M2177.003100
Deep Learning

[15: Generative Adversarial Network (GAN)]

Electrical and Computer Engineering
Seoul National University

(last compiled at 11:47:00 on 2020/11/21)

Outline

Generative Adversarial Networks
- Introduction
- Training
- Historical Developments
- Challenges and Analyses

Summary

Appendix: Wasserstein GAN

References

- *Deep Learning* by Goodfellow, Bengio and Courville Link
  - Chapter 20: Deep Generative Models
- survey:
  - *How GANs and Their Variants Work* Link
- online resources:
  - *Stanford CS231n: CNN for Visual Recognition* Link
  - *CVPR 2018 GAN Tutorial* Link
  - *NIPS 2016 GAN Tutorial* Link
  - *GAN Blog at Medium* Link
- Wasserstein GAN
  - *ICML 2017 paper* Link
  - *Read-through: Wasserstein GAN* Link
  - *From GAN to WGAN* Link

Outline

Generative Adversarial Networks
- Introduction
- Training
- Historical Developments
- Challenges and Analyses

Summary

Appendix: Wasserstein GAN

Comparison

- autoregressive models: define a tractable density function
  - optimize likelihood of training data:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \prod_{i=1}^{n} p_{\boldsymbol{\theta}}(x_i \mid x_1, \ldots, x_{i-1}) \tag{1}$$

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

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \int p_{\boldsymbol{\theta}}(\boldsymbol{z}) p_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{z}) d\boldsymbol{z} \tag{2}$$

  - cannot optimize (2) directly

  $\Rightarrow$ derive and optimize a lower bound on the likelihood instead

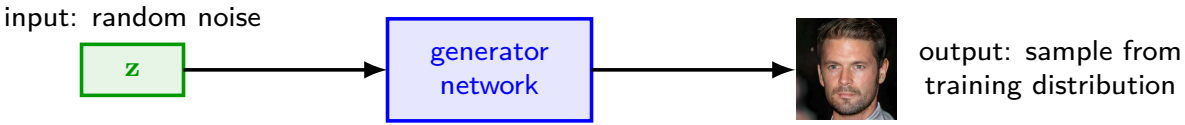
- what if we give up on explicitly modeling density
  - and just want ability to sample ?

Main problem and solution

- problem:
  - want to sample from a complex/high-dimensional training distribution
  - directly doing this: very challenging (if not impossible)

- a solution: two-step approach
  1. sample from a simple distribution (*e.g.* random noise)
  2. learn a transformation to the training distribution

     ↑ how to represent this complex transformation?
  - use a neural net



face image source: Karras et al. "Progressive growing of GANs for improved quality, stability, and variation." *arXiv preprint arXiv:1710.10196* (2017)

Generative adversarial networks (GANs)

- do not work with any explicit density function
  - instead, take a game-theoretic approach
  - learn to generate from training distribution through 2-player game

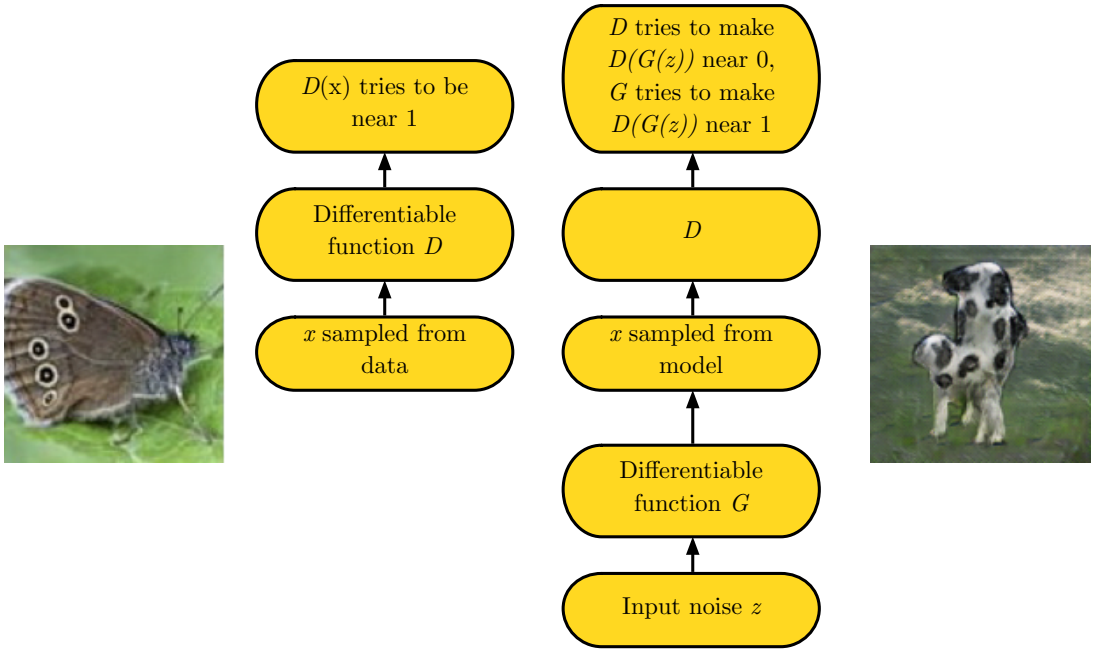


image source: CVPR 2018 Tutorial on GANs: Goodfellow (2018)
http://efrosgans.eecs.berkeley.edu/CVPR18_slides/Introduction_by_Goodfellow.pdf

- GAN: based on a game-theoretic scenario
  - generator network must compete against an adversary (= discriminator)
- generator network
  - directly produces samples $\tilde{\boldsymbol{x}} = G(\boldsymbol{z}; \boldsymbol{\theta}_g) = G_{\boldsymbol{\theta}_g}(\boldsymbol{z})$
- its adversary: discriminator network
  - attempts to distinguish between
    - samples drawn from training data
      and
    - samples drawn from generator
  - emits a probability value given by $\underbrace{D(\boldsymbol{x}; \boldsymbol{\theta}_d) = D_{\boldsymbol{\theta}_d}(\boldsymbol{x})}$

probability that $\boldsymbol{x}$ is a real training example rather than a fake sample drawn from the model

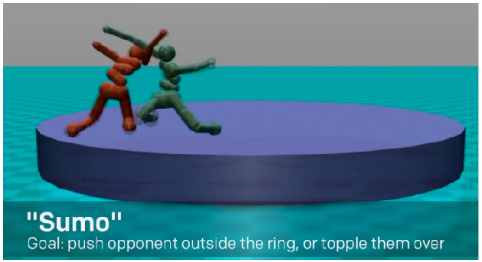
Related idea

- self-play
  - adversarial players improve together through self-play training

1959: Arthur Samuel's checkers agent

(Silver et al, 2017)

(OpenAI, 2017)



(Bansal et al, 2017)

image source: CVPR 2018 Tutorial on GANs: Goodfellow (2018)
http://efrosgans.eecs.berkeley.edu/CVPR18_slides/Introduction_by_Goodfellow.pdf

Explosion of GAN applications

- cumulative number of named GAN papers by month ▸ Link

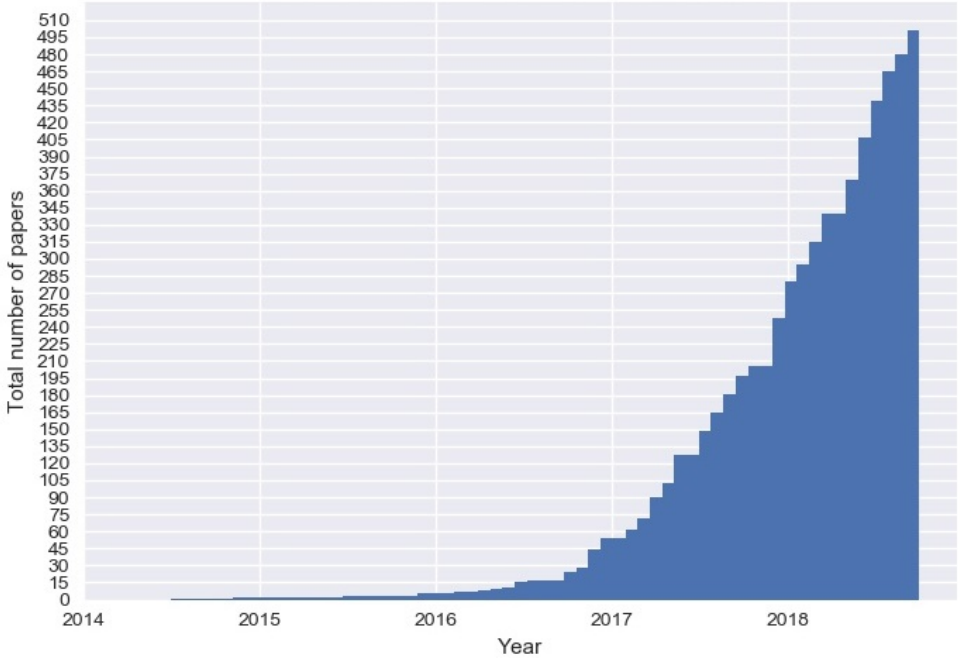


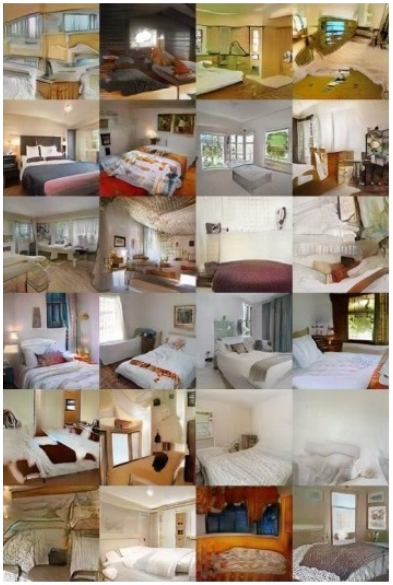
image source: https://github.com/hindupuravinash/the-gan-zoo

