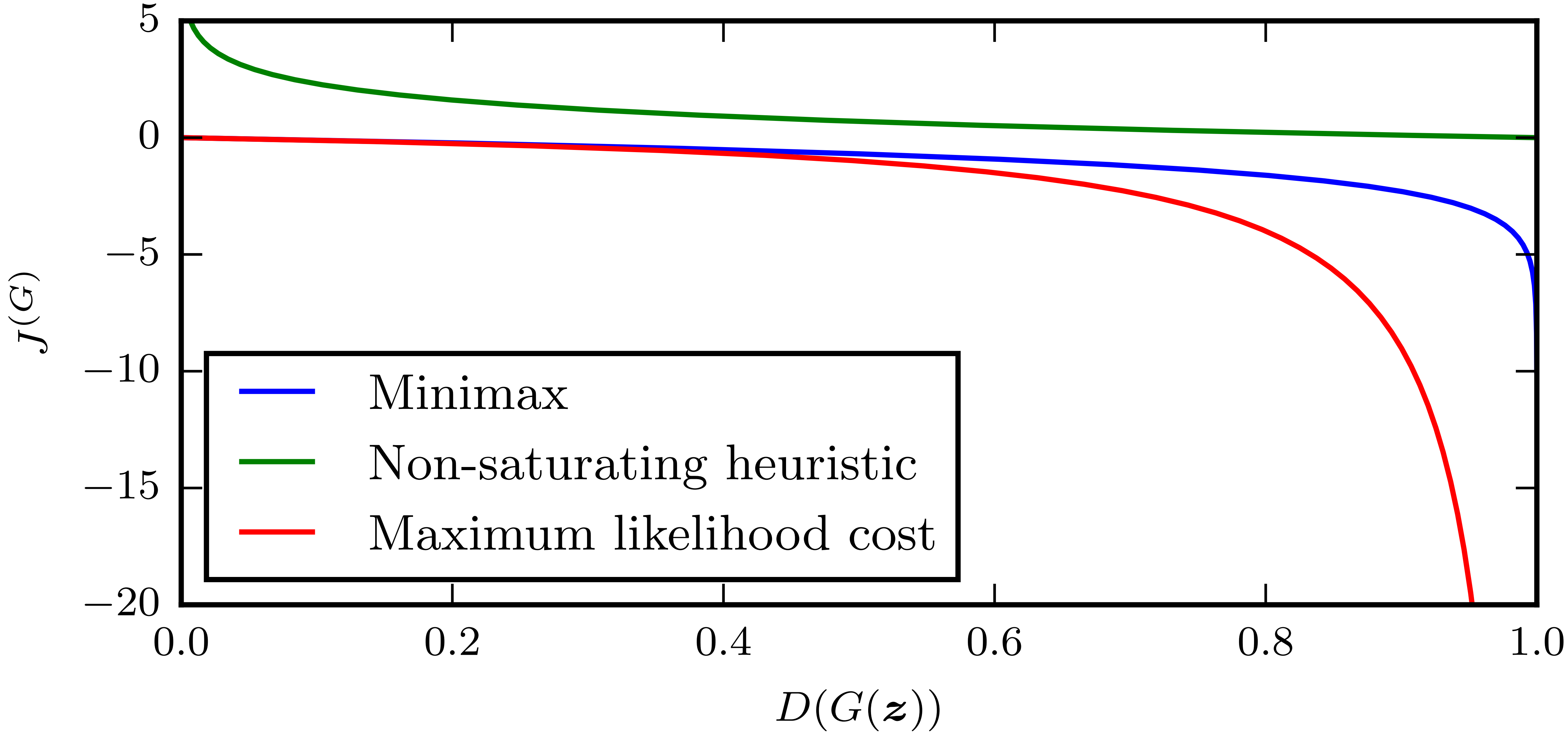
Heuristic

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

Maximum Likelihood

