Deep Learning
lecture 6
Generative Adversarial Network

Yi Wu, IIIS

Spring 2025

Mar-24

Today’s Topic

- Generative Adversarial Network
  - Math
  - Techniques

- Extensions, Variants and Applications

Generative Models (Recap)

- Goal: learn $p(x;\theta)$

- Energy-Based Model
  - $p(x;\theta) = \frac{1}{Z}\exp(-E(x;\theta))$
  - General representation + non-trivial sampling for both training and inference

- Variational Autoencoder (VAE)
  - Latent Variable Model: $p(x,z) = p(z)p(x|z)$
  - Evidence Lower Bound (ELBO) and Variational Inference

$$J(\theta,\phi;x) = E_{z\sim q(z|x;\phi)}[\log p(x|z;\theta)] - KL(q(z|x;\phi)||p(z))$$

Generative Model

- Variational Autoencoder
  - Pros
    - Flexible and stable training
    - Nice mathematical properties (ELBO) for inference
  - Cons
    - Approximate inference
    - Mode collapse and due to KL objective
    - Blurry samples due to Gaussian parameterization
- Core challenge: approximate $p(x)$ for MLE training
- Why do we even care about the density function $p(x)$?
  - We just need a function $g(\cdot)$ that can produce good samples!

Implicit Generative Model

- Goal: a sampler $g(\cdot)$ to generate images
- A simple generator $g(z;\theta)$
  - $z \sim N(0, I)$
  - $x = g(z;\theta)$

- Likelihood-free learning
  - Goal: $g(z;\theta) \approx p_{data}$
  - Idea: minimize $D(g(z;\theta), p_{data})$
    - D is some distance metric
    - D does not involve likelihood

Implicit Generative Model

- Choose a distance measure D
  - Goal: choose $D(x)$ for $x = g(z;\theta)$
  - Intuition: $D(x)$ measures how close x is to training images
  - Natural choice: a discriminative model $p(y|x)$!
    - Objective $L(\theta) = p(y|g(z;\theta))$
  - Learning generative models via discriminative approaches (Zhuowen Tu, CVPR2007)

Deep Learning?

Implicit Generative Model

- Choose a distance measure D
  - Goal: choose a differentiable $D(x;\phi)$ for $g(z;\theta)$
  - Objective $L(\theta) = D(g(z;\theta);\phi)$
    - We can optimize θ by gradient descent
  - How to get ϕ ?
    - Goal: $D(x;\phi)$ how likely x is from p_{data}
    - A binary classification problem!
      - $D(x;\phi) = 1$: $x \sim p_{data}$
      - $D(x;\phi) = 0$: x not from p_{data}
      - Let's train a neural classifier!
        - How to choose the negative samples?
          - Random samples are too easy …
          - We have a generator! (negative samples)

Implicit Generative Model

- Choose a distance measure D
  - Goal: choose a differentiable $D(x;\phi)$ for $g(z;\theta)$
  - Objective $L(\theta) = D(g(z;\theta);\phi)$
  - Train a neural classifier $D(x;\phi)$
    - Binary classification: $D(x;\phi)$ how likely x is from p_{data}
      - $D(x;\phi): x \to [0,1]$ (logistic regression)
      - Training data $\{(x,1)|x\sim p_{data}\} \cup \{(\hat{x},0)|\hat{x}\sim g(z;\theta)\}$
        - $g(z;\theta)$ to sample negative samples
      - MLE learning

$$L(\phi) = E_x[\log D(x;\phi)] + E_{\hat{x}}[\log\big(1 - D(\hat{x};\phi)\big)$$

- Overall objective
  - $\theta^\star = \max_\theta D(g(z;\theta);\phi)$
  - $\phi^\star = \max_\phi E_{x\sim p_{data}}[\log D(x;\phi)] + E_{\hat{x}\sim g}[\log(1 - D(\hat{x};\phi))]$

Implicit Generative Model

- Choose a distance measure D
  - Goal: choose a differentiable $D(x;\phi)$ for $g(z;\theta)$
  - Objective $L(\theta) = D(g(z;\theta);\phi)$
  - Train a neural classifier $D(x;\phi)$
    - Binary classification: $D(x;\phi)$ how likely x is from p_{data}
      - $D(x;\phi): x \to [0,1]$ (logistic regression)
      - Training data $\{(x,1)|x\sim p_{data}\} \cup \{(\hat{x},0)|\hat{x}\sim g(z;\theta)\}$
        - $g(z;\theta)$ to sample negative samples
      - MLE learning

$$L(\phi) = E_x[\log D(x;\phi)] + E_{\hat{x}}[\log\big(1 - D(\hat{x};\phi)\big)$$

- Overall objective
  - $\theta^\star = \max_\theta D(g(z;\theta);\phi)$
  - $\phi^\star = \max_\phi E_{x\sim p_{data}}[\log D(x;\phi)] + E_{\hat{x}\sim g}[\log(1 - D(\hat{x};\phi))]$

Implicit Generative Model

- Choose a distance measure D
  - Goal: choose a differentiable $D(x;\phi)$ for $g(z;\theta)$
  - Objective $L(\theta) = D(g(z;\theta);\phi)$
  - Train a neural classifier $D(x;\phi)$
    - Binary classification: $D(x;\phi)$ how likely x is from p_{data}
      - $D(x;\phi): x \to [0,1]$ (logistic regression)
      - Training data $\{(x,1)|x\sim p_{data}\} \cup \{(\hat{x},0)|\hat{x}\sim g(z;\theta)\}$
        - $g(z;\theta)$ to sample negative samples
      - MLE learning

$$L(\phi) = E_x[\log D(x;\phi)] + E_{\hat{x}}[\log\big(1 - D(\hat{x};\phi)\big)$$

- Overall objective
  - $\theta^\star = \max_\theta D(g(z;\theta);\phi) = \min_\theta 1 - D(g(z;\theta);\phi)$
  - $\phi^\star = \max_\phi E_{x\sim p_{data}}[\log D(x;\phi)] + E_{\hat{x}\sim g}[\log(1 - D(\hat{x};\phi))]$

Generative Adversarial Network

- GAN (Ian Goodfellow et al, NIPS 2014)
  - 79k citations, NeurIPS 2024 test-of-time award
  - Generator $G(z;\theta)$ ($z \sim p(z) = N(0, I)$)
    - generate realistic images
  - Discriminator $D(x;\phi)$
    - Classify the data is from p_{data} or G
  - Objective

$$L(\theta,\phi) = \min_{\theta} \max_{\phi} E_{x\sim p_{data}}[\log D(x;\phi)] + E_{\hat{x}\sim G}[\log(1 - D(\hat{x};\phi))]$$

