CS 4803 / 7643: Deep Learning

Topics:
- Unsupervised Learning
- Generative Models (PixelRNNs, PixelCNNs)
- Generative Adversarial Networks

Zsolt Kira
Georgia Tech
Administrativia

- Projects!
  - Checkpoint April 7th
  - Schedule and details coming soon
Last Time:
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.

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Reconstruction

- Autoencoder + additional assumptions
  - Denoising Autoencoder
    - Assume the embedding should not encode the noise.

Minimize the difference (with MSE)
Tasks

Supervised Learning

\( x \xrightarrow{\text{Classification}} y \) Discrete

\( x \xrightarrow{\text{Regression}} y \) Continuous

Unsupervised Learning

\( x \xrightarrow{\text{Clustering}} c \) Discrete

\( x \xrightarrow{\text{Dimensionality Reduction}} z \) Continuous

\( x \xrightarrow{\text{Density Estimation}} p(x) \) On simplex
Overview for Today

- Generative Models
  - PixelRNN and PixelCNN
  - Generative Adversarial Networks (GAN)
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 $p_{\text{data}}(x)$

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Generative Classification vs Discriminative Classification vs Density Estimation

• Generative Classification
  – Model $p(x, y)$; estimate $p(x|y)$ and $p(y)$
  – Use Bayes Rule to predict $y$
  – E.g Naïve Bayes

• Discriminative Classification
  – Estimate $p(y|x)$ directly
  – E.g. Logistic Regression

• Density Estimation
  – Model $p(x)$
  – E.g. VAEs
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

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
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

- Data augmentation!

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

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Taxonomy of Generative Models

Generative models

Explicit density

Tractable density
- Fully Visible Belief Nets
  - NADE
  - MADE
  - PixelRNN/CNN
- Change of variables models (nonlinear ICA)

Implicit density

Approximate density
- Variational Autoencoder
- Boltzmann Machine

Markov Chain

Direct
- GAN
- GSN

Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.
We will discuss 3 most popular types of generative models.

- Fully Visible Belief Nets
  - NADE
  - MADE
  - PixelRNN/CNN
- Change of variables models (nonlinear ICA)

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

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
PixelRNN and PixelCNN
Fully Observable Model

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

Then maximize likelihood of training data

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Fully Observable Model

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

Then maximize likelihood of training data

Complex distribution over pixel values

$\Rightarrow$ Express using a neural network!

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Fully Observable Model

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

Complex distribution over pixel values

Will need to define ordering of “previous pixels”

Then maximize likelihood of training data

=> Express using a neural network!

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Example:
Character-level Language Model

Vocabulary: [h,e,l,o]

Example training sequence: “hello”
PixelRNN [van der Oord et al. 2016]

Generate image pixels starting from corner

Dependency on previous pixels modeled using an RNN (LSTM)
PixelRNN \cite{van_der_Oord_2016}

Generate image pixels starting from corner

Dependency on previous pixels modeled using an RNN (LSTM)

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
PixelRNN [van der Oord et al. 2016]

Generate image pixels starting from corner

Dependency on previous pixels modeled using an RNN (LSTM)
Test Time: Sample / Argmax / Beam Search

Example:
Character-level Language Model Sampling

Vocabulary:
[h,e,l,o]

At test-time sample characters one at a time, feed back to model
Test Time: Sample / Argmax / Beam Search

Example:
Character-level Language Model Sampling

Vocabulary: [h,e,l,o]

At test-time sample characters one at a time, feed back to model

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Test Time: Sample / Argmax / Beam Search

Example:
Character-level Language Model Sampling

Vocabulary: [h,e,l,o]

At test-time sample characters one at a time, feed back to model

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Test Time: Sample / Argmax / Beam Search

Example:
Character-level Language Model Sampling

Vocabulary: [h,e,l,o]

At test-time sample characters one at a time, feed back to model

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
PixelRNN \cite{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 \cite{van_der_Oord_2016}

Still generate image pixels starting from corner

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

• Apply masks so that a pixel does not see “future” pixels
PixelCNN \cite{van_der_oord_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.
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

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

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Image Completion

Figure 1. Image completions sampled from a PixelRNN.
Results from generating sounds

- https://deepmind.com/blog/wavenet-generative-model-raw-audio/
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++)
Generative Adversarial Networks (GAN)
So far...