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

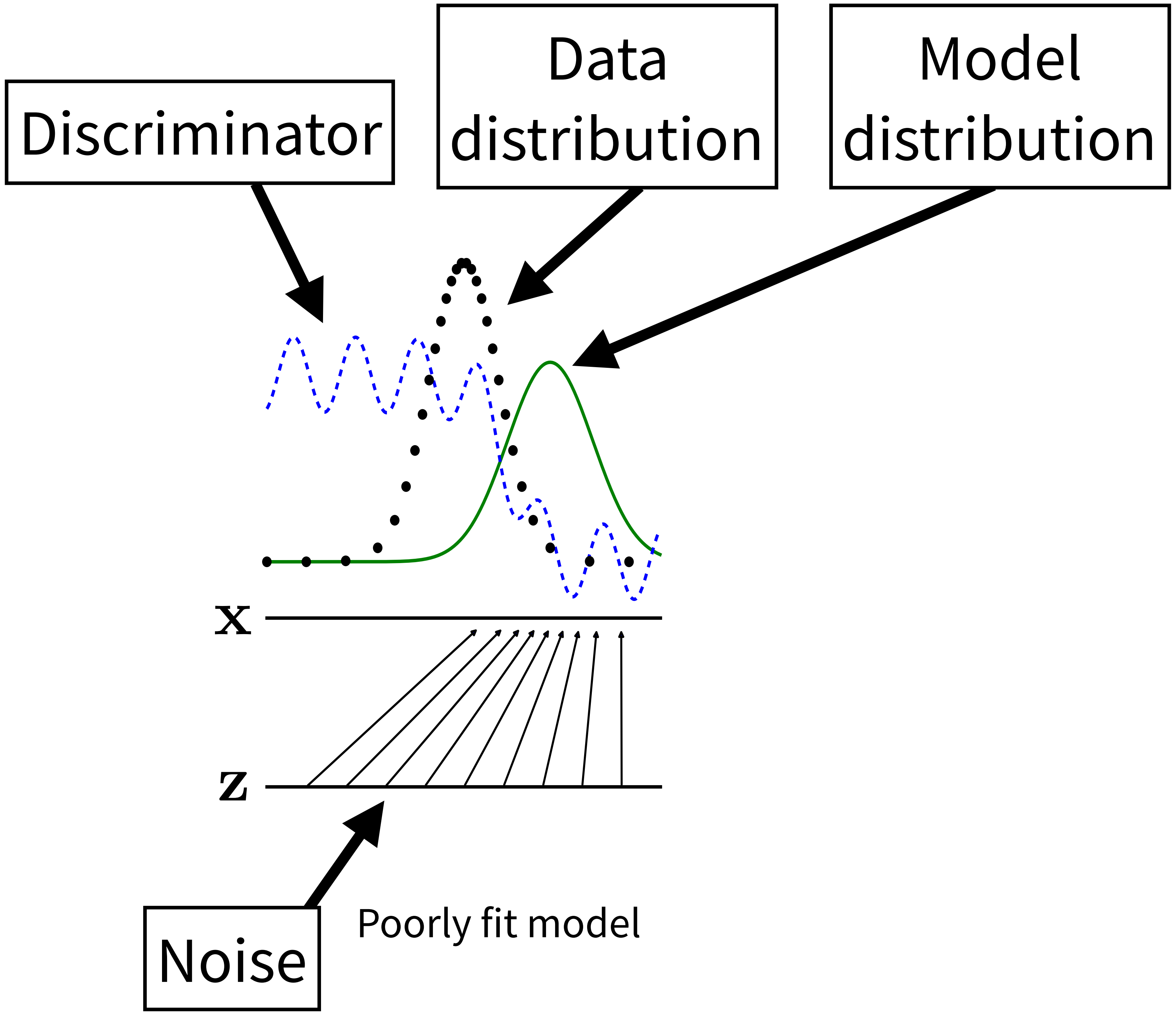
Comparison of Generator Losses



Goodfellow et al. (2014)

