Security and Fairness of Deep Learning

Deep generative models

Anupam Datta

CMU

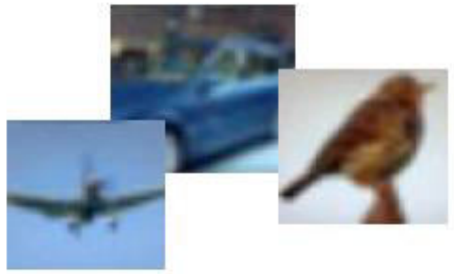
Spring 2018

Generative models

- Collect large amount of data in some domain
- Train generative model to generate data like it

Generative model

- Given training data generate samples from same distribution (density estimation)



Training data ~ $p_{data}(x)$

Generated samples ~ $p_{model}(x)$

Want to learn $p_{model}(x)$ similar to $p_{data}(x)$

Source: Fei-Fei Li et al.

Key idea

- Generative models cannot memorize training data since they do not have enough parameters
- Forced to learn higher-level features from which they can reconstruct data

Why generative models?

- Long-term hope
  - Learn the natural features of a dataset
- Current applications
  - image denoising, inpainting, super-resolution, and neural network pretraining in cases where labeled data is expensive (will discuss today)

Generative models

- Generative adversarial networks (GAN)

- Other models
  - Variational autoencoders (see also: Variational fair autoencoder)
  - Boltzmann machines

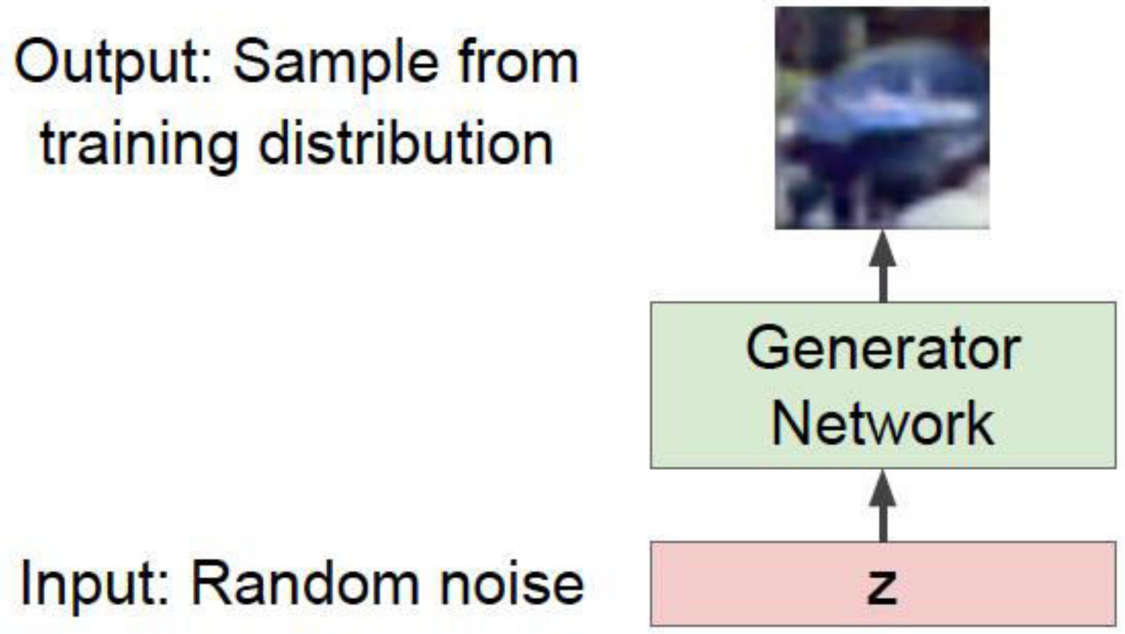
GANs

- Goal: Sample from complex, high-dimensional training distribution

- Approach
  - Sample from a simple distribution (e.g., random noise)
  - Learn transformation to training distribution

- Question
  - How to represent this complex transformation?
  - A neural network!

