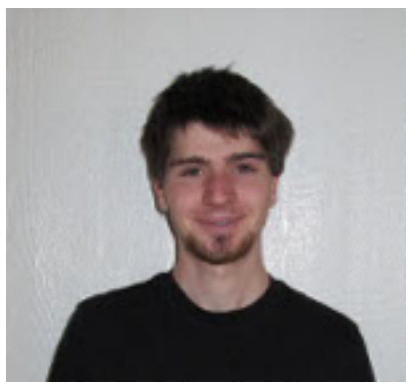
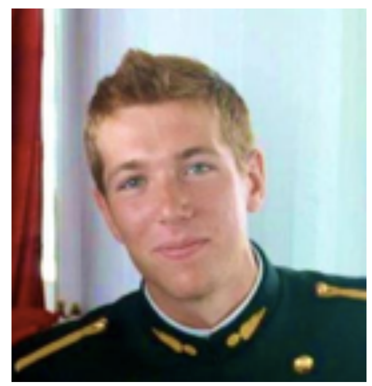
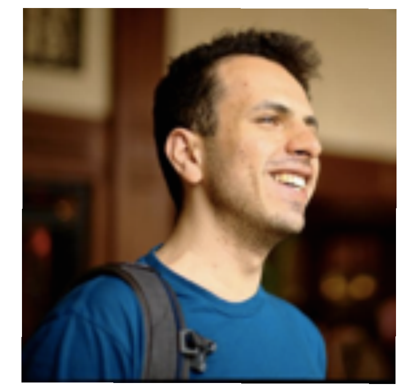
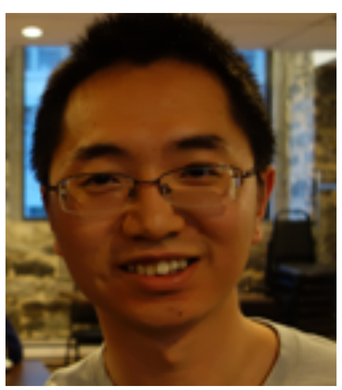
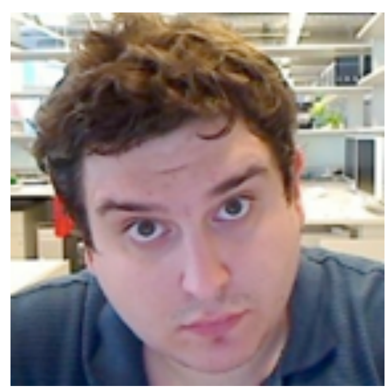
Generative adversarial networks