- better training and sample generation

church outdoor

kitchen

LSGAN[1]



WGAN[2], improved WGAN[3]

progressive GAN[4]

---

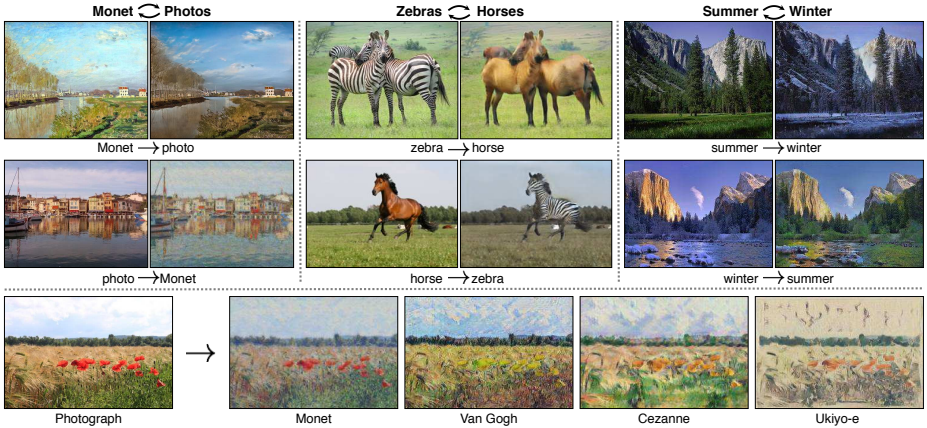
[1] Mao, X., Li, Q., Xie, H., Lau, R. Y., Wang, Z., and Smolley, S. P. (2017). Least squares generative adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2813–2821. IEEE

[2] Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223

[3] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017). Improved training of wasserstein gans. In *Advances in neural information processing systems*, pages 5767–5777

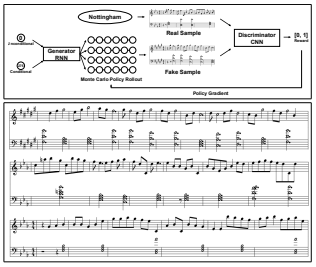
[4] Karras, T., Aila, T., Laine, S., and Lehtinen, J. (2017). Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*

- source → target domain transfer: CycleGAN[5]



- text → image synthesis[6]

- music generation[7]



[5] Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint*

[6] Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., and Lee, H. (2016). Generative adversarial text to image synthesis. *arXiv preprint arXiv:1605.05396*

[7] Lee, S.-g., Hwang, U., Min, S., and Yoon, S. (2017). A seqgan for polyphonic music generation. *arXiv preprint arXiv:1710.11418*

Outline

Generative Adversarial Networks
- Introduction
- Training
- Historical Developments
- Challenges and Analyses

Summary

Appendix: Wasserstein GAN

Training GANs

- two-player game
  - generator: try to fool discriminator by generating real-looking images
  - discriminator: try to distinguish between real and fake images

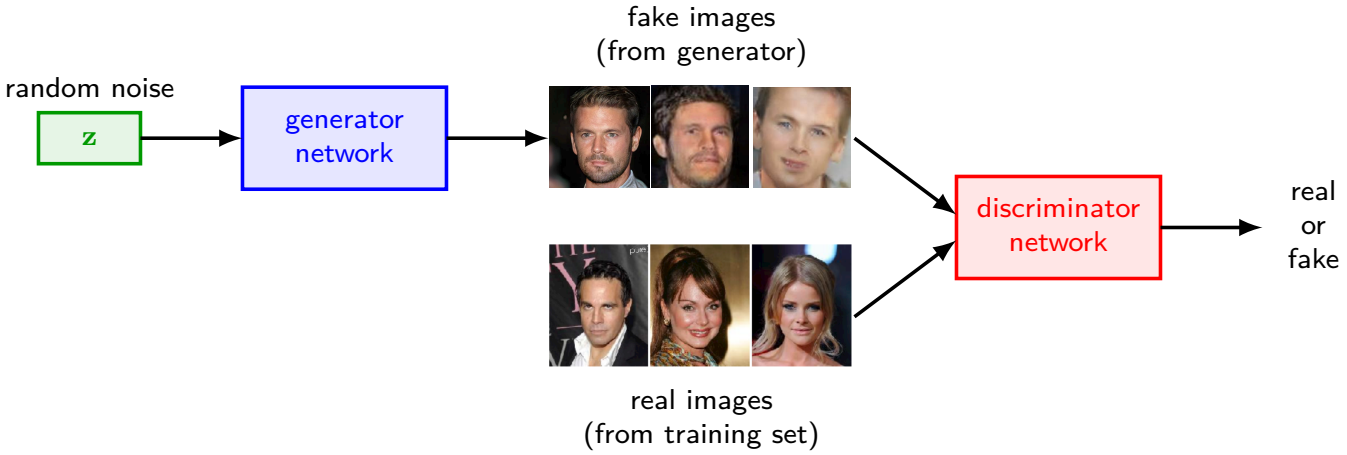


image sources: Goodfellow (2016) https://media.nips.cc/Conferences/2016/Slides/6202-Slides.pdf; Karras et al. "Progressive growing of GANs for improved quality, stability, and variation." *arXiv preprint arXiv:1710.10196* (2017); Brundage et al. (2017)

Training objectives

- assume:
  - discriminator D outputs the likelihood of input being real (range: $[0, 1]$)

- discriminator: maximize J_D wrt $\boldsymbol{\theta}_d$

$$J_D = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}} \log \overbrace{D_{\boldsymbol{\theta}_d}(\boldsymbol{x})}^{\text{how likely } \boldsymbol{x} \text{ is real}} + \mathbb{E}_{z\sim p(z)} \overbrace{\log(1 - D_{\boldsymbol{\theta}_d}(G_{\boldsymbol{\theta}_g}(\boldsymbol{z})))}^{\text{how likely } G(\boldsymbol{z}) \text{ is fake}}$$

$$\approx \underbrace{\frac{1}{m}\sum_{i=1}^{m} \log D_{\boldsymbol{\theta}_d}(\boldsymbol{x}^{(i)})}_{m \text{ real samples}} + \underbrace{\frac{1}{m}\sum_{i=1}^{m} \log(1 - D_{\boldsymbol{\theta}_d}(G_{\boldsymbol{\theta}_g}(\boldsymbol{z}^{(i)})))}_{m \text{ synthetic samples}}$$

- generator: minimize J_G wrt $\boldsymbol{\theta}_g$

$$J_G = \mathbb{E}_{z\sim p(z)} \log(1 - D_{\boldsymbol{\theta}_d}(G_{\boldsymbol{\theta}_g}(\boldsymbol{z})))$$

$$\approx \underbrace{\frac{1}{m}\sum_{i=1}^{m} \log(1 - D_{\boldsymbol{\theta}_d}(G_{\boldsymbol{\theta}_g}(\boldsymbol{z}^{(i)})))}_{m \text{ synthetic samples}}$$

Minimax game

- generator/discriminator: trained jointly in a minimax **game**
  - minimax objective:

$$\min_{\boldsymbol{\theta}_g} \max_{\boldsymbol{\theta}_d} [\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}} \log \underbrace{D_{\boldsymbol{\theta}_d}(\boldsymbol{x})}_{\substack{\text{discriminator output} \\ \text{for real data } \boldsymbol{x}}} + \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})} \log(1 - \underbrace{D_{\boldsymbol{\theta}_d}(G_{\boldsymbol{\theta}_g}(\boldsymbol{z}))}_{\substack{\text{discriminator output for} \\ \text{generated fake data } G(\boldsymbol{z})}})]$$

- discriminator ($\boldsymbol{\theta}_d$) wants to maximize objective s.t.
  - $D(\boldsymbol{x})$ is close to 1 (real) and
  - $D(G(\boldsymbol{z}))$ is close to 0 (fake)

- generator ($\boldsymbol{\theta}_g$) wants to minimize objective s.t.
  - $D(G(\boldsymbol{z}))$ is close to 1

  i.e. discriminator is fooled into thinking that generated $G(\boldsymbol{z})$ is real

remarks:

- Nash equilibrium[8] of this minimax game: a parameter setting in which
  - distribution of points created by G = distribution of real data samples
  - $D = 0.5$ for all of its input

- important: for this approach to work well
  - D should be a high-capacity model and have access to a lot of data

- jointly training two networks: challenging/can be unstable
  - choosing objectives with better loss landscapes helps training

  $\Rightarrow$ active area of research

  e.g. Wasserstein GAN (WGAN):[9] earth-mover distance bet'n p_{data} and p_{model}
  - based on the notion of optimal transport (see Appendix)

[8] a proposed solution of a non-cooperative game involving two or more players in which each player is assumed to know the equilibrium strategies of the other players, and no player has anything to gain by changing only their own strategy (source: Wikipedia)

[9] Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223

Training details

- minimax objective:

$$\min_{\boldsymbol{\theta}_g} \max_{\boldsymbol{\theta}_d} [\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}} \log \overbrace{D_{\boldsymbol{\theta}_d}(\boldsymbol{x})}^{\text{discriminator output for real data } x} + \mathbb{E}_{\mathbf{z}\sim p(\boldsymbol{z})} \log(1 - \overbrace{D_{\boldsymbol{\theta}_d}(G_{\boldsymbol{\theta}_g}(\boldsymbol{z}))}^{\text{discriminator output for generated fake data } G(z)})]$$

- alternate between:

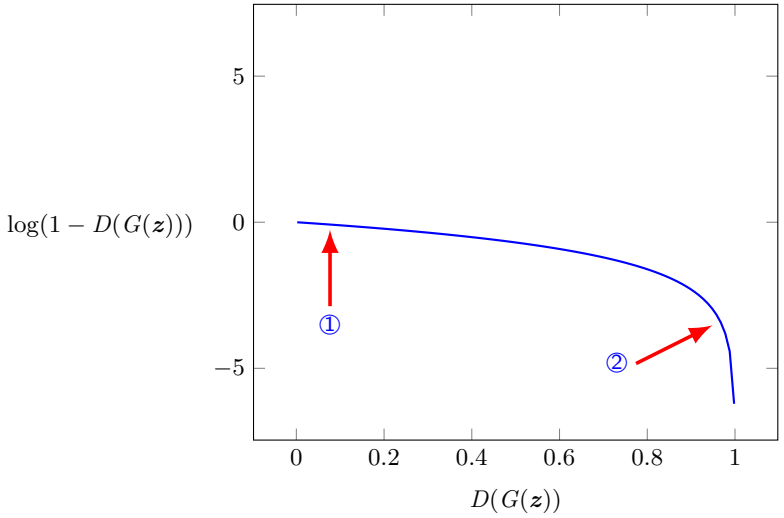
1. gradient ascent on discriminator

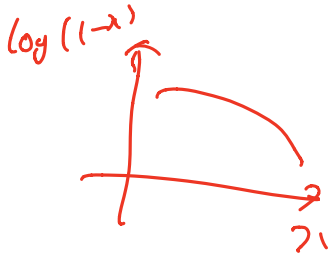
$$\max_{\boldsymbol{\theta}_d} [\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}} \log D_{\boldsymbol{\theta}_d}(\boldsymbol{x}) + \mathbb{E}_{\mathbf{z}\sim p(\boldsymbol{z})} \log(1 - D_{\boldsymbol{\theta}_d}(G_{\boldsymbol{\theta}_g}(\boldsymbol{z})))]$$

2. gradient descent on generator

$$\min_{\boldsymbol{\theta}_g} \mathbb{E}_{\mathbf{z}\sim p(\boldsymbol{z})} \log(1 - D_{\boldsymbol{\theta}_d}(G_{\boldsymbol{\theta}_g}(\boldsymbol{z})))$$

- in practice, optimizing this generator objective does not work well





- near ①: a sample is likely to be fake ⇒ want to learn from it to improve G
  - but gradient is relatively flat ⇒ little learning signal

- near ②: a sample is likely to be real
  - big gradient signal but of little use (the sample: already satisfactory)

image modified from: Fei-Fei Li, J. Johnson, S. Yeung, CS231n: Convolutional Neural Networks for Visual Recognition (2017), http://cs231n.stanford.edu/2017/index.html

- minimax objective function:

$$\min_{\boldsymbol{\theta}_g} \max_{\boldsymbol{\theta}_d} [\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}} \log \overbrace{D_{\boldsymbol{\theta}_d}(\boldsymbol{x})}^{\text{discriminator output for real data } \boldsymbol{x}} + \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})} \log(1 - \overbrace{D_{\boldsymbol{\theta}_d}(G_{\boldsymbol{\theta}_g}(\boldsymbol{z}))}^{\text{discriminator output for generated fake data } G(\boldsymbol{z})})]$$

- alternate between:

1. gradient ascent on discriminator

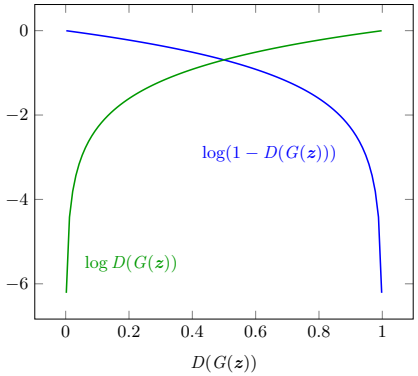
$$\max_{\boldsymbol{\theta}_d} [\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}} \log D_{\boldsymbol{\theta}_d}(\boldsymbol{x}) + \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})} \log(1 - D_{\boldsymbol{\theta}_d}(G_{\boldsymbol{\theta}_g}(\boldsymbol{z})))]$$

2. instead: gradient ascent on generator (different objective)

$$\max_{\boldsymbol{\theta}_g} \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})} \log(D_{\boldsymbol{\theta}_d}(G_{\boldsymbol{\theta}_g}(\boldsymbol{z})))$$

- ⁕ not minimizing likelihood of D being correct, but maximize likelihood of D being wrong
- ⁕ same objective of fooling D, but now higher gradient signal for bad samples

$\Rightarrow$ works much better! (standard in practice)



Each gradient update

1. discriminator update (repeat k times per iteration):
   - prepare m synthetic samples (using generator) and m real samples
   - feed these $2m$ samples to discriminator
   - gradient ascent: backprop on discriminator network to update $\boldsymbol{\theta}_d$

2. generator update (once per iteration):
   - prepare m synthetic samples using generator
   - feed these m samples to discriminator
   - gradient a/descent: backprop on whole network
     - backward gradient from discriminator output $\rightarrow$ generator
     - but update $\boldsymbol{\theta}_g$ only

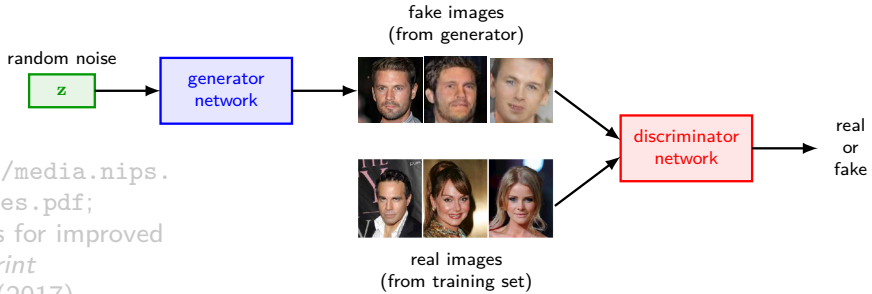


image sources: Goodfellow (2016) https://media.nips.cc/Conferences/2016/Slides/6202-Slides.pdf; Karras et al. "Progressive growing of GANs for improved quality, stability, and variation." *arXiv preprint arXiv:1710.10196* (2017); Brundage et al. (2017)

- putting it together[10]

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter.

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

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

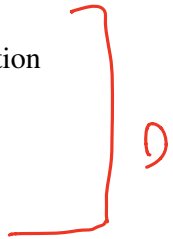
end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule.



- no best rule for k (*i.e.* # steps to apply to discriminator D)
  - some find $k = 1$ more stable; others use $k > 1$ (*e.g.* $k = 5$)
  - recent work (*e.g.* Wasserstein GAN) alleviates this problem $\Rightarrow$ better stability

[10] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., et al. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680

After training

- use generator network to generate new images
  - discard discriminator network

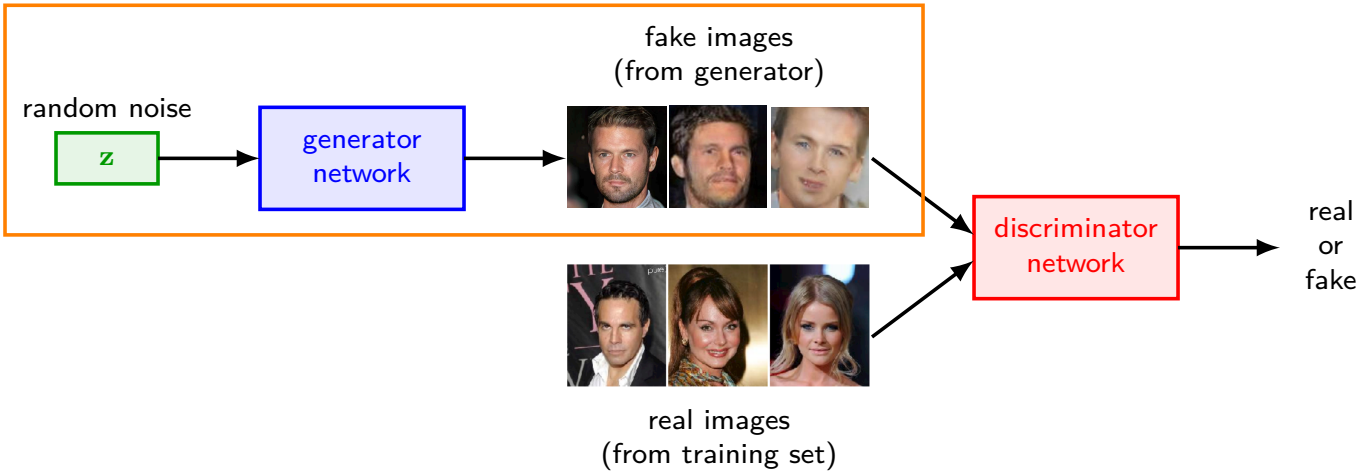


image sources: Goodfellow (2016) https://media.nips.cc/Conferences/2016/Slides/6202-Slides.pdf; Karras et al. "Progressive growing of GANs for improved quality, stability, and variation." *arXiv preprint arXiv:1710.10196* (2017); Brundage et al. (2017)

Performance evaluation

- recall: entropy
  - high entropy = high randomness (less predictable)
  - low entropy = low randomness (more predictable)
- desideratum #1: quality
  - low entropy (highly predictable) conditional label distribution $p(y \mid \boldsymbol{x})$

  i.e. given an image $\boldsymbol{x}$, should know its label y easily
  - use neural net to classify generated images $\Rightarrow$ gives distribution $p(y \mid \boldsymbol{x})$
- desideratum #2: diversity
  - high entropy (less predictable) $p(y)$

$$p(y) = \int p(y \mid \boldsymbol{x} = G(\boldsymbol{z}))\, d\boldsymbol{z}$$

i.e. if generated images are diverse, $p(y)$ should be uniform

① approx - gen M*
② quality - diverse
③ exploitation - exploration

⁂ there is often trade-off between quality and diversity[11]

[11] Karras, T., Aila, T., Laine, S., and Lehtinen, J. (2017). Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*