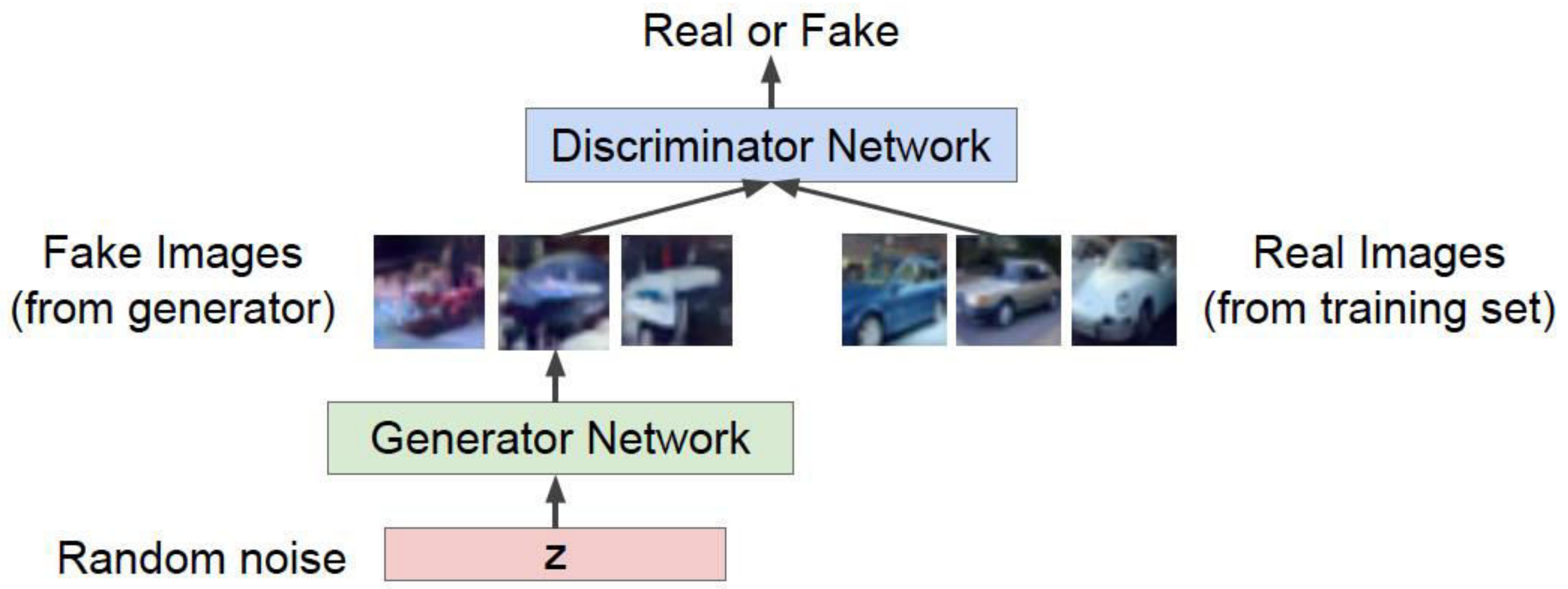
GANs



- Implicit density estimation
  - Can sample from training distribution without explicitly representing it

Training GANs

- Two player game
  - Generator: try to fool discriminator by generating real-looking images
  - Discriminator: try to distinguish between real and fake images



Training GANs

- Two player game
  - Generator: try to fool discriminator by generating real-looking images
  - Discriminator: try to distinguish between real and fake images

Train jointly in **minimax game**

Discriminator outputs likelihood in (0,1) of real image

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log \underbrace{D_{\theta_d}(x)}_{\text{Discriminator output for real data x}} + \mathbb{E}_{z \sim p(z)} \log(1 - \underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\text{Discriminator output for generated fake data G(z)}}) \right]$$

Training GANs

Train jointly in **minimax game**

Minimax objective function:

Discriminator outputs likelihood in (0,1) of real image

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log \underbrace{D_{\theta_d}(x)}_{\text{Discriminator output for real data x}} + \mathbb{E}_{z \sim p(z)} \log(1 - \underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\text{Discriminator output for generated fake data G(z)}}) \right]$$

- Discriminator (θ_d) wants to **maximize objective** such that D(x) is close to 1 (real) and D(G(z)) is close to 0 (fake)

- Generator (θ_g) wants to **minimize objective** such that D(G(z)) is close to 1 (discriminator is fooled into thinking generated G(z) is real)

Training GANs

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))\right]$$

Alternate between:

1. **Gradient ascent** on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))\right]$$

2. **Gradient descent** on generator

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

Training GANs

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x\sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z\sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))\right]$$

Alternate between:

1. **Gradient ascent** on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x\sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z\sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))\right]$$

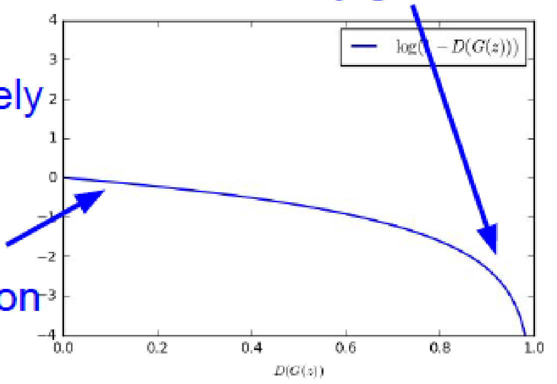
2. **Gradient descent** on generator

$$\min_{\theta_g} \mathbb{E}_{z\sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

In practice, optimizing this generator objective does not work well!

When sample is likely fake, want to learn from it to improve generator. But gradient in this region is relatively flat!

Gradient signal dominated by region where sample is already good



Training GANs

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:

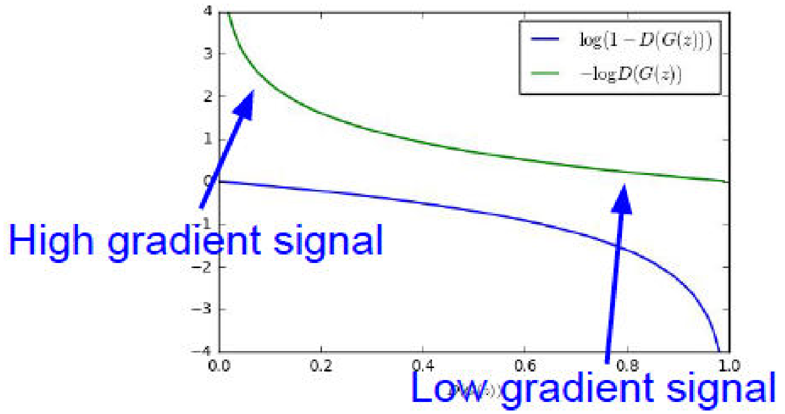
1. **Gradient ascent** on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