  - Training procedure
    - Collect dataset $\{(x,1)|x\sim p_{data}\} \cup \{(\hat{x},0)|\hat{x}\sim g(z;\theta)\}$
    - Train discriminator D: $L(\phi) = E_{x\sim p_{data}}[\log D(x;\phi)] + E_{\hat{x}\sim G}[\log(1 - D(\hat{x};\phi))]$
    - Train generator G: $L(\phi) = E_{z\sim p(z)}[\log D(G(z;\theta))]$
    - Repeat

Generative Adversarial Nets

Ian J. Goodfellow, Jean Pouget-Abadie,* Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair,† Aaron Courville, Yoshua Bengio‡
Département d'informatique et de recherche opérationnelle
Université de Montréal
Montréal, QC H3C 3J7

Generative Adversarial Network

- GAN (Ian Goodfellow et al, NIPS 2014)
  - Generator $G(z;\theta)$ & Discriminator $D(x;\phi)$

$$L(\theta,\phi) = \min_{\theta}\max_{\phi} E_{x\sim p_{data}}[\log D(x;\phi)] + E_{\hat{x}\sim G}[\log(1 - D(\hat{x};\phi))]$$

G

• G

a)

b)

c)

d)

(d)

End of Class?

- No, this is far from the end of story ☺

- Let’s begin with the math behind GAN

Generative Adversarial Network

- GAN Objective

$$L(\theta, \phi) = \min_{\theta} \max_{\phi} E_{x \sim p_{data}}[\log D(x; \phi)] + E_{\hat{x} \sim G}[\log(1 - D(\hat{x}; \phi))]$$

  - Let's Analyze the optimal solution $D^\star$ and $G^\star$ under $L(\theta, \phi)$
  - Optimal $D(x; \phi^\star)$ for x

$$L(D) = p_{data}(x) \cdot \log D + p_G(x) \log(1 - D)$$

    - Consider $\frac{\partial L}{\partial D} = 0$
      - We have $\frac{p_{data}(x)}{D^\star} - \frac{p_G(x)}{1 - D^\star} = 0$
      - So $(1 - D^\star) p_{data}(x) = P_G(x) D^\star$
      - $D^\star = \frac{p_{data}(x)}{p_{data(x)} + p_G(x)}$
    - Remark: when having a perfect generator, $D^\star = 0.5$

Generative Adversarial Network

- GAN Objective

$$L(\theta, \phi) = \min_{\theta} \max_{\phi} E_{x \sim p_{data}}[\log D(x; \phi)] + E_{\hat{x} \sim G}[\log(1 - D(\hat{x}; \phi))]$$

  - Optimal discriminator $D(x; \phi^{\star}) = p_{data}(x)/(p_{data}(x) + p_G(x))$
  - Optimal generator $G(z; \theta)$ with $\phi^{\star}$
    - $L(\theta, \phi) = E_{x \sim p_{data}}\left[\log \frac{p_{data}(x)}{p_{data}(x) + p_G(x)}\right] + E_{x \sim p_G}\left[\log \frac{p_G(x)}{p_{data}(x) + p_G(x)}\right]$

Generative Adversarial Network

- GAN Objective

$$L(\theta, \phi) = \min_{\theta} \max_{\phi} E_{x \sim p_{data}}[\log D(x; \phi)] + E_{\hat{x} \sim G}[\log(1 - D(\hat{x}; \phi))]$$

  - Optimal discriminator $D(x; \phi^{\star}) = p_{data}(x)/(p_{data}(x) + p_G(x))$
  - Optimal generator $G(z; \theta)$ with $\phi^{\star}$
    - $L(\theta, \phi) = E_{x \sim p_{data}}\left[\log \frac{p_{data}(x)}{p_{data}(x)+p_G(x)}\right] + E_{x \sim p_G}\left[\log \frac{p_G(x)}{p_{data}(x)+p_G(x)}\right]$
    - $= E_{x \sim p_{data}}\left[\log \frac{p_{data}(x)}{\frac{p_{data}(x)+p_G(x)}{2}}\right] + E_{x \sim p_G}\left[\log \frac{p_G(x)}{\frac{p_{data}(x)+p_G(x)}{2}}\right] - \log 4$

Generative Adversarial Network

- GAN Objective

$$L(\theta,\phi) = \min_{\theta}\max_{\phi} E_{x\sim p_{data}}[\log D(x;\phi)] + E_{\hat{x}\sim G}[\log(1 - D(\hat{x};\phi))]$$

  - Optimal discriminator $D(x;\phi^{\star}) = p_{data}(x)/(p_{data}(x) + p_G(x))$
  - Optimal generator $G(z;\theta)$ with $\phi^{\star}$
    - $L(\theta,\phi) = E_{x\sim p_{data}}\left[\log\frac{p_{data}(x)}{p_{data}(x)+p_G(x)}\right] + E_{x\sim p_G}\left[\log\frac{p_G(x)}{p_{data}(x)+p_G(x)}\right]$
    - $= E_{x\sim p_{data}}\left[\log\frac{p_{data}(x)}{\frac{p_{data}(x)+p_G(x)}{2}}\right] + E_{x\sim p_G}\left[\log\frac{p_G(x)}{\frac{p_{data}(x)+p_G(x)}{2}}\right] - \log 4$
    - $= KL\left(p_{data}(x)\,||\,\frac{1}{2}(p_{data} + p_G)\right) + KL\left(p_G\,||\,\frac{1}{2}(p_{data} + p_G)\right) - \log 4$

Generative Adversarial Network

- GAN Objective

$$L(\theta,\phi) = \min_{\theta}\max_{\phi} E_{x\sim p_{data}}[\log D(x;\phi)] + E_{\hat{x}\sim G}[\log(1 - D(\hat{x};\phi))]$$

  - Optimal discriminator $D(x;\phi^{\star}) = p_{data}(x)/(p_{data}(x) + p_G(x))$
  - Optimal generator $G(z;\theta)$ with $\phi^{\star}$
    - $L(\theta,\phi) = E_{x\sim p_{data}}\left[\log\frac{p_{data}(x)}{p_{data}(x)+p_G(x)}\right] + E_{x\sim p_G}\left[\log\frac{p_G(x)}{p_{data}(x)+p_G(x)}\right]$
    - $= E_{x\sim p_{data}}\left[\log\frac{p_{data}(x)}{\frac{p_{data}(x)+p_G(x)}{2}}\right] + E_{x\sim p_G}\left[\log\frac{p_G(x)}{\frac{p_{data}(x)+p_G(x)}{2}}\right] - \log 4$
    - $= KL\left(p_{data}(x)\|\frac{1}{2}(p_{data} + p_G)\right) + KL\left(p_G\|\frac{1}{2}(p_{data} + p_G)\right) - \log 4$
    - 2*Jenson-Shannon Divergence (JSD)

