High Level Computer Vision

Generative Models
@ June 12, 2019

Bernt Schiele & Mario Fritz

www.mpi-inf.mpg.de/hlcv/

Max Planck Institute for Informatics & Saarland University,
Saarland Informatics Campus Saarbrücken
Overview Today’s Lecture

• Unsupervised Learning

• Generative Models
  ‣ PixelRNN and PixelCNN
  ‣ Variational Autoencoder (VAE)
  ‣ Generative Adversarial Network (GAN)
  ‣ Conditional Generative Adversarial Network
Supervised vs. Unsupervised Learning

Supervised Learning

Data: (x, y)
x is data, y is label

Goal: Learn a function to map x -> y

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

Classification

This image is CC0 public domain

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Supervised Learning Examples

**Object Detection**
- *DOG, DOG, CAT*

**Semantic Segmentation**
- *GRASS, CAT, TREE, SKY*

**Image captioning**
- *A cat sitting on a suitcase on the floor*

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Supervised vs. Unsupervised Learning

Unsupervised Learning

Data: $\mathbf{x}$
Just data, no labels!

Goal: Learn some underlying hidden structure of the data

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

K-means clustering

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Unsupervised Learning Examples

1-d density estimation

2-d density estimation

L2 Loss function:
\[ \|x - \hat{x}\|^2 \]

Reconstructed input data
Decoder
Features
Encoder
Input data
Reconstructed data

Autoencoders (Feature learning)

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
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!

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

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
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

Figures from L-R are copyright: (1) Alec Radford et al. 2016; (2) David Bartheld et al. 2017; Phillip Isola et al. 2017. Reproduced with authors permission.

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Taxonomy of Generative Models

Today: discuss 3 most popular types of generative models today

Generative models
- Explicit density
- Implicit density

Tractable density
- Fully Visible Belief Nets
  - NADE
  - MADE
  - PixelRNN/CNN

Approximate density
- Variational Autoencoder
- Boltzmann Machine
-markov Chain

Markov Chain
- GSN

Direct
- GAN

Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
PixelRNN and PixelCNN
**Fully Visible Belief Network (FVBN)**

Explicit density model

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

\[
p(x) = \prod_{i=1}^{n} p(x_i | x_1, \ldots, x_{i-1})
\]

- Likelihood of image \( x \)
- Probability of \( i \)'th pixel value given all previous pixels

Will need to define ordering of “previous pixels”

Then maximize likelihood of training data

Complex distribution over pixel values => Express using a neural network!

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
PixelRNN \[\text{[van der Oord et al. 2016]}\]

Generate image pixels starting from corner

Dependency on previous pixels modeled using an RNN (LSTM)
PixelRNN  \[\text{[van der Oord et al. 2016]}\]

Generate image pixels starting from corner

Dependency on previous pixels modeled using an RNN (LSTM)

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
PixelRNN \cite{van_der ООrd_2016}

Generate image pixels starting from corner

Dependency on previous pixels modeled using an RNN (LSTM)

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
PixelRNN \[\textit{[van der Oord et al. 2016]}\]

Generate image pixels starting from corner

Dependency on previous pixels modeled using an RNN (LSTM)

\textbf{Drawback: sequential generation is slow!}
PixelCNN \[\text{[van der Oord et al. 2016]}\]

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

\[
p(x) = \prod_{i=1}^{n} p(x_i|x_1, \ldots, x_{i-1})
\]

Figure copyright van der Oord et al., 2016. Reproduced with permission.

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
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

Figure copyright van der Oord et al., 2016. Reproduced with permission.
Generation Examples

32x32 CIFAR-10

32x32 ImageNet

Figures copyright Aaron van der Oord et al., 2016. Reproduced with permission.

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
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:

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

VAEs define intractable density function with latent \( z \):

\[ p_\theta(x) = \int p_\theta(z)p_\theta(x|z)dz \]

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.

- \( \mathbf{z} \) usually smaller than \( \mathbf{x} \) (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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Some Background first: Autoencoders

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

Reconstructed input data

Features

Input data

Decoder

Encoder

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Some background first: Autoencoders

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

Reconstructed input data

Decoder

Features

Encoder

Input data

Reconstructed data

Encoder: 4-layer conv
Decoder: 4-layer upconv

Input data

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Some background first: Autoencoders

Train such that features can be used to reconstruct original data

L2 Loss function:
\[ \| x - \hat{x} \|^2 \]

Doesn’t use labels!