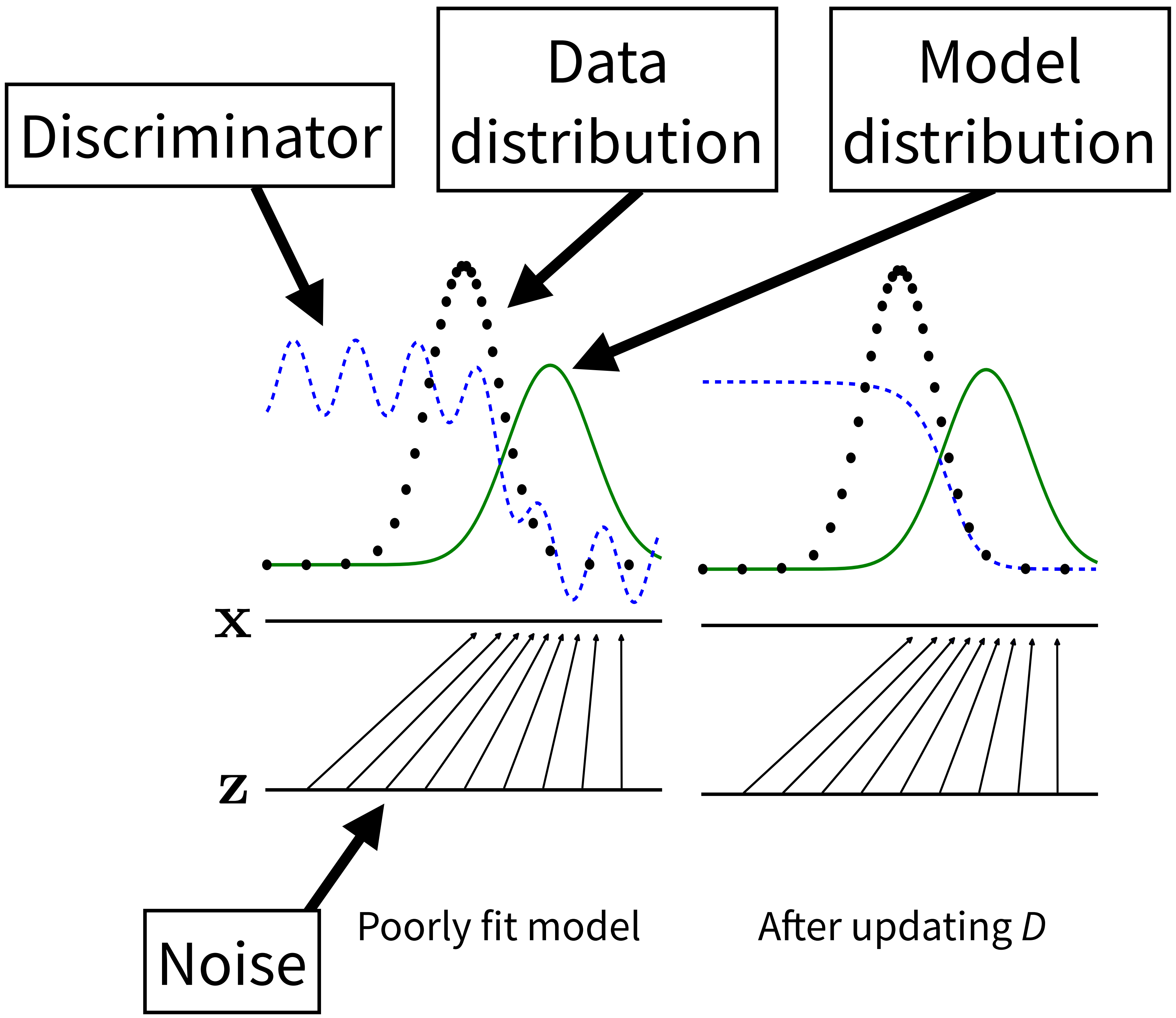
via Goodfellow et al. 2016

GAN - intuition



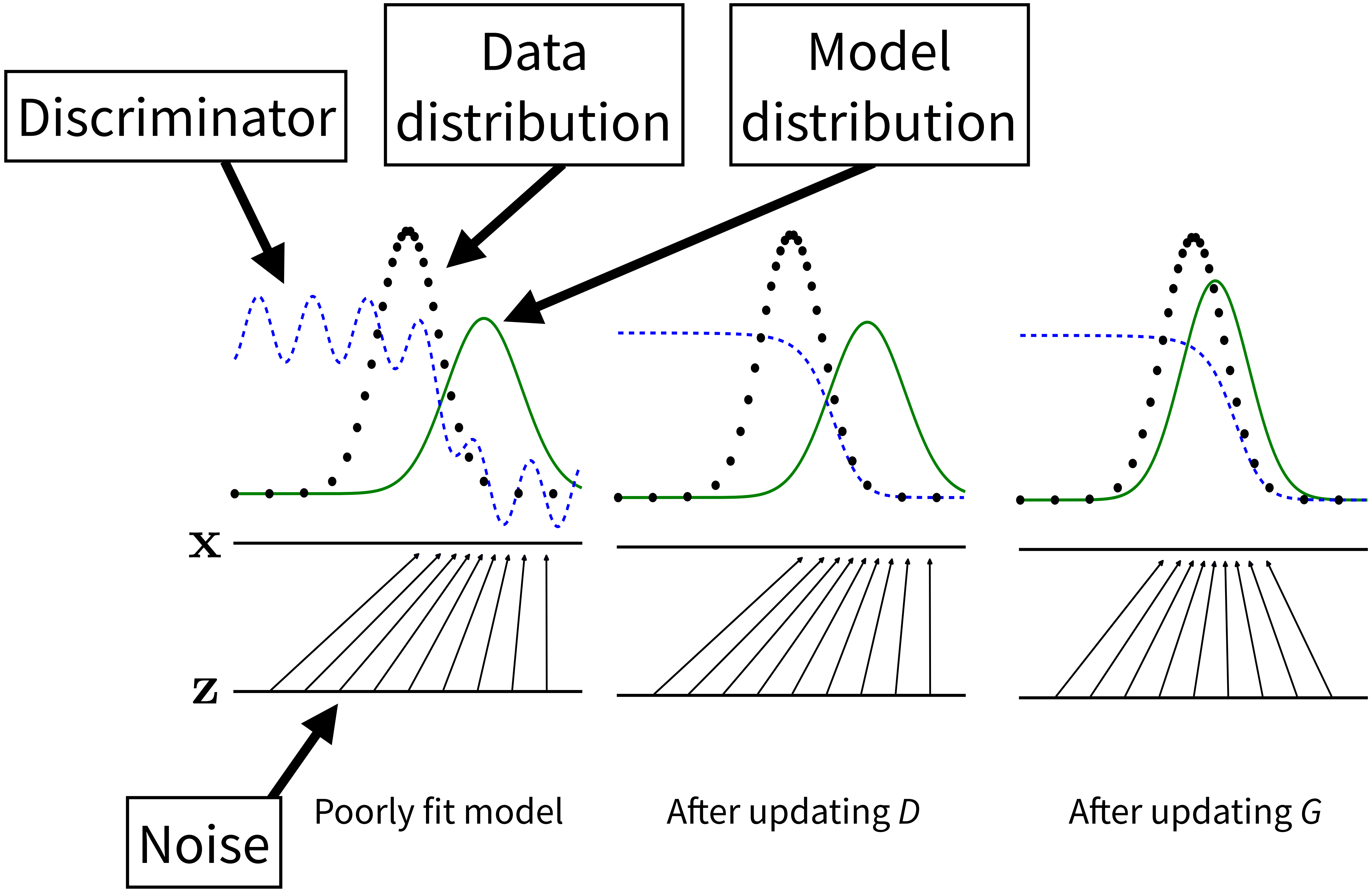
