

# Introduction to Computer Vision

## Generative models

Dr. Sergio Escalera  
University of Barcelona



UNIVERSITAT DE  
BARCELONA

# Supervised vs Unsupervised Learning

## Supervised Learning

**Data:**  $(x, y)$

$x$  is data,  $y$  is label

**Goal:** Learn a *function* to map  $x \rightarrow y$

**Examples:** Classification,  
regression, object detection,  
semantic segmentation, image  
captioning, etc.



→ Cat

Classification

# Supervised vs Unsupervised Learning

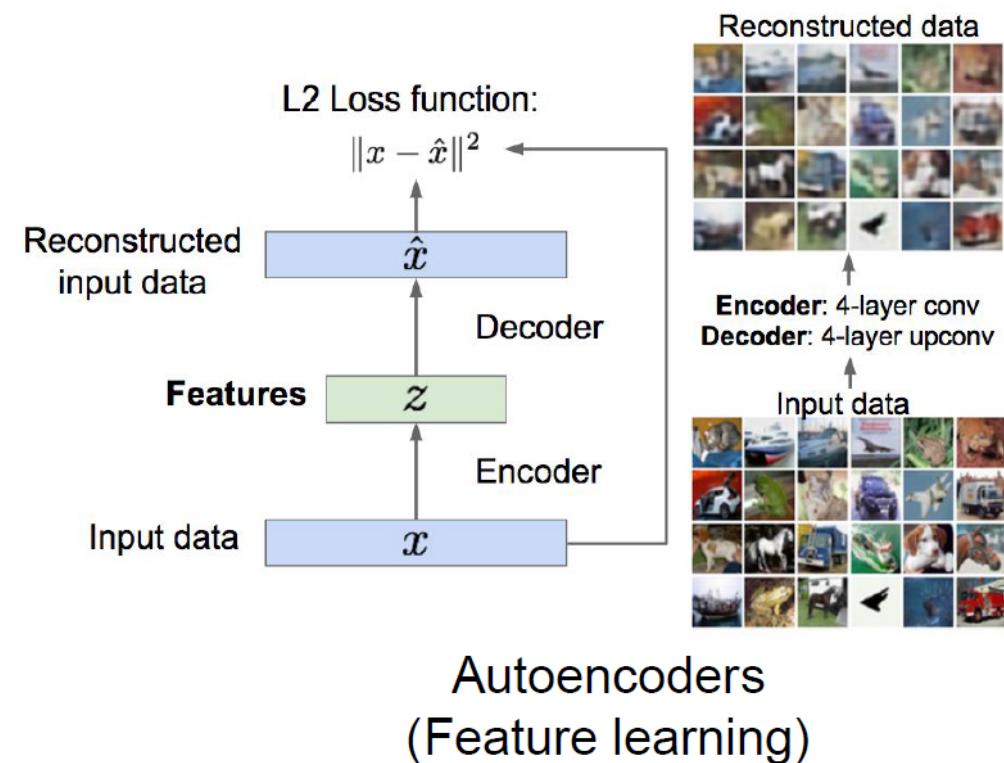
## Unsupervised Learning

**Data:**  $x$

Just data, no labels!

**Goal:** Learn some underlying hidden *structure* of the data

**Examples:** Clustering, dimensionality reduction, feature learning, density estimation, etc.



# Supervised vs Unsupervised Learning

## Supervised Learning

**Data:**  $(x, y)$

$x$  is data,  $y$  is label

**Goal:** Learn a *function* to map  $x \rightarrow y$

**Examples:** Classification, regression, object detection, semantic segmentation, image captioning, etc.

## Unsupervised Learning

Training data is cheap

**Data:**  $x$

Just data, no labels!

Holy grail: Solve unsupervised learning  
=> understand structure of visual world

**Goal:** Learn some underlying hidden *structure* of the data

**Examples:** Clustering, dimensionality reduction, feature learning, density estimation, etc.

# Generative Models

Given training data, generate new samples from same distribution



Training data  $\sim p_{\text{data}}(x)$



Generated samples  $\sim p_{\text{model}}(x)$

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

Addresses density estimation, a core problem in unsupervised learning

## Several flavors:

- Explicit density estimation: explicitly define and solve for  $p_{\text{model}}(x)$
- Implicit density estimation: learn model that can sample from  $p_{\text{model}}(x)$  w/o explicitly defining it

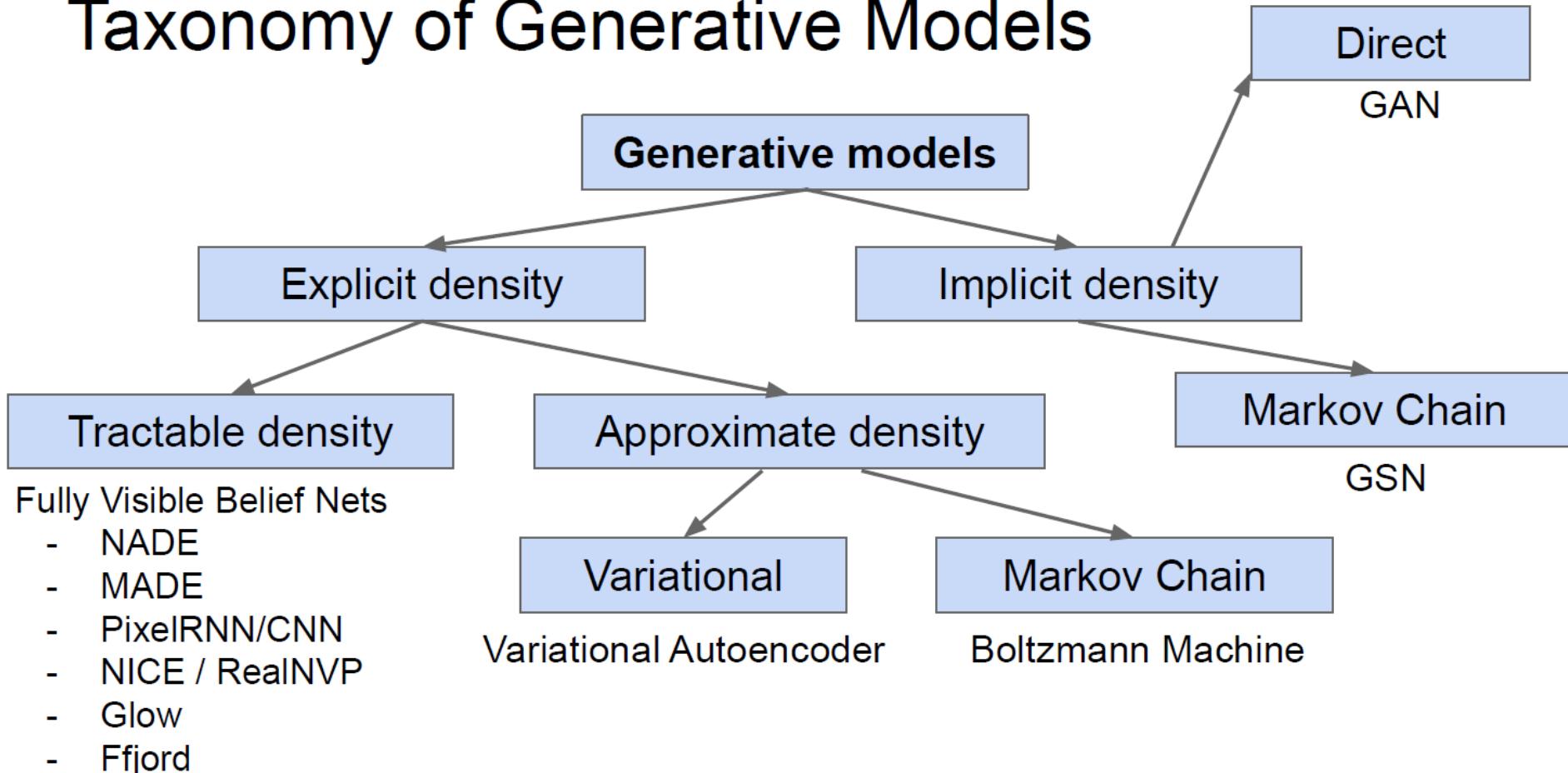
# Why Generative Models?

- Realistic samples for artwork, super-resolution, colorization, etc.