Reconstructed data

Encoder: 4-layer conv
Decoder: 4-layer upconv

Input data

Reconstructed input data

Features

Decoder

Encoder

Input data

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Some Background first: Autoencoders

![Autoencoder Diagram]

Reconstructed input data → Decoder → Features → Encoder

Input data

After training, throw away decoder

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Some Background first: Autoencoders

Encoder can be used to initialize a supervised model

Input data

Features

Predicted Label

Loss function (Softmax, etc)

Classifier

Fine-tune encoder jointly with classifier

Train for final task (sometimes with small data)

bird
plane
dog
deer
truck

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
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?

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Variational Autoencoders (VAEs)

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Variational Autoencoders (VAEs)

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 (VAEs)

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 (VAEs)

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

**How to train the model?**

Remember strategy for training generative models from FVBNs. 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

Q: What is the problem with this?

Intractable!

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Variational Autoencoders: Intractability

Data likelihood: \( p_\theta(x) = \int p_\theta(z)p_\theta(x|z)dz \)

Simple Gaussian prior

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Variational Autoencoders: Intractability

Data likelihood:  \[ p_\theta(x) = \int p_\theta(z)p_\theta(x|z)dz \]

Decoder neural network

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Variational Autoencoders: Intractability

Data likelihood: \[ p_\theta(x) = \int p_\theta(z)p_\theta(x|z)dz \]

Intractible to compute \( p(x|z) \) for every \( z \)!

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Variational Autoencoders: Intractability

Data likelihood: \( p_\theta(x) = \int p_\theta(z)p_\theta(x|z)dz \)

Posterior density also intractable: \( p_\theta(z|x) = \frac{p_\theta(x|z)p_\theta(z)}{p_\theta(x)} \)

Intractable data likelihood

Solution: In addition to decoder network modeling \( p_\theta(x|z) \), define additional encoder network \( q_\phi(z|x) \) that approximates \( p_\theta(z|x) \)

Will see that this allows us to derive a lower bound on the data likelihood that is tractable, which we can optimize

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Variational Autoencoders (VAEs)

Since we’re modeling probabilistic generation of data, encoder and decoder networks are probabilistic

Mean and (diagonal) covariance of $z \mid x$

Encoder network

$q_\phi(z \mid x)$

(parameters $\phi$)

$x$

Mean and (diagonal) covariance of $x \mid z$

Decoder network

$p_\theta(x \mid z)$

(parameters $\theta$)

$z$

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014
Variational Autoencoders (VAEs)

Since we’re modeling probabilistic generation of data, encoder and decoder networks are probabilistic.

Encoder and decoder networks also called “recognition”/“inference” and “generation” networks.

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014
Variational Autoencoders (VAEs)

Now equipped with our encoder and decoder networks, let’s work out the (log) data likelihood:

$$\log p_\theta(x^{(i)}) = \mathbb{E}_{z \sim q_\phi(z|x^{(i)})} \left[ \log p_\theta(x^{(i)}) \right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

Taking expectation wrt. $z$
(using encoder network) will come in handy later
Variational Autoencoders (VAEs)

Now equipped with our encoder and decoder networks, let’s work out the (log) data likelihood:

\[
\log p_\theta(x^{(i)}) = \mathbb{E}_{z \sim q_\phi(z|x^{(i)})} \left[ \log p_\theta(x^{(i)}) \right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)
\]

\[
= \mathbb{E}_z \left[ \log \frac{p_\theta(x^{(i)} | z)p_\theta(z)}{p_\theta(z | x^{(i)})} \right] \quad \text{(Bayes’ Rule)}
\]

\[
= \mathbb{E}_z \left[ \log \frac{p_\theta(x^{(i)} | z)p_\theta(z)}{p_\theta(z | x^{(i)})} q_\phi(z | x^{(i)}) \right] \quad \text{(Multiply by constant)}
\]

\[
= \mathbb{E}_z \left[ \log p_\theta(x^{(i)} | z) \right] - \mathbb{E}_z \left[ \log \frac{q_\phi(z | x^{(i)})}{p_\theta(z)} \right] + \mathbb{E}_z \left[ \log \frac{p_\theta(z | x^{(i)})}{q_\phi(z | x^{(i)})} \right] \quad \text{(Logarithms)}
\]

\[
= \mathbb{E}_z \left[ \log p_\theta(x^{(i)} | z) \right] - D_{KL}(q_\phi(z | x^{(i)}) \parallel p_\theta(z)) + D_{KL}(q_\phi(z | x^{(i)}) \parallel p_\theta(z | x^{(i)}))
\]