- Inception score (IS)
  - combines the two desiderata into one score:[12]
$$\mathrm{IS}(G) = \exp(\mathbb{E}_{\boldsymbol{x}\sim G}[D_{\mathrm{KL}}(p(y \mid \boldsymbol{x}) \,||\, p(y))])$$
  - used Inception for image classification; $\exp(\cdot)$: for easier comparison
  - the higher the better
  - misrepresents performance if G only generates one image per class[13]
- Fréchet Inception distance (FID)[14]
  - actually compares statistics of generated samples to real samples

$$\mathrm{FID}(\boldsymbol{x}_r, \boldsymbol{x}_g) = ||\boldsymbol{\mu}_r - \boldsymbol{\mu}_g||_2^2 + \mathrm{Tr}(\boldsymbol{\Sigma}_r + \boldsymbol{\Sigma}_g - 2(\boldsymbol{\Sigma}_r\boldsymbol{\Sigma}_g)^{1/2})$$

    - $\boldsymbol{x}_r \sim \mathcal{N}(\boldsymbol{\mu}_r, \boldsymbol{\Sigma}_r)$ and $\boldsymbol{x}_g \sim \mathcal{N}(\boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g)$: 2048-dim activations of Inception v3 pool3 layer for real and generated samples, respectively
  - the lower the better

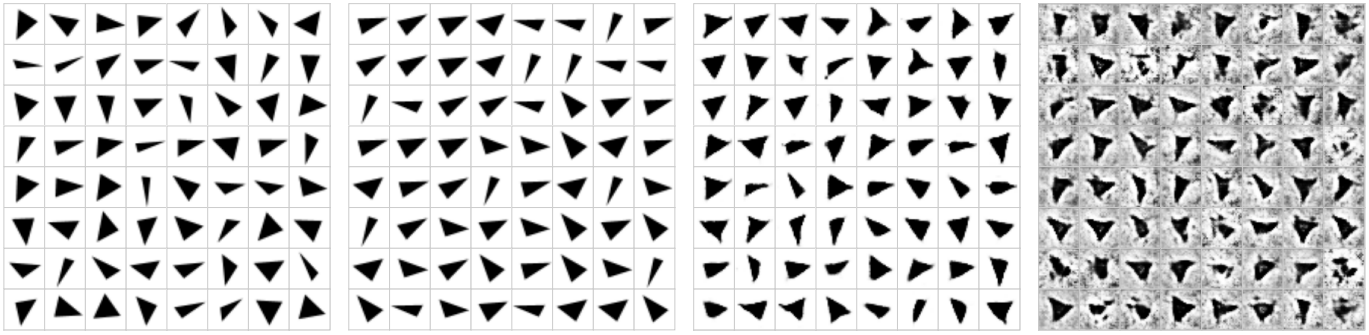
---

[12] Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242

[13] $p(y)$ will still be uniform despite low diversity

[14] Heusel, M., Ramsauer, H., Unterthiner, T., et al. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637

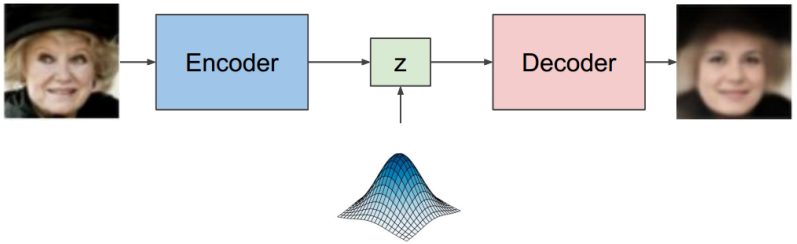
- precision, recall, and F_1 score
  - precision: high when generated images look similar to real on average
  - recall: high when generator can generate any sample in training data
  - F_1 score: harmonic mean of precision and recall



(a) High precision, high recall (b) High precision, low recall (c) Low precision, high recall (d) Low precision, low recall

image source: Lucic, M., Kurach, K., Michalski, M., Gelly, S., and Bousquet, O. (2017). Are GANs created equal? A large-scale study. *arXiv preprint arXiv:1711.10337*

VAE vs GAN

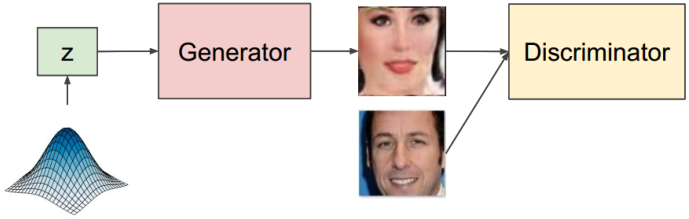


VAE

⊕ given x, easy to find z

⊕ interpretable probability $p(x)$

⊖ usually outputs blurry images



GAN

⊕ very sharp images

⊖ given x, difficult to find z

⊕/⊖ no explicit $p(x)$

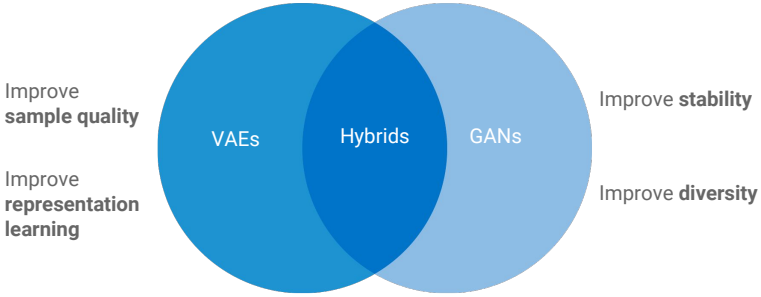
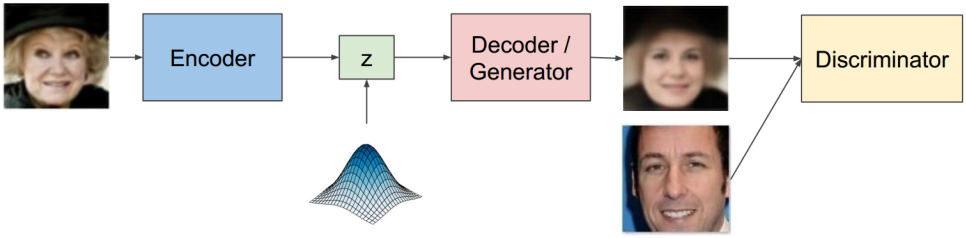
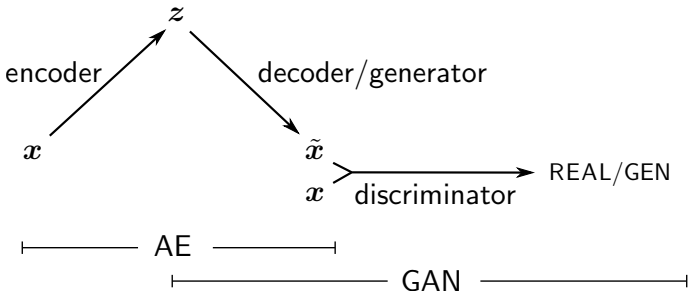


image sources: Larsen, A. B. L., Sønderby, S. K., Larochelle, H., and Winther, O. (2015). Autoencoding beyond pixels using a learned similarity metric. *arXiv preprint arXiv:1512.09300*; Rosca (2018); Yeh, Lou, and Lim (2017) `http://slazebni.cs.illinois.edu/spring17/lec12_vae.pdf`

VAE + GAN





- training objective:

$$\mathcal{L} = \mathcal{L}_{\text{prior}} + \mathcal{L}^{D}_{\text{like}} + \mathcal{L}_{\text{GAN}}$$

where

$$\mathcal{L}_{\text{prior}} = D_{\text{KL}}[q(\boldsymbol{z} \mid \boldsymbol{x}) \,||\, p(\boldsymbol{z})]$$
$$\mathcal{L}^{D}_{\text{like}} = -\mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}[\log p(D(\boldsymbol{x}) \mid \boldsymbol{z})]$$
$$\mathcal{L}_{\text{GAN}} = \log(D(\boldsymbol{x})) + \log(1 - D(G(\boldsymbol{z})))$$

- adversarial AE, adversarial variational Bayes, VEEGAN, ALI/BiGAN, AlphaGAN, ...

image sources: Larsen, A. B. L., Sønderby, S. K., Larochelle, H., and Winther, O. (2015). Autoencoding beyond pixels using a learned similarity metric. *arXiv preprint arXiv:1512.09300*; Yeh, Lou, and Lim (2017)
http://slazebni.cs.illinois.edu/spring17/lec12_vae.pdf

Outline

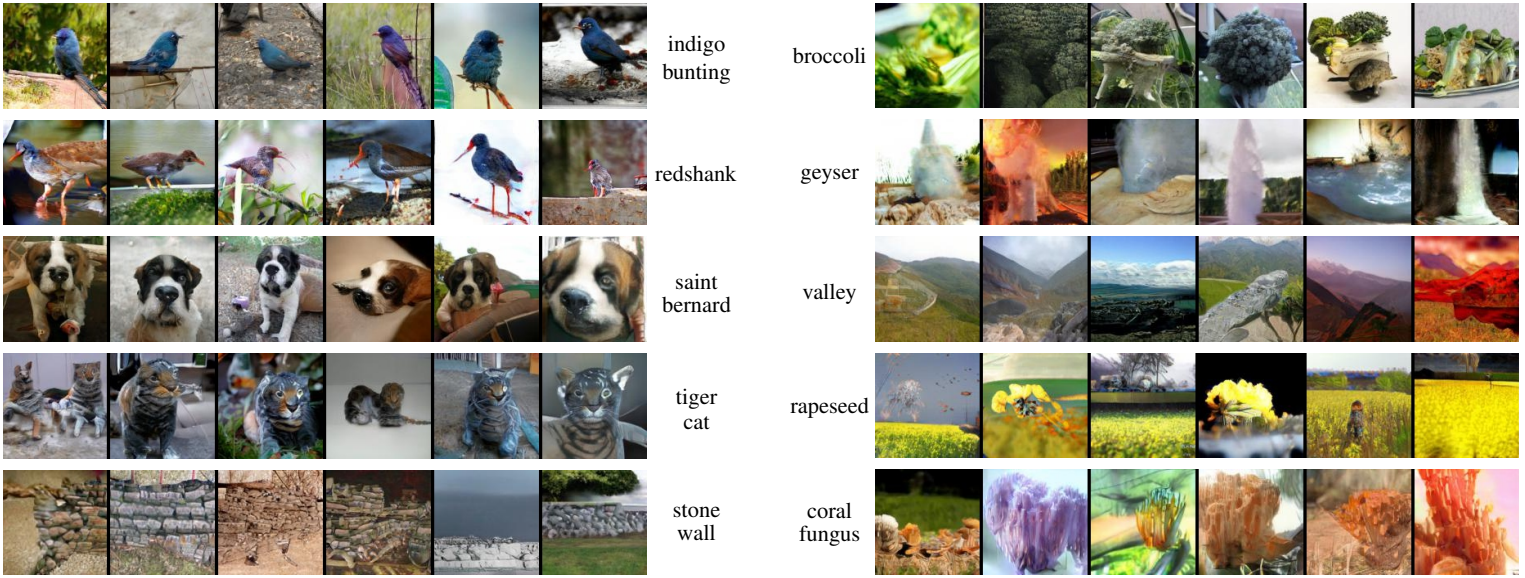
Generative Adversarial Networks
- Introduction
- Training
- Historical Developments
- Challenges and Analyses