via Goodfellow et al. 2014

GAN - intuition



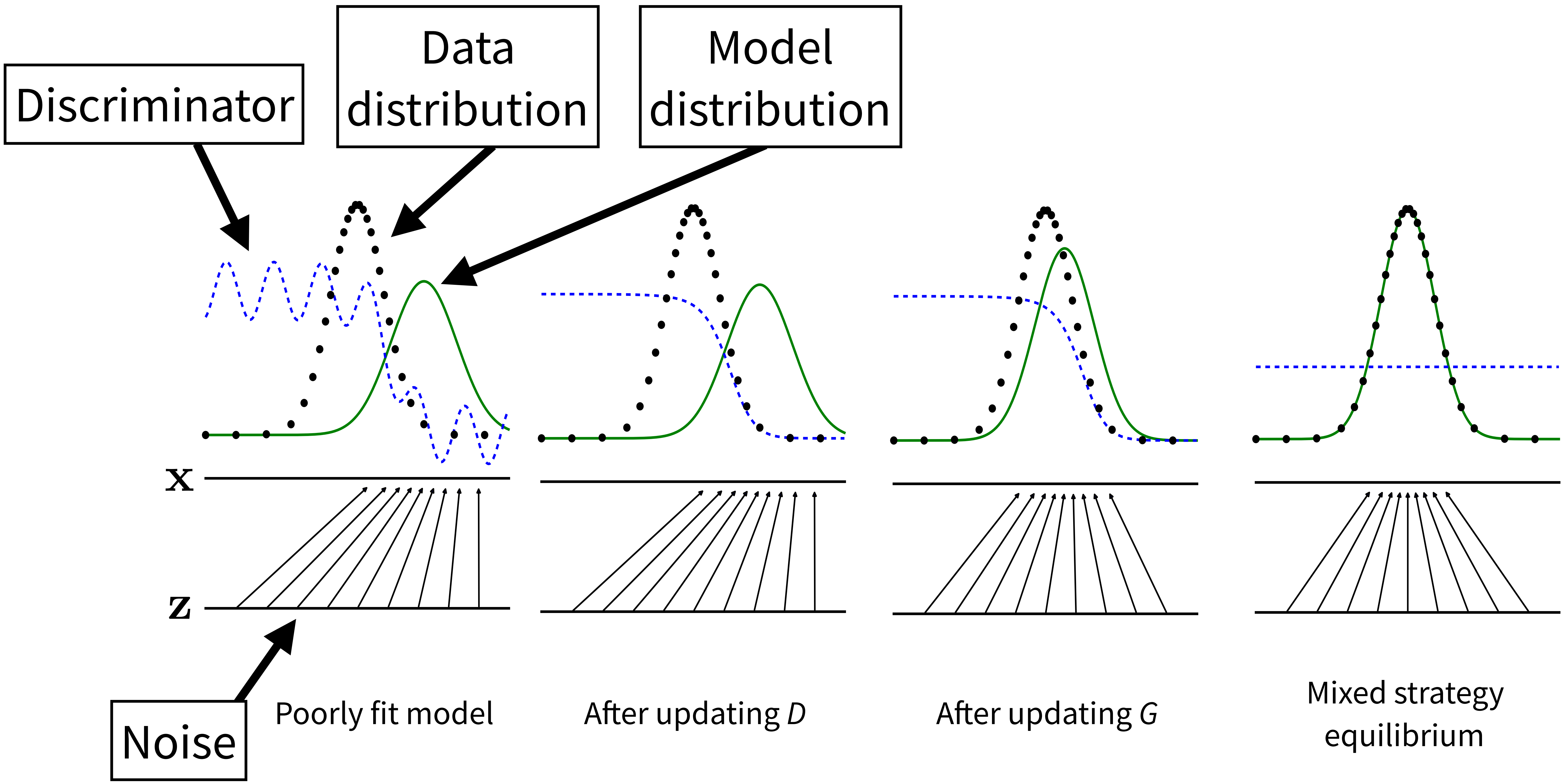
via Goodfellow et al. 2014

GAN - intuition