Ian Goodfellow


Jean Pouget-Abadie


Mehdi Mirza


Bing Xu


David Warde-Farley

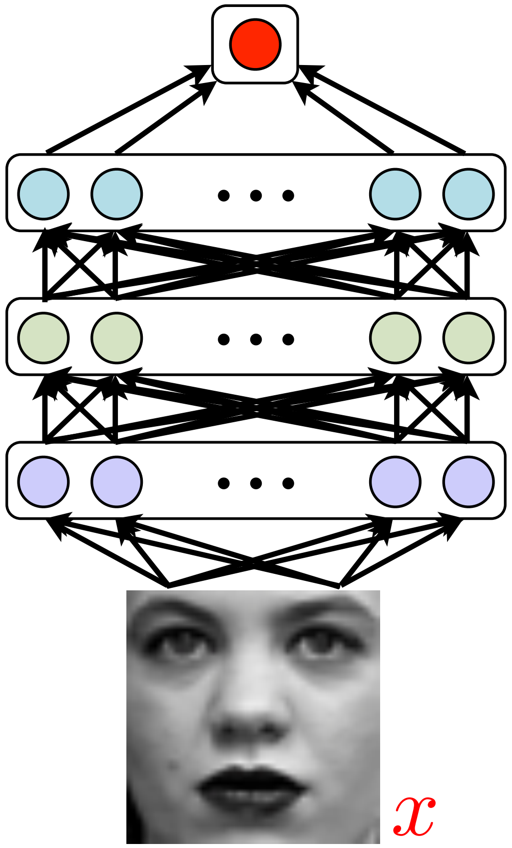

Sherjil Ozair


Aaron Courville


Yoshua Bengio

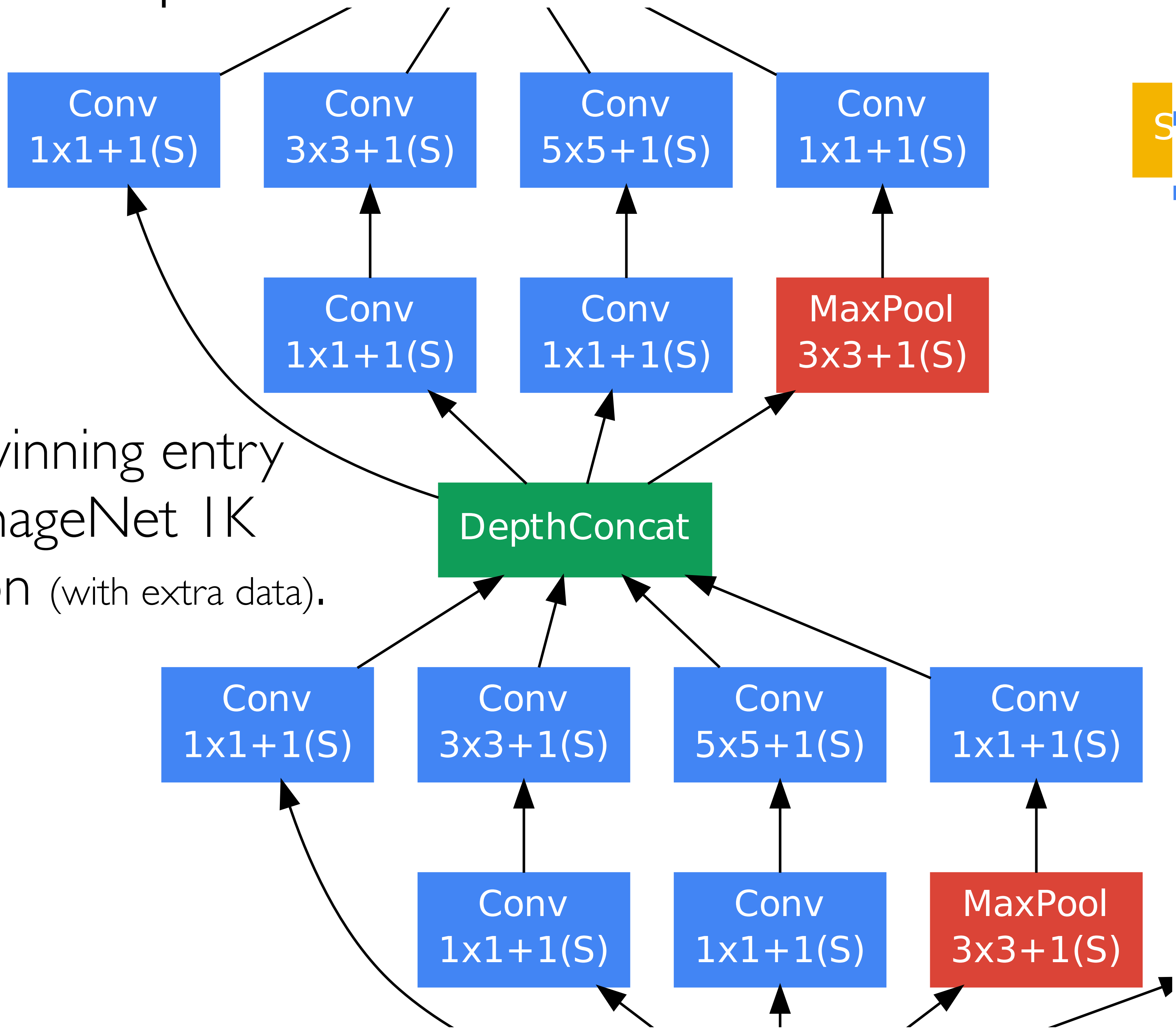
Discriminative deep learning

- Recipe for success

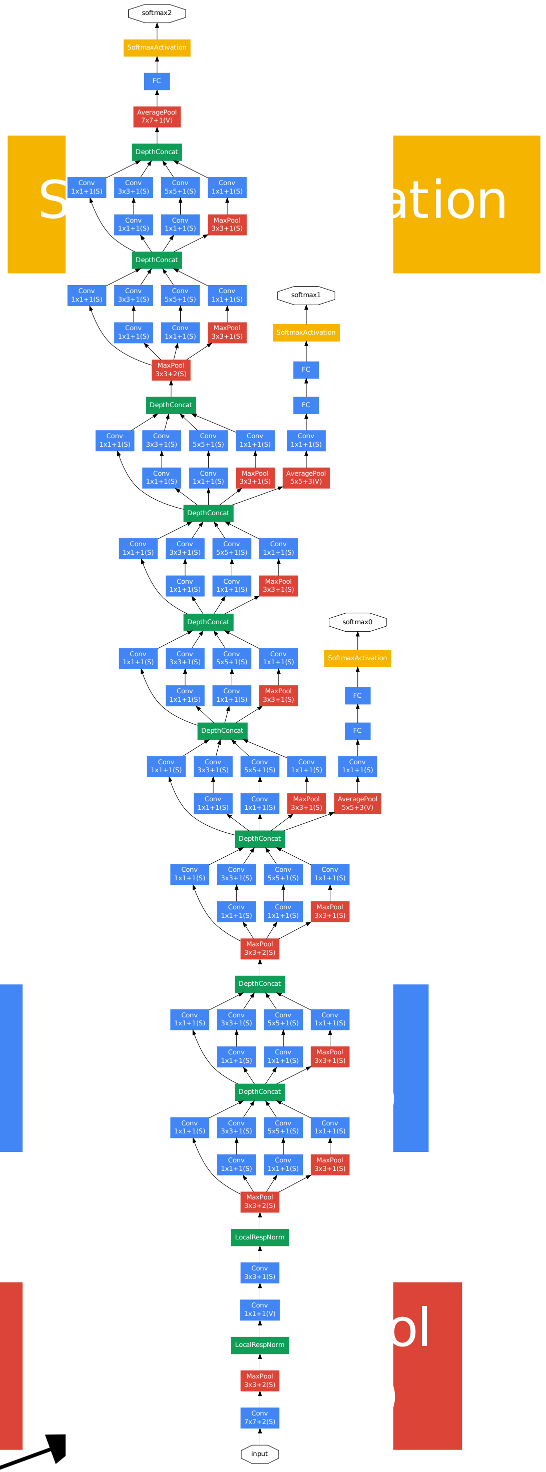


Discriminative deep learning

- Recipe for success:

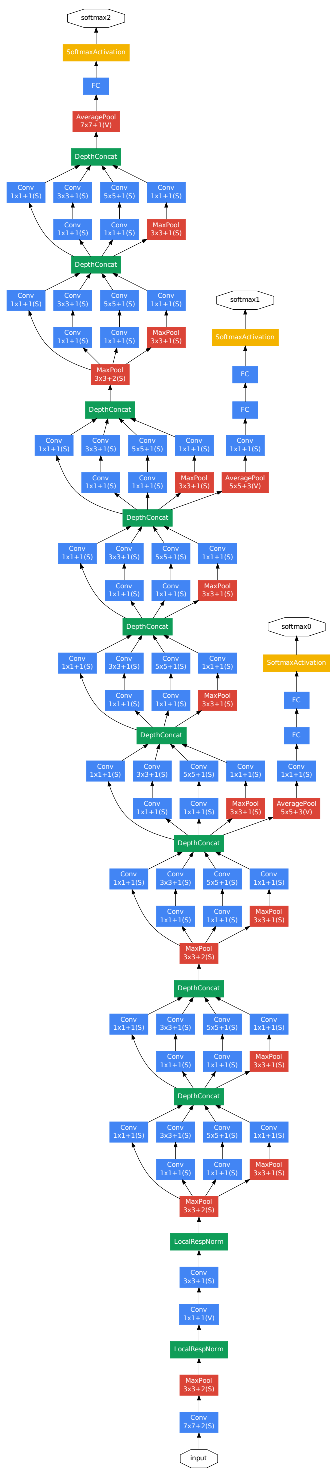


Google's winning entry into the ImageNet 1K competition (with extra data).



Discriminative deep learning

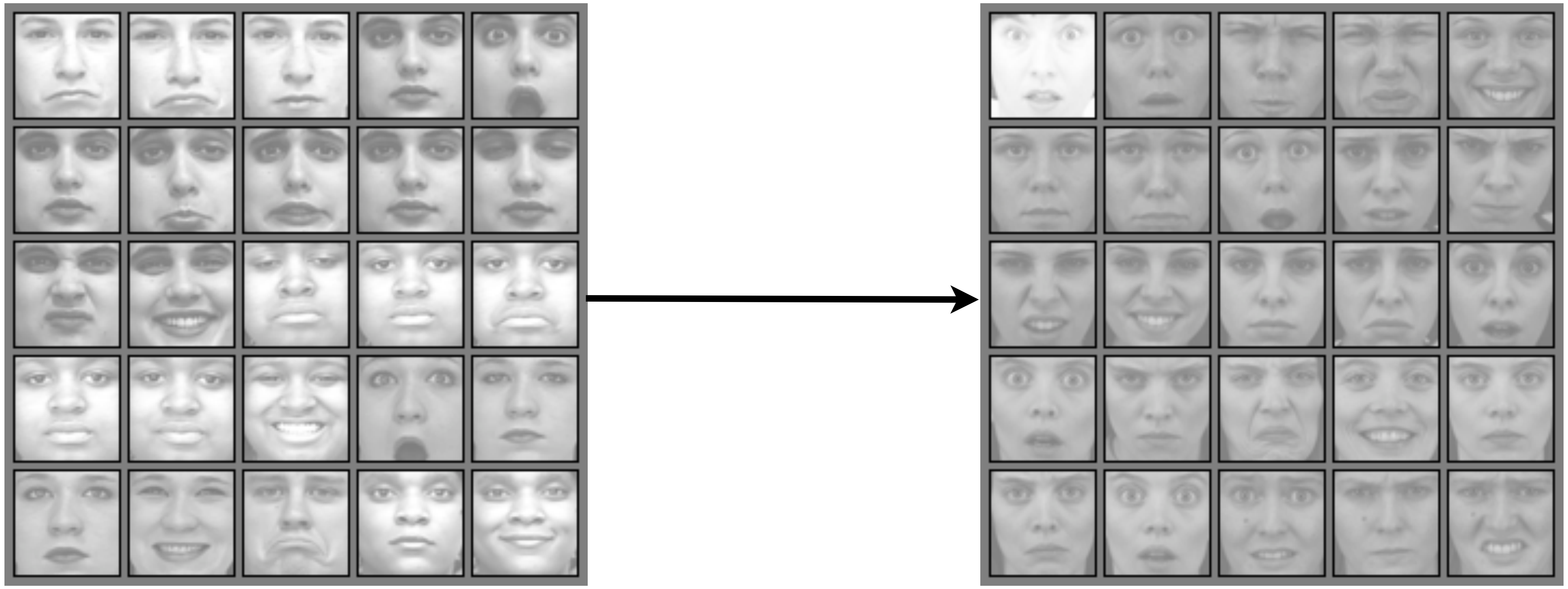
- Recipe for success:
  - Gradient backpropagation.
  - Dropout
  - Activation functions:
    - rectified linear
    - maxout



Google's winning entry into the ImageNet 1K competition (with extra data).

Generative modeling

- Have training examples $x \sim p_{\text{data}}(x)$
- Want a model that can draw samples: $x \sim p_{\text{model}}(x)$
- Where $p_{\text{model}} \approx p_{\text{data}}$



$x \sim p_{\text{data}}(x)$ $\qquad$ $x \sim p_{\text{model}}(x)$

Why generative models?

- Conditional generative models
  - Speech synthesis: Text ⇒ Speech
  - Machine Translation: French ⇒ English
    - French: Si mon tonton tond ton tonton, ton tonton sera tondu.
    - English: If my uncle shaves your uncle, your uncle will be shaved
  - Image ⇒ Image segmentation
- Environment simulator
  - Reinforcement learning
  - Planning
- Leverage unlabeled data

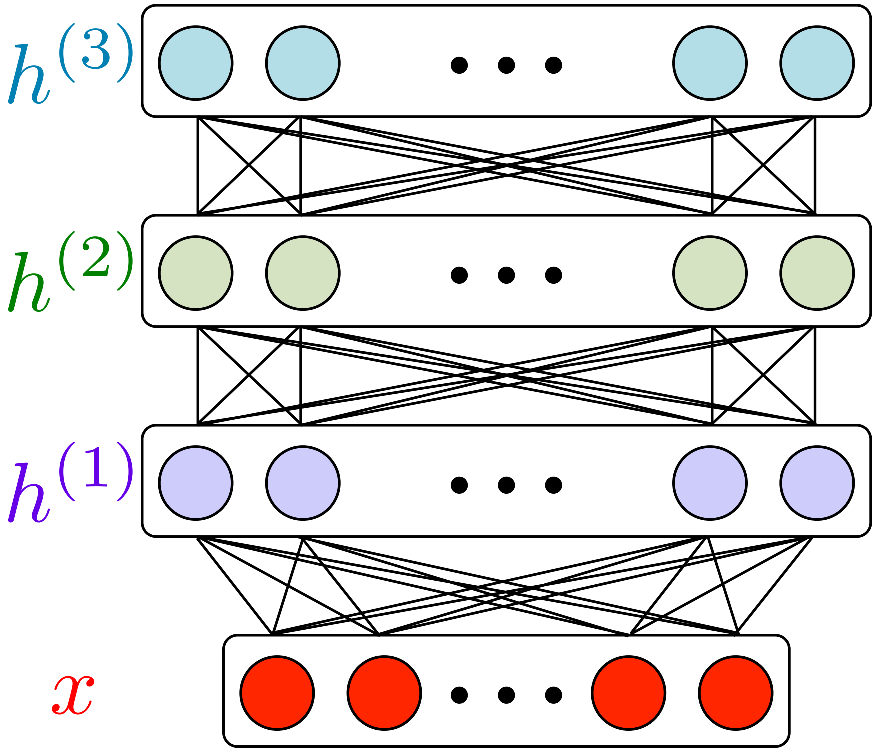
Maximum likelihood: the dominant approach

- ML objective function

$$\theta^* = \max_\theta \frac{1}{m} \sum_{i=1}^{m} \log p\left(x^{(i)}; \theta\right)$$