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

$$p_\theta(x) = \prod_{i=1}^{n} p_\theta(x_i | x_1, ..., x_{i-1})$$
### Taxonomy of Generative Models

Today & Next Week: discuss 3 most popular types of generative models today

- **Generative models**
  - Explicit density
    - Tractable density
      - Fully Visible Belief Nets
        - NADE
        - MADE
      - PixelRNN/CNN
    - Approximate density
      - Variational
      - Variational Autoencoder
  - Implicit density
    - Markov Chain
      - Markov Chain Variational
      - Boltzmann Machine
    - GSN

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

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

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

What if we give up on explicitly modeling density, and just want ability to sample?
So far...
PixelCNNs define tractable density function, optimize likelihood of training data:

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

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

GANs: don’t work with any explicit density function!
Generative Adversarial Networks (GANs)

GANs are a combination of the following ideas:

1. Learning to Sample
2. Adversarial Training
3. “Reparameterization” Trick
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?
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

Generative Adversarial Networks

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014
Generative Adversarial Networks (GANs)

GANs are a combination of the following ideas:

1. Learning to Sample
2. Adversarial Training
3. “Reparameterization” Trick
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

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

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
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.

---

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
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:

$$\min_{\theta_g} \max_{\theta_d} \left[ \mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$
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 outputs likelihood in (0,1) of real image

Discriminator output for real data \(x\)

Discriminator output for generated fake data \(G(z)\)

---

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

---

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
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)

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014
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. **Gradient descent** on generator
   $$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$
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. **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

---

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
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.

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
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.

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

---

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

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
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

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
GANs

• Demo
  – https://poloclub.github.io/ganlab/
Generative Adversarial Nets

Generated samples

Nearest neighbor from training set

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

Generated samples (CIFAR-10)

Nearest neighbor from training set

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

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
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.

Generative Adversarial Nets: Convolutional Architectures


Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Generative Adversarial Nets: Convolutional Architectures

Samples from the model look much better!

Radford et al, ICLR 2016

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Interpolating between random points in latent space

Radford et al, ICLR 2016
BigGAN
"We observe class leakage, where images from one class contain properties of another, as exemplified by Figure 4(d)"

Figure 4: Samples from our model with truncation threshold 0.5 (a-c) and an example of class leakage in a partially trained model (d).
2017: 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
- 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 Generative Networks
- 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 Networks
- 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
- CAMWN - 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
- JAN - 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
- LADGAN - Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks

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

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
2017: Explosion of GANs

See also: https://github.com/soumith/ganhacks for tips and tricks for training GANs

“The GAN Zoo”

- 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 Generative Adversarial Networks
- 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
- GANWAN - 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 Intraoperative 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

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Schemes to Ensure Samples Are Diverse
(Salimans et al., 2016)

- Use nearest neighbor on latent representation to detect samples in a minibatch that are too similar.
Using Labels Can Improve Generated Samples

- Denton et al. (2015)
- Salimans et al. (2016)
Using Labels Can Improve Generated Samples
(Denton et al., 2015)

- LAPGAN: multiresolution deconvolutional pyramid
- CC-LAPGAN: class conditional LAPGAN
Mode Collapse

- Optimization of GANs is tricky
  - Not guaranteed to find Nash equilibrium

- Large number of methods to combat:
  - Use history of discriminators
  - Regularization
  - Different divergence measures
Application: Data Augmentation

Low-Shot Learning from Imaginary Data, Yu-Xiong Wang, Ross Girshick, Martial Hebert, Bharath Hariharan
Application: Domain Adaptation

- **Idea**: Train a model on *source* data and adapt to *target* data using unlabeled examples from target.
Approach

<table>
<thead>
<tr>
<th>Method</th>
<th>MNIST $\rightarrow$ USPS</th>
<th>USPS $\rightarrow$ MNIST</th>
<th>SVHN $\rightarrow$ MNIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source only</td>
<td>$0.752 \pm 0.016$</td>
<td>$0.571 \pm 0.017$</td>
<td>$0.601 \pm 0.011$</td>
</tr>
<tr>
<td>Gradient reversal</td>
<td>$0.771 \pm 0.018$</td>
<td>$0.730 \pm 0.020$</td>
<td></td>
</tr>
<tr>
<td>Domain confusion</td>
<td>$0.791 \pm 0.005$</td>
<td>$0.665 \pm 0.033$</td>
<td>$0.681 \pm 0.003$</td>
</tr>
<tr>
<td>CoGAN</td>
<td>$0.912 \pm 0.008$</td>
<td>$0.891 \pm 0.008$</td>
<td>did not converge</td>
</tr>
<tr>
<td>ADDA (Ours)</td>
<td>$0.894 \pm 0.002$</td>
<td>$0.901 \pm 0.008$</td>
<td>$0.760 \pm 0.018$</td>
</tr>
</tbody>
</table>

Table 2: Experimental results on unsupervised adaptation among MNIST, USPS, and SVHN.
Aside: Other ways to Align

[Image of a digital SLR camera and a low-cost camera with flash]

[Diagram showing a feature extractor $G_f(\cdot; \theta_f)$, a label predictor $G_y(\cdot; \theta_y)$, and a domain classifier $G_d(\cdot; \theta_d)$, with gradient reversal layers and loss functions $L_y$ and $L_d$.]

[Reference to Ganin et al., JMLR 2016]
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 Li, Justin Johnson, Serena Yeung, CS 231n