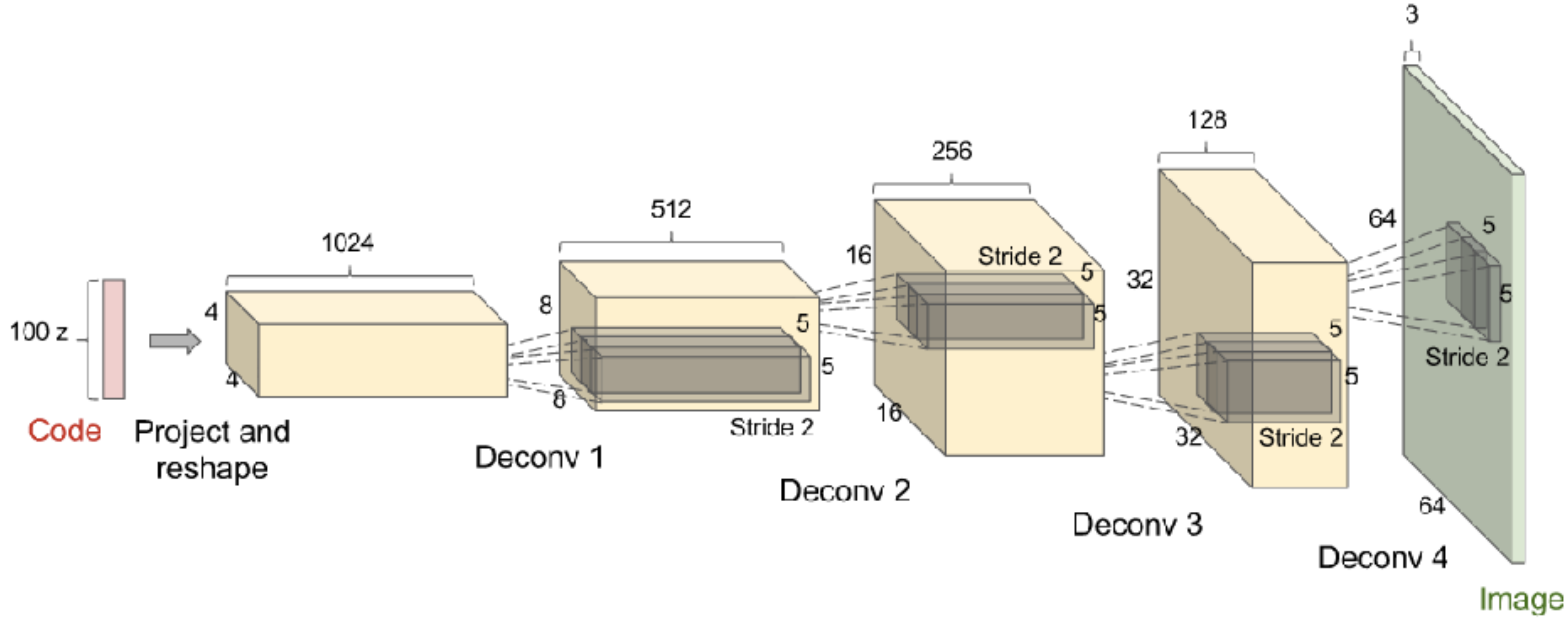
via Goodfellow et al. 2014

GAN - intuition



via Goodfellow et al. 2014

Deep Convolution GAN (DCGAN)