Undirected graphical models

- State-of-the-art general purpose undirected graphical model: **Deep Boltzmann machines**
- Several "hidden layers" h



$$p(h,x) = \frac{1}{Z}\tilde{p}(h,x)$$

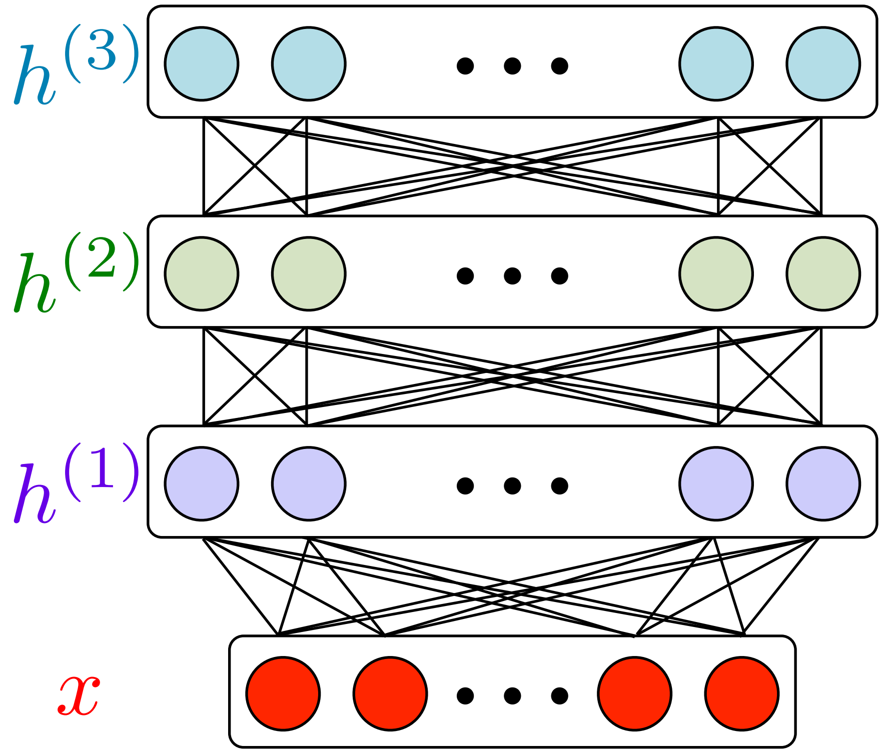
$$\tilde{p}(h,x) = \exp(-E(h,x))$$

$$Z = \sum_{h,x} \tilde{p}(h,x)$$

Undirected graphical models: disadvantage

- ML Learning requires that we draw samples:

$$\frac{d}{d\theta_i} \log p(x) = \frac{d}{d\theta_i} \left[\log \sum_h \tilde{p}(h, x) - \log Z(\theta) \right]$$



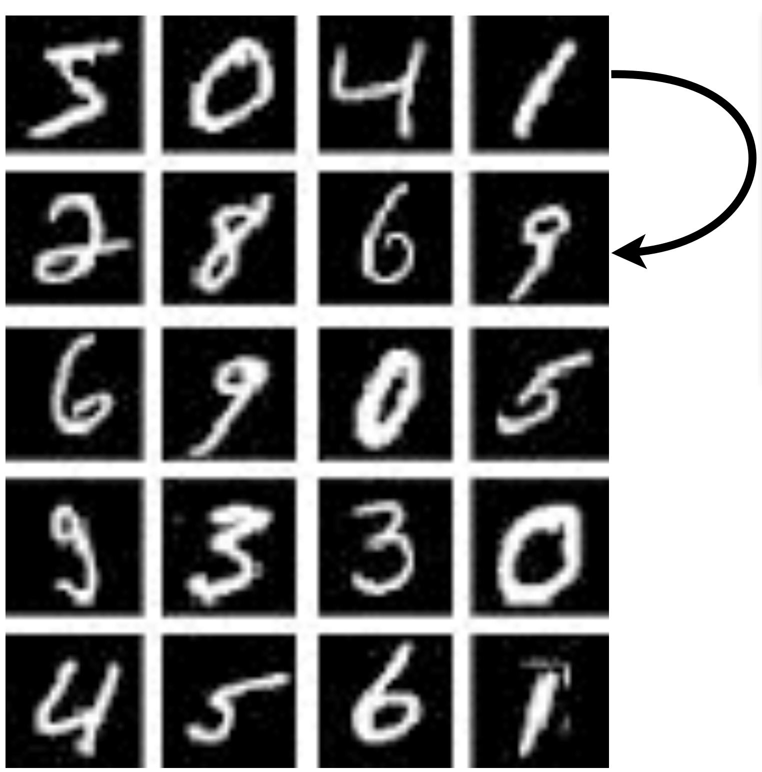
- Common way to do this is via MCMC (Gibbs sampling).

Boltzmann Machines: disadvantage

- Model is badly parameterized for learning high quality samples.
- Why?
  - Learning leads to large values of the model parameters.
    - Large valued parameters = peaky distribution.
  - Large valued parameters means slow mixing of sampler.
  - Slow mixing means that the gradient updates are correlated ⇒ leads to divergence of learning.

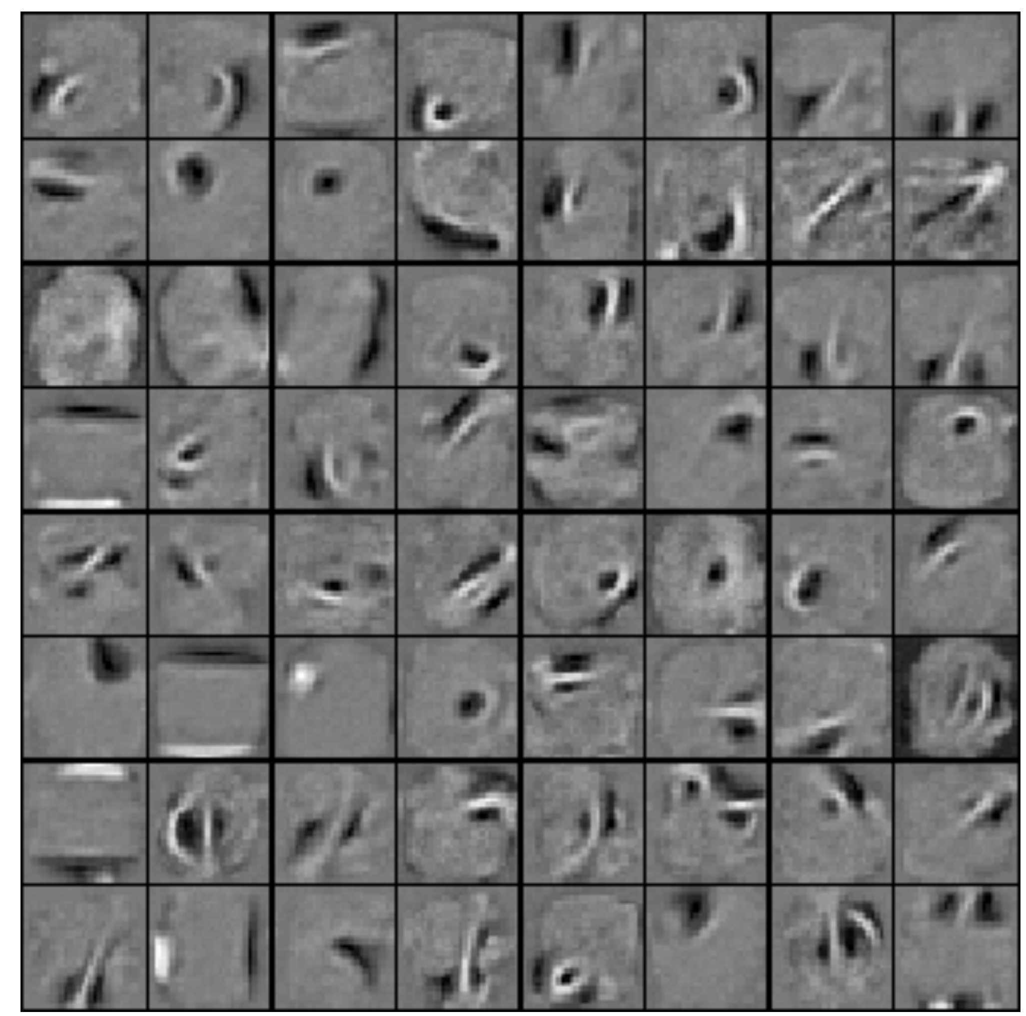
Boltzmann Machines: disadvantage

- Model is badly parameterized for learning high quality samples.
- Why poor mixing?