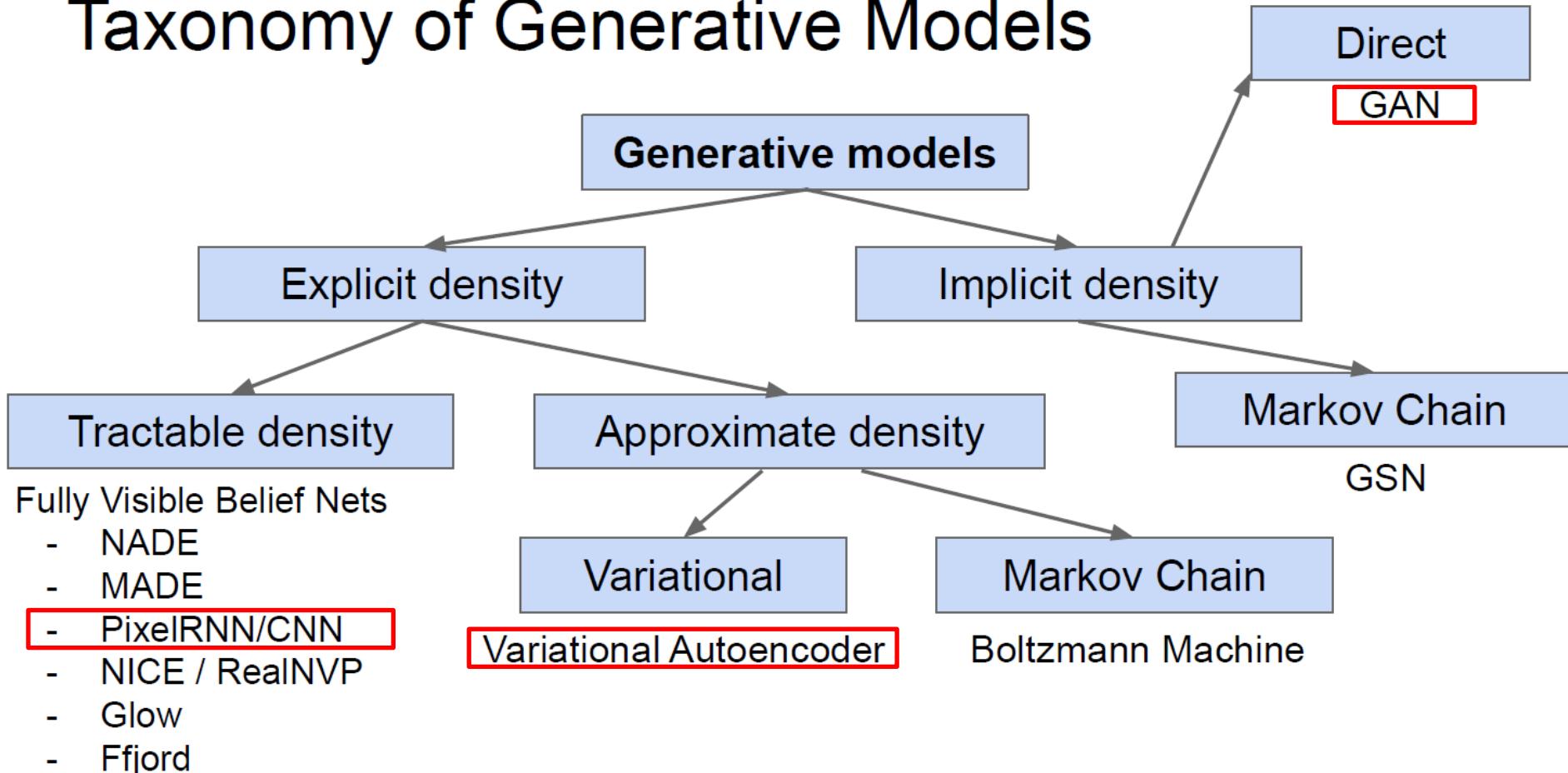


- Generative models of time-series data can be used for simulation and planning (reinforcement learning applications!)
- Training generative models can also enable inference of latent representations that can be useful as general features

# Taxonomy of Generative Models



# Taxonomy of Generative Models



# Fully visible belief network

## Explicit density model

Use chain rule to decompose likelihood of an image  $x$  into product of 1-d distributions:

Then maximize likelihood of training data



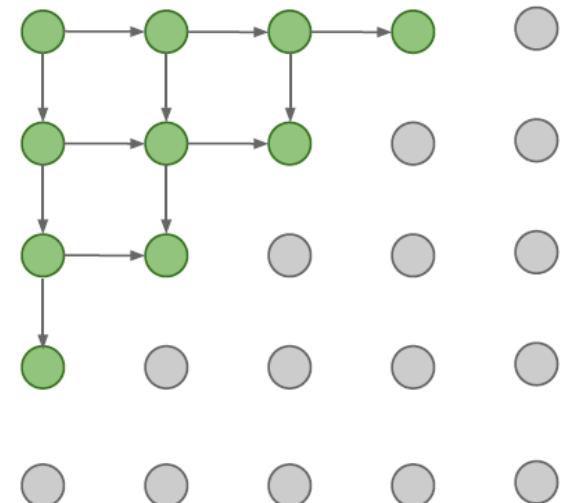
# PixelRNN

*[van der Oord et al. 2016]*

Generate image pixels starting from corner

Dependency on previous pixels modeled using an RNN (LSTM)

Drawback: sequential generation is slow!



# PixelCNN

*[van der Oord et al. 2016]*

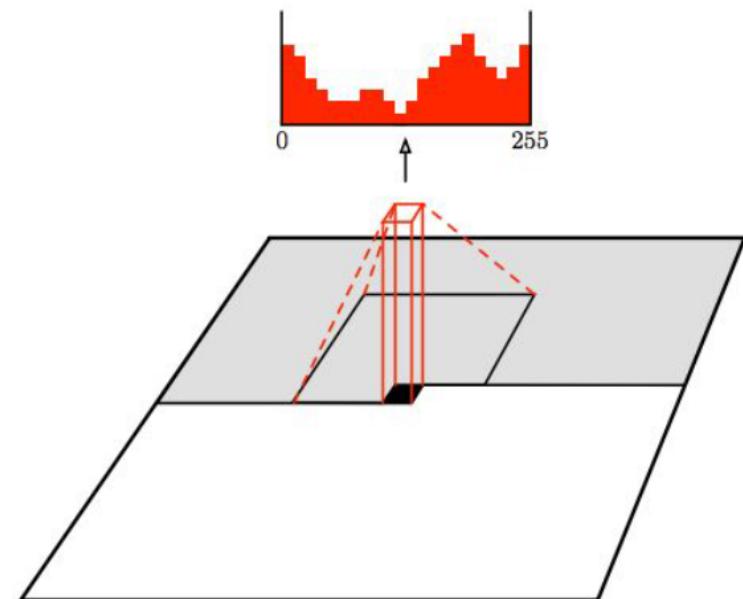
Still generate image pixels starting from corner

Dependency on previous pixels now modeled using a CNN over context region

Training: maximize likelihood of training images

$$p(x) = \prod_{i=1}^n p(x_i | x_1, \dots, x_{i-1})$$

Softmax loss at each pixel



# PixelCNN

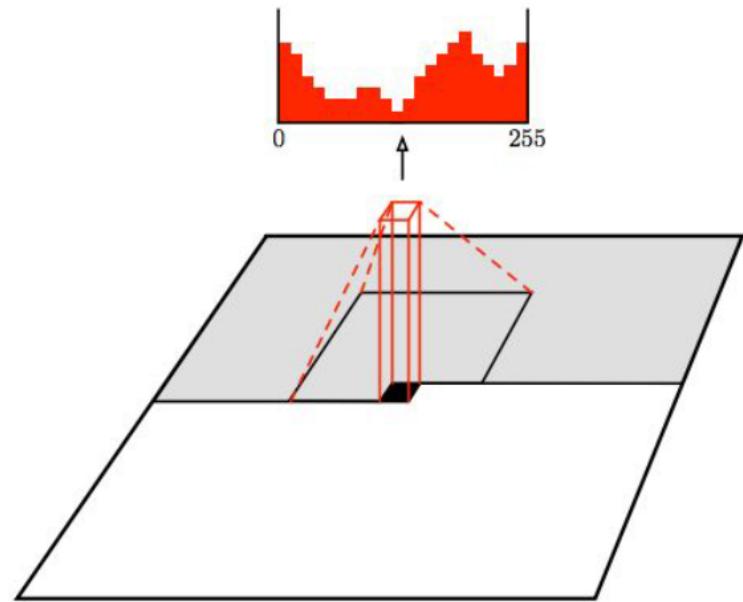
*[van der Oord et al. 2016]*

Still generate image pixels starting from corner

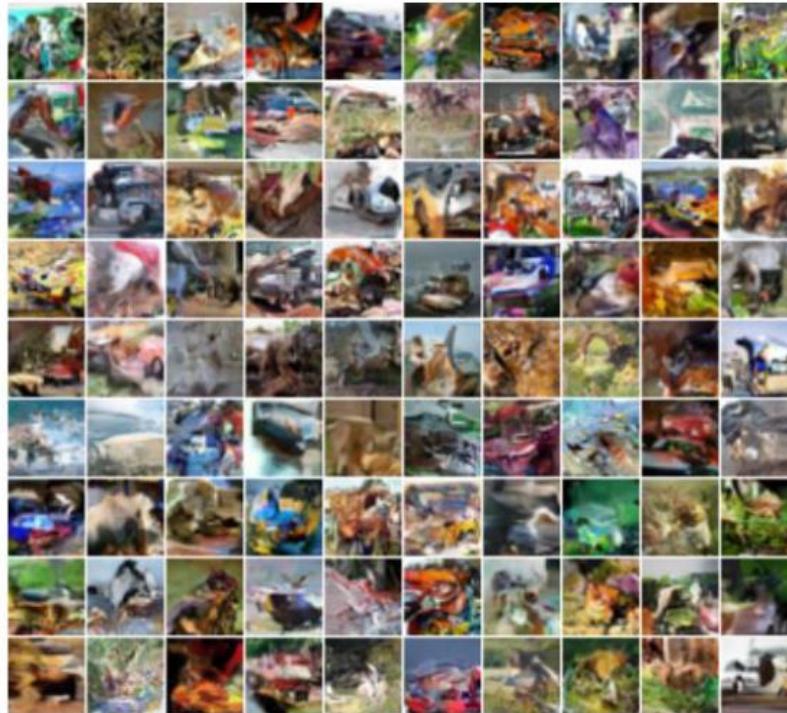
Dependency on previous pixels now modeled using a CNN over context region

Training is faster than PixelRNN  
(can parallelize convolutions since context region values known from training images)