Summary

Appendix: Wasserstein GAN

Self-attention GAN (SAGAN)[15]

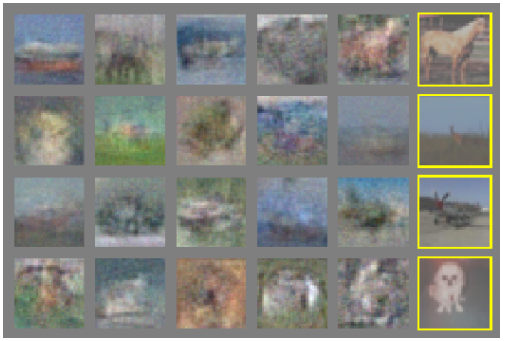
- state-of-the-art on ImageNet
  - 1000 categories, 128×128 pixels



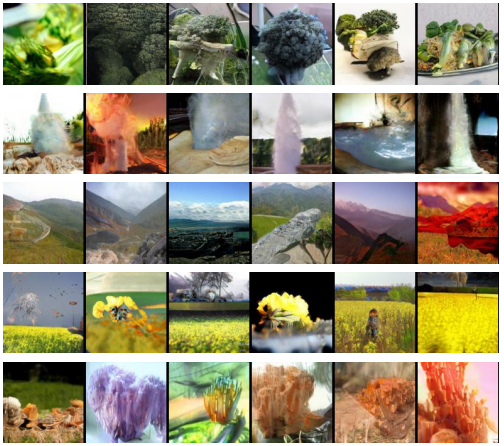
[15] Zhang, H., Goodfellow, I., Metaxas, D., and Odena, A. (2018). Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*

From GAN to SAGAN

- great ideas developed
  1. depth and convolution
  2. class-conditional generation
  3. spectral normalization
  4. two-timescale update rule
  5. self-attention



GAN [Goodfellow et al., 2014][16]



SAGAN [Zhang et al., 2018][17]

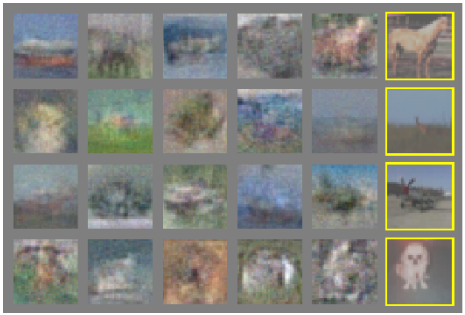
[16] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., et al. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680

[17] Zhang, H., Goodfellow, I., Metaxas, D., and Odena, A. (2018). Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*

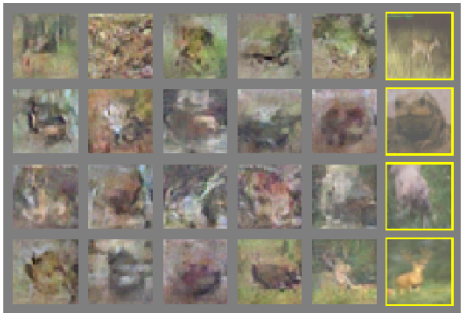
1. Depth and convolution

- difficulty matters
  - simple tasks: no convolution needed
  - harder tasks: depth and convolution needed

original GAN (CIFAR-10)



no convolution



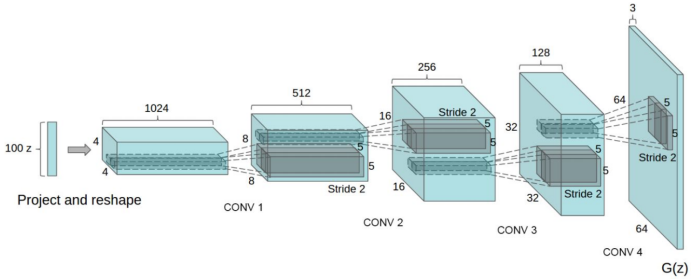
one conv layer

DCGAN (ImageNet)

many conv layers

image sources: Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, et al. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680; Radford, Metz, and Chintala (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*

- deep convolutional GAN (DCGAN)[18]
  - generator: an upsampling network with fractionally-strided convolutions



  - discriminator: a convolutional network

- architectural guidelines for stable DCGANs
  a) replace any pooling layers with
     - strided conv (discriminator) + fraction-strided conv (generator)

transposed conv.

  b) use batch norm in both generator and discriminator
  c) remove fully connected hidden layers for deeper architectures
  d) use ReLU in generator for all layers except for the output, which uses tanh
  e) use LeakyReLU in discriminator for all layers

[18] Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*

2. Class-conditional generation

- even after developing idea 1
  - ImageNet samples did not yet have recognizable objects

- key idea to generate recognizable objects
  - let generative model learn joint distribution over pixels ($\boldsymbol{x}$) and classes (y)

$$p(\boldsymbol{x}, y) = p(y)\underbrace{p(\boldsymbol{x} \mid y)}_{\text{conditional GAN learns this}}$$

- first work on conditional GAN:[19]
  - discriminator gets to see both class label and pixel input, saying
    - whether pixels match requested class
    - not whether they are realistic
  - could generate 10 MNIST digits

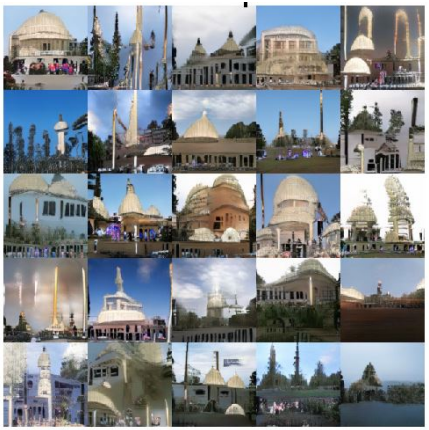
[19] Mirza, M. and Osindero, S. (2014). Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*

- auxiliary classifier GAN (AC-GAN)[20]
  - could generate class-conditional ImageNet samples
- trick: an ensemble of specialized classifiers
  - a group of 100 different generators each specialized in 10 classes
- generated images: recognizable, coming from the desired class
- limitations
  - not yet high-quality samples
  - not yet a single generator for all 1000 classes

monarch butterfly | goldfinch | daisy | redshank | grey whale

[20] Odena, A., Olah, C., and Shlens, J. (2016). Conditional image synthesis with auxiliary classifier gans. *arXiv preprint arXiv:1610.09585*

- spectrally normalized GAN (SN-GAN)[21]
  - a single generator can make recognizable samples from every class
  - generated images: recognizable and somewhat high quality



- the idea of spectral normalization
  - important beyond just the context of class-conditional generation

[21] Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. (2018). Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*

3. Spectral normalization (SN)[22]

- main idea:
  - regularize the eigen/singular value spectrum of each weight matrix
- how it works:
  - for every layer of the net (that is represented by a linear mapping $\boldsymbol{W}$)
  - estimate the largest singular value of that mapping and divide by it

  $\Rightarrow$ each of the weight matrices has a bounded spectral radius

$$\bar{\boldsymbol{W}}_{\mathrm{SN}} \triangleq \boldsymbol{W} / \underbrace{\sigma(\boldsymbol{W})}_{\substack{\text{spectral norm of } \boldsymbol{W} \\ = \text{ largest singular value}}}$$

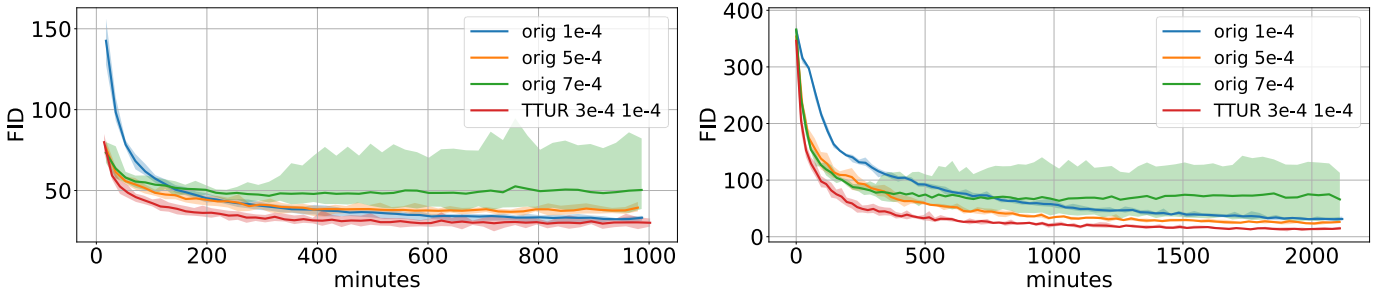
- effects:
  - makes learning process much more stable
    - despite no direct connection to game theory
  - makes it more possible to use a lot of computation with GANs