Coordinated flipping of low-level features

MNIST dataset



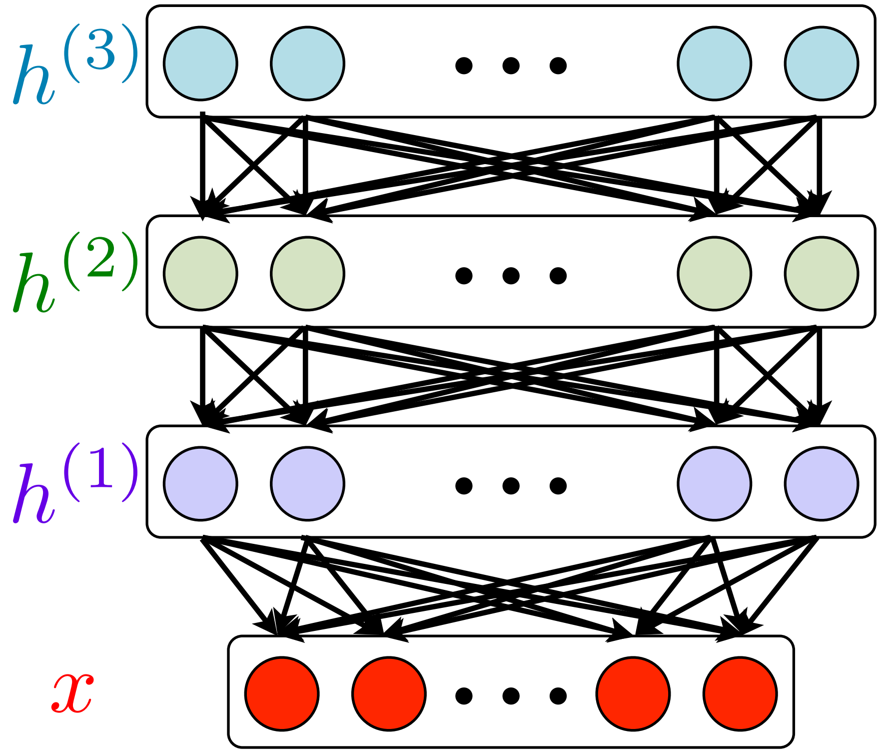
1st layer features (RBM)

Directed graphical models

$$p(x, h) = p(x \mid h^{(1)})p(h^{(1)} \mid h^{(2)}) \ldots p(h^{(L-1)} \mid h^{(L)})p(h^{(L)})$$

$$\frac{d}{d\theta_i} \log p(x) = \frac{1}{p(x)} \frac{d}{d\theta_i} p(x)$$

$$p(x) = \sum_h p(x \mid h)p(h)$$



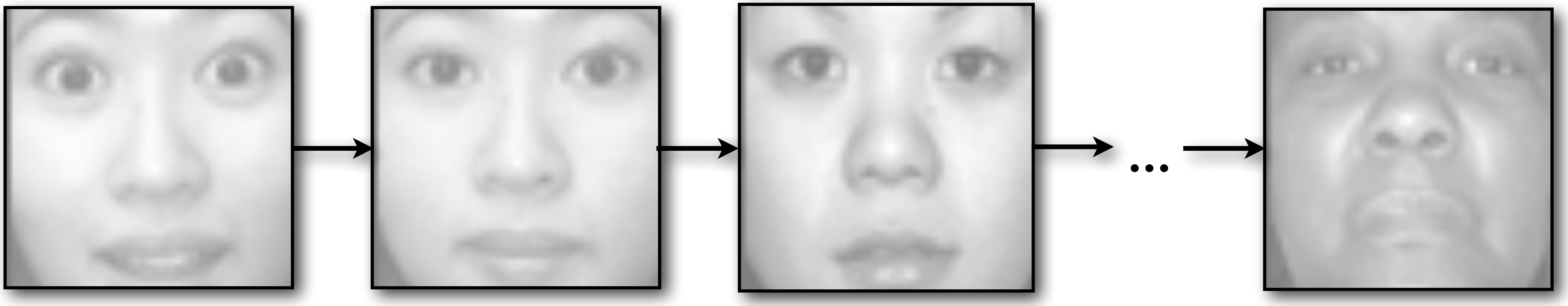
- Two problems:
  1. Summation over exponentially many states in ***h***
  2. Posterior inference, i.e. calculating ***p*(*h* | *x*)**, is intractable.

Directed graphical models: New approaches

- The Variational Autoencoder model:
  - Kingma and Welling, Auto-Encoding Variational Bayes, International Conference on Learning Representations (ICLR) 2014.
  - Rezende, Mohamed and Wierstra, Stochastic back-propagation and variational inference in deep latent Gaussian models. ArXiv.
  - Use a reparametrization that allows them to train very efficiently with gradient backpropagation.

Generative stochastic networks

- General strategy: Do not write a formula for $p(x)$, just learn to sample incrementally.



- Main issue: Subject to some of the same constraints on mixing as undirected graphical models.

Generative adversarial networks

- Don't write a formula for $p(x)$, just learn to sample directly.
- No summation over all states.
- How? By playing a game.

Two-player zero-sum game

- Your winnings + your opponent’s winnings = 0
- Minimax theorem: a rational strategy exists for all such finite games

Two-player zero-sum game

- Strategy: specification of which moves you make in which circumstances.
- Equilibrium: each player's strategy is the best possible for their opponent's strategy.
- Example: Rock-paper-scissors:
  - *Mixed strategy equilibrium*
  - Choose you action at random

You \ Your opponent	Rock	Paper	Scissors
Rock	0	-1	1
Paper	1	0	-1
Scissors	-1	1	0

Generative modeling with game theory?

- Can we design a game with a mixed-strategy equilibrium that forces one player to learn to generate from the data distribution?

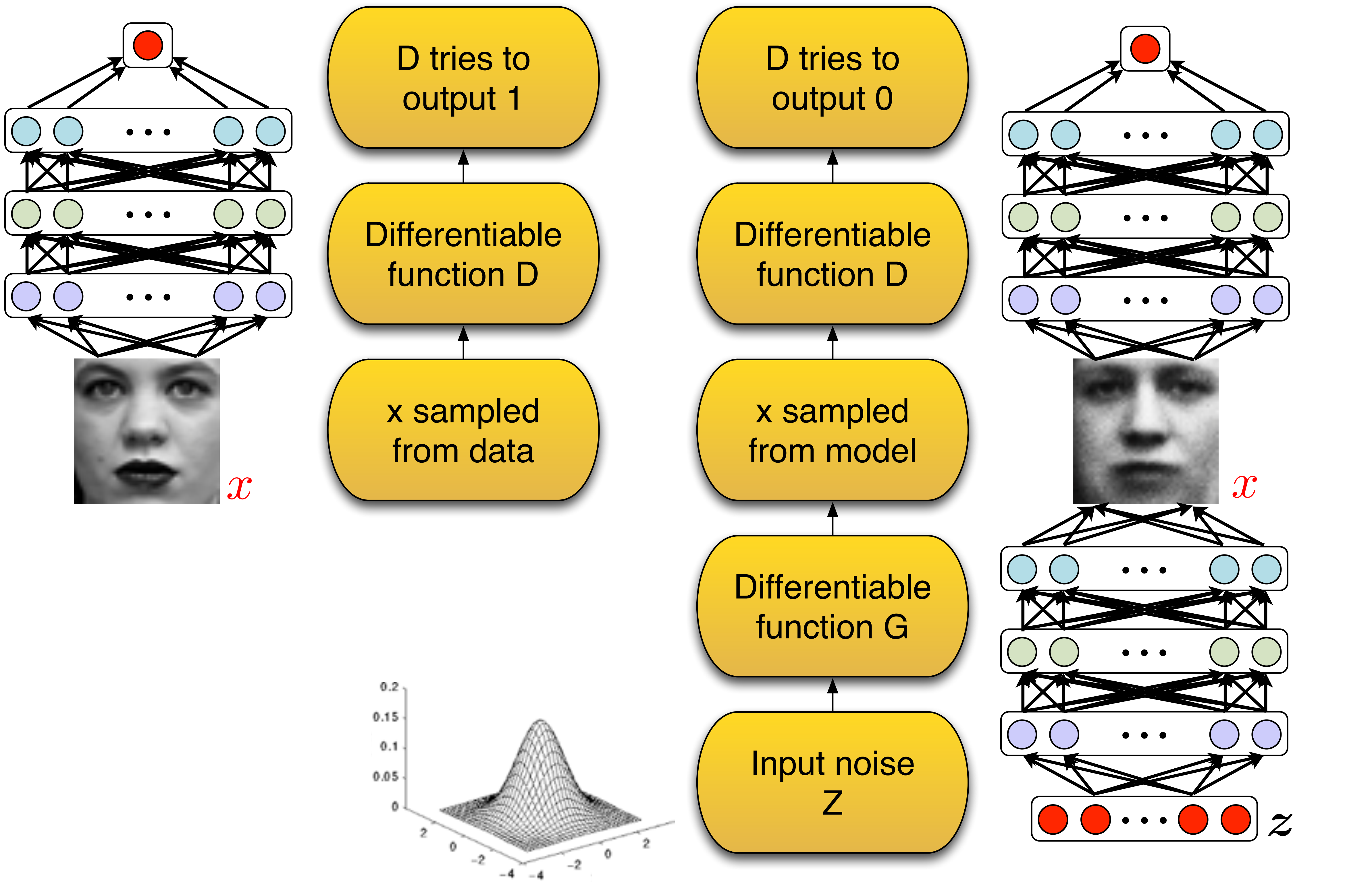
Adversarial nets framework

- A game between two players:
  1. Discriminator D
  2. Generator G

- D tries to discriminate between:
  - A sample from the data distribution.
  - And a sample from the generator G.

- G tries to "trick" D by generating samples that are hard for D to distinguish from data.

