Generative models

- Explicit density
  - Tractable density
    - Fully Visible Belief Nets
      - NADE
      - MADE
      - PixelRNN/CNN
    - Change of variables models (nonlinear ICA)
  - Approximate density
    - Variational
      - Variational Autoencoder
  - Implicit density
    - Markov Chain
      - GSN
    - Direct
      - GAN

Figure from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017
Taxonomy of generative models

Explicit density

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

- Approximate density
  - Variational Autoencoder
  - Variational

Implicit density

- Direct GAN
- Markov Chain
  - GSN
  - Boltzmann Machine
Generative Adversarial Networks
Generative Adversarial Networks (GANs)

Cumulative number of named GAN papers by month

https://github.com/hindupuravinash/the-gan-zoo
Convolution and Deconvolution

Convolution
no padding, no stride

Transposed convolution
no padding, no stride

https://github.com/vdumoulin/conv_arithmetic
Autoencoder

encoder

bottleneck layer

decoder

Conv

Deconv
Reconstruction: Autoencoder

- **Input Image**
- **Encoder**
- **Bottleneck Layer**
- **Decoder**
- **Conv**
- **Deconv**
- **Output Image**

Reconstruction Loss (often L2)
Training Autoencoders

Input $x$  

Encoder  

Latent space $z$

$\text{dim}(z) < \text{dim}(x)$

Decoder  

Reconstruction $x'$

Input images

Reconstructed images
Decoder as Generative Model

Latent space $z$
$\text{dim (z)} < \text{dim (x)}$

"Test time":
$\rightarrow$ reconstruction from 'random' vector

Reconstruction Loss (often L2)

Output Image
Decoder as Generative Model

Interpolation between two chair models

[Dosovitsky et al. 14] Learning to Generate Chairs
Decoder as Generative Model

Morphing between chair models

[Dosovitsky et al. 14] Learning to Generate Chairs
Decoder as Generative Model

"Test time":
- reconstruction from 'random' vector

Latent space $z$
- $\text{dim}(z) < \text{dim}(x)$

Reconstruction Loss
- Often L2, i.e., sum of squared dist.
- $\Rightarrow$ L2 distributes error equally
- $\Rightarrow$ mean is opt.
- $\Rightarrow$ res. is blurry

Instead of L2, can we "learn" a loss function?
Generative Adversarial Networks (GANs)

\[ G(z) \]

\[ D(G(z)) \]

[Goodfellow et al. 14] GANs (slide McGuinness)
Generative Adversarial Networks (GANs)

\[
G(z) \rightarrow D(G(z)) \\
\]

[Goodfellow et al. 14] GANs (slide McGuinness)
Generative Adversarial Networks (GANs)

\[ D(x) \text{ tries to be near } 1 \]

Differentiable function $D$

$x$ sampled from data

$D$ tries to make $D(G(z))$ near 0, $G$ tries to make $D(G(z))$ near 1

$x$ sampled from model

Differentiable function $G$

Input noise $z$
GANs: Loss Functions

Discriminator loss

\[ J(D) = -\frac{1}{2} \mathbb{E}_{x \sim p_{data}} \log D(x) - \frac{1}{2} \mathbb{E}_z \log (1 - D(G(z))) \]

Generator loss

\[ J(G) = -J(D) \]

- Minimax Game:
  - G minimizes probability that D is correct
  - Equilibrium is saddle point of discriminator loss

\[ \rightarrow D \text{ provides supervision (i.e., gradients) for } G \]

[Goodfellow et al. 14/16] GANs
GANs: Loss Functions

Discriminator loss

\[ J^{(D)} = -\frac{1}{2} \mathbb{E}_{x \sim p_{data}} \log D(x) - \frac{1}{2} \mathbb{E}_z \log (1 - D(G(z))) \]

Generator loss

\[ J^{(G)} = -\frac{1}{2} \mathbb{E}_z \log D(G(z)) \]

- Heuristic Method (often used in practice)
  - G maximizes the log-probability of D being mistaken
  - G can still learn even when D rejects all generator samples

[Goodfellow et al. 14/16] GANs
Alternating Gradient Updates

- Step 1: Fix $G$, and perform gradient step to

$$J^{(D)} = -\frac{1}{2} \mathbb{E}_{x \sim p_{\text{data}}} \log D(x) - \frac{1}{2} \mathbb{E}_z \log (1 - D(G(z)))$$