<u>Recipe for success:</u>

- Use batch normalization layers in most layers
- Use “All-convolutional net”-style architecture, neither pooling nor unpooling layers, no fully connected layers
- Use ReLU layers in generator, except for tanh in the output layer
- Use leaky ReLU layers in discriminator
- Use of Adam optimizer over SGD with momentum

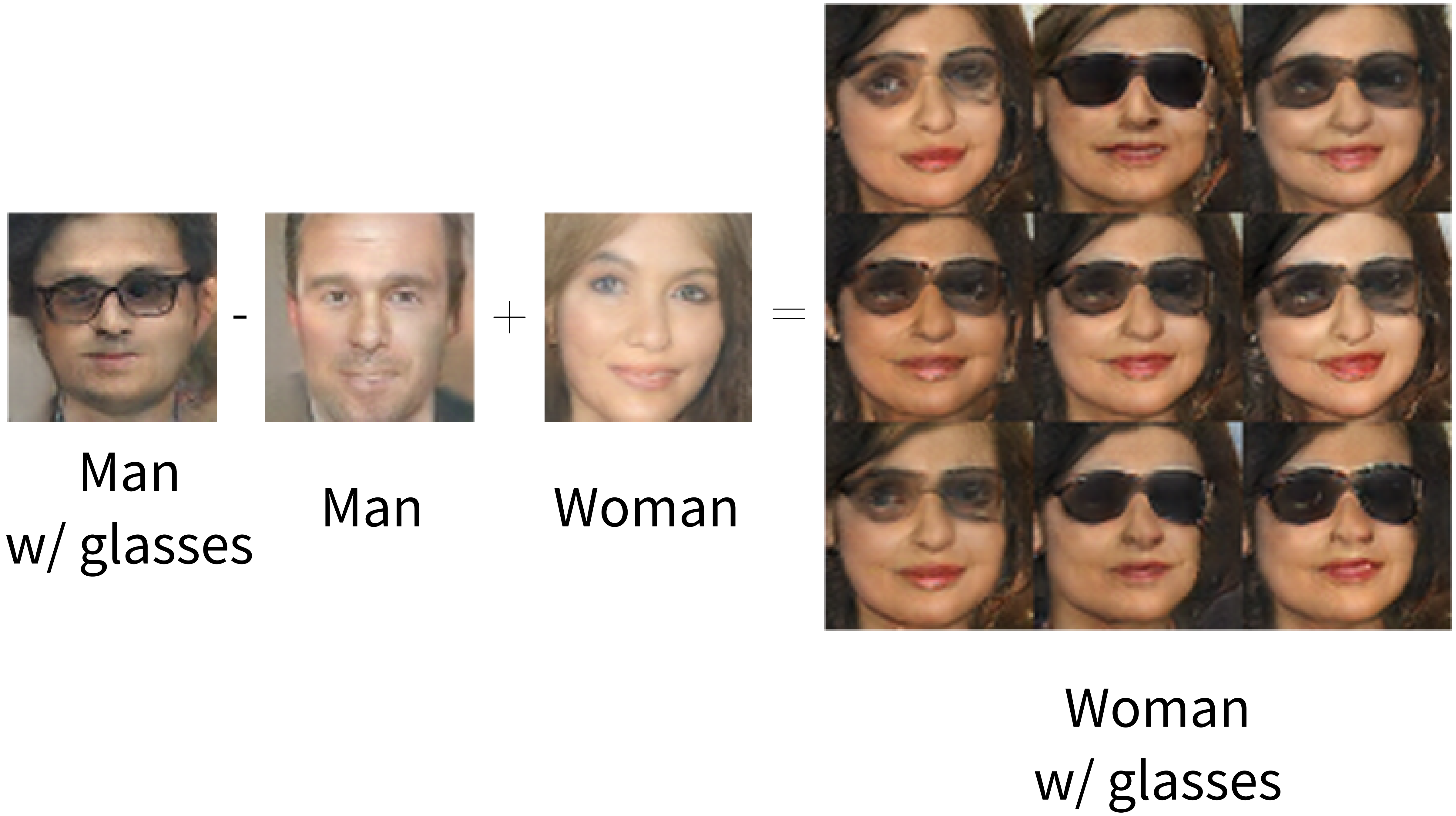
via Radford et al. 2015

DCGANs for LSUN Bedrooms

Radford et al. (2015)