Adversarial nets framework



Zero-sum game

- Minimax objective function:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

- In practice, to estimate G we use:

$$\max_G \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log D(G(\boldsymbol{z}))]$$

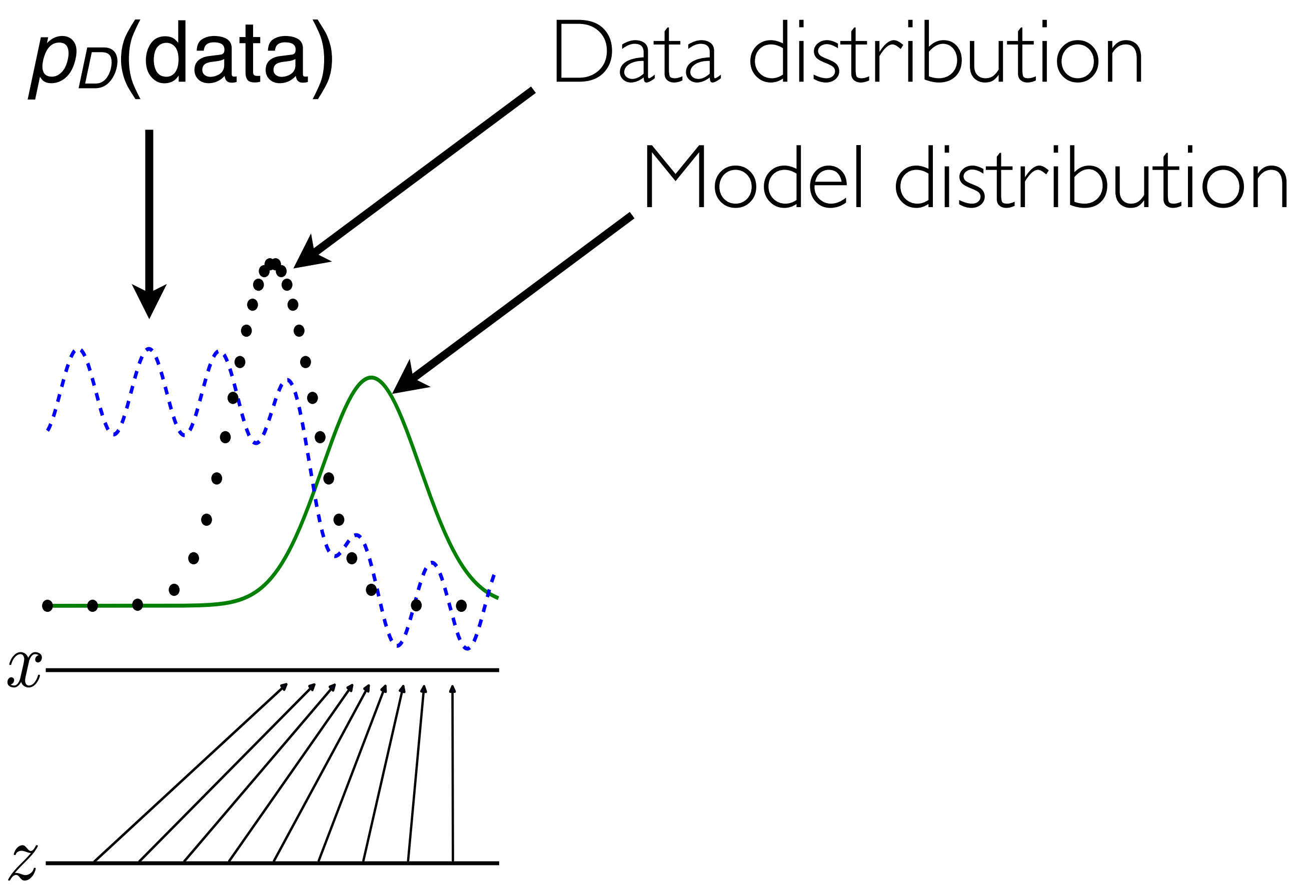
Why? Stronger gradient for G when D is very good.

Discriminator strategy

- Optimal strategy for any $p_{\text{model}}(x)$ is always

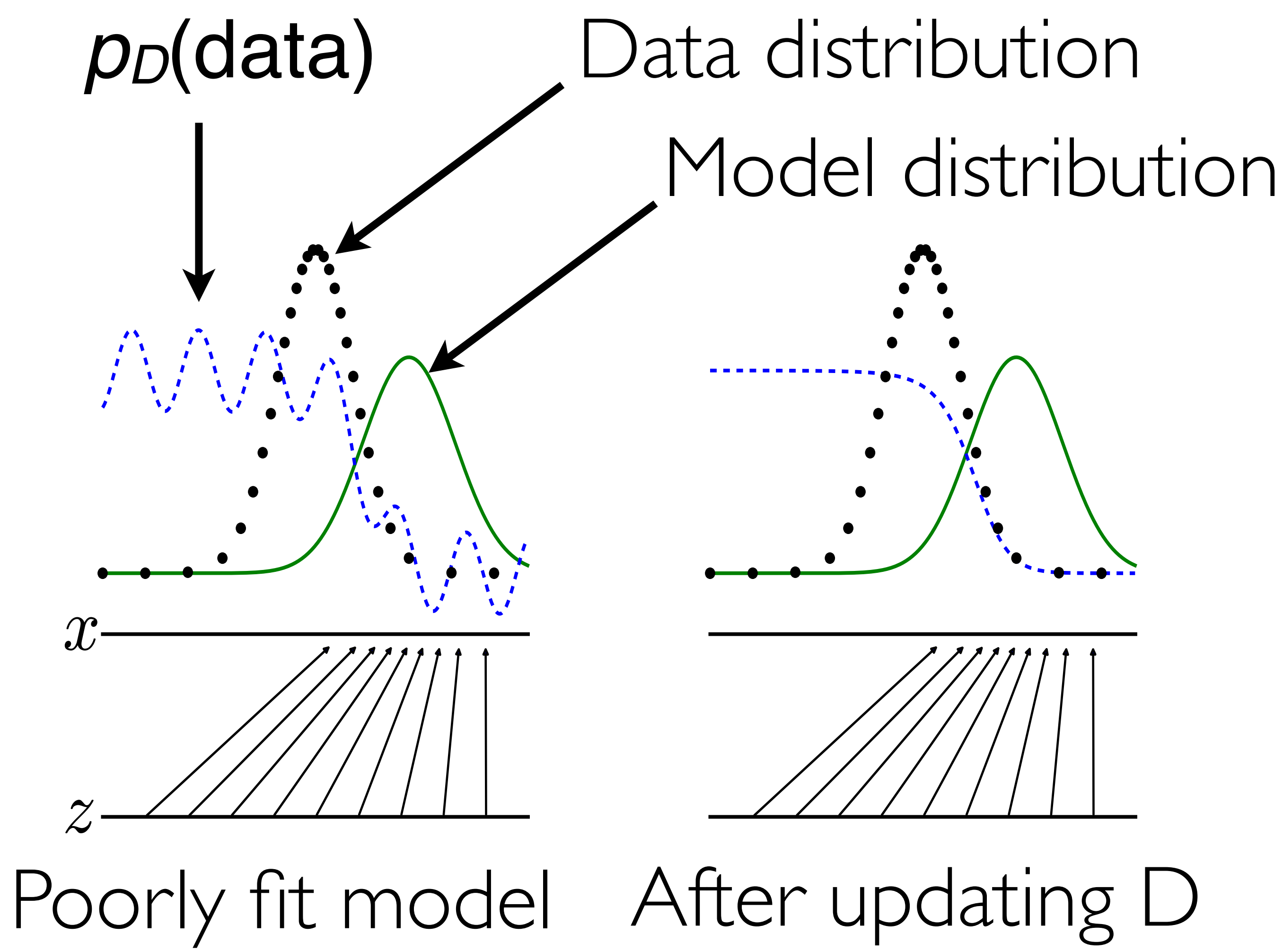
$$D(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{model}}(x)}$$