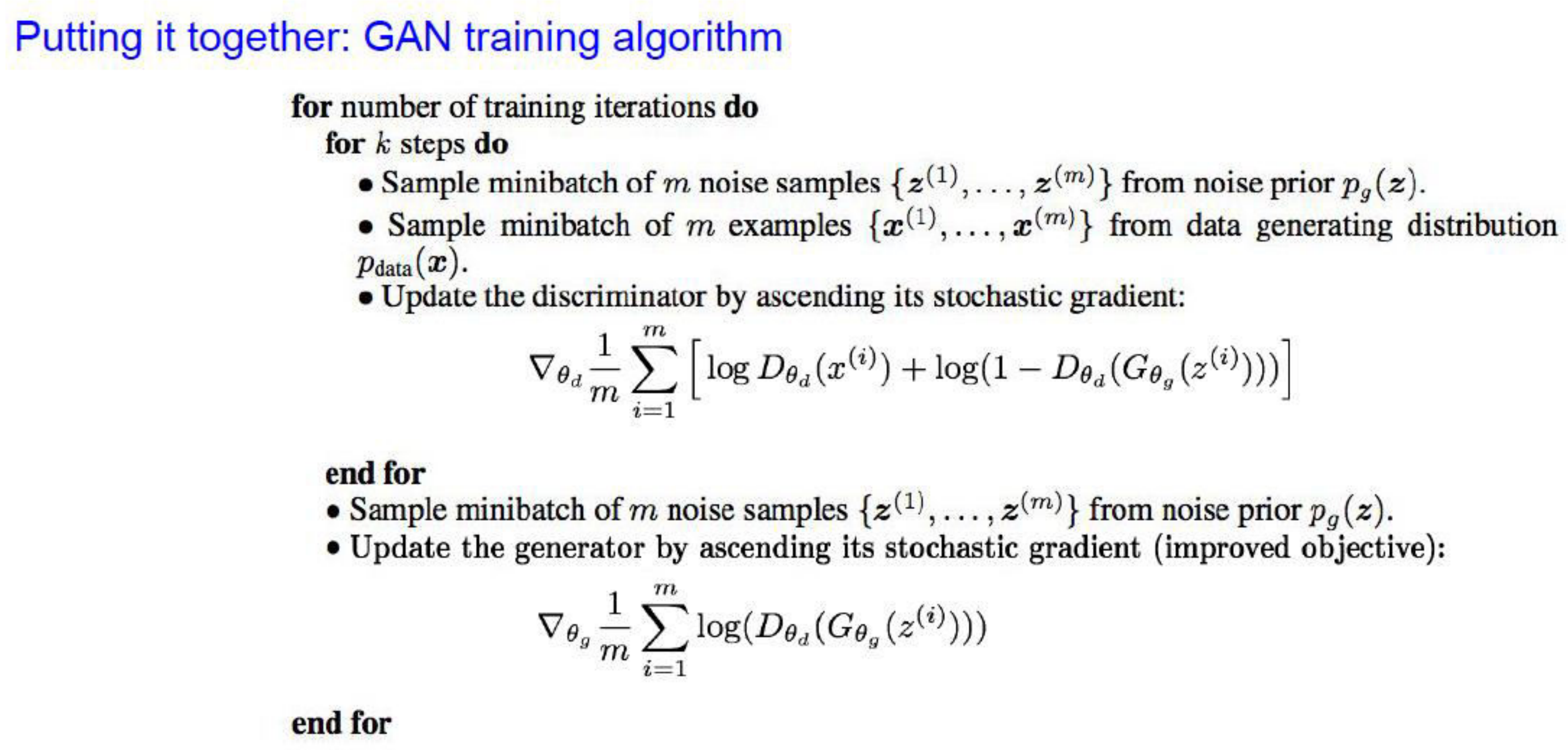
2. **Instead: Gradient ascent** on generator, different objective

$$\max_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(D_{\theta_d}(G_{\theta_g}(z)))$$

Instead of minimizing likelihood of discriminator being correct, now maximize likelihood of discriminator being wrong.
Same objective of fooling discriminator, but now higher gradient signal for bad samples => works much better! Standard in practice.



Training GANs

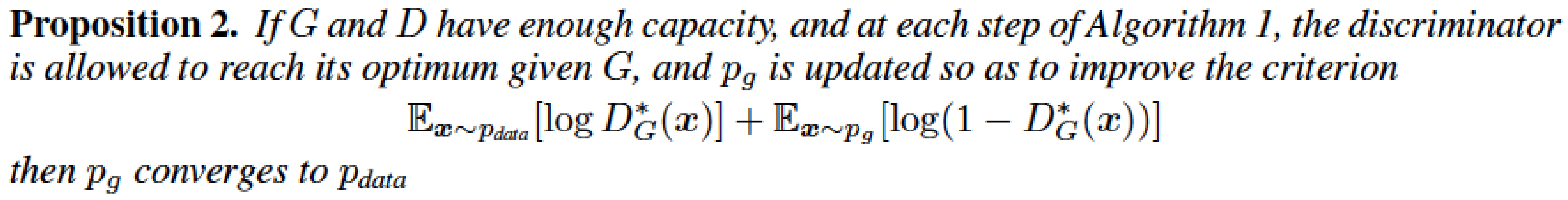


Putting it together: GAN training algorithm

```
for number of training iterations do
  for $k$ steps do
    • Sample minibatch of $m$ noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
    • Sample minibatch of $m$ examples $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
    • Update the discriminator by ascending its stochastic gradient:
        $\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[ \log D_{\theta_d}(x^{(i)}) + \log(1 - D_{\theta_d}(G_{\theta_g}(z^{(i)}))) \right]$
  end for
  • Sample minibatch of $m$ noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  • Update the generator by ascending its stochastic gradient (improved objective):
      $\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(D_{\theta_d}(G_{\theta_g}(z^{(i)})))$
end for
```

Convergence theorem

- The training criterion allows one to recover the data generating distribution as G and D are given enough capacity