The expectation wrt. \(z\) (using encoder network) let us write nice KL terms.

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Variational Autoencoders (VAEs)

Now equipped with our encoder and decoder networks, let’s work out the (log) data likelihood:

$$\log p_\theta(x^{(i)}) = E_z \sim q_\phi(z|x^{(i)}) \left[ \log p_\theta(x^{(i)}) \right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= E_z \left[ \log \frac{p_\theta(x^{(i)} | z) p_\theta(z)}{p_\theta(z | x^{(i)})} \right] \quad \text{(Bayes’ Rule)}$$

$$= E_z \left[ \log \frac{p_\theta(x^{(i)} | z) p_\theta(z)}{p_\theta(z | x^{(i)})} \right] \frac{q_\phi(z | x^{(i)})}{q_\phi(z | x^{(i)})} \quad \text{(Multiply by constant)}$$

$$= E_z \left[ \log p_\theta(x^{(i)} | z) \right] - E_z \left[ \log \frac{q_\phi(z | x^{(i)})}{p_\theta(z)} \right] + E_z \left[ \log \frac{q_\phi(z | x^{(i)})}{p_\theta(z | x^{(i)})} \right] \quad \text{(Logarithms)}$$

$$= E_z \left[ \log p_\theta(x^{(i)} | z) \right] - D_{KL}(q_\phi(z | x^{(i)}) || p_\theta(z)) + D_{KL}(q_\phi(z | x^{(i)}) || p_\theta(z | x^{(i)}))$$

Decoder network gives $p_\theta(x|z)$, can compute estimate of this term through sampling. (Sampling differentiable through reparam. trick, see paper.)

This KL term (between Gaussians for encoder and $z$ prior) has nice closed-form solution!

$p_\theta(z|x)$ intractable (saw earlier), can’t compute this KL term :( But we know KL divergence always >= 0.

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Variational Autoencoders (VAEs)

Now equipped with our encoder and decoder networks, let's work out the (log) data likelihood:

\[
\log p_\theta(x^{(i)}) = E_{z \sim q_\phi(z|x^{(i)})} \left[ \log p_\theta(x^{(i)}) \right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)
\]

\[
= E_z \left[ \log \frac{p_\theta(x^{(i)} | z)p_\theta(z)}{p_\theta(z | x^{(i)})} \right] \quad (\text{Bayes' Rule})
\]

\[
= E_z \left[ \log \frac{p_\theta(x^{(i)} | z)p_\theta(z)}{p_\theta(z | x^{(i)})} \frac{q_\phi(z | x^{(i)})}{p_\theta(z | x^{(i)})} \right] \quad (\text{Multiply by constant})
\]

\[
= E_z \left[ \log p_\theta(x^{(i)} | z) \right] - E_z \left[ \log \frac{q_\phi(z | x^{(i)})}{p_\theta(z)} \right] + E_z \left[ \log \frac{q_\phi(z | x^{(i)})}{p_\theta(z | x^{(i)})} \right] \quad (\text{Logarithms})
\]

\[
= E_z \left[ \log p_\theta(x^{(i)} | z) \right] - D_{KL}(q_\phi(z | x^{(i)}) \| p_\theta(z)) + D_{KL}(q_\phi(z | x^{(i)}) \| p_\theta(z | x^{(i)})) \\
\leq 0
\]

**Tractable lower bound** which we can take gradient of and optimize! \((p_\theta(x|z)\text{ differentiable, KL term differentiable})\)
Variational Autoencoders (VAEs)