Learning process

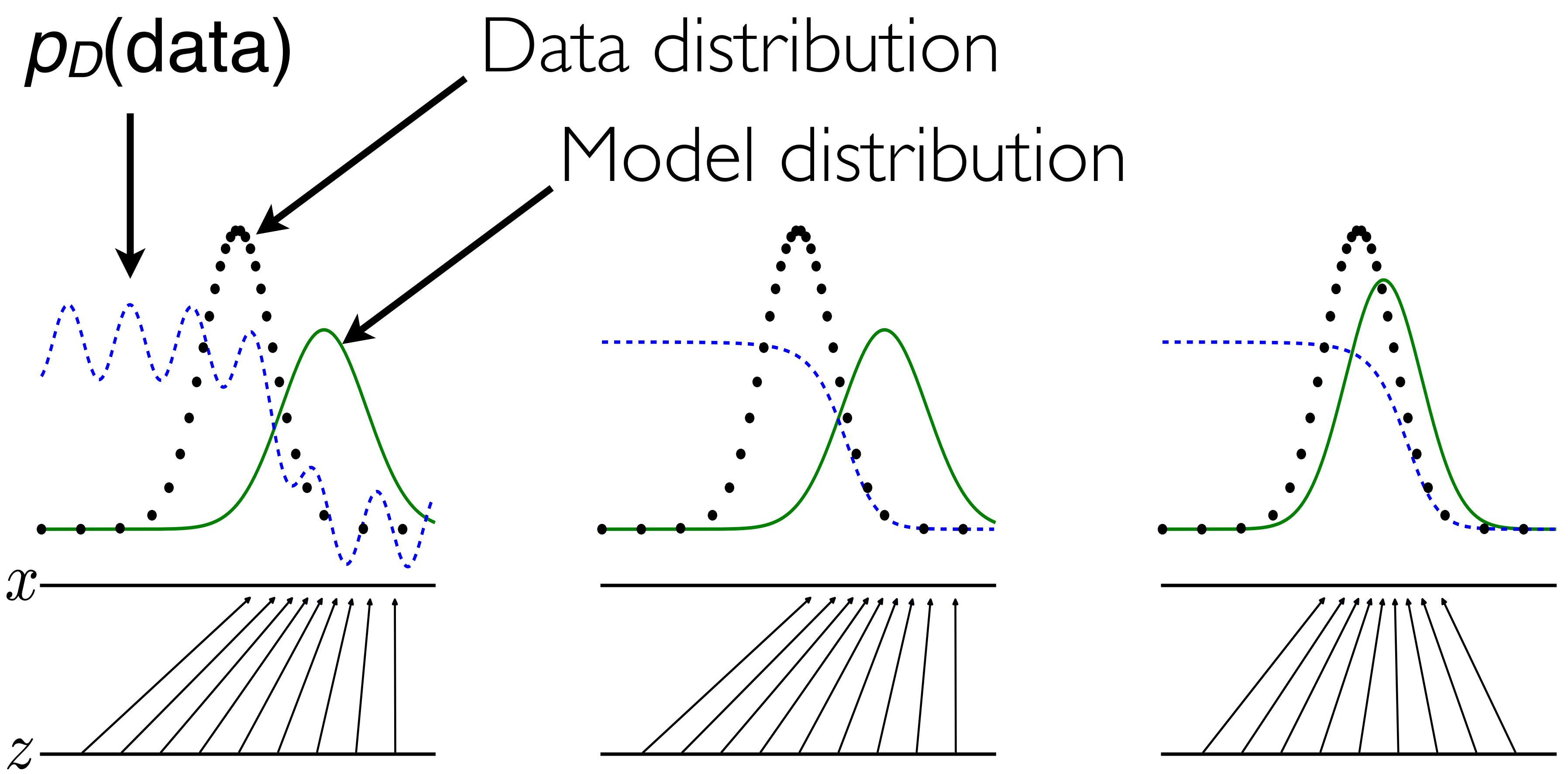


Poorly fit model

Learning process

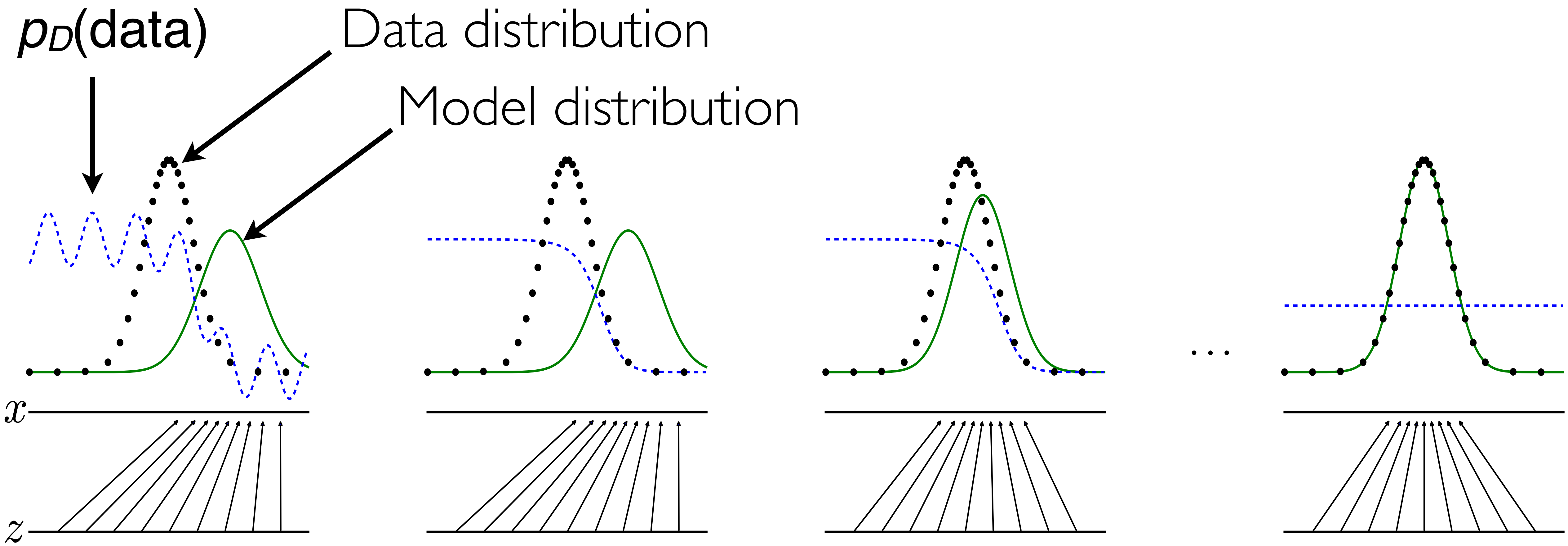


Learning process



Poorly fit model | After updating D | After updating G

Learning process



Poorly fit model | After updating D | After updating G | Mixed strategy equilibrium

Theoretical properties

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

- Theoretical properties (assuming infinite data, infinite model capacity, direct updating of generator's distribution):
  - Unique global optimum.
  - Optimum corresponds to data distribution.
  - Convergence to optimum guaranteed.

Quantitative likelihood results

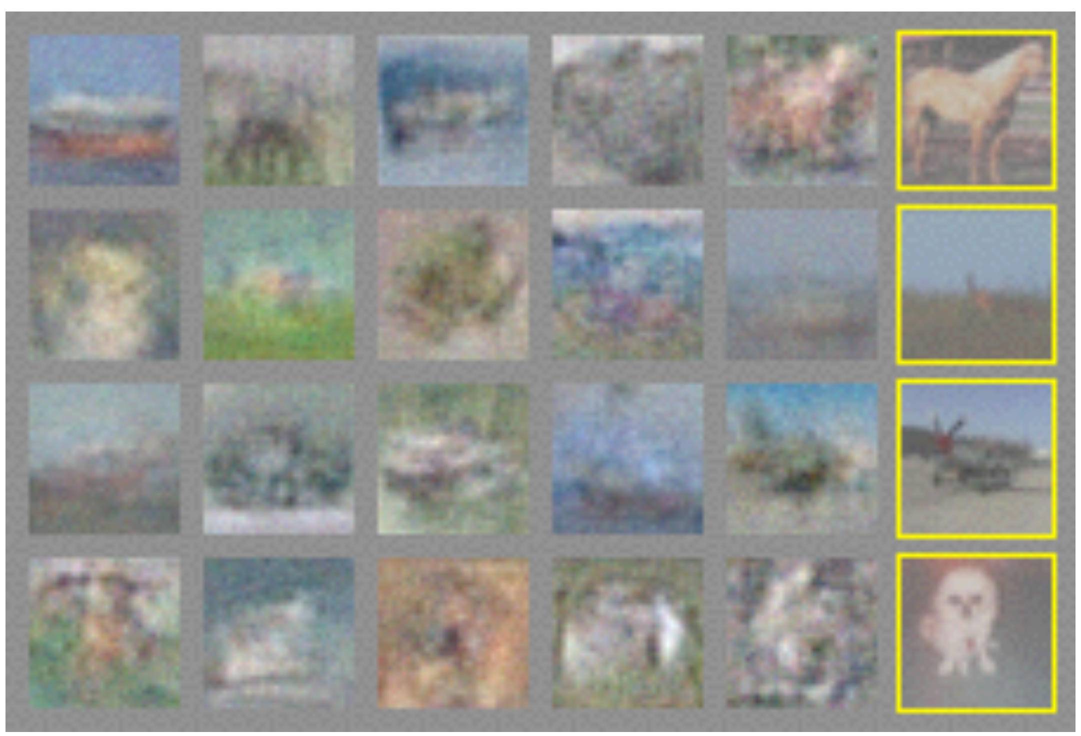
- Parzen window-based log-likelihood estimates.
  - Density estimate with Gaussian kernels centered on the samples drawn from the model.