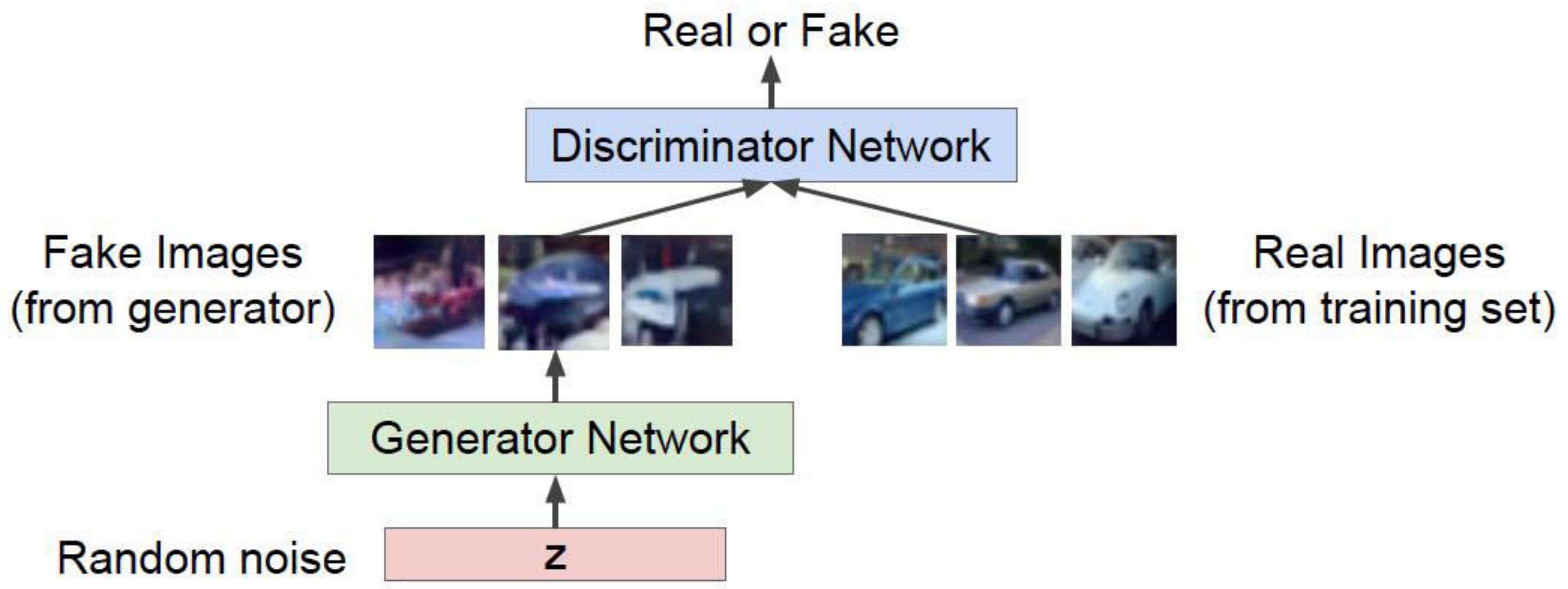
Proposition 2. *If G and D have enough capacity, and at each step of Algorithm 1, the discriminator is allowed to reach its optimum given G, and p_g is updated so as to improve the criterion*

$$\mathbb{E}_{\boldsymbol{x}\sim p_{data}}[\log D^*_G(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log(1 - D^*_G(\boldsymbol{x}))]$$

then p_g converges to p_{data}

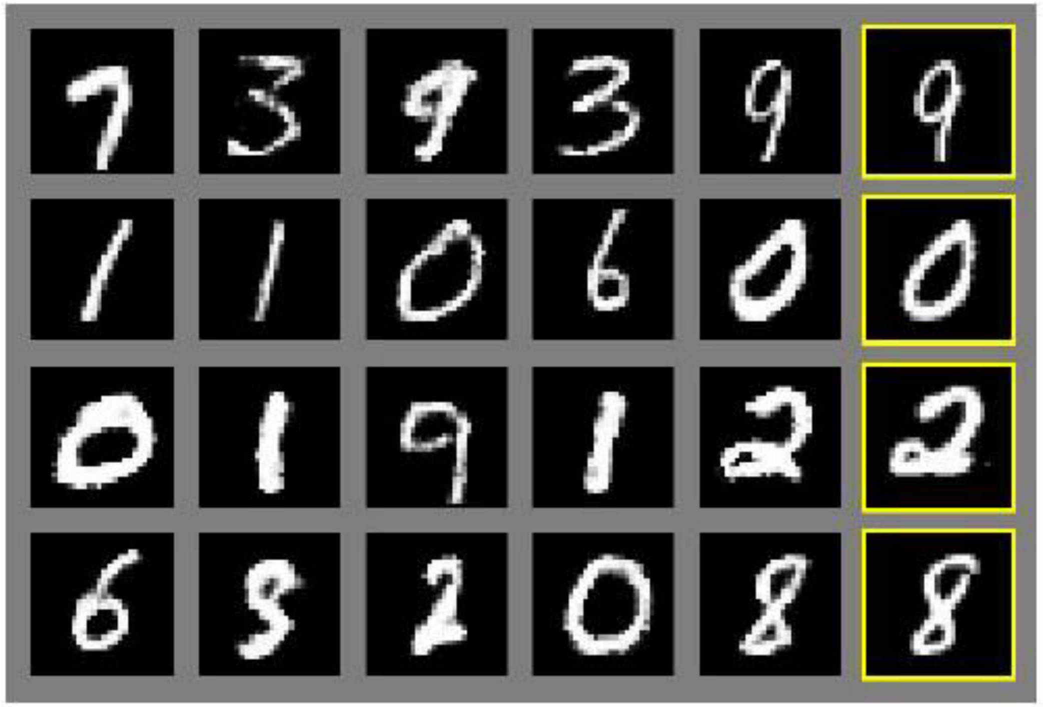
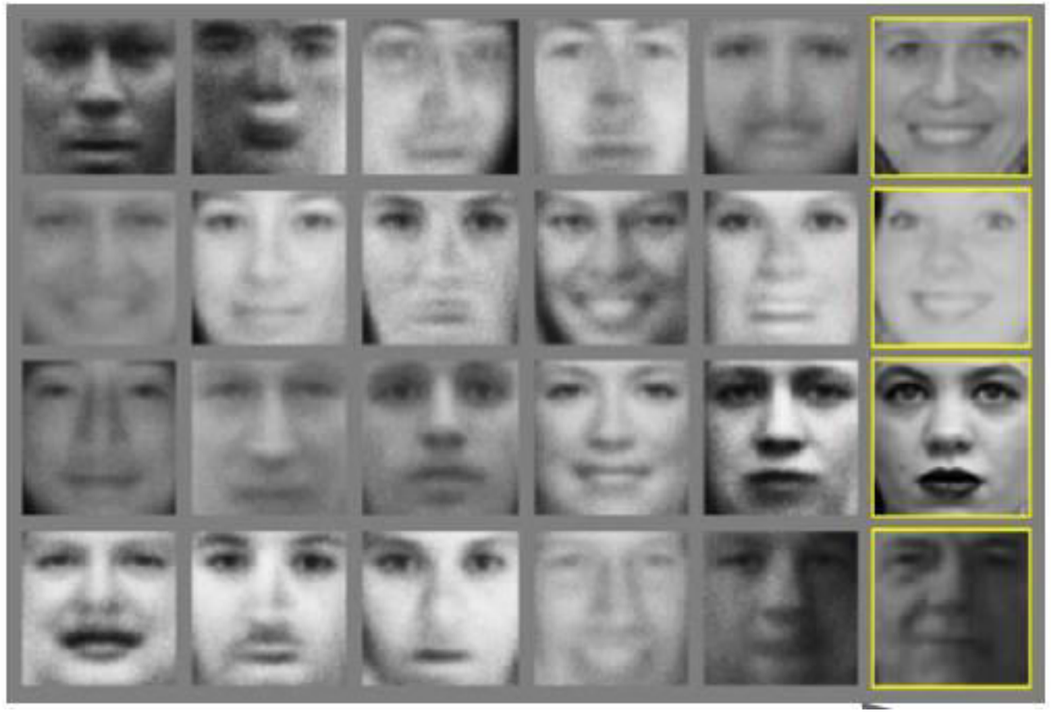
Training GANs