Now equipped with our encoder and decoder networks, let's work out the (log) data likelihood:

\[
\log p_\theta(x^{(i)}) = \mathbb{E}_{z \sim q_\phi(z|x^{(i)})} \left[ \log p_\theta(x^{(i)}) \right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)
\]

\[
= \mathbb{E}_z \left[ \log \frac{p_\theta(x^{(i)} | z)p_\theta(z)}{p_\theta(z | x^{(i)})} \right] \quad \text{(Bayes’ Rule)}
\]

\[
= \mathbb{E}_z \left[ \log p_\theta(x^{(i)} | z) \right] - \mathbb{E}_z \left[ \log \frac{q_\phi(z | x^{(i)})}{p_\theta(z)} \right] + \mathbb{E}_z \left[ \log \frac{q_\phi(z | x^{(i)})}{p_\theta(z | x^{(i)})} \right] \quad \text{(Logarithms)}
\]

\[
= \mathbb{E}_z \left[ \log p_\theta(x^{(i)} | z) \right] - D_{KL}(q_\phi(z | x^{(i)}) \| p_\theta(z)) + D_{KL}(q_\phi(z | x^{(i)}) \| p_\theta(z | x^{(i)}))
\]

\[
\mathcal{L}(x^{(i)}, \theta, \phi) \geq 0
\]

\[
\log p_\theta(x^{(i)}) \geq \mathcal{L}(x^{(i)}, \theta, \phi) \quad \text{Variational lower bound (“ELBO”)}
\]

\[
\theta^*, \phi^* = \arg \max_{\theta, \phi} \sum_{i=1}^{N} \mathcal{L}(x^{(i)}, \theta, \phi)
\]

Training: Maximize lower bound

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Variational Autoencoders (VAEs)

Putting it all together: maximizing the likelihood lower bound

$$
\mathbb{E}_z \left[ \log p_{\theta}(x^{(i)} \mid z) \right] - D_{KL}(q_{\phi}(z \mid x^{(i)}) \parallel p_{\theta}(z))
$$

$$
\mathcal{L}(x^{(i)}, \theta, \phi)
$$

Let’s look at computing the bound (forward pass) for a given minibatch of input data

Input Data $\mathcal{X}$

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Variational Autoencoders (VAEs)

Putting it all together: maximizing the likelihood lower bound

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

Maximize likelihood of original input being reconstructed

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

Sample \( x|z \) from \( x|z \sim \mathcal{N}(\mu_{x|z}, \Sigma_{x|z}) \)

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

Sample \( z \) from \( z|x \sim \mathcal{N}(\mu_{z|x}, \Sigma_{z|x}) \)

Encoder network
\( q_\phi(z|x) \)

Input Data
\( x \)

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Variational Autoencoders: Generating Data

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

Data manifold for 2-d $z$

Sample $z$ from $z \sim \mathcal{N}(0, I)$

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
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 $q_\phi(z|x)$!

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Variational Autoencoders: Generating Data

32x32 CIFAR-10

Labeled Faces in the Wild


slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Variational Autoencoders (VAEs)

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)

Active areas of research:
- More flexible approximations, e.g. richer approximate posterior instead of diagonal Gaussian
- Incorporating structure in latent variables

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Generative Adversarial Networks (GAN)
So far...

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

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

VAEs define intractable density function with latent \( z \):

\[ p_\theta(x) = \int p_\theta(z)p_\theta(x|z)dz \]

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Goal of Generative Adversarial Networks

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

slide credit: Ian Goodfellow
Goal of Generative Adversarial Networks

- Density estimation

- Sample generation

Training examples  Model samples

slide credit: Ian Goodfellow
Generative Adversarial Networks

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

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
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

Fake and real images copyright Emily Denton et al. 2015. Reproduced with permission.

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
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**

Minimax objective function:

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

- 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)

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014
$p_D(data)$

Data distribution

Model distribution

Poorly fit model

After updating $D$

After updating $G$

Mixed strategy equilibrium
Training GANs: Two-Player Game

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

Alternate between:
1. **Gradient ascent** on discriminator
   \[
   \max_{\theta_d} \left[ \mathbb{E}_{x \sim p_{\text{data}}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]
   \]

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!

Gradient signal dominated by region where sample is already good.

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

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Training GANs: Two-Player Game

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.