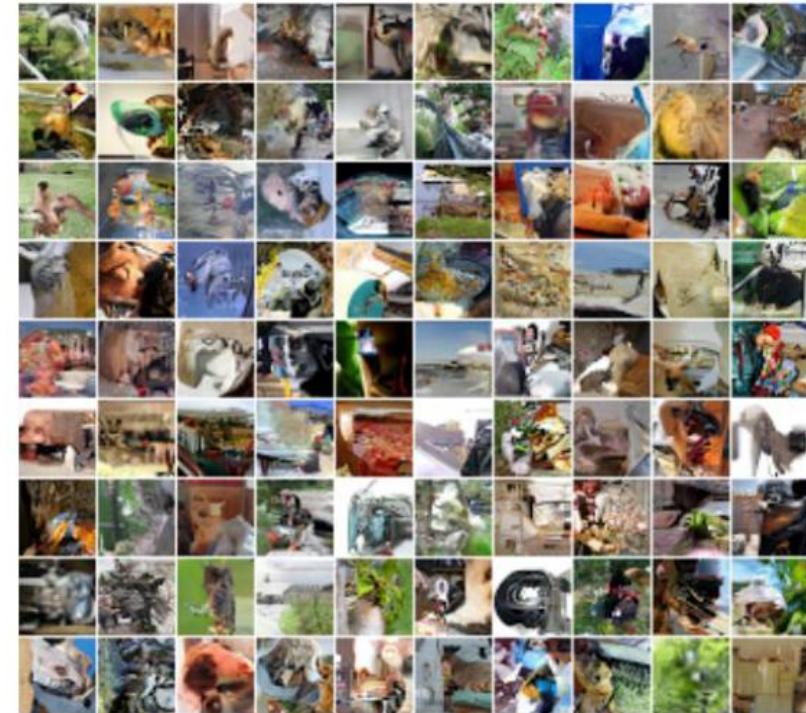
Generation must still proceed sequentially  
=> still slow



# Generation Samples



32x32 CIFAR-10



32x32 ImageNet

# PixelRNN and PixelCNN

## Pros:

- Can explicitly compute likelihood  $p(x)$
- Explicit likelihood of training data gives good evaluation metric
- Good samples

## Con:

- Sequential generation => slow

## Improving PixelCNN performance

- Gated convolutional layers
- Short-cut connections
- Discretized logistic loss
- Multi-scale
- Training tricks
- Etc...

## See

- Van der Oord et al. NIPS 2016
- Salimans et al. 2017  
(PixelCNN++)

# Variational Autoencoders (VAE)

# So far...

PixelCNNs define tractable density function, optimize likelihood of training data:

$$p_{\theta}(x) = \prod_{i=1}^n p_{\theta}(x_i|x_1, \dots, x_{i-1})$$

VAEs define intractable density function with latent  $\mathbf{z}$ :

$$p_{\theta}(x) = \int p_{\theta}(z)p_{\theta}(x|z)dz$$

Cannot optimize directly, derive and optimize lower bound on likelihood instead

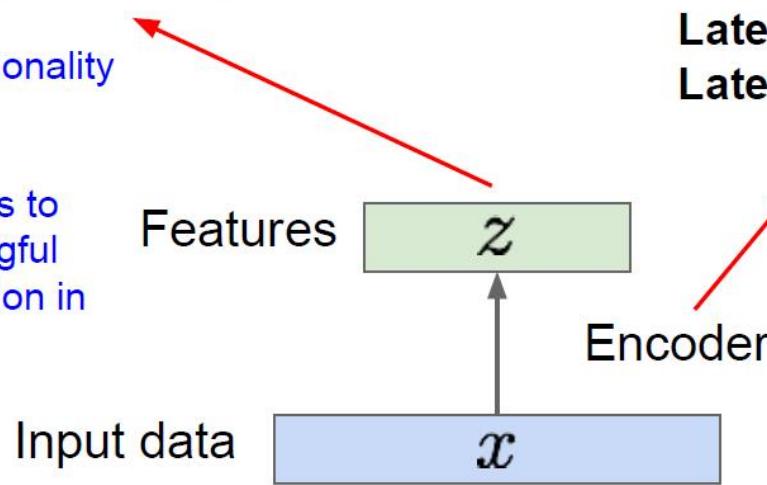
# Some background first: Autoencoders

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

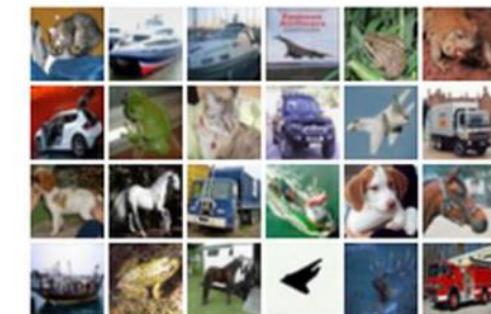
**z** usually smaller than **x**  
(dimensionality reduction)

Q: Why dimensionality reduction?

A: Want features to capture meaningful factors of variation in data



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

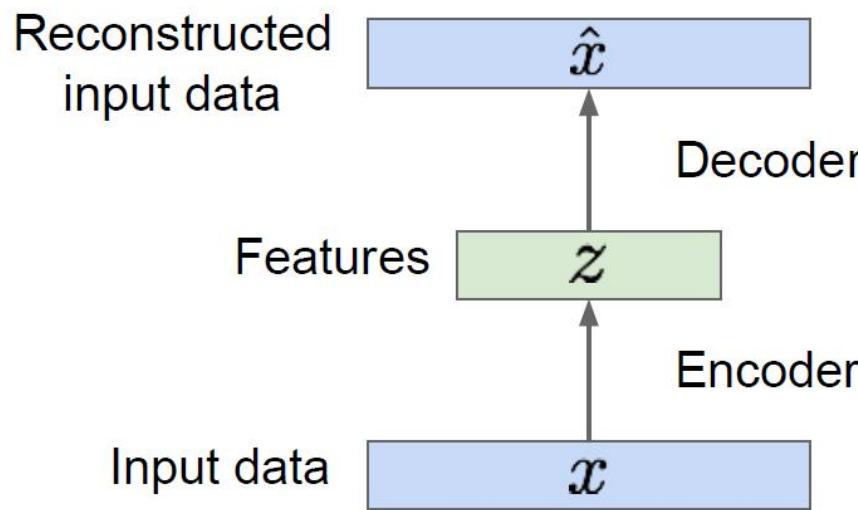


# Some background first: Autoencoders

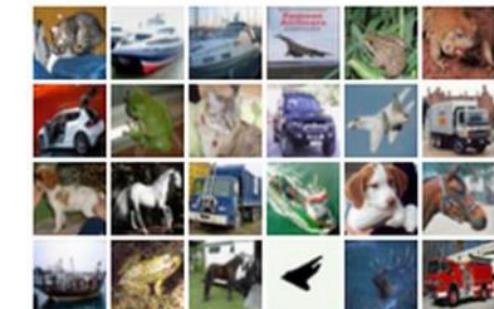
How to learn this feature representation?

Train such that features can be used to reconstruct original data

“Autoencoding” - encoding itself



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



# Some background first: Autoencoders

Train such that features  
can be used to  
reconstruct original data

Reconstructed  
input data

Input data

Features

$x$

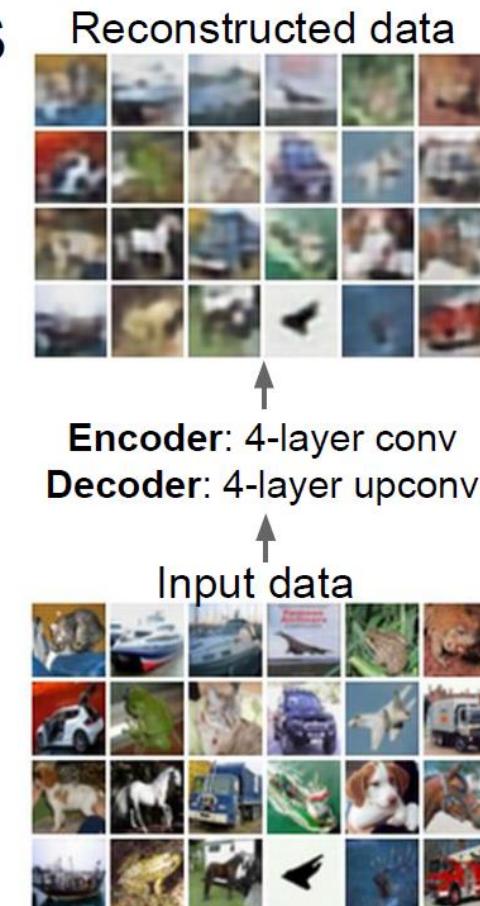
Decoder

$\hat{x}$

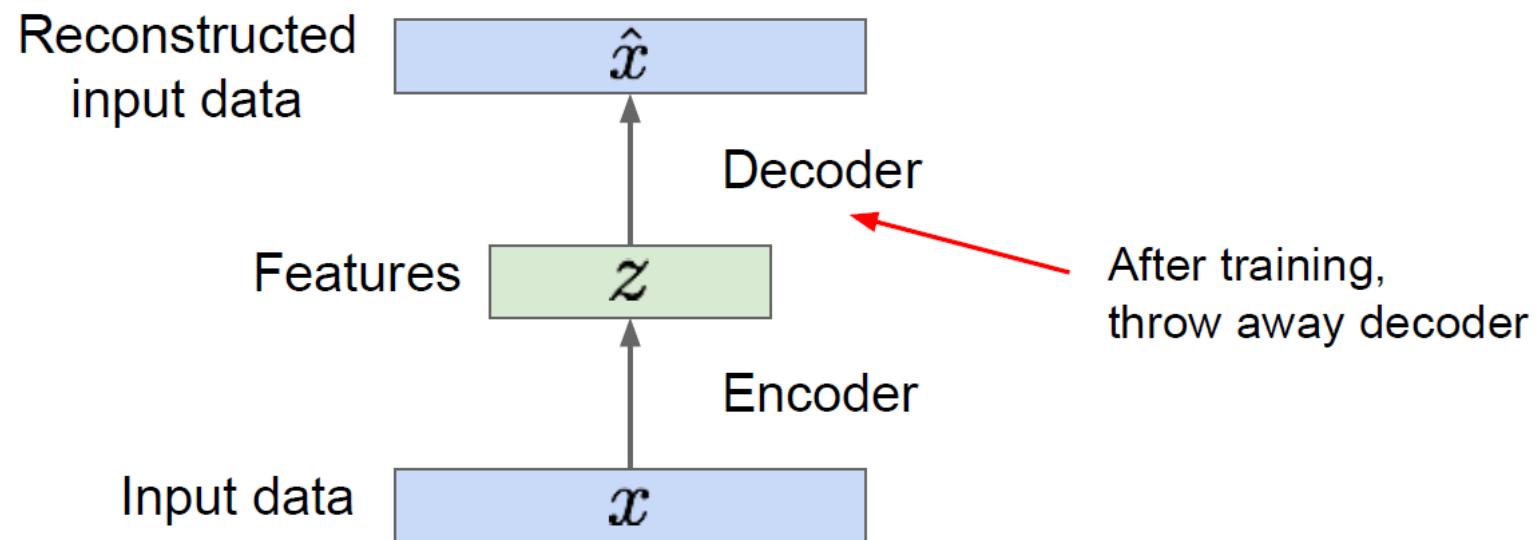
$\|x - \hat{x}\|^2$

Doesn't use labels!