- Two player game
  - Generator: try to fool discriminator by generating real-looking images
  - Discriminator: try to distinguish between real and fake images



Generated samples

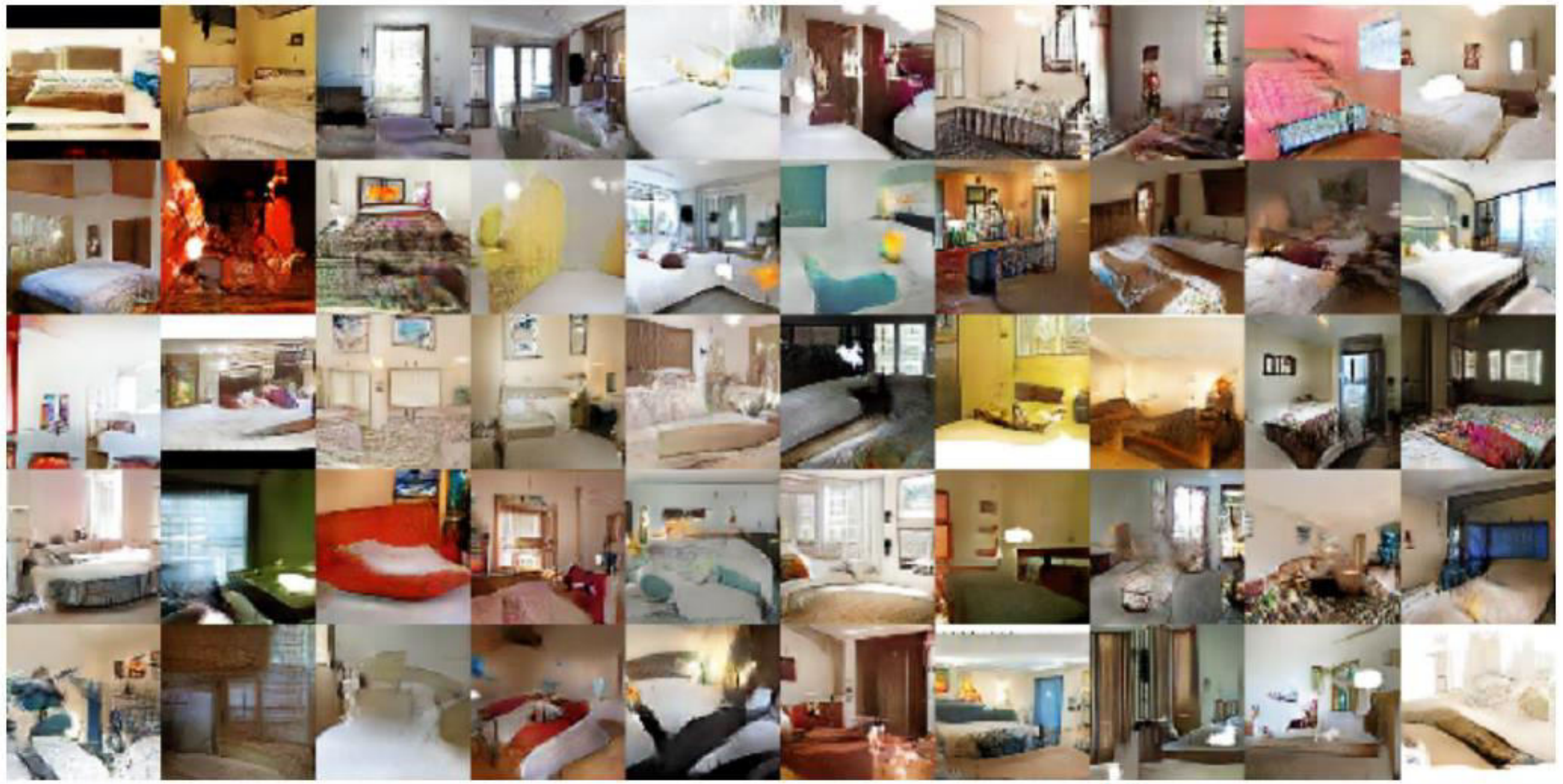
Generated samples

Generated samples

Generated samples (CIFAR-10)