- Step 2: Fix $D$, and perform gradient step to

$$J^{(G)} = -\frac{1}{2} \mathbb{E}_z \log D(G(z))$$
Vanilla GAN

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

https://papers.nips.cc/paper/5423-generative-adversarial-nets
Training a GAN

GANs: Loss Functions

Minimax

Heuristic
DCGAN: Generator

Generator of Deep Convolutional GANs

DCGAN: https://github.com/carpedm20/DCGAN-tensorflow
DCGAN: Results

Results on MNIST

Prof. Leal-Taixé and Prof. Niessner

DCGAN: https://github.com/carpedm20/DCGAN-tensorflow
DCGAN: Results

Results on CelebA (200k relatively well aligned portrait photos)

DCGAN: https://github.com/carpedm20/DCGAN-tensorflow
DCGAN: Results

Asian face dataset

DCGAN: https://github.com/carpedm20/DCGAN-tensorflow
DCGAN: Results

https://github.com/carpedm20/DCGAN-tensorflow
DCGAN: Results

Loss of D and G on custom dataset
“Bad” Training Curves

“Good” Training Curves

Generator’s Error through Time

Discriminator’s Error through Time

“Good” Training Curves

Training Schedules

• Adaptive schedules

• For instance:
  
  while loss_discriminator > t_d:
  train discriminator
  
  while loss_generator > t_g:
  train generator
Weak vs Strong Discriminator

Need balance 😊

• Discriminator too weak?
  – No good gradients (cannot get better than teacher…)

• Generator too weak?
  – Discriminator will always be right
Mode Collapse

- $\min_G \max_D V(G, D) \neq \max_D \min_G V(G, D)$

- $D$ in inner loop -> convergence to correct dist.
- $G$ in inner loop -> easy to convergence to one sample
Mode Collapse

- Data dim. Fixed (512)
- Performance correlates with # of modes

- More modes, smaller recovery rate!
- part of the reason, why we often see GAN-results on specific domains (e.g., faces)

Prof. Leal-Taixé and Prof. Niessner
Mode Collapse

• Performance correlates with dim of manifold

-> Larger latent space, more mode collapse
Problems with Global Structure
Problems with Counting
Evaluation of GAN Performance
Evaluation of GAN Performance

• Main difficulty of GANs: we don't know how good they are

• People cherry pick results in papers -> some of them will always look good, but how to quantify?

• Do we only memorize or do we generalize?

• GANs are difficult to evaluate! [This et al., ICLR 2016]
Evaluation of GAN Performance

Human evaluation:
- Every n updates, show a series of predictions
- Check train curves
- What does ‘look good’ mean at the beginning?
  - Need variety!
  - But don't have ‘realistic’ predictions yet…
- If it doesn't look good? Go back, try different hyperparameters…
Evaluation of GAN Performance

Inception Score (IS)
- Measures saliency and diversity
- Train an accurate classifier
- Train a image generation model (conditional)
- Check how accurate the classifier can recognize the generated images
- Makes some assumptions about data distributions…
Evaluation of GAN Performance

Inception Score (IS)

- Saliency: check whether the generated images can be classified with high confidence (i.e., high scores only on a single class)

- Diversity: check whether we obtain samples from all classes

What if we only have one good image per class?
Evaluation of GAN Performance

• Could also look at discriminator
  – If we end up with a strong discriminator, then generator must also be good
    – Use D features, for classification network
    – Only fine-tune last layer
    – If high class accuracy -> we have a good D and G

Caveat: not sure if people do this... Couldn’t find paper
Next: Making GANs Work in Practice

• Training / Hyperparameters (most important)

• Choice of loss function

• Choice of architecture
GAN Hacks: Normalize Inputs

• Normalize the inputs between -1 and 1

• Tanh as the last layer of the generator output

• No-brainer 😊
GAN Hacks: Sampling

- Use a spherical z
- Don't sample from a uniform distribution
- Sample from a Gaussian Distribution

- When doing interpolations, do the interpolation via a great circle, rather than a straight line from point A to point B

- Tom White's Sampling Generative Networks ref code [https://github.com/dribnet/plat](https://github.com/dribnet/plat) has more details
GAN Hacks: BatchNorm

- Use Batch Norm
- Construct different mini-batches for real and fake, i.e. each mini-batch needs to contain only all real images or all generated images.
GAN Hacks: Use ADAM