L2 Loss function:

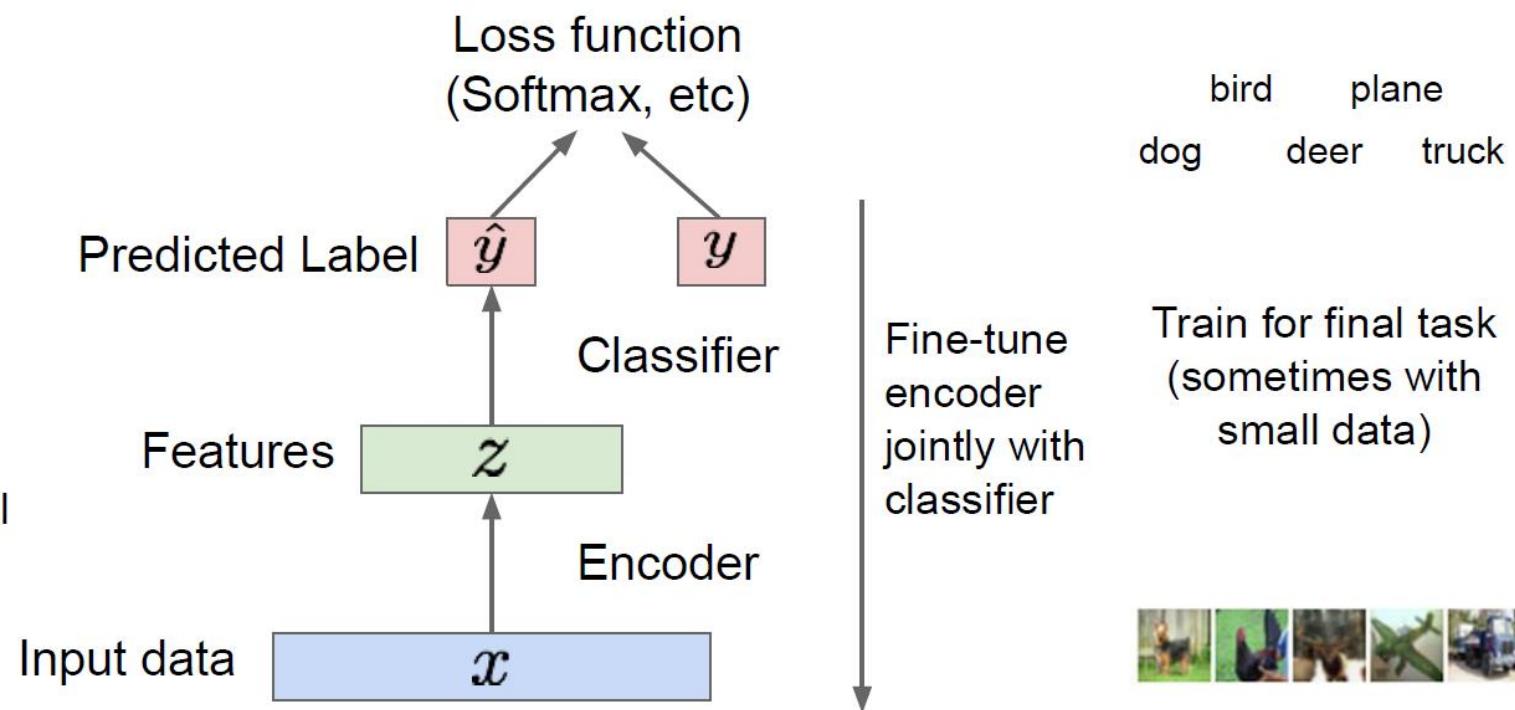


# Some background first: Autoencoders

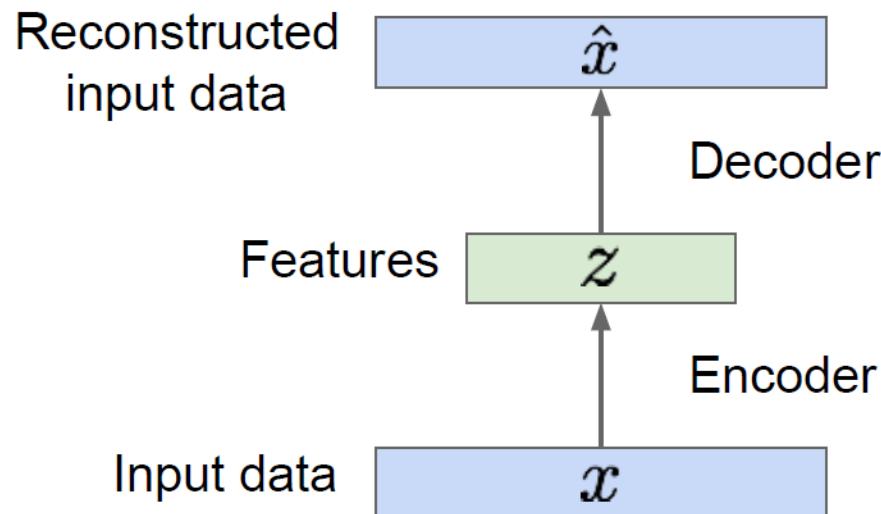


# Some background first: Autoencoders

Encoder can be used to initialize a **supervised** model



# Some background first: Autoencoders



Autoencoders can reconstruct data, and can learn features to initialize a supervised model

Features capture factors of variation in training data. Can we generate new images from an autoencoder?

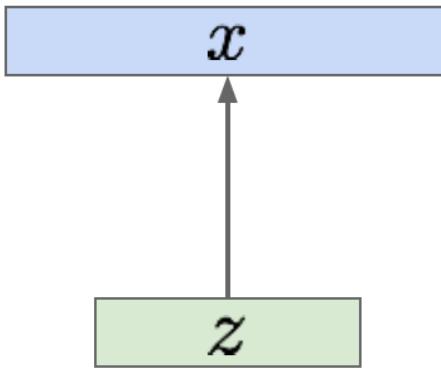


# Variational Autoencoders

Probabilistic spin on autoencoders - will let us sample from the model to generate data!

Assume training data  $\{x^{(i)}\}_{i=1}^N$  is generated from underlying unobserved (latent) representation  $z$

Sample from  
true conditional  
 $p_{\theta^*}(x \mid z^{(i)})$



Sample from  
true prior  
 $p_{\theta^*}(z)$

**Intuition** (remember from autoencoders!):  
 $x$  is an image,  $z$  is latent factors used to  
generate  $x$ : attributes, orientation, etc.

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

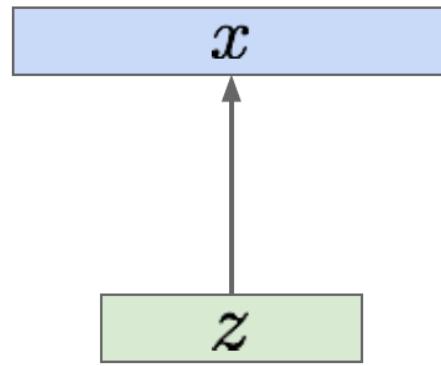
# Variational Autoencoders

Sample from  
true conditional

$$p_{\theta^*}(x \mid z^{(i)})$$

Sample from  
true prior

$$p_{\theta^*}(z)$$



We want to estimate the true parameters  $\theta^*$  of this generative model.

How should we represent this model?

Choose prior  $p(z)$  to be simple, e.g. Gaussian. Reasonable for latent attributes, e.g. pose, how much smile.

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

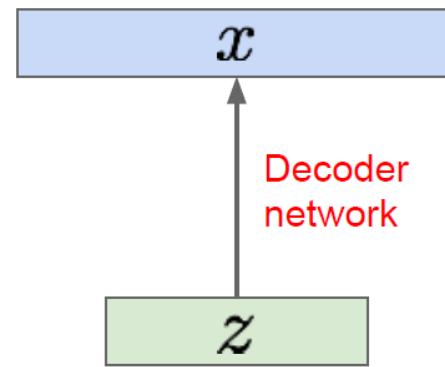
# Variational Autoencoders

Sample from  
true conditional

$$p_{\theta^*}(x \mid z^{(i)})$$

Sample from  
true prior

$$p_{\theta^*}(z)$$



We want to estimate the true parameters  $\theta^*$  of this generative model.

How should we represent this model?

Choose prior  $p(z)$  to be simple, e.g. Gaussian.