via Goodfellow et al. 2016

Vector Space Arithmetic

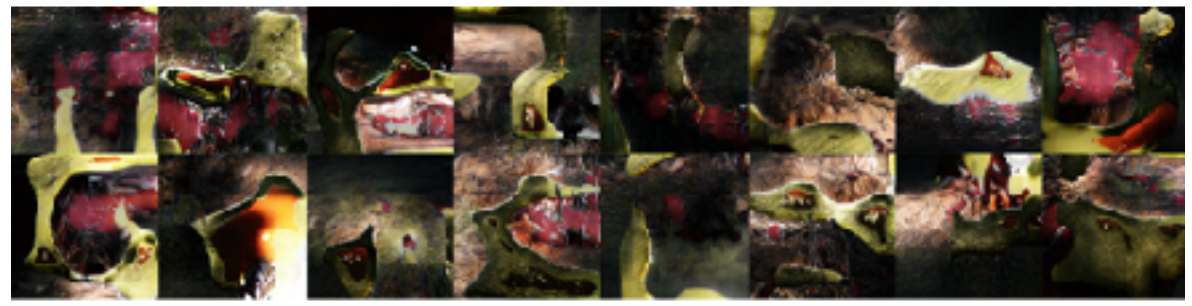


Radford et al. (2015)

via Goodfellow et al. 2016

Tips and Tricks

- Train with labels
  - either as class-conditional (Denton et al. 2015)
  - or train discriminator on classes (Salimans et al.)
- One-sided label smoothing
- Virtual batch normalization



Fluctuations in mean and standard deviation of a mini batch can have a greater effect than the z codes for the images in the mini-batch.
Examples in mini-batch should be independent of each other, but batch-norm has caused them to be correlated!

via Goodfellow et al. 2016

Outstanding Issues: Non-convergence

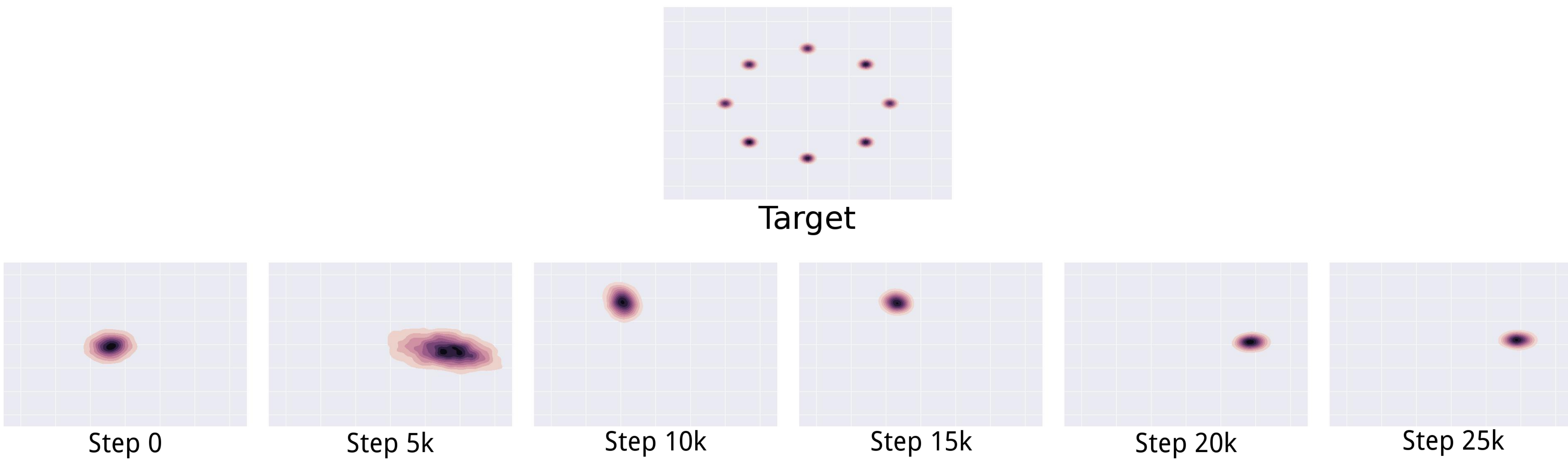
- Most deep learning algorithms are trained by optimization
  - often approach a saddle point, or local minimum rather than global minimum
- Game solving algorithms may not approach equilibrium at all
- Subject to oscillation: can train for a very long time generating different categories of samples without clearly generating better samples