  $\Rightarrow$ complex GAN training runs for weeks without diverging $\Rightarrow$ beneficial

[22] Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. (2018). Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*

4. Two-timescale update rule (TTUR)[23]

- key idea:
  - use different learning rates (ϵ_{g}, ϵ_{d}) for generator and discriminator
- in theory: we can prove
  - this helps with convergence if you decay ϵ_{g} faster than ϵ_{d}
- in practice: no need for decaying ϵ_{g}
  - just using two different learning rates: sufficient



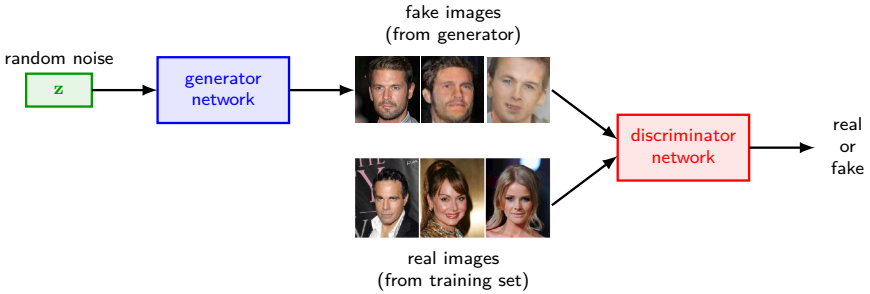
left: CIFAR-10, right: LSUN bedrooms; TTUR ϵ_{d} ϵ_{g}

[23] Heusel, M., Ramsauer, H., Unterthiner, T., et al. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637

- known tip for getting better GAN performance:
  - run D for several steps before running G (*e.g.* $k = 5$)
  - reason: D generates signals G uses to learn

- unfortunate consequence: to calculate a D step, we need
  - go forward through $G \rightarrow D$; then go backward through D

  $\Rightarrow$ much computation wasted if you do multiple D and G steps



- TTUR benefits
  - allows a much higher learning rate for D and for G
  - makes both players doing simultaneous steps and keeps the learning stable

image sources: Goodfellow (2016) https://media.nips.cc/Conferences/2016/Slides/6202-Slides.pdf; Karras et al. "Progressive growing of GANs for improved quality, stability, and variation." *arXiv preprint arXiv:1710.10196* (2017); Brundage et al. (2017)

5. Self-attention

- self-attention (= intra-attention)[24]
  - relates different positions of a single sequence (*e.g.* a sentence)
    - in order to compute a representation of the sequence
  - computes the response at a position in a sequence by
    - attending to all positions and
    - taking their weighted average in an embedding space
  - can be viewed as a form of the non-local mean

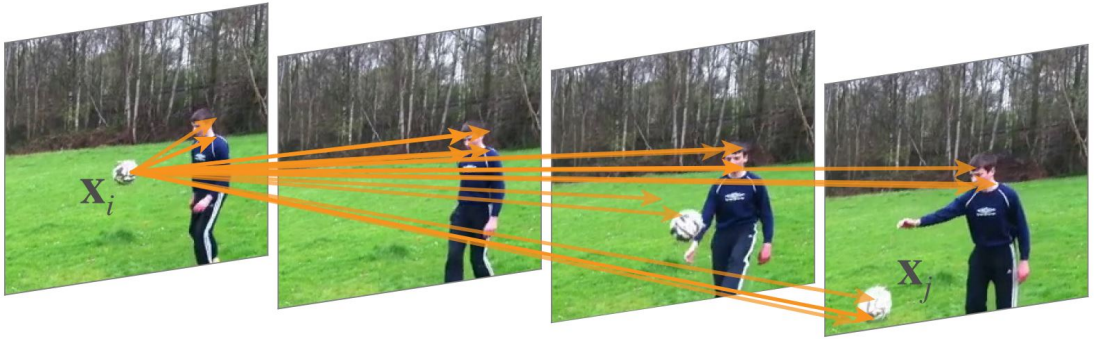
a weighted mean of all pixels in an image

[24] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008

- non-local neural nets[25]
  - bridges self-attention to more general class of non-local filtering ops

for machine translation ↑ self-attention

non-local filtering ops ↑ for image/video problems



a space-time non-local operation [Wang et al., 2017]

[25] Wang, X., Girshick, R., Gupta, A., and He, K. (2017). Non-local neural networks. *arXiv preprint arXiv:1711.07971*, 10

- self-attention GAN (SAGAN)[26]
  - uses layers from the non-local neural nets
  - main idea: during generation process
    - when generating each pixel, we can look around the image
    - and see what the other pixels look like

e.g. when generating the eye of a bear
    - look around the other eye and make sure that they are symmetric
  - we can focus attention in very unusually shaped regions too

    e.g. narrow tail of a bird (square/circle-focused attention would miss)

[26] Zhang, H., Goodfellow, I., Metaxas, D., and Odena, A. (2018). Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*

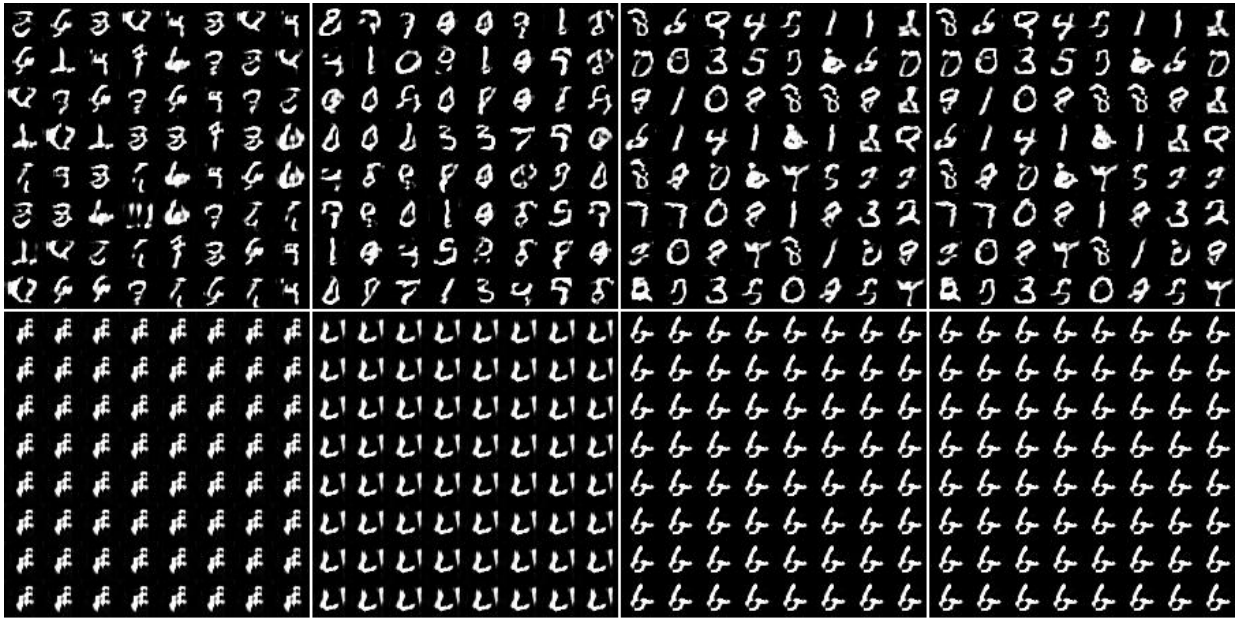
Outline

Generative Adversarial Networks
- Introduction
- Training
- Historical Developments
- Challenges and Analyses

Summary

Appendix: Wasserstein GAN

Mode collapse



- top: no mode collapsing (all digits appear)
- bottom: mode collapsing (to digit 6)

image source: Metz, L., Poole, B., Pfau, D., and Sohl-Dickstein, J. (2016). Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163*

Sample sharpness

- GAN produces sharp samples
  - image generation

  - manifold estimation (left: VAE, right: GAN)

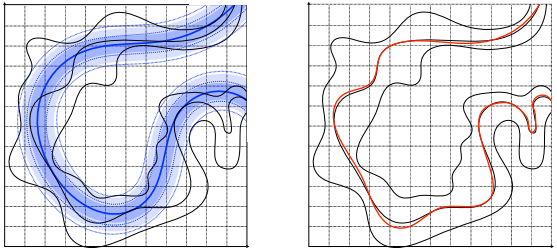


image sources: Larsen, A. B. L., Sønderby, S. K., Larochelle, H., and Winther, O. (2015). Autoencoding beyond pixels using a learned similarity metric. *arXiv preprint arXiv:1512.09300*; Courville (2017)

Why?

- cost function?
  - NO.