• See Adam usage [Radford et al. 15]

• SGD for discriminator

• ADAM for generator
GAN Hacks: One-sided Label Smoothing

• Prevent discriminator from giving too large gradient signal to generator:

\[
J^{(D)} = -\frac{1}{2} \mathbb{E}_{x \sim p_{\text{data}}} \lambda \log D(x) - \frac{1}{2} \mathbb{E}_z \log (1 - D(G(z)))
\]

Some value smaller than 1; e.g., 0.9

-> reduces confidence; i.e., makes disc. ‘weaker’
-> encourages ‘extreme samples’ (prevents extrapolating)
GAN Hacks: Historical Generator Batches

Help stabilize discriminator training in early stage

Srivastava et al. 17 “Learning from Simulated and Unsupervised Images through Adversarial Training”
GAN Hacks: Avoid Sparse Gradients

- Stability of GAN game suffers if gradients are sparse
- LeakyReLU -> good in both G and D
- Downsample -> use average pool, conv+stride
- Upsample -> deconv+stride, PixelShuffle

Exponential Averaging of Weights

• Problem: discriminator is noisy due to SGD

• Rather than taking final result of a GAN, would be biased on last latest iterations (i.e., latest training samples),
  -> exponential average of weights
  -> keep second ‘vector’ of weights that are averaged
  -> almost no cost, average of weights from last n iters
New Objective Functions
New Objective Functions

“heuristic is standard...”

EBGAN: “Energy-based Generative Adversarial Networks”
BEGAN: “Boundary Equilibrium GAN”
WGANGP: “Wasserstein Generative Adversarial Networks”
LSGAN: “Least Squares Generative Adversarial Networks”

... The loss function alone will not make it suddenly work!
GAN Losses: EBGAN

- Discriminator is AE (Energy-based GAN)
- a good autoencoder: we want the reconstruction cost $D(x)$ for real images to be low.
- a good critic: we want to penalize the discriminator if the reconstruction error for generated images drops below a value $m$.

$$D(x) = \|\text{Dec(Enc(x))} - x\|$$

$$\mathcal{L}_D(x, z) = D(x) + [m - D(G(z))]^+$$

$$\mathcal{L}_G(z) = D(G(z))$$

where $[u]^+ = \max(0, u)$
GAN Losses: BEGAN

- Similar to EBGAN
- Instead of reconstruction loss, measure difference in data distribution of real and generated images
GAN Losses: WGAN

- Earth Mover Distance / Wasserstein Distance

Minimum amount of work to move earth from $p(x)$ to $q(x)$

https://medium.com/@jonathan_hui/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490
GAN Losses: WGAN

- Formulate EMD via its dual:

\[
W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta}[f(x)]
\]

\[
|f(x_1) - f(x_2)| \leq |x_1 - x_2|.
\]

1-Lipschitz function: upper bound between densities
GAN Losses: WGAN

\[ |f(x_1) - f(x_2)| \leq |x_1 - x_2|. \]

f is a critic function, defined by a neural network

\[ \rightarrow f \text{ needs to be 1-Lipschitz; WGAN restricts max weight value in } f; \]

weights of the discriminator must be within a certain range controlled by hyperparameters c

\[
\begin{align*}
w & \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w) \\
& \leftarrow \text{clip}(w, -c, c)
\end{align*}
\]

https://medium.com/@jonathan_hui/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490
GAN Losses: WGAN

\[ \nabla_w \left[ \frac{1}{m} \sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m} \sum_{i=1}^{m} f_w(g_\theta(z^{(i)})) \right] \]

\[ -\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} f_w(g_\theta(z^{(i)})) \]
GAN Losses: WGAN

**Discriminator/Critic**

\[ \nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[ \log D(x^{(i)}) + \log (1 - D(G(z^{(i)}))) \right] \]

**Generator**

\[ \nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} -\log (D(G(z^{(i)}))) \]

GAN

\[ \nabla_{w} \frac{1}{m} \sum_{i=1}^{m} \left[ f(x^{(i)}) - f(G(z^{(i)})) \right] \]

WGAN

\[ \nabla_{\theta} \frac{1}{m} \sum_{i=1}^{m} -f(G(z^{(i)})) \]
GAN Losses: WGAN

**Algorithm 1** WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

**Require:** $\alpha$, the learning rate. $c$, the clipping parameter. $m$, the batch size. $n_{\text{critic}}$, the number of iterations of the critic per generator iteration.