Model	MNIST	TFD
DBN [3]	138 ± 2	1909 ± 66
Stacked CAE [3]	121 ± 1.6	$\mathbf{2110 \pm 50}$
Deep GSN [6]	214 ± 1.1	1890 ± 29
Adversarial nets	$\mathbf{225 \pm 2}$	$\mathbf{2057 \pm 26}$

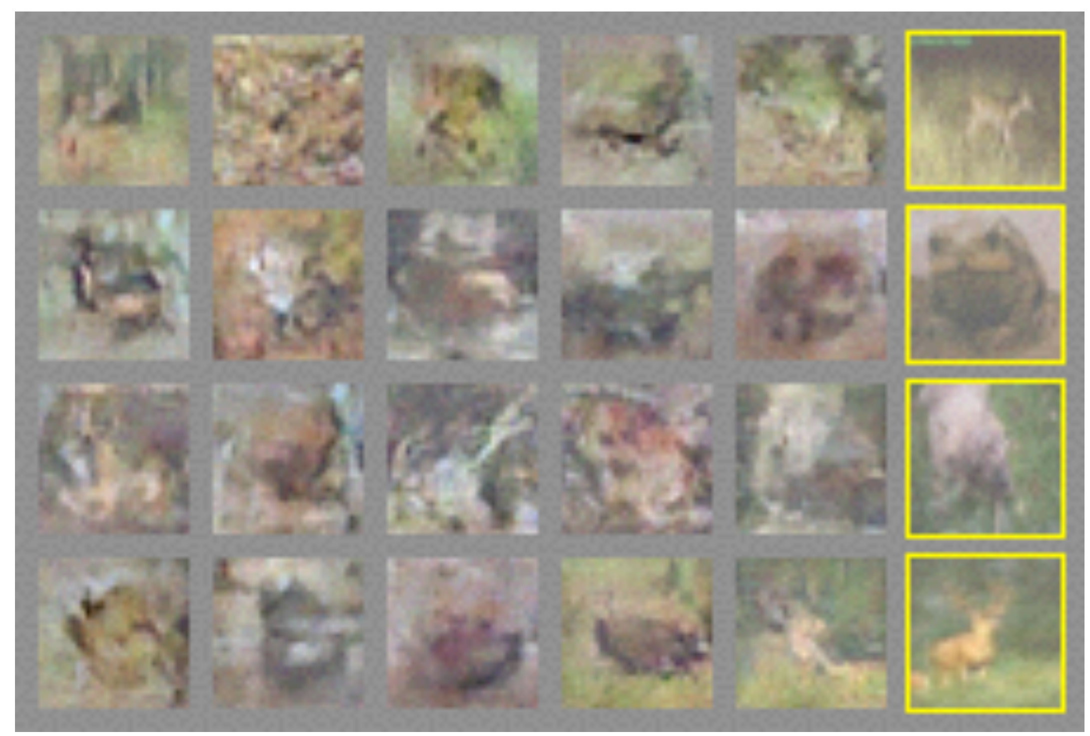
Visualization of model samples

MNIST

TFD

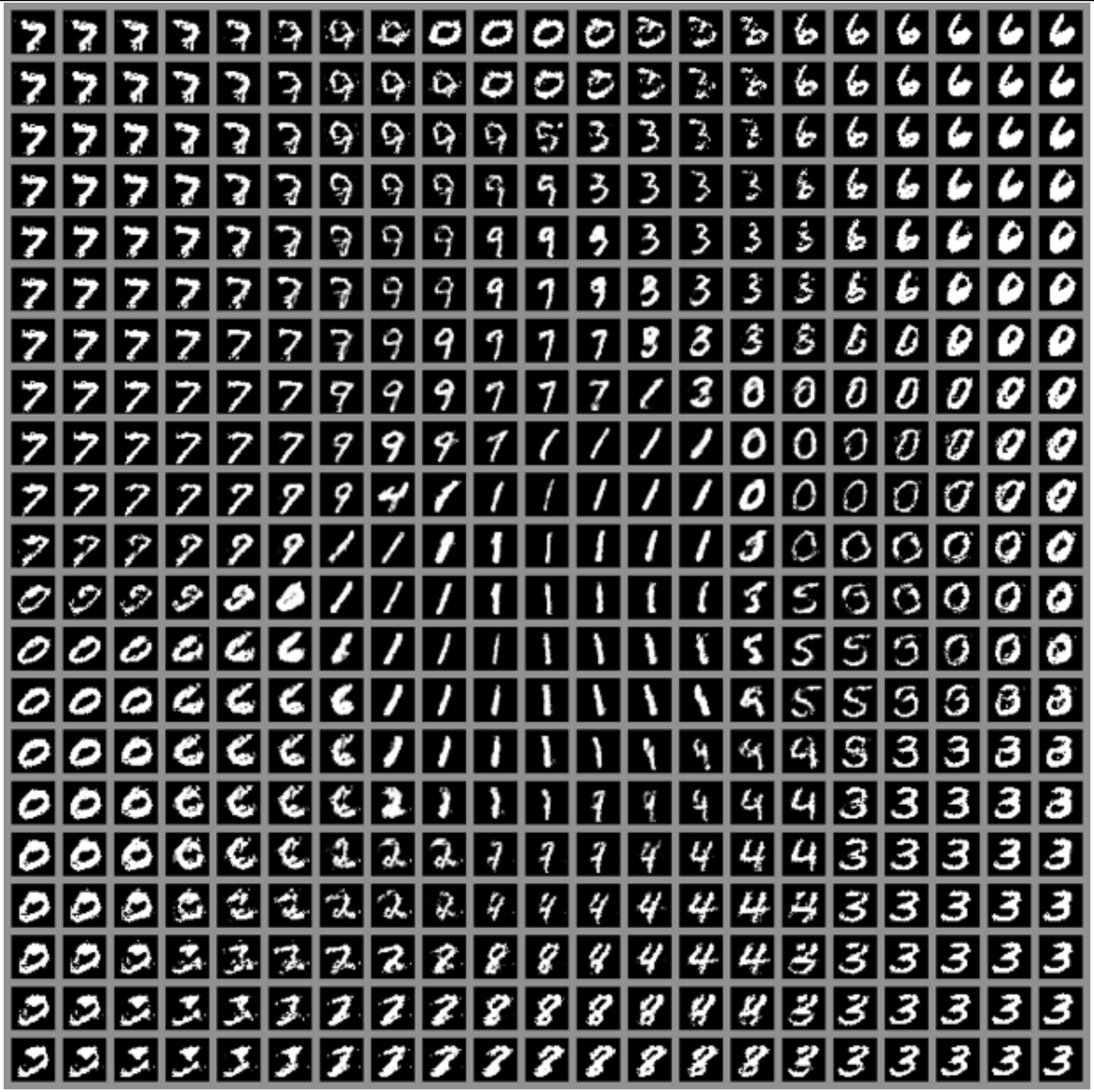


CIFAR-10 (fully connected)



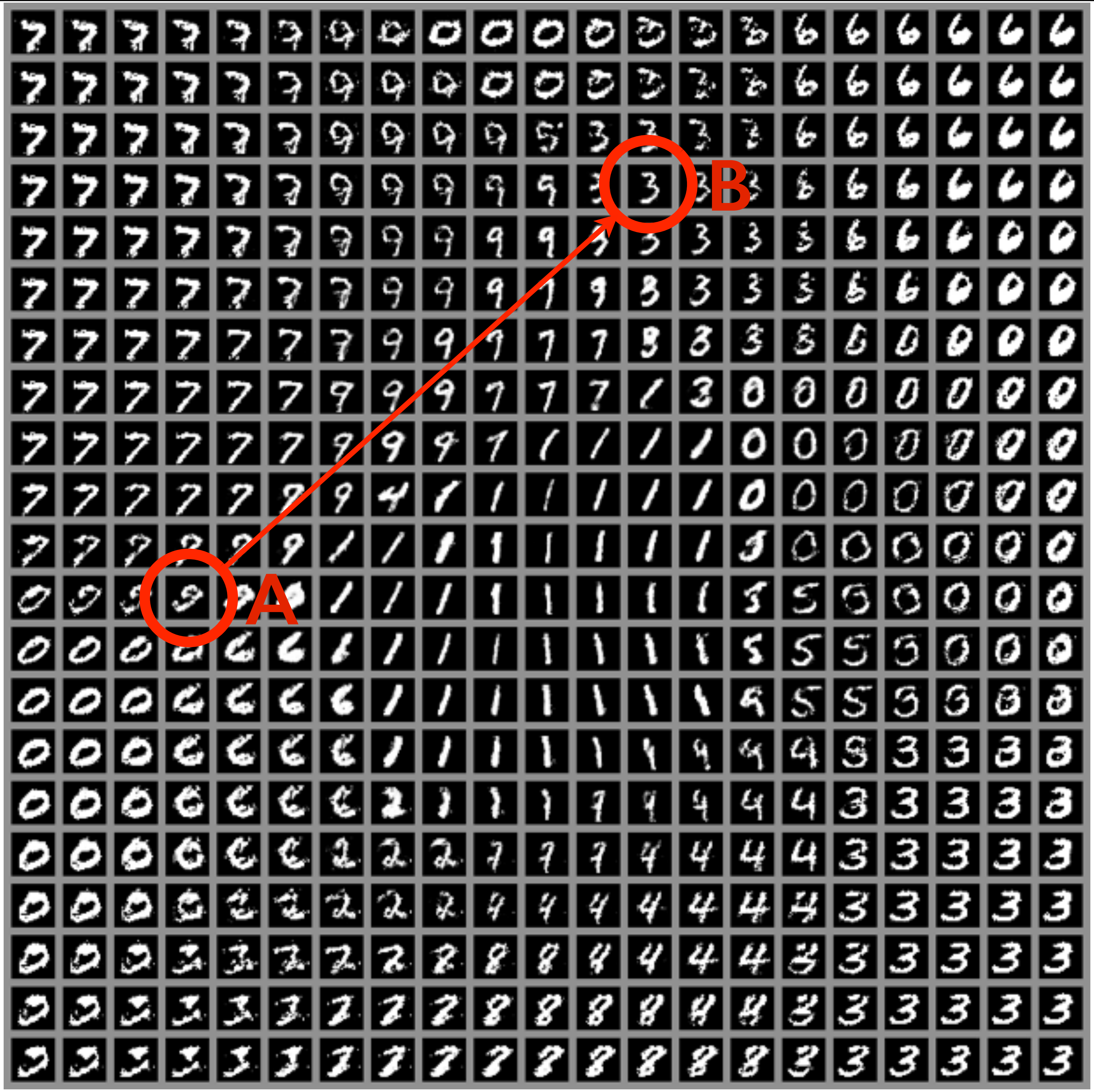
CIFAR-10 (convolutional)