GANs: Convolutional architectures

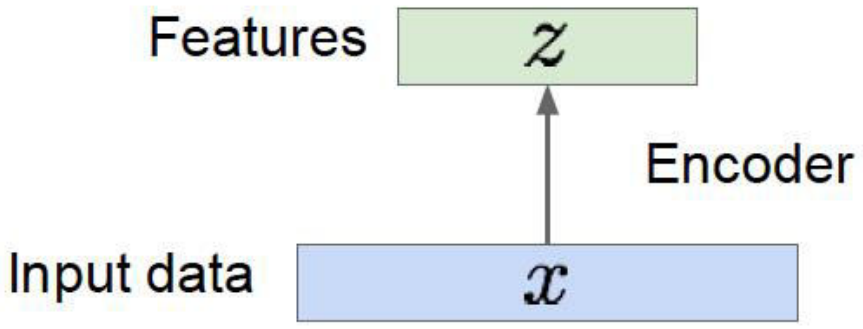


Radford et al, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”, ICLR 2016

Unsupervised pretraining

Autoencoders

Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data



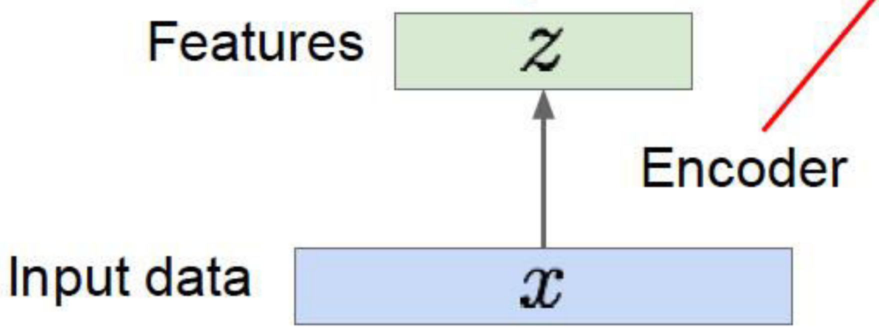
Autoencoders

Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data

z usually smaller than **x** (dimensionality reduction)

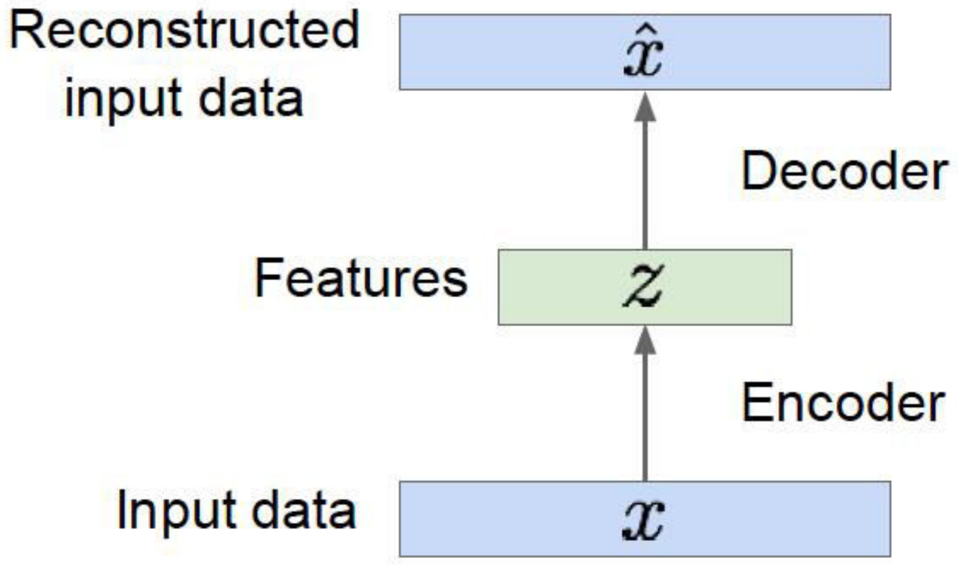
Q: Why dimensionality reduction?

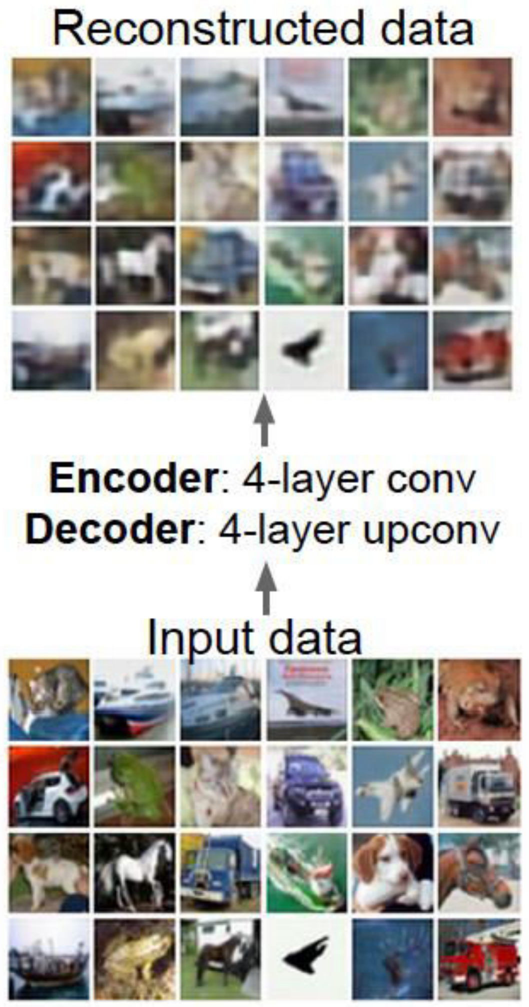
Originally: Linear + nonlinearity (sigmoid)
Later: Deep, fully-connected
Later: ReLU CNN