Generative Adversarial Network

- Kullback–Leibler Divergence (Recap)
  - $KL(p||q) = E_{x\sim p}[\log p(x)/q(x)]$
  - Asymmetric measure
    - $KL(p||q)$ forward KL (inclusive)
    - $KL(q||p)$ reverse KL (exclusive)
- Jensen-Shannon Divergence (JSD)
  - $JSD(p||q) = \frac{1}{2}\left(KL\left(p||\frac{1}{2}(p+q)\right) + KL\left(q||\frac{1}{2}(p+q)\right)\right)$
  - Properties
    - Symmetric: $JSD(p||q) = JSD(q||p)$
    - $JSD(p||q) \geq 0$ and $JSD(p||q) = 0$ iff $p = q$
    - Jenson-Shannon distance: $\sqrt{JSD(p||q)}$ satisfies triangular inequality

Generative Adversarial Network

- GAN Objective

$$L(\theta, \phi) = \min_{\theta} \max_{\phi} E_{x \sim p_{data}}[\log D(x; \phi)] + E_{\hat{x} \sim G}[\log(1 - D(\hat{x}; \phi))]$$

  - Optimal discriminator $D(x; \phi^{\star}) = p_{data}(x)/(p_{data}(x) + p_G(x))$
  - Loss function with optimal discriminator

$$L(\theta) = 2JSD(p_G || p_{data}) - \log 4$$

  - Global optimum
    - $G^{\star} = p_{data}$
    - $L^{\star} = -\log 4$

Generative Adversarial Network

- Pros of GAN
  - Likelihood-free learning
  - Focus on generation
  - Powerful loss function (neural net!)
  - Flexible mathematical framework
  - Connection to EBM, Game Theory (Minimax game between G and D) and even RL (connection to actor-critic method)
    - Further reading: https://arxiv.org/abs/1611.03852, https://arxiv.org/abs/1610.01945
- There is no free lunch!
- Cons
  - evaluation; sample diversity & training instability

Generative Adversarial Network

- Evaluation of GAN
  - No $p(x)$ at all!
    - Not possible for any likelihood-based evaluation
  - Idea: use a trained neural classifier $f(y|x)$
    - If $x \sim p_{data}$, $f(y|x)$ should have low entropy
      - Otherwise, $f(y|x)$ should be close to uniform
    - Samples from G should also vary!
      - $p_f(y) = E_{x \sim G}[f(y|x)]$ should be close to uniform

Generative Adversarial Network

- Evaluation of GAN
  - No $p(x)$ at all!
    - Not possible for any likelihood-based evaluation
  - Idea: use a trained neural classifier $f(y|x)$
    - If $x \sim p_{data}$, $f(y|x)$ should have low entropy
      - Otherwise, $f(y|x)$ should be close to uniform
    - Samples from G should also vary!
      - $p_f(y) = E_{x \sim G}[f(y|x)]$ should be close to uniform

Let's combine them!

Generative Adversarial Network

- Evaluation of GAN
  - Idea: use a trained neural classifier f
    - If $x \sim p_{data}, f(y|x)$ should have low entropy
    - Samples should also vary: $p_f(y)$ should be close to uniform
  - Inception Score (IS, Salimans et al, 2016)
    - Use Inception v3 trained on ImageNet as $f(y|x)$
    - $IS = \exp\left(E_{x\sim G}\left[KL\left(f(y|x)||p_f(y)\right)\right]\right)$ *(higher the better)*

Generative Adversarial Network

- Evaluation of GAN
  - Idea: use a trained neural classifier f
    - If $x \sim p_{data}, f(y|x)$ should have low entropy
    - Samples should also vary: $p_f(y)$ should be close to uniform
  - Inception Score (IS, Salimans et al, 2016)
    - Use Inception v3 trained on ImageNet as $f(y|x)$
    - $IS = \exp\left(E_{x\sim G}\left[KL\left(f(y|x)||p_f(y)\right)\right]\right)$
  - Issue of IS?
    - No p_{data} involved in IS!
    - What if G memorizes a single image per label y from p_{data}?
      - Additional reading: https://arxiv.org/abs/1801.01973

Generative Adversarial Network

- Evaluation of GAN
  - Inception Score (IS, Salimans et al, 2016)
    - $IS = \exp\left(E_{x\sim G}\left[KL\left(f(y|x)||p_f(y)\right)\right]\right)$
  - IS only measures the quality of generated samples
    - We also want G to fully cover p_{data}
    - Idea: statistics of $G(z)$ should be similar to p_{data}
      - Statistics: measure the distribution of extracted features of p_{data}
  - Fréchet Inception Distance (FID, Heusel et al, NIPS 2017)
    - Compute μ_p, Σ_p and μ_G, Σ_G for p_{data} and $G(z)$ using inception v3 pool3 layer (2048-d)
    - Compute Wasserstein distance between two Gaussians (more on this later)
    - $FID = \left|\mu_p - \mu_G\right|^2 + trace\left(\Sigma_p + \Sigma_G - 2\left(\Sigma_p\Sigma_G\right)^{1/2}\right)$
      - *Lower the better*
      - *https://arxiv.org/abs/1706.08500*

Note: IND is a "distance version" of inception score. Details can be found at the FID paper's Appendix A.1

Note: IND is a "distance version" of inception score. Details can be found at the FID paper's Appendix A.1

Generative Adversarial Network

- Sample Diversity
  - GAN suffer from the mode collapse issue
    - The generator converges to a few samples

Arjovsky et al., 2017

Generative Adversarial Network

- Sample Diversity
  - GAN suffer from the mode collapse issue
    - The generator converges to a few samples
    - Or keep oscillating over a few modes

Source: Metz et al., 2017

Generative Adversarial Network

- Sample Diversity
  - GAN suffer from the mode collapse issue
    - The generator converges to a few samples
    - Or keep oscillating over a few modes
  - This is a cheating strategy for $G(z)$
    - No fundamental solution for this. ☹
  - Intrinsic issue of JSD

Figure 1: An isotropic Gaussian distribution was fit to data drawn from a mixture of Gaussians by either minimizing Kullback-Leibler divergence (KLD), maximum mean discrepancy (MMD), or Jensen-Shannon divergence (JSD). The different fits demonstrate different tradeoffs made by the three measures of distance between distributions.

["A note on the evaluation of generative models" -- Theis, Van den Oord, Bethge 2015]

Generative Adversarial Network

- Training Instability
  - Discriminator and generator may keep oscillating

Source: Mirantha Jayathilaka

Generative Adversarial Network

- Training Instability
  - Discriminator and generator may keep oscillating
    - Simple gradient decent on the minimax game may not converge at all!
    - Intuitive example: $J = -xy$
      - Generator: x; discriminator: y
      - Nash Equilibrium: $x = y = 0$
      - Gradient decent converges to the orbit

Figure 3: A simulation of our example for updating x to minimize xy and updating y to minimize $-xy$. The learning rate $\eta = 0.1$. With more iterations, the oscillation grows more and more unstable.