Conditional  $p(x|z)$  is complex (generates image) => represent with neural network

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

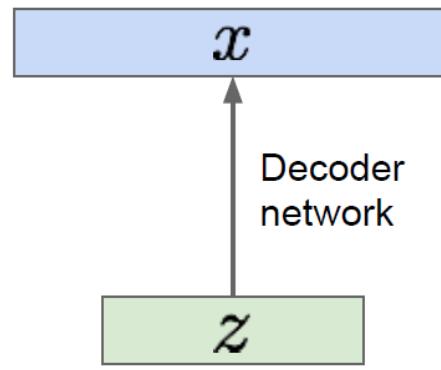
# Variational Autoencoders

Sample from  
true conditional

$$p_{\theta^*}(x | z^{(i)})$$

Sample from  
true prior

$$p_{\theta^*}(z)$$



We want to estimate the true parameters  $\theta^*$  of this generative model.

How to train the model?

Learn model parameters  
to maximize likelihood of training data

$$p_{\theta}(x) = \int p_{\theta}(z)p_{\theta}(x|z)dz$$

Now with latent z

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

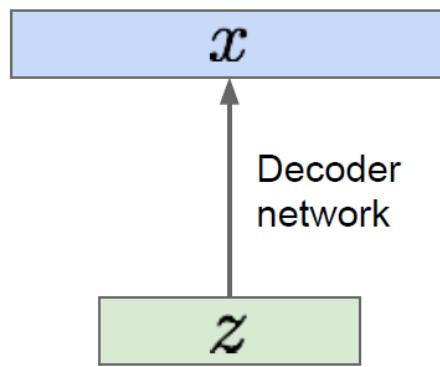
# Variational Autoencoders

Sample from  
true conditional

$$p_{\theta^*}(x \mid z^{(i)})$$

Sample from  
true prior

$$p_{\theta^*}(z)$$



We want to estimate the true parameters  $\theta^*$  of this generative model.

How to train the model?

Learn model parameters  
to maximize likelihood of training data

$$p_{\theta}(x) = \int p_{\theta}(z)p_{\theta}(x|z)dz$$

Q: What is the problem with this?

Intractable!

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

# Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

$$\underbrace{\mathbb{E}_z \left[ \log p_\theta(x^{(i)} | z) \right] - D_{KL}(q_\phi(z | x^{(i)}) || p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)}$$

Make approximate posterior distribution close to prior

For every minibatch of input data: compute this forward pass, and then backprop!

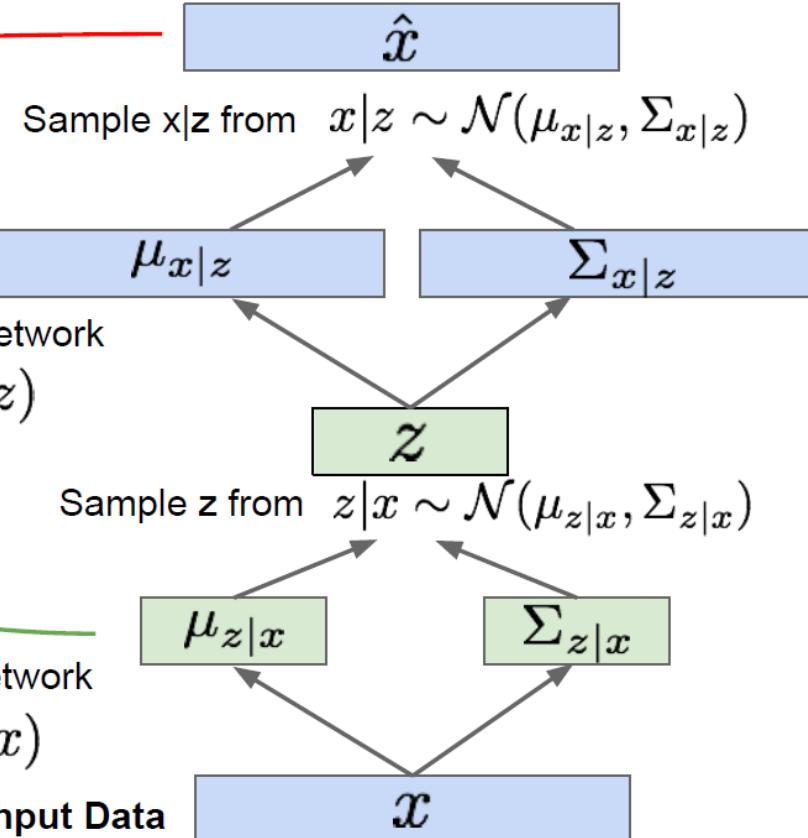
Maximize likelihood of original input being reconstructed

Decoder network  
 $p_\theta(x|z)$

Encoder network

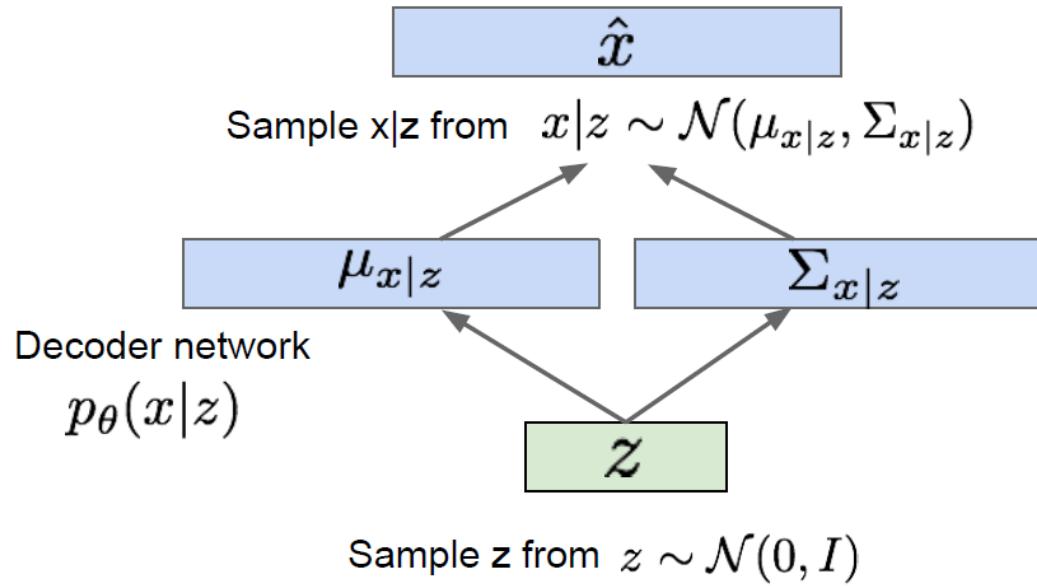
$q_\phi(z|x)$

Input Data  
 $x$

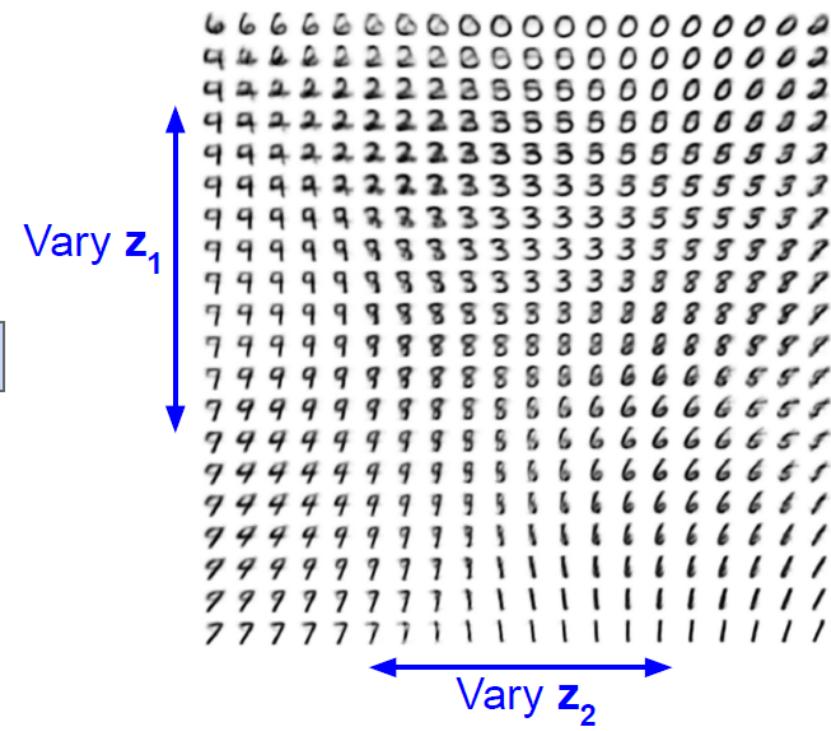


# Variational Autoencoders: Generating Data!

Use decoder network. Now sample  $z$  from prior!



Data manifold for 2-d  $z$



Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

# Variational Autoencoders: Generating Data!

Diagonal prior on  $z$   
=> independent  
latent variables

Different  
dimensions of  $z$   
encode  
interpretable factors  
of variation

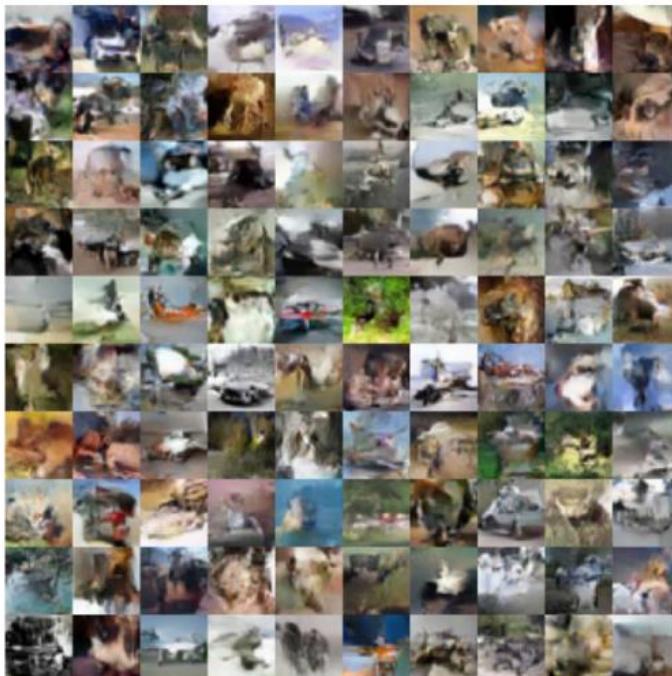
Also good feature representation that  
can be computed using  $q_\phi(z|x)$ !