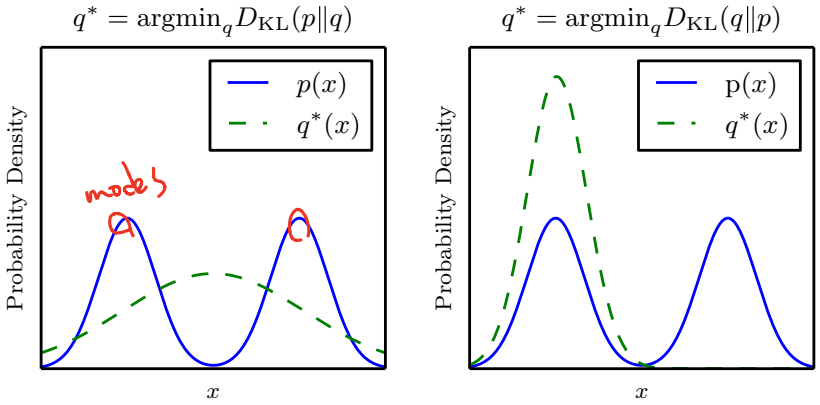


image source: Goodfellow et al. (2016)

p: p_{data}, q: p_{model}	$D_{\text{KL}}(p \,\|\| \, q)$	$D_{\text{KL}}(q \,\|\| \, p)$
name key property	maximum likelihood mean-seeking	reverse KL mode-seeking
requires	high $q(\boldsymbol{x})$ anywhere $p(\boldsymbol{x})$ is high	rarely high $q(\boldsymbol{x})$ anywhere $p(\boldsymbol{x})$ is low
penalize generator if = high penalty when	generated image $\tilde{\boldsymbol{x}}$ misses modes $p(\boldsymbol{x}) > 0$ but $q(\boldsymbol{x}) \to 0$	$\tilde{\boldsymbol{x}}$ does not look real $p(\boldsymbol{x}) \to 0$ but $q(\boldsymbol{x}) > 0$
acceptable even if = low penalty when	$\tilde{\boldsymbol{x}}$ does not look real $p(\boldsymbol{x}) \to 0$ but $q(\boldsymbol{x}) > 0$	less variety in $\tilde{\boldsymbol{x}}$ $p(\boldsymbol{x}) > 0$ but $q(\boldsymbol{x}) \to 0$
bottom line	lower quality but more diverse samples	higher quality but less diverse samples

- mode collapsing/sample sharpness: potentially related/traded-off[27]
  - maximum likelihood models minimize $D_{\text{KL}}(p \,||\, q)$
  - reverse KL models minimize $D_{\text{KL}}(q \,||\, p)$

[27] Karras, T., Aila, T., Laine, S., and Lehtinen, J. (2017). Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*

How about GAN? [Goodfellow, 2016][29]

- recall: Jensen-Shannon divergence

$$D_{\mathrm{JS}}(p \,||\, q) = \frac{1}{2} D_{\mathrm{KL}}\left(p \,||\, \frac{p+q}{2}\right) + \frac{1}{2} D_{\mathrm{KL}}\left(q \,||\, \frac{p+q}{2}\right)$$

  - symmetric : $D_{\mathrm{JS}}(p \,||\, q) = D_{\mathrm{JS}}(q \,||\, p)$
  - penalizes poor images badly (*i.e.* when $p(\boldsymbol{x}) \to 0$ but $q(\boldsymbol{x}) > 0$)

  $\Rightarrow$ similar property to reverse KL $D_{\mathrm{KL}}(q \,||\, p)$

- GAN: optimizing generator = minimizing JS divergence
  - [Goodfellow et al., 2014][28]: if discriminator is optimal
  - generator's objective becomes minimizing

$$J(D^*, G) = 2 D_{\mathrm{JS}}(p_{\mathrm{data}} \,||\, p_{\mathrm{model}}) - \log 4$$

  $\Rightarrow$ potentially results in sharp samples/mode collapsing

---

[28] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., et al. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680

[29] Goodfellow, I. (2016). Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*

- previously many researchers believed:
  - GAN produces sharp/realistic images because it minimize D_{JS}
  - VAE produces blurry samples because it minimize D_{KL}

- newer evidence suggests: D_{JS} does not explain why GANs give sharper samples
  1. maximum likelihood GAN (*i.e.* minimizing forward D_{KL}, not D_{JS}) still
     - ▷ generates sharp samples
     - ▷ selects a small number of modes
  2. reverse KL does not prefer fewer modes in general
     - ▷ it prefers to generate from as many modes of p_{data} as the model can

- reason for mode collapsing: due to
  - a defect in training rather than the divergence minimized

- reason for sharp samples: not entirely clear
  - models/approximations in GAN may be different from others (*e.g.* VAE)

GAN training challenges

1. non-convergence
   - ▷ model parameters oscillate/destabilize/never converge
2. mode collapse
   - ▷ produce limited varieties of samples
3. diminished gradient
   - ▷ D too successful $\Rightarrow$ G gradient vanishes/learns nothing
4. generator/discriminator unbalance
   - ▷ causes overfitting
5. hyperparameter sensitivity
   - ▷ difficult to train robustly

Non-convergence

- largest problem in GAN training: simultaneous gradient descent of two players
  - downhill move of one player $\Rightarrow$ may push the other player uphill
  - a general problem with games not unique to GAN
- in theory[30]
  - updates in function space: guaranteed convergence (a convex problem)
- in practice
  - updates are made in parameter space (non-convex): no such guarantee
  - consequences
    - ▷ oscillation: generating mode $m \rightarrow$ mode $m' \rightarrow \cdots$ (no equilibrium)
    - ▷ mode collapse: most common form of harmful non-convergence

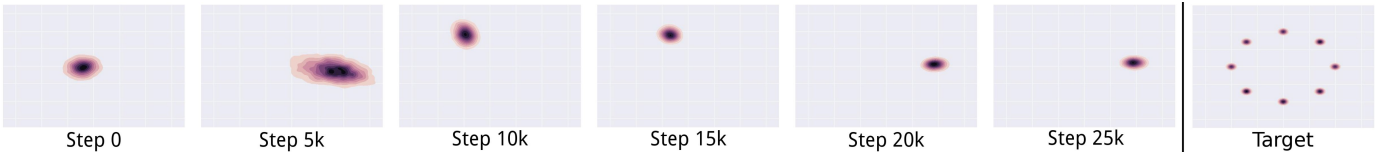
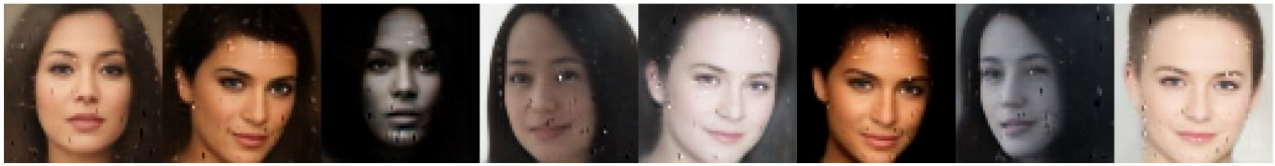


image source: Metz et al. (2016) Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163*

[30] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., et al. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680

Mode collapse

- occurs when generator learns to map
  - several different input z values to the same output point
- one of the most important issues with GAN to address
- in practice: *complete* mode collapse is rare
  - more common: *partial* mode collapse[31]



- possible cause of mode collapsing[32]
  - maximin solution $\neq$ minimax solution (see next page)
  - not any particular cost function (*e.g.* D_{KL} vs D_{JS})

image source: Berthelot et al. (2017) BEGAN: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717* https://arxiv.org/pdf/1703.10717.pdf

[31] generator makes multiple images that contain the same color/texture themes, or multiple images containing different views of the same person

[32] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., et al. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680

- minimax game
  - generator in outer loop of optimization

$$G^* = \min_G \max_D J(G, D) \tag{3}$$

  - G^* draws samples from p_{data}

- maximin game
  - generator in inner loop

$$G^* = \max_D \min_G J(G, D) \tag{4}$$

  - G^* is asked to map every $\boldsymbol{z}$ to $\underbrace{\boldsymbol{x}}_{\text{discriminator believes this is most likely to be real rather than fake}}$

- simultaneous gradient descent: no clear privilege
  - sometimes behave like minimax (3) but often behaves like maximin (4)

- ramification of mode collapse
  - GAN applications: often problems with a small number of distinct outputs

  i.e. tasks aiming to map some input to one of many acceptable outputs

- example: text-to-image synthesis
  - input = a caption for an image
  - output = an image matching that description
  - plain GAN has low output diversity because of mode collapsing

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.

the flower has petals that are bright pinkish purple with white stigma

this white and yellow flower have thin white petals and a round yellow stamen

image source: Reed et al. (2016) Generative adversarial text to image synthesis. *arXiv preprint arXiv: 1605.05396*

Methods to address mode collapse [Hong et al., 2019][33]

- augmented/different objective function
  - unrolled GAN: G predicts D's response by unrolling D's updates
  - deep regret analytic GAN (DRAGAN): constrain gradients of D around real data manifold (also in WGAN-GP and improved WGAN)
  - energy based GAN (EBGAN): a repelling regularizer loss to generator
- architecture change
  - multi-agent diverse GAN (MAD-GAN): multiple G's to capture diversity
  - mode regularized GAN (MRGAN): penalizes G for missing modes
- minibatch discrimination: reflects sample diversity to discriminator
  - discriminator considers multiple examples in a minibatch simultaneously
  - progressive GAN: simplified version of minibatch discrimination

[33] Hong, Y., Hwang, U., Yoo, J., and Yoon, S. (2019). How generative adversarial networks and their variants work: An overview. *ACM Computing Surveys (CSUR)*, 52(1):10 https://dl.acm.org/doi/10.1145/3301282

General tips and tricks [Goodfellow, 2016][36]

1. train with labels: almost always dramatic improvement
   - class-conditional GAN[34]
   - [Salimans et al., 2016][35]
     - ▷ generator needs not explicitly incorporate class information
     - ▷ training discriminator to recognize real object's class is sufficient

2. one-sided label smoothing
   - replace the label $(= 1)$ of real samples with $1 - \epsilon$
   - leave the label of fake samples unchanged

   $\Rightarrow$ reduce confidence in correct class $\Rightarrow$ regularization effect

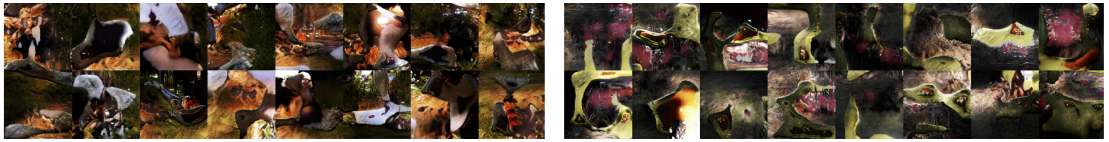
[34] Denton, E. L., Chintala, S., Fergus, R., et al. (2015). Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*, pages 1486–1494