Generative Adversarial Network

- Training Instability
  - Discriminator and generator may keep oscillating
    - Simple gradient decent on the minimax game may not converge at all!
    - Intuitive example: $J = -xy$
      - Generator: x; discriminator: y
      - Nash Equilibrium: $x = y = 0$
      - Gradient decent converges to the orbit
    - No stopping criteria!
      - Unlike MLE learning
      - Sometimes GAN may suffer from immediate crashing!
    - Additional reading:
      - GANs May Have No Nash Equilibria (Farnia et al, ICML2020)
      - https://arxiv.org/abs/2002.09124

(a) G

(b) D

Generative Adversarial Network

- Training Instability
  - Discriminator and generator may keep oscillating
  - Gradient Vanishing Issue
    - Generator: $z \rightarrow x$, low-d to high-d
      - The intrinsic dimensionality of G is small
    - Discriminator: $x \rightarrow z$, a hyper-plan in the high-d
      - It is almost impossible for two low-dimensional manifold to fully overlap in high-d
      - D can always achieve 100%

Figure 4: Low dimensional manifolds in high dimension space can hardly have overlaps. (Left) Two lines in a three-dimension space. (Right) Two surfaces in a three-dimension space.

Generative Adversarial Network

- Tra
  -
  -

eep

mall

the l

imen

ps. (Left) Two

- When D is overly confident
  - Gradient vanishes for Sigmoid output
  - Discriminator cannot learn too fast!

Generative Adversarial Network

- A lot of issues in GAN
  - Evaluation
    - IS & FID
  - Sample Diversity
    - Mode collapse
    - No fundamental solution
  - Training Instability
    - Tricky balancing between G and D

- Let's make it work!
  - … and a lot of tricks are coming!

GAN Techniques

- Deep Convolutional GAN (DCGAN, Radford et al, ICLR2016)
  - The first milestone paper to make GAN really work
  - Trick Suggestions
    - Use fully convolutional network
      - No pooling or MLP layer
      - Supervised CNNs cannot be directly used as discriminators

GAN Techniques

- Deep Convolutional GAN (DCGAN, Radford et al, ICLR2016)
  - The first milestone paper to make GAN really work
  - Trick Suggestions
    - Use fully convolutional network
    - Batch normalization should be used
      - To stabilize training
      - But NO batchnorm for output of G or input of D
      - Separate batchnorm for p_{data} and G

GAN Techniques

- Deep Convolutional GAN (DCGAN, Radford et al, ICLR2016)
  - The first milestone paper to make GAN really work
  - Trick Suggestions
    - Use fully convolutional network
    - Batch normalization should be used
    - Avoid ReLU activation in discriminator
      - Avoid gradient vanishing
      - ReLU in generator except output layer, which should use tanh
      - Leaky ReLU in discriminator (slope = 0.2)

GAN Techniques

- Deep Convolutional GAN (DCGAN, Radford et al, ICLR2016)
  - The first milestone paper to make GAN really work
  - Trick Suggestions
    - Use fully convolutional network
    - Batch normalization should be used
    - Avoid ReLU activation in discriminator
    - Small learning rate and momentum
      - Adam with learning rate $\eta = 0.0002$
      - Momentum $\beta = 0.5$
      - Small batch size = 128 (which helps prevent memorizing the training data)
  - DCGAN can learn interesting features and patterns

GAN Tec

- Deep Convo .6)
  - The first m
  - Trick Sugg
    - Use full
    - Batch n
    - Avoid R
    - Small le
  - DCGAN ca

GAN Tec

- Deep Conv 5)
  - The first r
  - Trick Sugg
    - Use ful
    - Batch r
    - Avoid F
    - Small l
  - DCGAN ca

GAN Techniques

- Deep Convo)
  - The first m
  - Trick Sugg
    - Use full
    - Batch n
    - Avoid R
    - Small le
  - DCGAN ca

Figure 5: On the right, guided backpropagation visualizations of maximal axis-aligned responses for the first 6 learned convolutional features from the last convolution layer in the discriminator. Notice a significant minority of features respond to beds - the central object in the LSUN bedrooms dataset. On the left is a random filter baseline. Comparing to the previous responses there is little to no discrimination and random structure.

GAN Techniques

Figure 6: Top row: un-modified samples from model. Bottom row: the same samples generated with dropping out "window" filters. Some windows are removed, others are transformed into objects with similar visual appearance such as doors and mirrors. Although visual quality decreased, overall scene composition stayed similar, suggesting the generator has done a good job disentangling scene representation from object representation. Extended experiments could be done to remove other objects from the image and modify the objects the generator draws.

GAN Techniques

- Deep Convolutional GAN (DCGAN, Radford et al, ICLR2016)
  - The first milestone paper to make GAN really work
  - Trick Suggestions
    - Use fully convolutional network
    - Batch normalization should be used
    - Avoid ReLU activation in discriminator
    - Small learning rate and momentum
  - DCGAN can learn interesting features and patterns

GAN Techniques

- Improved Training Techniques for GAN (Salimans et al, NIPS2016)
  - The paper primarily about GAN tricks
    - also IS and semi-supervised learning; more to cover later
  - Trick Suggestions
    - Feature matching
      - Directly optimizing $D(G(z))$ for generator may suffer from gradient vanishing
      - Idea: $G(z)$ should match the image statistics for p_{data}
      - Features are more informative than the final score

$$L(\theta) = \left|E_{\hat{x}\sim G}[f(\hat{x};\phi)] - E_{x\sim p_{data}}[f(x;\phi)]\right|^2$$

      - $f(x)$ is the final activation layer in D

GAN Techniques

- Improved Training Techniques for GAN (Salimans et al, NIPS2016)
  - The paper primarily about GAN tricks
    - also IS, and semi-supervised learning, more to cover later
  - Trick Suggestions
    - Feature matching
    - Minibatch discrimination
      - Avoid mode collapse
        - if G has collapsed mode, then samples of G will cluster to each other in the minibatch
      - Idea: compute clustering feature as additional side information per mini-batch
        - Similar to batchnorm
      - Additional feature $o(x^i)$
        - $M = f(x^i) \times T$ (T: projection tensor)
        - $c(x^i, x^j)_b = \exp(|M_{i,b} - M_{j,b}|_1)$
        - $o(x^i)_b = \sum_j c(x^i, x^j)_b$

GAN Techniques

- Improved Training Techniques for GAN (Salimans et al, NIPS2016)
  - The paper primarily about GAN tricks
    - also IS, and semi-supervised learning, more to cover later
  - Trick Suggestions
    - Feature matching
    - Minibatch discrimination
    - Historical averaging
      - Include term $\left|\theta - \frac{1}{T}\sum_{t=1}^{T}\theta\right|^2$ for averaging historical weights
        - Inspired by the fictitious self-play algorithm in game theory
      - Exponential weight averaging for efficient computation
      - Similar to how DQN stabilizes training
        - Most stabilizing tricks in RL works here as well