Degree of smile  
Vary  $z_1$

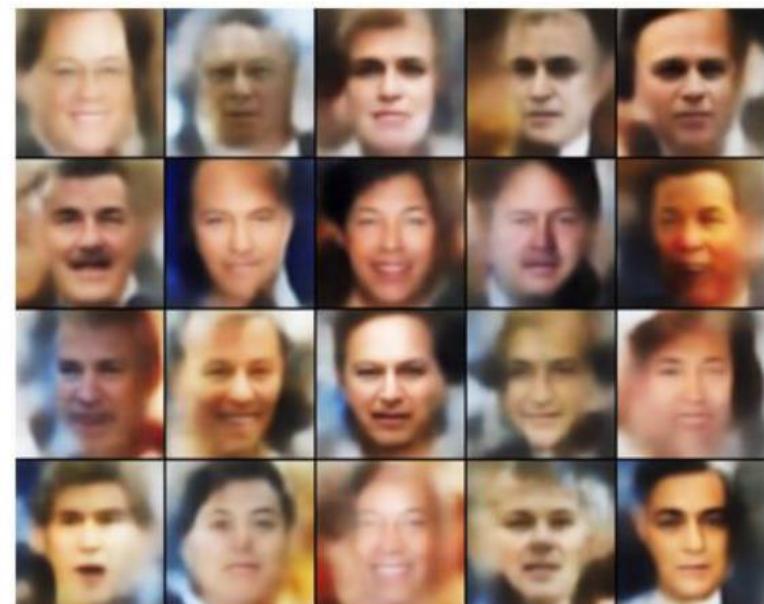


Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

# Variational Autoencoders: Generating Data!



32x32 CIFAR-10



Labeled Faces in the Wild

# Variational Autoencoders

Probabilistic spin to traditional autoencoders => allows generating data

Defines an intractable density => derive and optimize a (variational) lower bound

## Pros:

- Principled approach to generative models
- Allows inference of  $q(z|x)$ , can be useful feature representation for other tasks

## Cons:

- Maximizes lower bound of likelihood: okay, but not as good evaluation as PixelRNN/PixelCNN
- Samples blurrier and lower quality compared to state-of-the-art (GANs)

# Generative Adversarial Networks (GAN)

# So far...

PixelCNNs define tractable density function, optimize likelihood of training data:

$$p_{\theta}(x) = \prod_{i=1}^n p_{\theta}(x_i|x_1, \dots, x_{i-1})$$

VAEs define intractable density function with latent  $\mathbf{z}$ :

$$p_{\theta}(x) = \int p_{\theta}(z)p_{\theta}(x|z)dz$$

Cannot optimize directly, derive and optimize lower bound on likelihood instead

What if we give up on explicitly modeling density, and just want ability to sample?

GANs: don't work with any explicit density function!

Instead, take game-theoretic approach: learn to generate from training distribution through 2-player game

# Generative Adversarial Networks

Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

Problem: Want to sample from complex, high-dimensional training distribution. No direct way to do this!

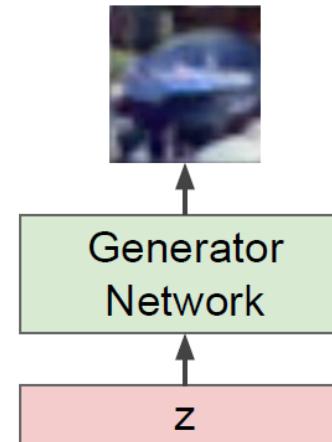
Solution: Sample from a simple distribution, e.g. random noise. Learn transformation to training distribution.

Q: What can we use to represent this complex transformation?

A: A neural network!

Output: Sample from training distribution

Input: Random noise

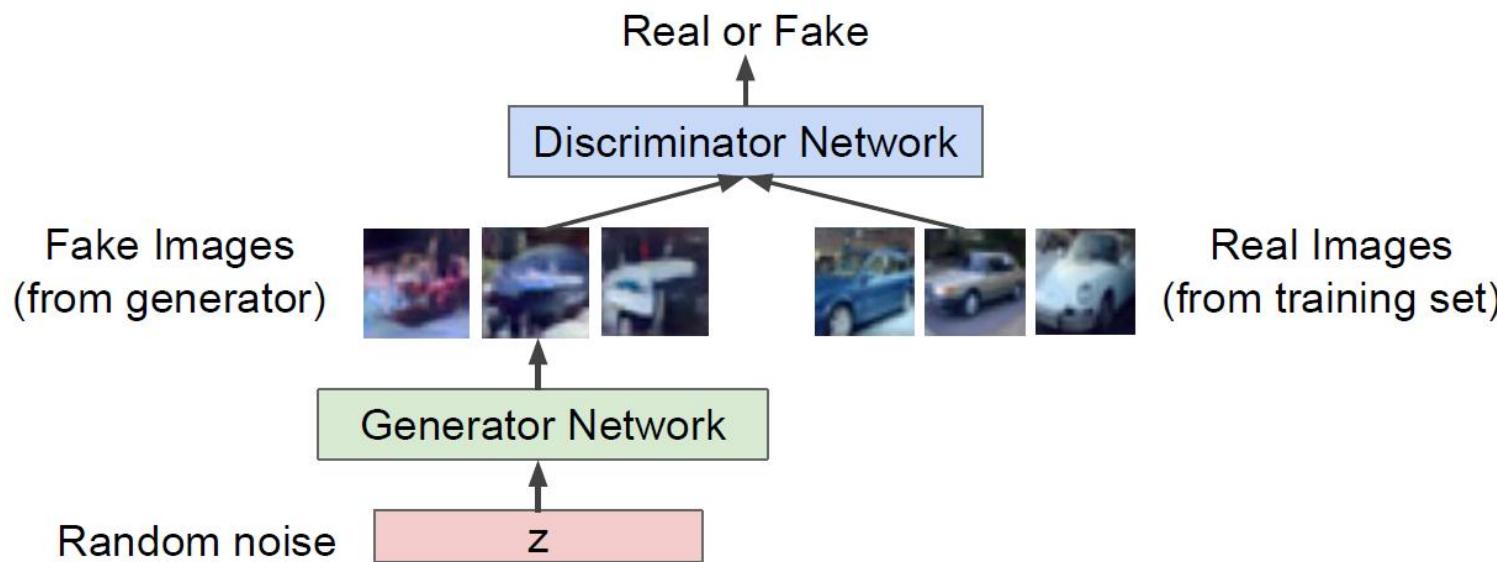


# Training GANs: Two-player game

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

**Generator network:** try to fool the discriminator by generating real-looking images

**Discriminator network:** try to distinguish between real and fake images



Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

# Training GANs: Two-player game

**Generator network:** try to fool the discriminator by generating real-looking images

**Discriminator network:** 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:

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

- Discriminator ( $\theta_d$ ) wants to **maximize objective** such that  $D(x)$  is close to 1 (real) and  $D(G(z))$  is close to 0 (fake)
- Generator ( $\theta_g$ ) 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: Two-player game

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

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

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

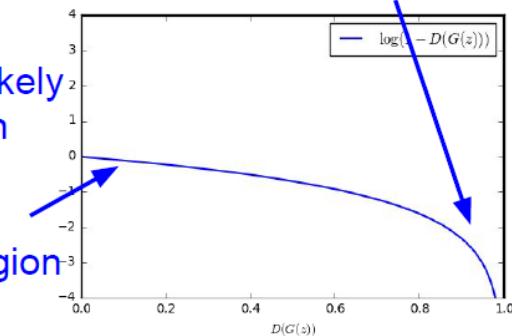
Gradient signal dominated by region where sample is already good

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!



# Training GANs: Two-player game

Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

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

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

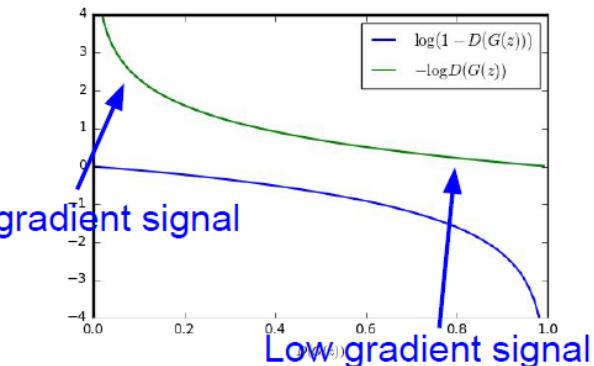
2. **Instead: Gradient ascent** on generator, different objective

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

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.

Aside: Jointly training two networks is challenging, can be unstable. Choosing objectives with better loss landscapes helps training, is an active area of research.



# Training GANs: Two-player game

Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

Putting it together: GAN training algorithm

```
for number of training iterations do
  for k steps do
    • Sample minibatch of  $m$  noise samples  $\{z^{(1)}, \dots, z^{(m)}\}$  from noise prior  $p_g(z)$ .
    • Sample minibatch of  $m$  examples  $\{x^{(1)}, \dots, x^{(m)}\}$  from data generating distribution  $p_{\text{data}}(x)$ .
    • 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  $\{z^{(1)}, \dots, z^{(m)}\}$  from noise prior  $p_g(z)$ .
  • 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
```

Some find  $k=1$  more stable, others use  $k > 1$ , no best rule.