Autoencoders

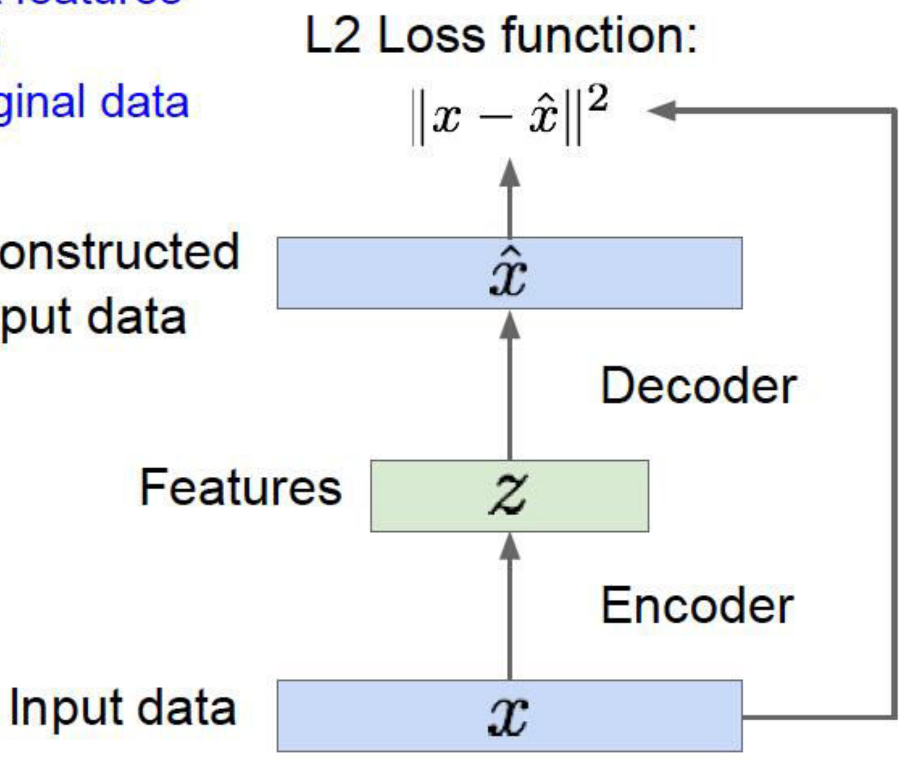
How to learn this feature representation?
Train such that features can be used to reconstruct original data
“Autoencoding” - encoding itself