GAN Techniques

- Improved Training Techniques for GAN (Salimans et al, NIPS2016)
  - The paper primarily about GAN tricks
    - also IS, and semi-supervised learning, more to cover later
  - Trick Suggestions
    - Feature matching
    - Minibatch discrimination
    - Historical averaging
    - One-sided label smoothing
      - Change positive label to 0.9

Default discriminator cost:

```
cross_entropy(1., discriminator(data))
+ cross_entropy(0., discriminator(samples))
```

One-sided label smoothed cost (Salimans et al 2016):

```
cross_entropy(.9, discriminator(data))
+ cross_entropy(0., discriminator(samples))
```

GAN Techniques

- Improved Training Techniques for GAN (Salimans et al, NIPS2016)
  - The paper primarily about GAN tricks
    - also IS, and semi-supervised learning, more to cover later
  - Trick Suggestions
    - Feature matching
    - Minibatch discrimination
    - Historical averaging
    - One-sided label smoothing
      - Change positive label to 0.9
      - Why only one-sided?
        - Data label (positive) α, sample label (negative) β
        - Optimal classifier $D^\star = \frac{\alpha \cdot p_{data}(x) + \beta \cdot p_G(x)}{p_{data}(x) + p_G(x)}$
        - If $\beta > 0$, then for x with $p_{data}(x) = 0$, G has no incentive to move $p_G(x)$ to 0

GAN Techniques

- Improved Training Techniques for GAN (Salimans et al, NIPS2016)
  - The paper primarily about GAN tricks
    - also IS, and semi-supervised learning, more
  - Trick Suggestions
    - Feature matching
    - Minibatch discrimination
    - Historical averaging
    - One-sided label smoothing
    - Virtual batch normalization
      - Issue of batch norm in DCGAN
        - Output of $G(x)$ can be highly dependent on each other
      - Idea: select a fixed batch B to compute batch statistics
        - Only applied when training generator G for computation purpose
        - Practical: combine B and current batch to avoid overfitting

GAN Techniques

- Improved Training Techniques for GAN (Salimans et al, NIPS2016)
  - The paper primarily about GAN tricks
    - also IS, and semi-supervised learning, more to cover later
  - Trick Suggestions
    - Feature matching
    - Minibatch discrimination
    - Historical averaging
    - One-sided label smoothing
    - Virtual batch normalization
  - And more!
    - Check https://github.com/soumith/ganhacks and more on Google
    - "*Keep calm and train a GAN*" ☺

GAN Techniques

- Some mathematical technique
  - $L(\theta) = 2JSD(p_G || p_{data}) - \log 4$ with $\phi^\star$
  - Issue of JSD:
    - Gradient vanishing when p_{data} and p_G are fully disjoint
    - Mode collapse
  - We need a better metric!
  - Idea: Wasserstein distance
    - It is also called *Earth Mover's distance*

$$W(P,Q) = \inf_{\gamma \sim \Pi(P,Q)} E_{(x,y)\sim\gamma}[|x-y|]$$

    - Intuition:
      - Two distribution of dirt
      - We want to move the dirt to change P to Q
      - $W(P,Q)$: the minimum energy consumed to transport dirt from P to Q

GAN Techniques

- Wasserstein Distance (*Earth Mover's distance*)

$$W(P,Q) = \inf_{\gamma \sim \Pi(P,Q)} E_{(x,y)\sim\gamma}[|x-y|]$$

  - Examples: discrete case
    - $P = \frac{1}{Z}\text{Categorical}(3,2,1,4)$
    - $Q = \frac{1}{Z}\text{Categorical}(1,2,4,3)$

GAN Techniques

- Wasserstein Distance (*Earth Mover's distance*)

$$W(P,Q) = \inf_{\gamma \sim \Pi(P,Q)} E_{(x,y)\sim\gamma}[|x-y|]$$

  - Examples: discrete case
    - $P = \frac{1}{Z}\text{Categorical}(3,2,1,4)$
    - $Q = \frac{1}{Z}\text{Categorical}(1,2,4,3)$
    - Let's change P to Q!
      - $\delta = 2 + 2 + |-1| = 5$
      - So $W(P,Q) = \frac{1}{Z} \cdot 5$

  - Continuous case
    - $W(P,Q) = \inf_{\gamma \sim \Pi(P,Q)} \sum_{(x,y)\sim\gamma} p(x,y)|x-y|$
    - $p(x,y)$ the amount of dirty transported from x to y, $|x-y|$ is the cost of the transport

GAN Techniques

- Wasserstein Distance (*Earth Mover's distance*)

$$W(P,Q) = \inf_{\gamma \sim \Pi(P,Q)} E_{(x,y)\sim\gamma}[|x-y|]$$

  - Compare with JSD
    - $P: x = 0, 0 \le y \le 1$, uniform
    - $Q: x = \theta, 0 \le y \le 1$, uniform
    - Assume $\theta \neq 0$
      - $KL(P||Q) = KL(Q||P) = \infty$
      - $JSD(P,Q) = \frac{1}{2}\left(\sum_{x,y} \log\left(\frac{1}{\frac{1}{2}}\right) + \sum_{x,y} \log\left(\frac{1}{\frac{1}{2}}\right)\right) = \log 2$
      - $W(P,Q) = |\theta|$
  - Wasserstein Distance is a smooth measure!

- Let's make Wasserstein Distance as a GAN objective!

Wasserstein GAN

- WGAN (Arjovsky et al, 18.5k citations, ICML2017)
  - Goal: $W(p_{data}, p_G)$
    - Intractable
  - Kantorovich Rubinstein Duality
    - $W(p_{data}, p_G) = \frac{1}{K} \sup_{|f|_L \leq K} E_{x \sim p_{data}}[f(x)] - E_{x \sim p_G}[f(x)]$
    - $|f|_L \leq K$: f is K-Lipschitz continuous
      - $|f(x) - f(y)| \leq K|x - y|$
    - Search over all K-continuous functions!
  - WGAN objective $(K = 1)$
    - $f(x; \phi)$ is called a critic
    - $L(\phi) = E_{x \sim p_{data}}[f(x; \phi)] - E_{x \sim p_G}[f(x; \phi)]$
    - $L(\theta) = E_{x \sim G(z;\theta)}[f(x; \phi)]$
    - Subject to $|f(x; \phi)|_L \leq 1$