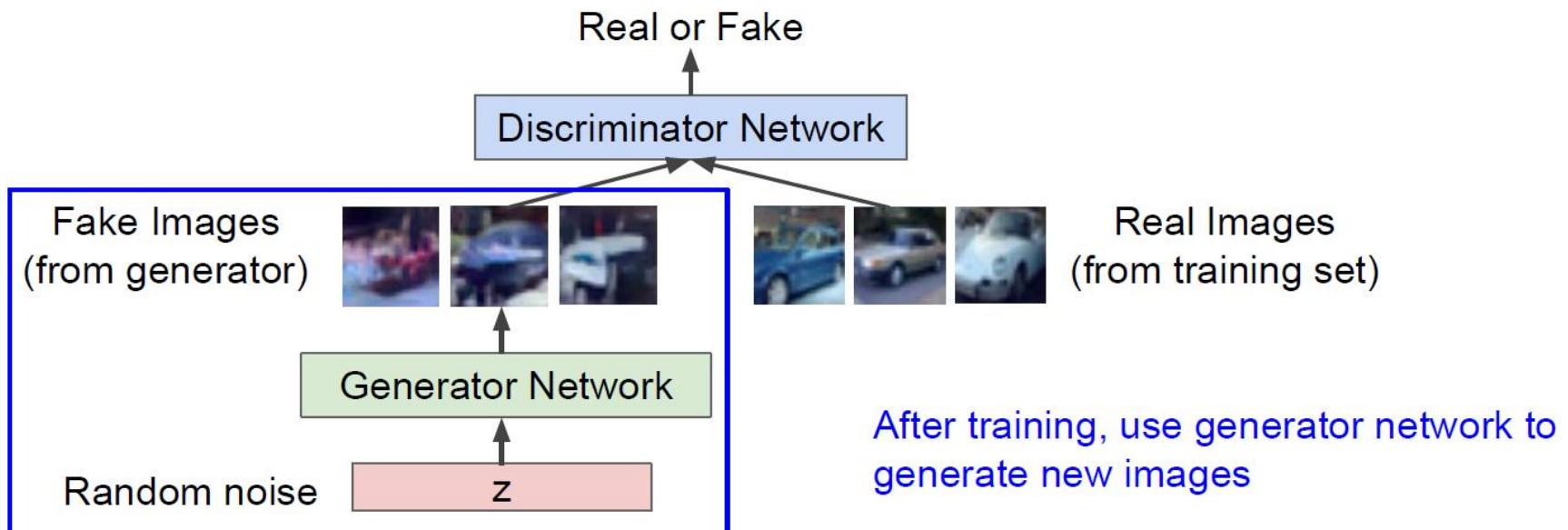
Recent work (e.g. Wasserstein GAN) alleviates this problem, better stability!

# Training GANs: Two-player game

Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

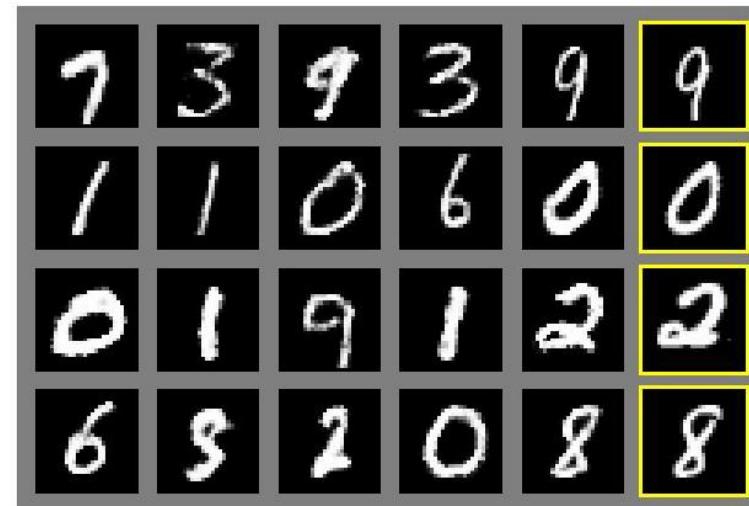
**Generator network**: try to fool the discriminator by generating real-looking images

**Discriminator network**: try to distinguish between real and fake images



# Generative Adversarial Nets

Generated samples



Nearest neighbor from training set

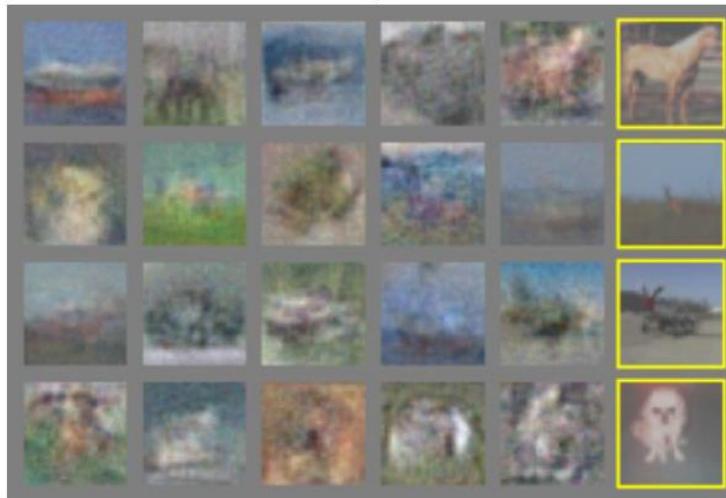
Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014



# Generative Adversarial Nets

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

Generated samples (CIFAR-10)



Nearest neighbor from training set

# Generative Adversarial Nets: Convolutional Architectures

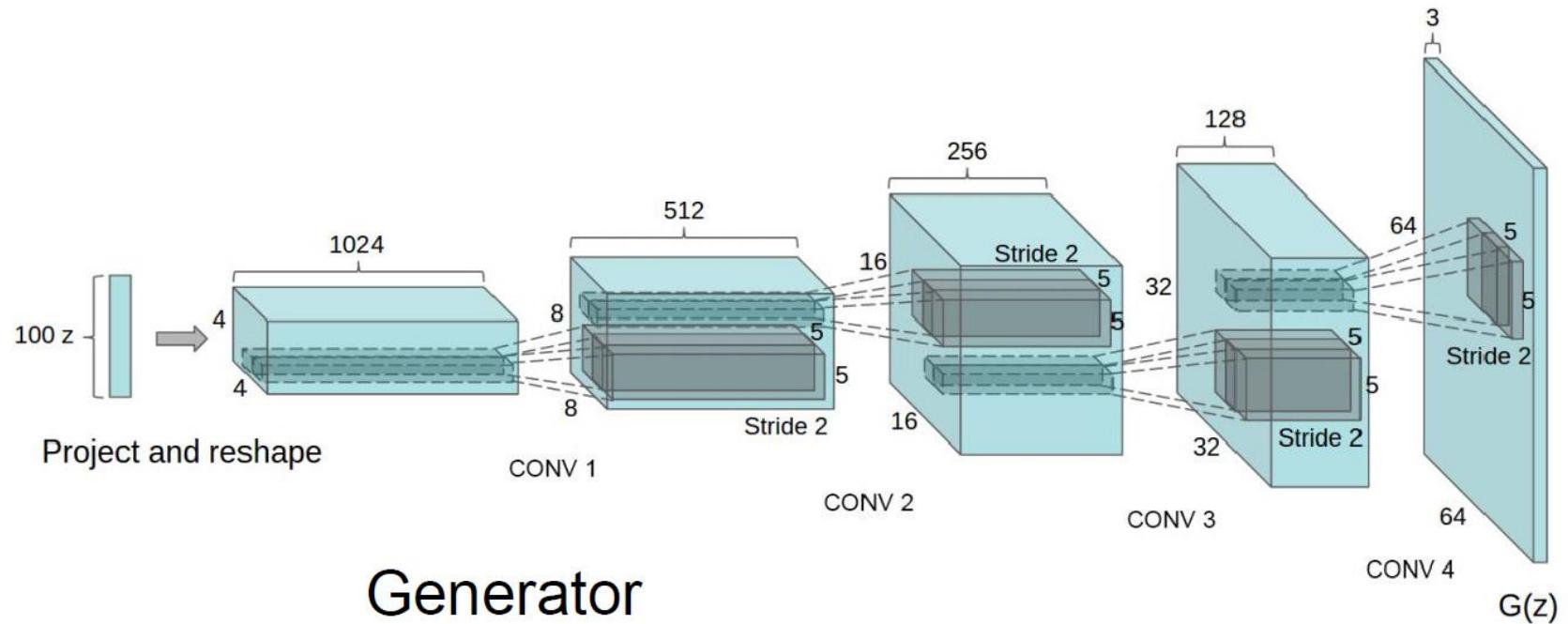
Generator is an upsampling network with fractionally-strided convolutions  
Discriminator is a convolutional network

Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

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

# Generative Adversarial Nets: Convolutional Architectures



## Generator

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

# Generative Adversarial Nets: Convolutional Architectures

Samples from the model look much better!