**Require:** $w_0$, initial critic parameters. $\theta_0$, initial generator’s parameters.

1: while $\theta$ has not converged do
2:   for $t = 0, ..., n_{\text{critic}}$ do
3:     Sample $\{x^{(i)}\}_{i=1}^m \sim P_r$ a batch from the real data.
4:     Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
5:     $g_w \leftarrow \nabla_w \left[ \frac{1}{m} \sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m} \sum_{i=1}^m f_w(g_\theta(z^{(i)})) \right]$  
6:     $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
7:     $w \leftarrow \text{clip}(w, -c, c)$
8:   end for
9:   Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
10:  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m} \sum_{i=1}^m f_w(g_\theta(z^{(i)}))$
11:  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
12: end while

https://medium.com/@jonathan_hui/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490
GAN Losses: WGAN

\[ \frac{1}{m} \sum_{i=1}^{m} \log \left( 1 - D \left( G \left( z^{(i)} \right) \right) \right) \]

Prof. Leal-Taixé and Prof. Niessner

https://medium.com/@jonathan_hui/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490
GAN Losses: WGAN

\[ \frac{1}{m} \sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m} \sum_{i=1}^{m} f_w(g_\theta(z^{(i)})) \]

Prof. Leal-Taixé and Prof. Niessner

https://medium.com/@jonathan_hui/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490
GAN Losses: WGAN

+ mitigates mode collapse
+ generator still learns when critic performs well
+ actual convergence

- Enforcing Lipschitz constraint is difficult
- Weight clipping is “terrible”
  -> too high: takes long time to reach limit; slow training
  -> too small: vanishing gradients when layers are big
GAN Losses

- Many more variations!!!
- High-level understanding: “loss” is a meta loss to train the actual loss (i.e., D) to provide gradients for G
- Always start simple: if things don’t converge, don’t randomly shuffle loss around; always try easy things first (AE, VAE, ‘simple heuristic’ GAN)
GAN Architectures
Multiscale GANs

Multiscale GANs

Discriminators work at every scale!

Denton et al, NIPS 2015

Prof. Leal-Taixé and Prof. Niessner
Progressive Growing GANs

https://github.com/tkarras/progressive_growing_of_gans [Karras et al. 17]
Latent $4 \times 4$ \rightarrow \mathbf{64} \times \mathbf{64}$ \rightarrow Generated image $64 \times 64$ \rightarrow \mathbf{4} \times \mathbf{4}$ \rightarrow Real or fake

\textbf{G}

\textbf{D}
Latent $\rightarrow$ G $\rightarrow$ D $\rightarrow$ Real or fake

Latent $\rightarrow$ 4×4 $\rightarrow$ 1024×1024 $\rightarrow$ 4×4

Generated image
There’s waves everywhere!

But where’s the shore?
There it is!
Nearest-neighbor upsampling

Replicated block

3x3 convolution
G

1×1 convolution

toRGB

4×4
4×4

2x
8×8
8×8

2x
16×16
16×16

2x
32×32
32×32

81
G

4x4
4x4

2x
8x8
8x8

2x
16x16
16x16

2x
32x32
32x32

toRGB

toRGB
Linear crossfade
fromRGB

G

4x4
4x4

2x
8x8
8x8

toRGB

2x
16x16
16x16

2x
32x32
32x32

32x32
32x32
0.5x

D

16x16
16x16
0.5x

8x8
8x8
0.5x

4x4
4x4
Progressive Growing GANs

CelebA-HQ
1024 × 1024

Latent space interpolations

https://github.com/tkarras/progressive_growing_of_gans [Karras et al. 17]
Lots of GAN Variations

- Hundreds of GAN papers in the last two years
  - > Mostly with different losses
  - > Extremely hard to train and evaluate

Are GANs Created Equal? A Large-Scale Study

Mario Lucic*    Karol Kurach*    Marcin Michalski    Sylvain Gelly    Olivier Bousquet
Google Brain

Abstract

Generative adversarial networks (GAN) are a powerful subclass of generative models. Despite a very rich research GAN algorithm(s) perform objectively better than the others. That's partially due to the lack of robust and consistent metric, as well as limited comparisons which put all algorithms on equal footing, including the computational
Next lectures

• Next Monday 16th, more on Generative models
  – Conditional GANs (cGANs)!

• Keep working on the projects!
Thanks 😊