via Goodfellow et al. 2016

Outstanding Issues: Mode Collapse

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

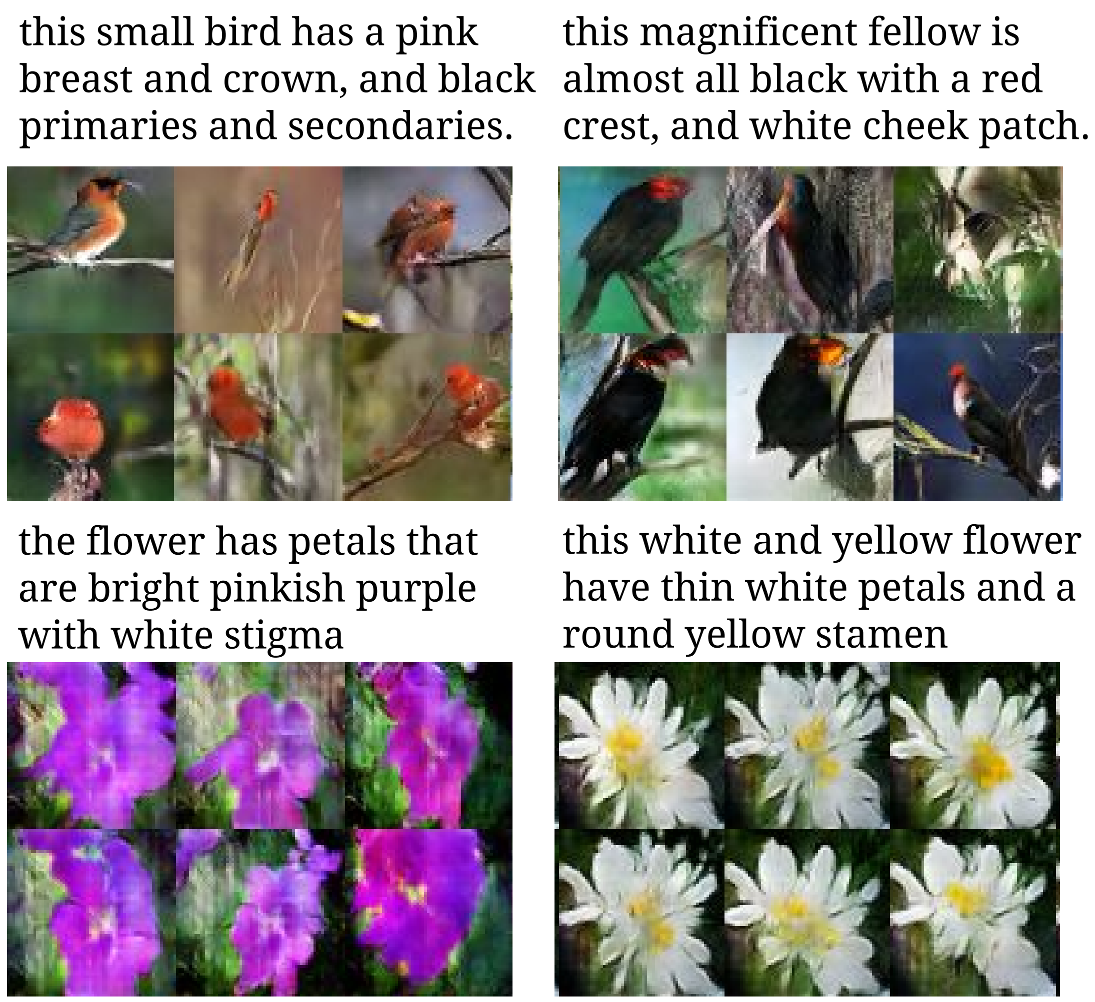
- D in inner loop: convergence to correct distribution
- G in inner loop: place all mass on most likely point
- Simultaneous gradient descent - which one does it do?



Radford et al. (2016)

via Goodfellow et al. 2016

Mode Collapse and Low Output Diversity



Reed et al. (2016)

Reed et al. (2017)

via Goodfellow et al. 2016

Outstanding Issues: Evaluation

- There is no single best way to evaluate generative models (Theis et al., 2016)
- Likelihood is the most popular, but:
  - Models with good likelihood can produce poor samples
  - Models with good samples can have poor likelihood
- For GANs, it's hard to even estimate the likelihood (see Wu et al. 2017 for an attempt)

via Goodfellow et al. 2016