Wasserstein GAN

- WGAN (Arjovsky et al, ICML2017)
  - Goal: $W(p_{data}, p_G)$
    - Intractable
  - Kantorovich Rubinstein Duality
    - $W(p_{data}, p_G) = \frac{1}{K} \sup_{|f|_L \leq K} E_{x \sim p_{data}}[f(x)] - E_{x \sim p_G}[f(x)]$
    - $|f|_L \leq K$: f is K-Lipschitz continuous
      - $|f(x) - f(y)| \leq K|x - y|$
    - Search over all K-continuous functions!
  - WGAN objective $(K = 1)$
    - $f(x; \phi)$ is called a critic
    - $L(\phi) = E_{x \sim p_{data}}[f(x; \phi)] - E_{x \sim p_G}[f(x; \phi)]$
    - $L(\theta) = E_{x \sim G(z;\theta)}[f(x; \phi)]$
    - Subject to $|f(x; \phi)|_L \leq 1$

Wasserstein GAN

- WGAN (Arjovsky et al, ICML2017)
  - Objective
    - $L(\phi) = E_{x \sim p_{data}}[f(x;\phi)] - E_{x \sim p_G}[f(x;\phi)]$
    - $L(\theta) = E_{x \sim G(z;\theta)}[f(x;\phi)]$
    - Subject to $|f(x;\phi)|_L \leq 1$
  - Tricks to enforce $|f(x;\phi)|_L \leq 1$
    - For each ϕ_i, $\phi_i \leftarrow \mathrm{clip}(\phi_i, -c, c)$ (e.g., with $c = 0.01$)
    - No momentum! RMSProp suggested ($\alpha = 5e - 5$)
    - Update critic for n_{critic} batches before update G ($n_{critic} = 5$)
  - Results

Wasserstein GAN

- WGAN (Arjovsky et al, ICML2017)

Wasserstein GAN

- WGAN (Arjovsky et al, ICML2017)

Top: WGAN with the same DCGAN architecture. *Bottom:* DCGAN

Top: WGAN with DCGAN architecture, no batch norm. *Bottom:* DCGAN, no batch norm.

Wasserstein GAN

- Summary of WGAN (Arjovsky et al, ICML2017)
  - Original GAN objective

$$\min_G \max_D E_{x\sim p_{data}}[\log D(x)] + E_{\hat{x}\sim p_G}[\log(1 - D(\hat{x}))]$$

  - Wasserstein GAN objective

$$\min_G \max_{D:|D|_L\leq 1} E_{x\sim p_{data}}[D(x)] - E_{\hat{x}\sim p_G}[D(\hat{x})]$$

  - Pros
    - Address instability and gradient vanishing due to JSD, also help mode collapse
    - Robust to architecture choice
    - Introduce the idea of Lipschitzness for stabilizing GAN training
  - Cons
    - Cited from the paper

Weight clipping is a clearly terrible way to enforce a Lipschitz constraint. If the clipping parameter is large, then it can take a long time for any weights to reach their limit, thereby making it harder to train the critic till optimality. If the clipping is small, this can easily lead to vanishing gradients when the number of layers is big, or batch normalization is not used (such as in RNNs). We experimented with simple variants (such as projecting the weights to a sphere) with little difference, and we stuck with weight clipping due to its simplicity and already good performance.

Wasserstein GAN

- Improved Training of Wasserstein GANs (Gulrajani et al, NIPS2017)
  - Wasserstein GAN objective

$$\min_{G} \max_{f:|f|_L \leq 1} E_{x \sim p_{data}}[f(x)] - E_{\hat{x} \sim p_G}[f(\hat{x})]$$

  - Corollary: $f^\star$ have $|\nabla f| = 1$ almost everywhere under p_{data} and p_G
  - WGAN-GP (Gradient Penalty)
    - $\tilde{x} \leftarrow (1-\epsilon)x + \epsilon \cdot \hat{x}$ with $\epsilon \sim \mathrm{Unif}\,(0,1)$
    - $L(\phi) = E_{x \sim p_{data}}[f(x;\phi)] - E_{\hat{x} \sim p_G}[f(\hat{x})] + \lambda \cdot E_{\tilde{x}}[(|\nabla_{\tilde{x}} f(\tilde{x};\phi)|^2 - 1)^2]$

Wasserstein GAN

- Improved Training of Wasserstein GANs (Gulrajani et al, NIPS2017)
  - Wasserstein GAN objective

$$\min_{G} \max_{f:|f|_L \le 1} E_{x\sim p_{data}}[f(x)] - E_{\hat{x}\sim p_G}[f(\hat{x})]$$

  - Corollary: $f^\star$ have $|\nabla f| = 1$ almost everywhere under p_{data} and p_G
  - WGAN-GP (Gradient Penalty)
    - $\tilde{x} \leftarrow (1-\epsilon)x + \epsilon \cdot \hat{x}$ with $\epsilon \sim \mathrm{Unif}\,(0,1)$
    - $L(\phi) = E_{x\sim p_{data}}[f(x;\phi)] - E_{\hat{x}\sim p_G}[f(\hat{x})] + \lambda \cdot E_{\tilde{x}}[(|\nabla_{\tilde{x}} f(\tilde{x};\phi)|^2 - 1)^2]$

Wasserstein GAN

- Improved Training of Wasserstein GANs (Gulrajani et al, NIPS2017)
  - Wasserstein GAN objective

$$\min_{G} \max_{f:|f|_L \leq 1} E_{x \sim p_{data}}[f(x)] - E_{\hat{x} \sim p_G}[f(\hat{x})]$$

  - Corollary: $f^\star$ have $|\nabla f| = 1$ almost everywhere under p_{data} and p_G
  - WGAN-GP (Gradient Penalty)
    - $\tilde{x} \leftarrow (1-\epsilon)x + \epsilon \cdot \hat{x}$ with $\epsilon \sim \mathrm{Unif}\,(0,1)$
    - $L(\phi) = E_{x \sim p_{data}}[f(x;\phi)] - E_{\hat{x} \sim p_G}[f(\hat{x})] + \lambda \cdot E_{\tilde{x}}[(|\nabla_{\tilde{x}} f(\tilde{x};\phi)|^2 - 1)^2]$
  - Practical issue
    - No batch norm for $f(x;\phi)$ (since we compute $\nabla_{\tilde{x}} f(\tilde{x})$ for each $\tilde{x}$)
    - Layer norm or instance norm recommended as a drop-in replacement
  - Remark:
    - Stable training with various architecture (even ResNet) and Adam
    - But expensive due to gradient computation over gradient
    - Also might be unstable due to the heuristic distribution $\tilde{x}$ when learning rate is high

W

• In

)

DCGAN
LSGAN
WGAN (clipping)
WGAN-GP (ours)
Baseline (G: DCGAN, D: DCGAN)
G: No BN and a constant number of filters, D: DCGAN
G: 4-layer 512-dim ReLU MLP, D: DCGAN
No normalization in either G or D
Gated multiplicative nonlinearities everywhere in G and D
tanh nonlinearities everywhere in G and D
101-layer ResNet G and D