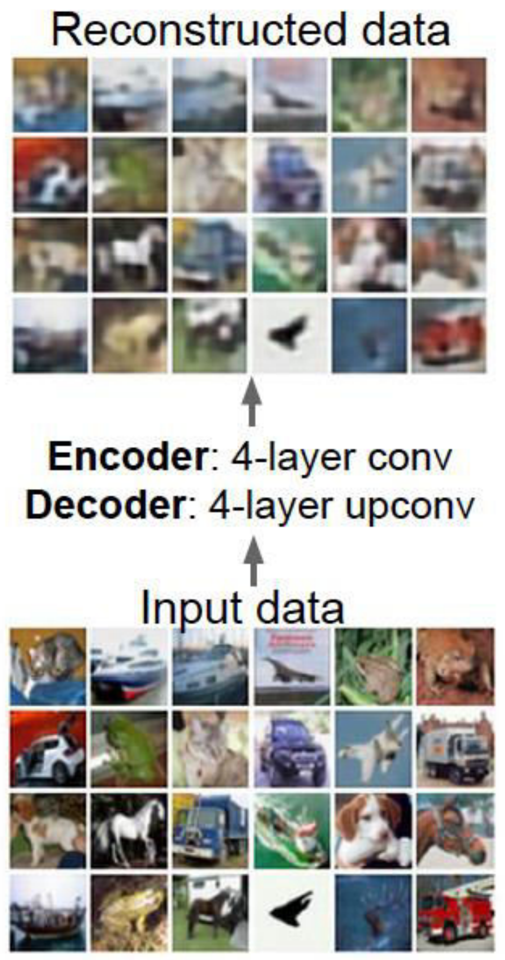




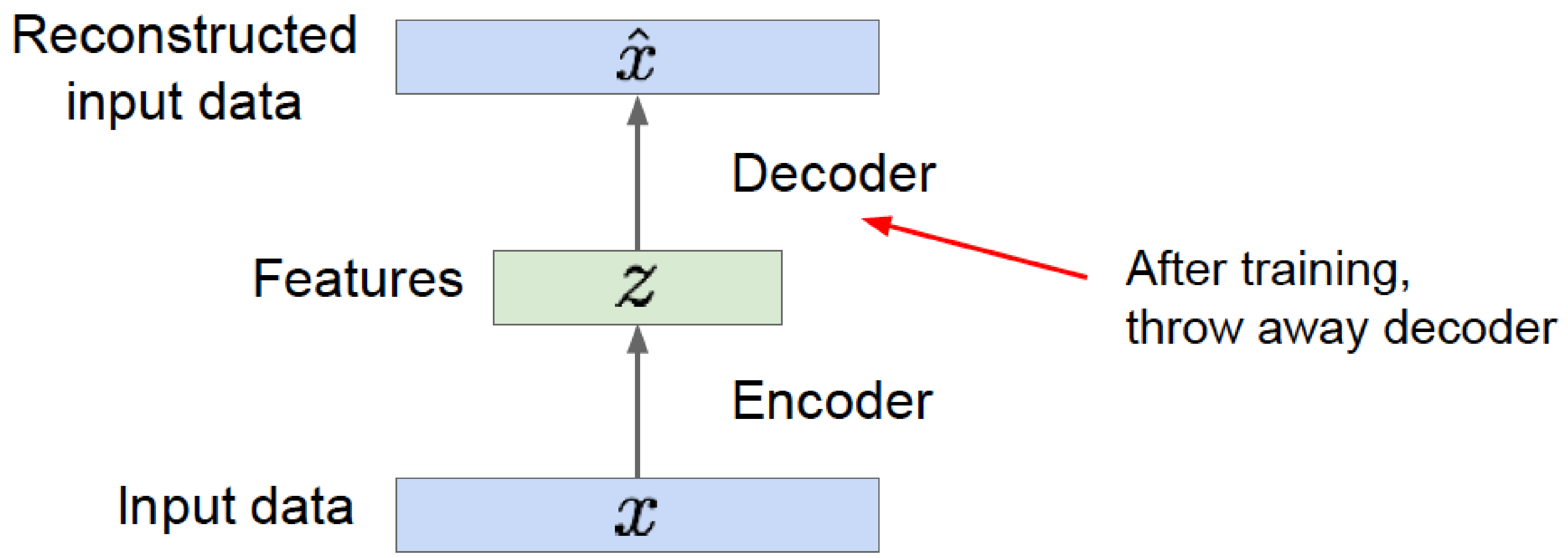
Autoencoders

Train such that features can be used to reconstruct original data

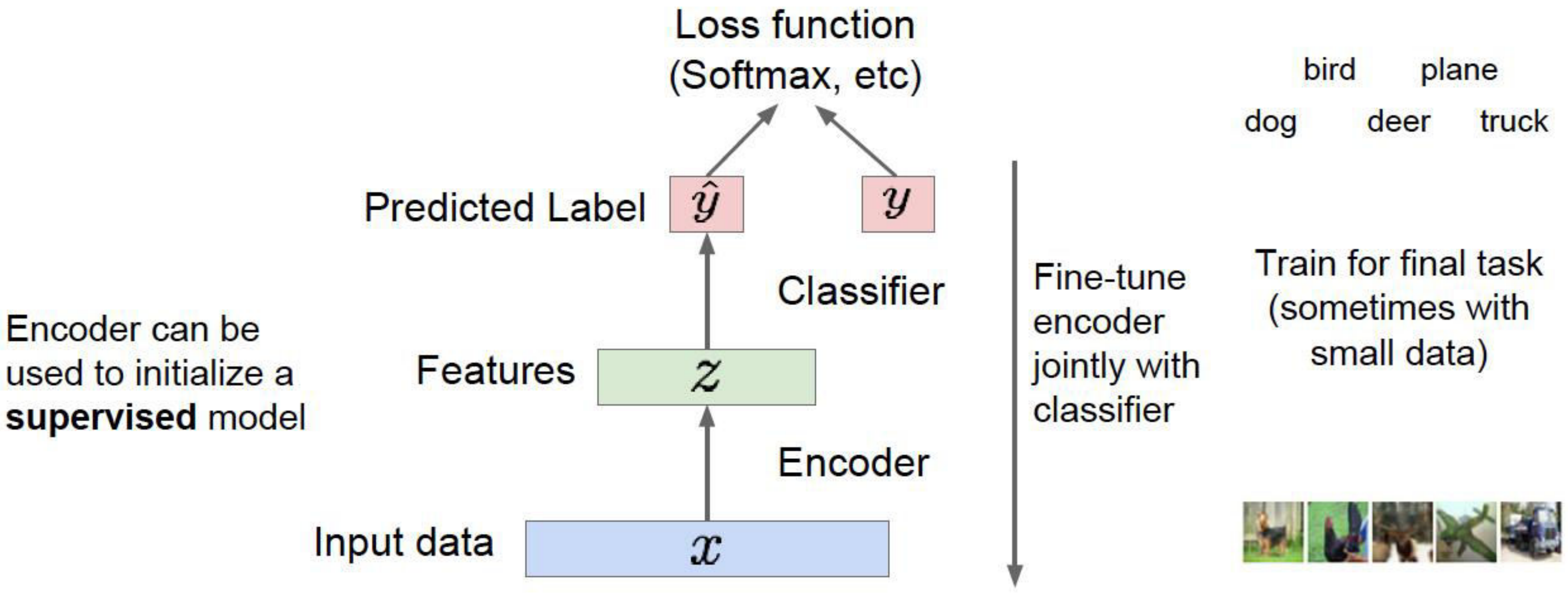




Autoencoders for unsupervised pretraining



Autoencoders for unsupervised pretraining



Loss function
(Softmax, etc)
Predicted Label
$\hat{y}$
y
Classifier
Features
z
Encoder
Input data
x
Encoder can be used to initialize a **supervised** model
Fine-tune encoder jointly with classifier
bird
plane
dog
deer
truck
Train for final task (sometimes with small data)

Stacked auto-encoders: Bengio et al., Greedy Layer-Wise Training of Deep Networks, NIPS 2006

References and acknowledgments

- Fei-Fei Li et al.: Generative models
- OpenAI blog post on Generative Models
- Goodfellow et al.: Generative Adversarial Networks

- Unsupervised pretraining reference
  - Erhan et al., Why Does Unsupervised Pre-training Help Deep Learning?, JMLR 11 (2010) 625-660