Radford et al,  
ICLR 2016

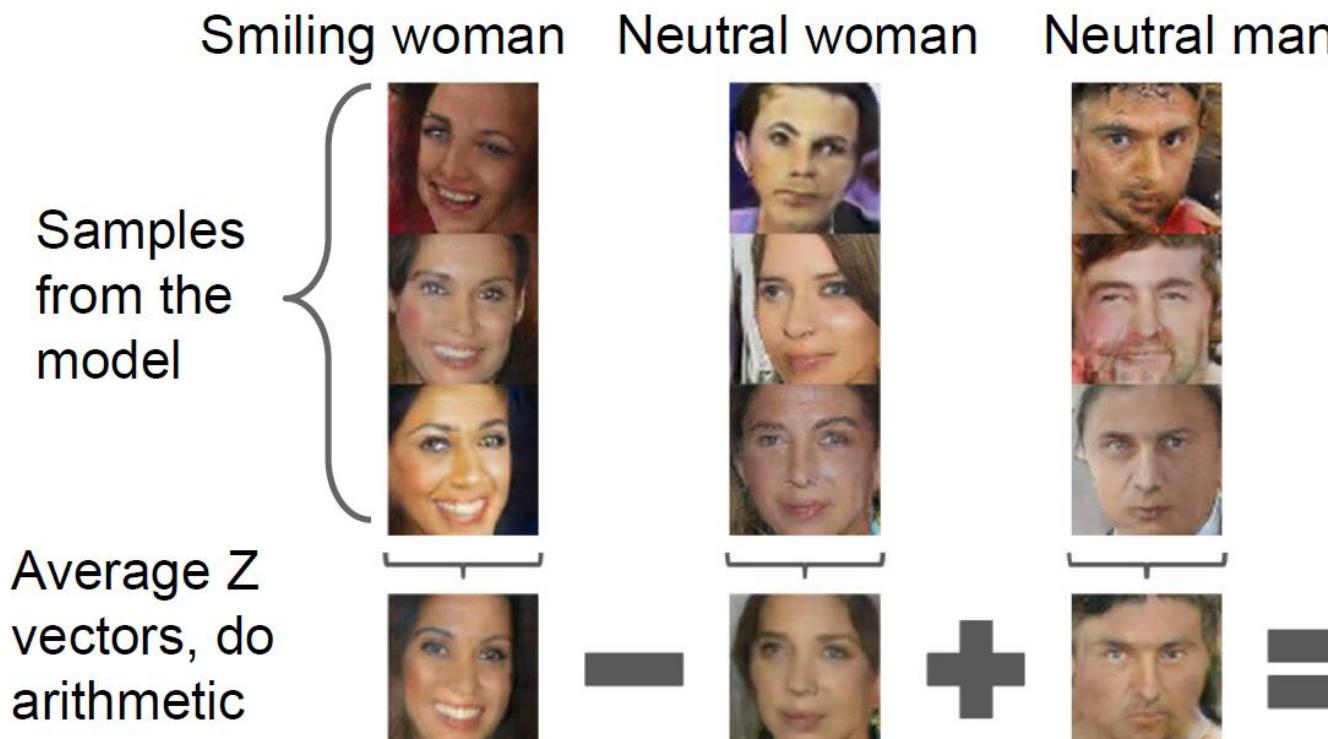
# Generative Adversarial Nets: Convolutional Architectures

Interpolating  
between  
random  
points in latent  
space



Radford et al,  
ICLR 2016

# Generative Adversarial Nets: Interpretable Vector Math



Radford et al, ICLR 2016

Smiling Man



# Generative Adversarial Nets: Interpretable Vector Math

Glasses man



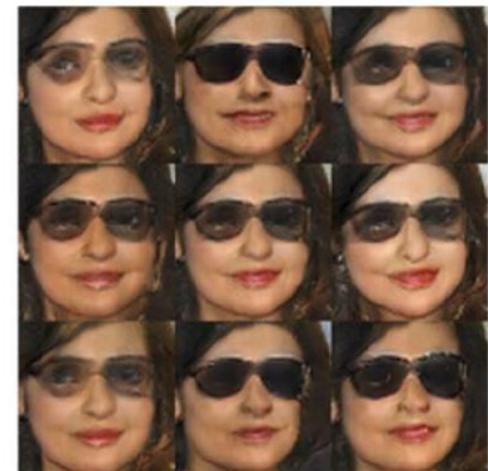
No glasses man



No glasses woman

Radford et al,  
ICLR 2016

Woman with glasses



# 2017: Explosion of GANs

## “The GAN Zoo”

See also: <https://github.com/soumith/ganhacks> for tips and tricks for trainings GANs

- GAN - Generative Adversarial Networks
- 3D-GAN - Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling
- acGAN - Face Aging With Conditional Generative Adversarial Networks
- AC-GAN - Conditional Image Synthesis With Auxiliary Classifier GANs
- AdaGAN - AdaGAN: Boosting Generative Models
- AEGAN - Learning Inverse Mapping by Autoencoder based Generative Adversarial Nets
- AffGAN - Amortised MAP Inference for Image Super-resolution
- AL-CGAN - Learning to Generate Images of Outdoor Scenes from Attributes and Semantic Layouts
- ALI - Adversarially Learned Inference
- AM-GAN - Generative Adversarial Nets with Labeled Data by Activation Maximization
- AnoGAN - Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery
- ArtGAN - ArtGAN: Artwork Synthesis with Conditional Categorical GANs
- b-GAN - b-GAN: Unified Framework of Generative Adversarial Networks
- Bayesian GAN - Deep and Hierarchical Implicit Models
- BEGAN - BEGAN: Boundary Equilibrium Generative Adversarial Networks
- BiGAN - Adversarial Feature Learning
- BS-GAN - Boundary-Seeking Generative Adversarial Networks
- CGAN - Conditional Generative Adversarial Nets
- CaloGAN - CaloGAN: Simulating 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters with Generative Adversarial Networks
- CCGAN - Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks
- CatGAN - Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks
- CoGAN - Coupled Generative Adversarial Networks
- Context-RNN-GAN - Contextual RNN-GANs for Abstract Reasoning Diagram Generation
- C-RNN-GAN - C-RNN-GAN: Continuous recurrent neural networks with adversarial training
- CS-GAN - Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets
- CVAE-GAN - CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training
- CycleGAN - Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks
- DTN - Unsupervised Cross-Domain Image Generation
- DCGAN - Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks
- DiscoGAN - Learning to Discover Cross-Domain Relations with Generative Adversarial Networks
- DR-GAN - Disentangled Representation Learning GAN for Pose-Invariant Face Recognition
- DualGAN - DualGAN: Unsupervised Dual Learning for Image-to-Image Translation
- EBGAN - Energy-based Generative Adversarial Network
- f-GAN - f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization
- FF-GAN - Towards Large-Pose Face Frontalization in the Wild
- GAWWN - Learning What and Where to Draw
- GeneGAN - GeneGAN: Learning Object Transfiguration and Attribute Subspace from Unpaired Data
- Geometric GAN - Geometric GAN
- GoGAN - Gang of GANs: Generative Adversarial Networks with Maximum Margin Ranking
- GP-GAN - GP-GAN: Towards Realistic High-Resolution Image Blending
- IAN - Neural Photo Editing with Introspective Adversarial Networks
- iGAN - Generative Visual Manipulation on the Natural Image Manifold
- IcGAN - Invertible Conditional GANs for image editing
- ID-CGAN - Image De-raining Using a Conditional Generative Adversarial Network
- Improved GAN - Improved Techniques for Training GANs
- InfoGAN - InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets
- LAGAN - Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis
- LAPGAN - Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks

<https://github.com/hindupuravinash/the-gan-zoo>

# 2017: Explosion of GANs

Better training and generation



LSGAN, Zhu 2017.



Wasserstein GAN,  
Arjovsky 2017.  
Improved Wasserstein  
GAN, Gulrajani 2017.

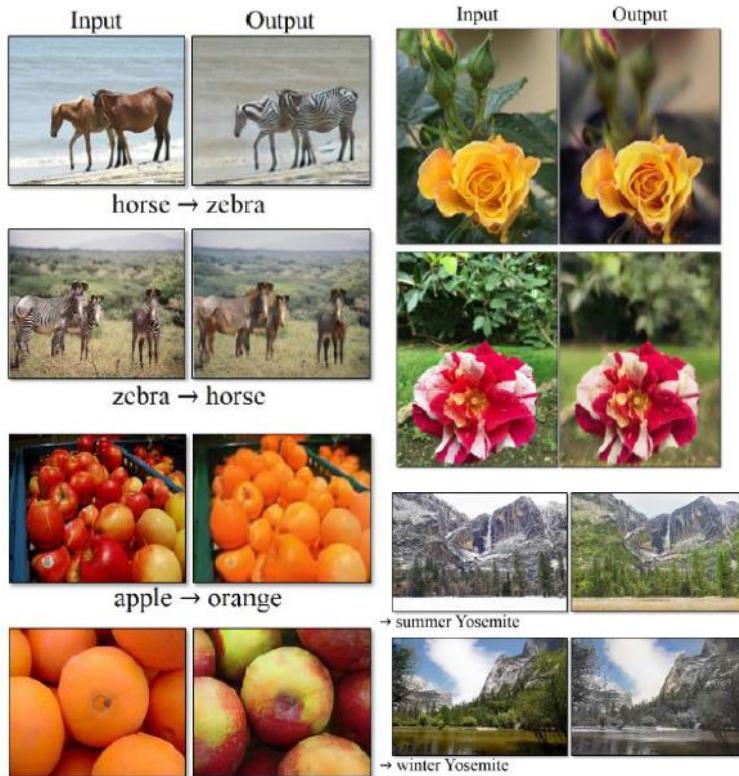


Progressive GAN, Karras 2018.

# Convolutional neural networks

## 2017: Explosion of GANs

Source->Target domain transfer



CycleGAN. Zhu et al. 2017.

### Text -> Image Synthesis

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.



Reed et al. 2017.

### Many GAN applications



Pix2pix. Isola 2017. Many examples at <https://phillipi.github.io/pix2pix/>

## Convolutional neural networks

# 2019: BigGAN



Brock et al., 2019

# Recap

## Generative Models

- PixelRNN and PixelCNN      Explicit density model, optimizes exact likelihood, good samples. But inefficient sequential generation.
- Variational Autoencoders (VAE)      Optimize variational lower bound on likelihood. Useful latent representation, inference queries. But current sample quality not the best.
- Generative Adversarial Networks (GANs)      Game-theoretic approach, best samples! But can be tricky and unstable to train, no inference queries.