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

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
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

![Diagram of GANs](image)

- **Fake Images** (from generator)
- **Real Images** (from training set)
- **Random noise**

**After training, use generator network to generate new images**

Fake and real images copyright Emily Denton et al. 2015. Reproduced with permission.

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Generative Adversarial Networks (GANs)

Generated samples

Nearest neighbor from training set

Figures copyright Ian Goodfellow et al., 2014. Reproduced with permission.

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Generative Adversarial Networks (GANs)

Generated samples (CIFAR-10)

Nearest neighbor from training set

Figures copyright Ian Goodfellow et al., 2014. Reproduced with permission.

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
GANs: Convolutional Architectures


slide credit: Fei-Fei, Justin Johnson, Serena Yeung
GANs: Convolutional Architectures

Samples from the model look much better!

Radford et al,
ICLR 2016

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
GANs: Convolutional Architectures

Interpolating between random points in latent space

Radford et al, ICLR 2016

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
GANs: Interpretable Vector Math

Samples from the model

Average Z vectors, do arithmetic

Smiling woman  Neutral woman  Neutral man

Smiling Man

Radford et al, ICLR 2016

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
GANs: Interpretable Vector Math

Glasses man  No glasses man  No glasses woman

Radford et al, ICLR 2016

Woman with glasses

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
2017 & 2018: Explosion of GANs...

“The GAN Zoo”

- GAN - Generative Adversarial Networks
- 3D-GAN - Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling
- scGAN - 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 Categorial 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 Prospective 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
- LAGAN - Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
2017 & 2018: Explosion of GANs...

Better training and generation

- Progressive GAN, Karras 2018.

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Conditional GANs:

Source -> Target domain transfer

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Learning What and Where to Draw

Scott Reed¹,³, Zeynep Akata², Santosh Mohan¹, Samuel Tenka¹, Bernt Schiele², Honglak Lee¹
The discriminator $D$ tries to distinguish real (text, image) pairs from synthetic. The generator $G$ tries to fool $D$. 

$$\min_G \max_D V(D, G) = \mathbb{E}_{x, t \sim p_{data}(x, t)}[\log D(x, t)] + \mathbb{E}_{z \sim p_z(z), t \sim p_{data}(t)}[\log(1 - D(G(z, t)))]$$
Text-conditional GAN

This flower has small, round violet petals with a dark purple center

$z \sim \mathcal{N}(0, 1)$

Generator Network
Text-conditional GAN

This flower has small, round violet petals with a dark purple center

$z \sim \mathcal{N}(0, 1)$

Generator Network
Text-conditional GAN

This flower has small, round violet petals with a dark purple center

$z \sim \mathcal{N}(0, 1)$

Generator Network

Discriminator Network

This flower has small, round violet petals with a dark purple center
Idea: condition on location as well as sentence/text

1. Bounding box

This bird is completely black.

2. Keypoints, e.g. 15 parts of a bird

This bird is bright blue.
Conditioning on bounding box

Spatial replicate, crop to bbox

A red bird with a black face

$z \sim \mathcal{N}(0, 1)$

Generator Network
Conditioning on bounding box

Generator Network

Discriminator Network

A red bird with a black face

\( z \sim \mathcal{N}(0, 1) \)
Moving the bird around with bounding box (noise z fixed)

Caption
This bird has a black head, a long orange beak and yellow body

GT
Moving the bird around with bounding box (noise z fixed)

<table>
<thead>
<tr>
<th>Caption</th>
<th>GT</th>
<th>Shrinking</th>
</tr>
</thead>
<tbody>
<tr>
<td>This bird has a black head, a long orange beak and yellow body</td>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
</tr>
</tbody>
</table>
Moving the bird around with bounding box (noise $z$ fixed)

<table>
<thead>
<tr>
<th><strong>Caption</strong></th>
<th><strong>GT</strong></th>
<th><strong>Translation</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>This bird has a black head, a long orange beak and yellow body</td>
<td>![Image of a bird]</td>
<td>![Translation images]</td>
</tr>
</tbody>
</table>
Moving the bird around with bounding box (noise z fixed)