Wassers

- Improved T PS2017)
  - Wasserst
  - Corollary d p_G
  - WGAN-G
    - $\tilde{x} \leftarrow ($
    - $L(\phi)$]
  - Practical
    - No ba
    - Layer
  - Remark:
    - Stable
    - But ex
    - Also m te is high

GAN Techniques

- BigGAN (DeepMind, ICLR2019)
  - Large-Scale Training of GAN!
    - ~100M params
    - TPU training (48h)
    - Large batch size

  - Trick Suggestions
    - Synced cross-replica class-conditioned batch-norm (linear projected from class id)
    - Large batch size (as much as you can) and wider model!
    - Sample z from truncated Gaussian $N(0,1,-c,c)$
    - Orthogonal initialization & spectral normalization for weights
    - Hinge loss $l(z) = \max(0, 1 - t * z)$:
      - Ignore samples when D make a correct output with high confidence (z is too high)
      - Adaptive hinge loss margin t to include sufficient training data

GAN Techniques

)

oned batch-norm (linear projected from class id)

can) and wider model!

n $N(0,1,-c,c)$

al normalization for weights

correct output with high confidence

clude sufficient training data

GAN Techniques

- GigaGAN (CMU & Adobe Reserch, CVPR 2023)
  - A larger GAN model (https://mingukkang.github.io/GigaGAN/)
    - 0.13s for 512px & 3.7s for 4k
    - ~650M params
    - 64 ~ 128x A100 GPU for training
    - Text to image generation
      - (more on lecture 10 & 12)

a blue Porsche 356 parked in front of a yellow brick wall.

Eiffel Tower, landscape photography

Figure A2. Uncurated images (above: Lorikeet and below: Arctic fox) from ADM-G-U [15], LDM-4-G [79], GigaGAN (ours), and StyleGAN-XL [86]. FID values of each generative model are 4.01, 3.60, 3.45, and 2.32, respectively.

GAN Techniques

- GigaGAN (CMU & Adobe Reserch, CVPR 2023)
  - A larger GAN model (https://mingukkang.github.io/GigaGAN/)
    - 0.13s for 512px & 3.7s for 4k
    - ~650M params
    - 64 ~ 128x A100 GPU for training
    - Text to image generation
  - A lot of tricks…
    - StyleGAN2 Backbone
    - Attention + adaptive convolution
    - Multi-scale training
    - Smart designs of G & D
    - …

Model	FID-10k ↓	CLIP Score ↑	# Param.
StyleGAN2	29.91	0.222	27.8M
+ Larger (5.7×)	34.07	0.223	158.9M
+ Tuned	28.11	0.228	26.2M
+ Attention	23.87	0.235	59.0M
+ Matching-aware D	27.29	0.250	59.0M
+ Matching-aware G and D	21.66	0.254	59.0M
+ Adaptive convolution	19.97	0.261	80.2M
+ Deeper	19.18	0.263	161.9M
+ CLIP loss	14.88	0.280	161.9M
+ Multi-scale training	14.92	0.300	164.0M
+ Vision-aided GAN	13.67	0.287	164.0M
+ Scale-up (**GigaGAN**)	9.18	0.307	652.5M

GAN Techniques

- GigaGAN (2023)
  - Super-resolution

Figure 2. **Our GAN-based upsampler** can serve in the upsampling pipeline of many text-to-image models that often generate initial

GAN Techniques

- GigaGAN (2023)
  - Super-resolution

Figure 3. **Our GAN-based upsampler**, similar to Figure 2, can also be used as an off-the-shelf superresolution model for real images

GAN Techniques

- GigaGAN (2023): Latent-Space Interpolation

GAN Techniques

- GigaGAN (2023)
  - Latent-Space Mixing

GAN Techniques

The GAN is dead; long live the GAN!
A Modern Baseline GAN

Yiwen Huang, Brown University
Aaron Gokaslan, Cornell University
Volodymyr Kuleshov, Cornell University
James Tompkin, Brown University

- R3GAN (Brown & Cornell, NeurIPS 2024)
  - Latest GAN with modern architectures & (simplified) convergence guarantee
  - A pairwise loss (from RpGAN, NeurIPS2020) for mode coverage

$$\mathcal{L}(\theta,\psi) = \mathbb{E}_{\substack{z\sim p_z \\ x\sim p_{\mathcal{D}}}}\left[f\left(D_\psi(G_\theta(z)) - D_\psi(x)\right)\right]$$

  - Zero-centered gradient penalty to ensure convergence

$$R_1(\psi) = \frac{\gamma}{2}\mathbb{E}_{x\sim p_{\mathcal{D}}}\left[\left\|\nabla_x D_\psi\right\|^2\right] \qquad R_2(\theta,\psi) = \frac{\gamma}{2}\mathbb{E}_{x\sim p_\theta}\left[\left\|\nabla_x D_\psi\right\|^2\right]$$

  - Modern architectures (i.e., ConvNeXT) & very few tricks

Loss	# modes↑	D_{KL}↓
RpGAN $+R_1+R_2$	**1000**	**0.0781**
GAN $+R_1+R_2$	693	0.9270
RpGAN $+R_1$	Fail	Fail
GAN $+R_1$	Fail	Fail

Model	NFE↓	FID↓
BigGAN-deep [3]	1	4.06
DDPM [20]	250	11.0
DDIM [76]	50	13.7
ADM [7]	§250	2.91
EDM [33]	79	2.23
CT [79]	2	11.1
CD [79]	3	4.32
iCT-deep [77]	2	2.77
DMD [96]	1	2.62
Ours—Config E	1	2.09
With ImageNet feature leakage [41]:		
StyleGAN-XL* [69]	1	1.52

Table 8: ImageNet-64. §:deterministic sampling.