Learned 2-D manifold of MNIST

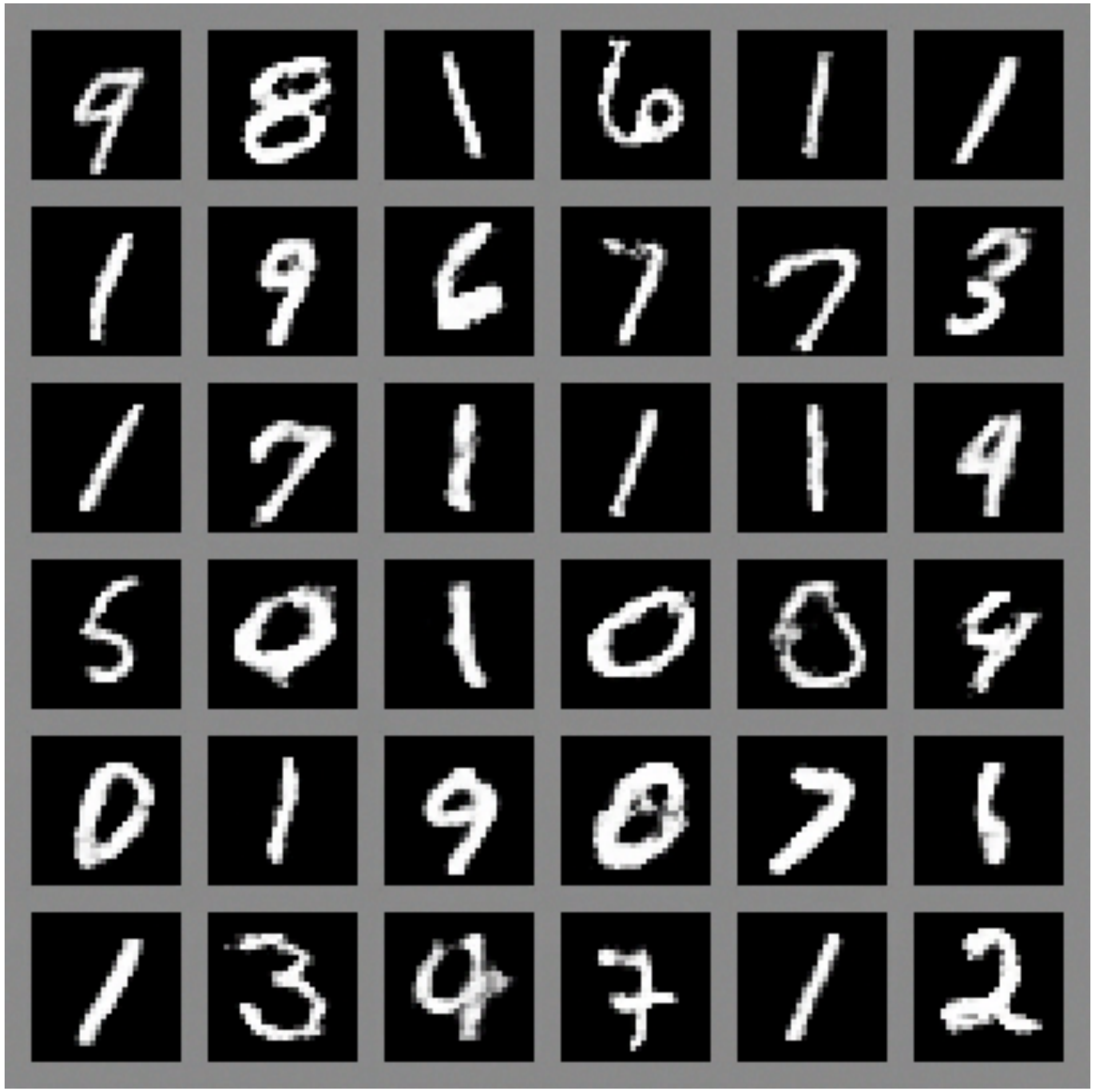


Visualizing trajectories

1. Draw sample (A)
2. Draw sample (B)
3. Simulate samples along the path between A and B
4. Repeat steps 1-3 as desired.



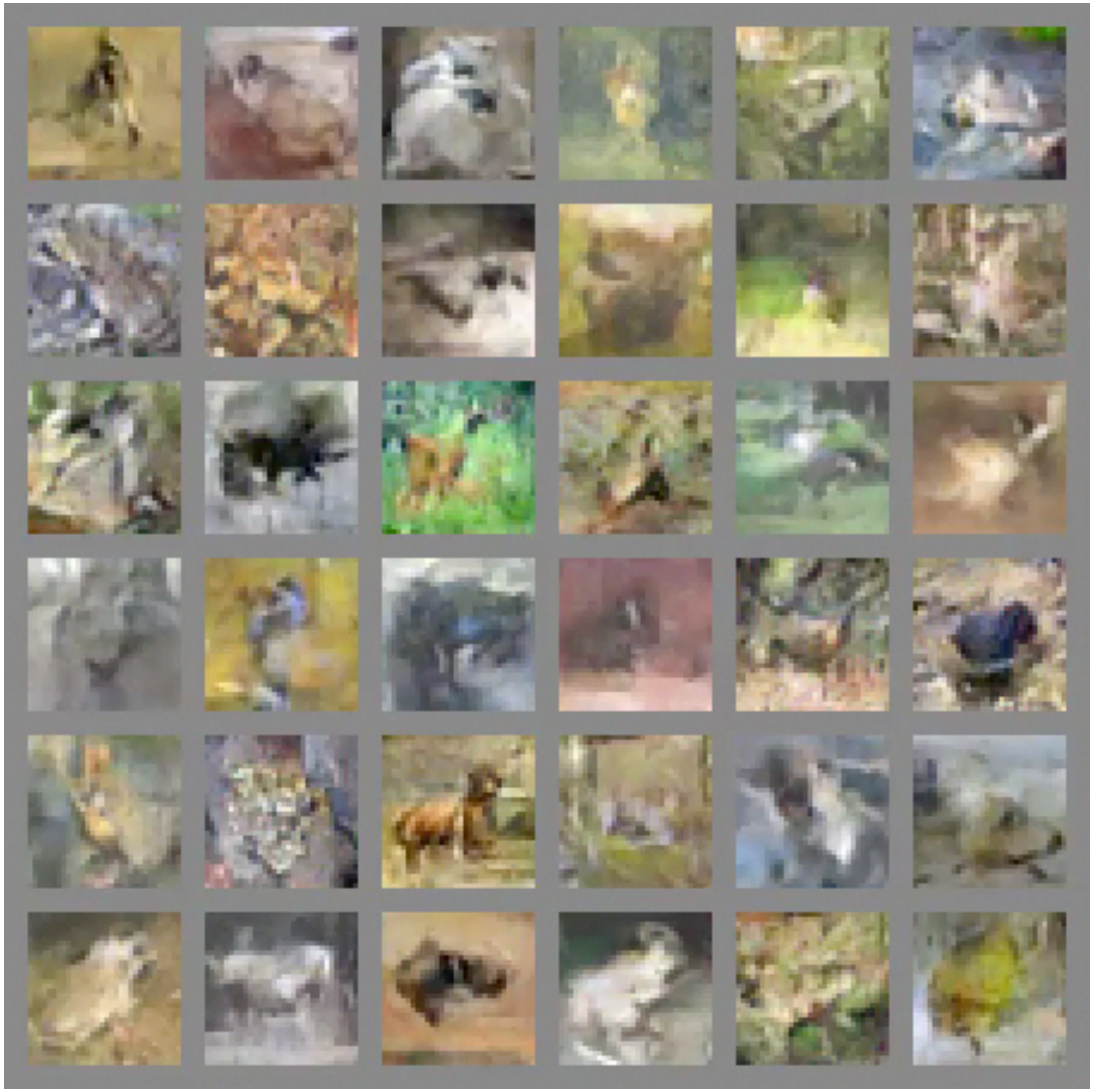
Visualization of model trajectories



MNIST digit dataset

Toronto Face Dataset (TFD)

Visualization of model trajectories



CIFAR-10
(convolutional)

Extensions

- Conditional model:
  - Learn $p(x \mid y)$
  - Discriminator is trained on (x,y) pairs
  - Generator net gets y and z as input
  - Useful for: Translation, speech synth, image segmentation.

Extensions

- Inference net:
  - Learn a network to model $p(z \mid x)$
  - Infinite training set!

Extensions

- Take advantage of high amounts of unlabeled data using the generator.
- Train G on a large, unlabeled dataset
- Train G' to learn $p(z|x)$ on an infinite training set
- Add a layer on top of G', train on a small labeled training set

Extensions

- Take advantage of unlabeled data using the discriminator
- Train G and D on a large amount of unlabeled data
    - Replace the last layer of D
    - Continue training D on a small amount of labeled data

Thank You.

Questions?