<table>
<thead>
<tr>
<th>Caption</th>
<th>GT</th>
<th>Stretching</th>
</tr>
</thead>
<tbody>
<tr>
<td>This bird has a black head, a long orange beak and yellow body</td>
<td><img src="image1.jpg" alt="Bird Image" /></td>
<td><img src="image2.jpg" alt="Bounding Box Images" /></td>
</tr>
</tbody>
</table>
Moving the bird around with bounding box (noise $z$ fixed)

<table>
<thead>
<tr>
<th>Caption</th>
<th>GT</th>
<th>Shrinking</th>
<th>Translation</th>
<th>Stretching</th>
</tr>
</thead>
<tbody>
<tr>
<td>This bird has a black head, a long orange beak and yellow body</td>
<td><img src="image1" alt="Bird Image" /></td>
<td><img src="image2" alt="Shrinking Image" /></td>
<td><img src="image3" alt="Translation Image" /></td>
<td><img src="image4" alt="Stretching Image" /></td>
</tr>
<tr>
<td>This large black bird has a pointy beak and black eyes</td>
<td><img src="image5" alt="Bird Image" /></td>
<td><img src="image6" alt="Shrinking Image" /></td>
<td><img src="image7" alt="Translation Image" /></td>
<td><img src="image8" alt="Stretching Image" /></td>
</tr>
<tr>
<td>This small blue bird has a short pointy beak and brown patches on its wings</td>
<td><img src="image9" alt="Bird Image" /></td>
<td><img src="image10" alt="Shrinking Image" /></td>
<td><img src="image11" alt="Translation Image" /></td>
<td><img src="image12" alt="Stretching Image" /></td>
</tr>
</tbody>
</table>
Moving the bird around with key points (noise \( z \) fixed)

Caption
This bird has a black head, a long orange beak and yellow body

<table>
<thead>
<tr>
<th>Caption</th>
<th>GT</th>
<th>Shrinking</th>
</tr>
</thead>
<tbody>
<tr>
<td>This bird has a black head, a long orange beak and yellow body</td>
<td>![Image of bird in GT]</td>
<td>![Image of shrinking bird]</td>
</tr>
</tbody>
</table>
Moving the bird around with key points (noise $z$ fixed)

Caption

This bird has a black head, a long orange beak and yellow body

GT

Translation
Moving the bird around with key points (noise z fixed)

<table>
<thead>
<tr>
<th>Caption</th>
<th>GT</th>
</tr>
</thead>
<tbody>
<tr>
<td>This bird has a black head, a long orange beak and yellow body</td>
<td></td>
</tr>
</tbody>
</table>

Stretching
Moving the bird around with key points (noise $z$ fixed)

<table>
<thead>
<tr>
<th>Caption</th>
<th>GT</th>
<th>Shrinking</th>
<th>Translation</th>
<th>Stretching</th>
</tr>
</thead>
<tbody>
<tr>
<td>This bird has a black head, a long orange beak and yellow body</td>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
</tr>
<tr>
<td>This large black bird has a pointy beak and black eyes</td>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /></td>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Image" /></td>
</tr>
<tr>
<td>This small blue bird has a short pointy beak and brown patches on its wings</td>
<td><img src="image9.png" alt="Image" /></td>
<td><img src="image10.png" alt="Image" /></td>
<td><img src="image11.png" alt="Image" /></td>
<td><img src="image12.png" alt="Image" /></td>
</tr>
</tbody>
</table>
Generative Adversarial Networks (GANs)

Don’t work with an explicit density function
Take game-theoretic approach: learn to generate from training distribution through 2-player game

Pros:
- Beautiful, state-of-the-art samples!

Cons:
- Trickier / more unstable to train
- Can’t solve inference queries such as p(x), p(z|x)

Active areas of research:
- Better loss functions, more stable training (Wasserstein GAN, LSGAN, many others)
- Conditional GANs, GANs for all kinds of applications

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Recap

Generative Models

- **PixelRNN and PixelCNN**
  
  Explicit density model, optimizes exact likelihood, good samples. But inefficient sequential generation.

- **Variational Autoencoders (VAE)**
  

- **Generative Adversarial Networks (GANs)**
  
  Game-theoretic approach, best samples!
  But can be tricky and unstable to train, no inference queries.

Also recent work in combinations of these types of models! E.g. Adversarial Autoencoders (Makhanzi 2015) and PixelVAE (Gulrajani 2016)