GAN Techniques

- R3GAN (Brown & Cornell, Ne
  - Latest GAN with modern archi
  - A pairwise loss (from RpGAN, N

$$\mathcal{L}(\theta,\psi) = \mathbb{E}_{\substack{z\sim p_z \\ x\sim p_{\mathcal{D}}}}\Big[f$$

  - Zero-centered gradient penalty

$$R_1(\psi) = \frac{\gamma}{2}\mathbb{E}_{x\sim p_{\mathcal{D}}}\left[\left\|\nabla_x D_\psi\right\|^2\right]$$

  - Modern architectures (i.e., Co

GAN Extensions

- Semi-Supervised Learning with GAN (Salimans et al, 2016)
  - Dataset $\{(x, y)\} \cup \{x\}$, K labels
  - Conditioned GAN
    - $G(z, y) \to x$
    - $D(x) \to softmax(l_1, \dots, l_K, l_{K+1})$
      - Label $K + 1$: "fake" data
  - Semi-Supervised Learning
    - For x without label, $L(x; \phi) = 1 - p_D(y = K + 1|x)$
    - Manually set $l_{K+1} = 1$ for simplicity
      - $L_{sup}(x, k; \phi) = p_D\left(y = k \,\middle|\, x \& y \leq K\right) = softmax(k|l_1, \dots, l_K)$
      - $L_{uns}(x; \phi) = 1 - p_D(y = K + 1|x) = 1 - softmax(K + 1|l_1, \dots, l_K, 1)$
  - Tip: Labeled data help train a much better D, leading to higher sample quality!
    - Remark: minibatch discrimination breaks training in practice

Model	Percentage of incorrectly predicted test examples for a given number of labeled samples		
	500	1000	2000
DGN [21]		36.02±0.10	
Virtual Adversarial [22]		24.63	
Auxiliary Deep Generative Model [23]		22.86	
Skip Deep Generative Model [23]		16.61±0.24	
Our model	18.44 ± 4.8	8.11 ± 1.3	6.16 ± 0.58
Ensemble of 10 of our models		5.88 ± 1.0	

Figure 5: *(Left)* Error rate on SVHN. *(Right)* Samples from the generator for SVHN.

GAN Extensions

- Representation learning with GAN?
  - $G: z \rightarrow x$; can we learn $E(x) \rightarrow z$?
  - Idea: $(z, G(z))$ and $(E(x), x)$ should from the same distribution!
- Adversarial Feature Learning (BiGAN, Donahue et al, ICLR2017)
  - Bidirectional GAN: learn both $G(z)$ and $E(x)$
  - Discriminator $D(x, z)$
    - Input: either $(z, G(z))$ or $(E(x), x)$, $x \sim p_{data}$
    - Predict how likely a sample is from $(E(x), x)$
  - Learns better features than naïve AE

GAN Extensions

- F

- A

$P(y)$

Figure 4: Qualitative results for ImageNet BiGAN training, including generator samples $G(\mathbf{z})$, real data $\mathbf{x}$, and corresponding reconstructions $G(E(\mathbf{x}))$.

GAN Extensions

- BigBiGAN (Donahue et al, DeepMind, NIPS2019)
  - Large scale training of BiGAN based on BigGAN
  - Similar architecture and design as BigGAN
  - Modification
    - Additional unary constraint on $E(x)$ and $G(z)$
    - $J(x, z)$ takes in input of
      - (x, z), $x \sim p_{data}$, $z \sim N(0, I)$
      - $(G(z), E(z))$ (negative)
    - A total of 3 loss functions

GAN Extensions

- BigBiGAN (Donahue et al, DeepMind, NIPS2019)
  - Large scale training of BiGAN based on BigGAN
  - Similar architecture and design as BigGAN
  - Modification
    - Additional unary constraint on $E(x)$ and $G(z)$
    - $J(x, z)$ takes in input of
      - (x, z), $x \sim p_{data}$, $z \sim N(0, I)$
      - $(G(z), E(z))$ (negative)
    - A total of 3 loss functions
    - Reconstruction result
      - $G(E(x))$

Figure 2: Selected reconstructions from an unsupervised BigBiGAN model (Section 3.3). Top row

GAN Extensions

- Extend the generator G
  - Standard $G: z \rightarrow x$; $z \sim N(0, I)$
  - Extension: z can be any distribution!
    - Style transfer!

GAN Extensions

- Extend the generator G
  - Standard $G: z \to x$; $z \sim N(0, I)$
  - Extension: z can be any distribution!
    - Style transfer!
- Pix2Pix (Isola, Berkeley, CVPR2017)
  - Paired data: $\{(x, z)\}$, z: input; x: target
  - Generator $G: z \to x$;
  - Discriminator $D(z, x)$ v.s. $D(z, G(z))$
  - Tricks
    - U-Net architecture for generator with skip-connections for better conditioning
    - Additional L_1 loss $|x - G(z)|$

GAN Extensions

- Extenc
  - Star
  - Exte
  -
- Pix2Pi
  - Pair
  - Ger
  - Disc
  - Tric
  -
  -

GAN Extensions

- Pix2Pix (Isola, Berkeley, CVPR2017)
  - Paired data: $\{(x, z)\}$, z: input; x: target
  - Generator $G: z \rightarrow x$;
  - Discriminator $D(z, x)$ v.s. $D(z, G(z))$
- Let's draw cats!
  - https://affinelayer.com/pixsrv/

GAN Extensions

- Pix2Pix (
  - Paired
  - Gener
  - Discrir
- Let's dra
  - https:/

Ivy Tasi @ivymyt

Vitaly Vidmirov @vvid

GAN Extensions

- Pix2Pix (Isola, Berkeley, CVPR2017)
  - Paired data: $\{(x, z)\}$, z: input; x: target
  - Generator $G: z \to x$;
  - Discriminator $D(z, x)$ v.s. $D(z, G(z))$
- Let's dance!

GAN Extensions

- Pix2Pix (Isola, Berkeley, CVPR2017)
- Everybody Dance Now (Berkeley, ICCV 2019)

GAN Extensions

- Pix
- Eve

GAN Extensions

- What if we do not have paired data?
  - Pix2Pix require 1-on-1 mapped data pairs
  - E.g., P_X and P_Y but no pairing
- CycleGAN (Junyan Zhu et al, ICCV2017)
  - Idea: two generator and two discriminator
    - $G: x \to y$ and $F: y \to x$
    - Data: $D_X(x) \to [0,1]$ and $D_Y(y) \to [0,1]$
  - Cycle-consistency loss
    - $x \to G(x) \to F\big(G(x)\big) \to x$
    - $L_{cyc} = E_x\left[\left|F\big(G(x)\big) - x\right|_1\right] + E_y\left[\left|G\big(F(y)\big) - y\right|_1\right]$
  - Train two GANs jointly with additional loss $\lambda \cdot L_{cyc}$

GAN Extensions

GAN Extensions

- Funny Failures of CycleGAN

GAN Extensions

- DragGAN (Pan et al, SIGGRAPH 2023)
  - You can modify an image by dragging points
  - No fine-tuning required at all
- Key idea:
  - A trained generator $G(z) \rightarrow x$
  - Search over z' where key points move long the drag
  - "Gradient" estimate for z'

Summary

- Generative Adversarial Networks
  - Implicit generative model and likelihood-free learning
  - Flexible framework and powerful neural loss function
  - High quality generation and general applications
  - Fundamentally hard to train and lots of tricks to make it work

- Compare with other generative models
  - EBM: generic density, best math, sample via MCMC, slow converge
  - VAE: flexible model and sampling, stable training but approximate inference
  - Trade-offs: sampling, expressiveness and training
  - Applications: (conditioned) generation/inference, semi/un-supervised learning

End of Class

- Yes, it is true!