[35] Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242

[36] Goodfellow, I. (2016). Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*

3. virtual batch normalization

- problem of using BN in GAN
  - examples within a minibatch should be independent
  - but BN can cause them to become correlated



- a solution: virtual batch normalization which runs the net twice
  - once on a minibatch of reference examples
    ↑
    sampled once at start of training and never replaced
  - once on the current minibatch of examples to train on
  - BN statistics: computed using both current and reference batches

image source: Goodfellow, I. (2016). NIPS 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*

4. balancing generator/discriminator

- it is fine for D to overpower G
  - implicit density estimation in GAN: correct only when D is optimal
- practical tips based on this fact
  - update D k times while updating G only once
  - D has more layers/filters than G
- but when D becomes too confident/accurate:
  - gradient for G can explode/vanish
- the right way to solve this problem
  - not to limit the power of D but to use
  - (for exploding gradient) one-sided label smoothing
  - (for vanishing) parameterization where gradient does not vanish

Outline

Generative Adversarial Networks

Summary

Appendix: Wasserstein GAN

Summary

- GANs: do not work with an explicit density function
  - take a game-theoretic approach
  - learn to generate from training distribution through 2-player game
- advantages
  - beautiful, state-of-the-art samples
  - can parallelize sampling of generated data
  - no need to approximate likelihood by introducing lower bound
- disadvantages
  - tricker/more unstable to train (*e.g.* non-convergence, mode collapse)
  - cannot solve inference queries such as $p(\boldsymbol{x})$, $p(\boldsymbol{z} \mid \boldsymbol{x})$
- areas of research
  - better loss function, more stable training (Wasserstein GAN, LSGAN, ...)
  - handling sample quality-variance trade-off
  - novel applications of GAN

Recap: generative models

- autoregressive models (*e.g.* PixelRNN/PixelCNN, WaveNet)
  - explicit density model/optimizes exact likelihood
  - good samples
  - but inefficient sequential generation
- variational autoencoder (VAE)
  - optimize variational lower bounds on likelihood
  - useful latent representation/inference queries
  - but current sample quality is not the best
- generative adversarial network (GAN)
  - game-theoretic approach
  - best samples
  - but tricky and unstable to train/no inference queries

Outline

Generative Adversarial Networks

Summary

Appendix: Wasserstein GAN

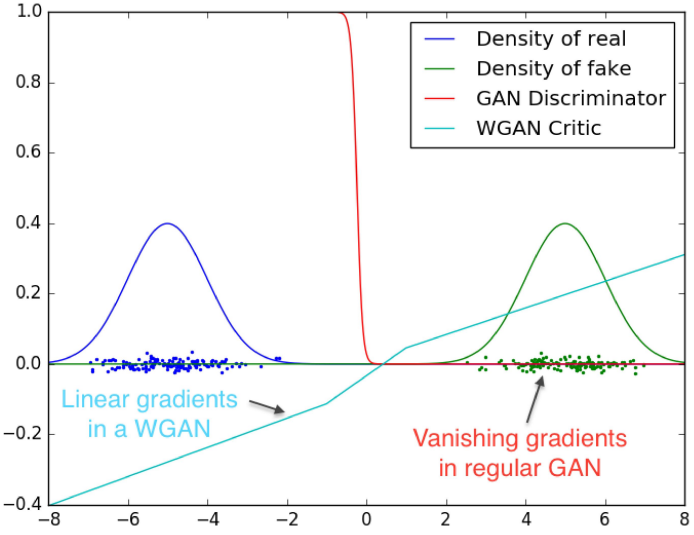
Wasserstein GAN (WGAN)[37]

- proposes a **new cost function** using Wasserstein distance
  - has smoother gradient everywhere
  - improves stability of training
  - correlates with quality of generated images
- name (and role) change: discriminator $\rightarrow$ critic
  - GAN discriminator: predict the probability of generated images being real
  - WGAN critic: scores the realness/fakeness of a given image
- additional properties
  - learns no matter generator is performing or not
  - less sensitive to model architecture/hyperparameters

[37] Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223

WGAN advantages

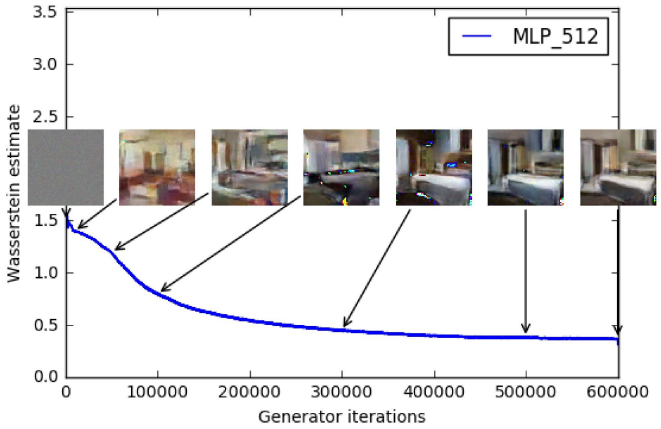
- **improved training stability**
  - suffer less from mode collapsing
  - generator can still learn even when critic performs well

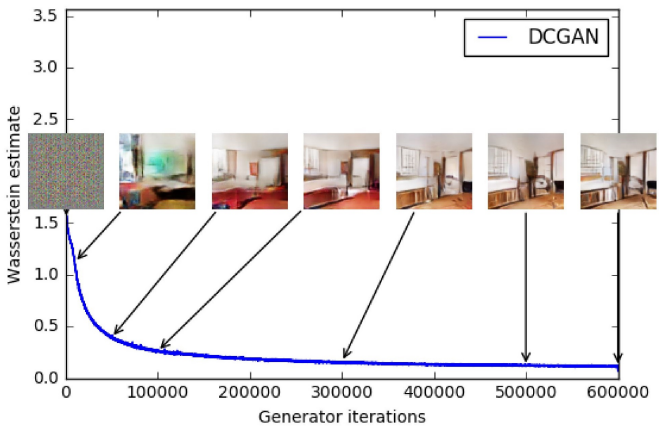


- optimal discriminator and critic
  - when learning to differentiate two Gaussians
- traditional GAN discriminator
  - saturates and results in vanishing gradients
- WGAN critic
  - provides very clean gradients on all parts of the space

image source: Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223

- Wasserstein loss: correlates well with image quality
  - loss curve over time along with generated images





  - left: MLP generator + DCGAN critic
  - right: DCGAN generator + DCGAN critic

image source: Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223

Wasserstein distance

- a measure of the distance between probability distributions

  aka earth mover's distance (EM distance)

- intuition: EM distance = min energy cost of moving/transforming
  - a pile of dirt in the shape of one distribution to the shape of the other
  - the cost = (amount of dirt moved) $\times$ (moving distance)

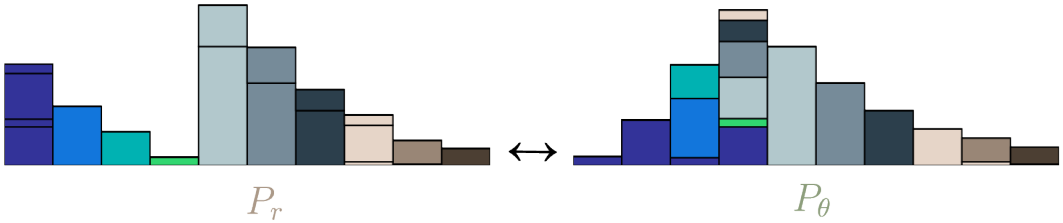


image source: https://vincentherrmann.github.io/blog/wasserstein/

- definition:

$$W(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} \mathbb{E}_{(x,y)\sim\gamma} \left[||x - y|| \right] \tag{5}$$

  - $\Pi(P_r, P_g)$: set of all joint distributions $\gamma(x, y)$[38]
  - $\gamma(x, y)$: how much "mass" must be transported from x to y in order to transform distribution P_r into distribution P_g

- note:
  - infimum in (5): highly intractable[39]

[38] whose marginals are respectively P_r and P_g

[39] Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223

Recall: Lipschitz continuity

- the function f in Wasserstein metric should satisfy $||f||_L \leq K$

  i.e. f should be K-Lipschitz continuous

- a real-valued function $f : \mathbb{R} \mapsto \mathbb{R}$ is called *K-Lipschitz continuous* if
  - there exists a real constant $K \geq 0$ s.t. for all $x_1, x_2 \in \mathbb{R}$,

$$|f(x_1) - f(x_2)| \leq K|x_1 - x_2|$$

  - K: Lipschitz constant for f
  - everywhere continuously differentiable $\overset{\Rightarrow}{\not\Leftarrow}$ Lipschitz continuous

  e.g. $f(x) = |x|$

Duality and parametrization

- by Kantorovich-Rubinstein duality, W is equivalent to

$$W(P_r, P_\theta) = \sup_{||f||_L \leq 1} \mathbb{E}_{x \sim P_r}[f(x)] - \mathbb{E}_{x \sim P_\theta}[f(x)] \tag{6}$$

  - $||f||_L \leq 1$ indicates f should be *1-Lipschitz continuous*
  - supremum in (6): still intractable but easier to approximate ▸ Link

- suppose a parameterized function family $\{f_w\}_{w \in \mathcal{W}}$ (w : weights)
  - assuming these functions are all K-Lipschitz for some K gives

$$\begin{aligned} \max_{w \in \mathcal{W}} \mathbb{E}_{x \sim P_r}[f_w(x)] - \mathbb{E}_{x \sim P_\theta}[f_w(x)] &\leq \sup_{||f||_L \leq K} \mathbb{E}_{x \sim P_r}[f(x)] - \mathbb{E}_{x \sim P_\theta}[f(x)] \\ &= K \cdot W(P_r, P_\theta) \end{aligned}$$

- in Wasserstein GAN
  - we will learn f_w (now called "critic" instead of discriminator)
  - no need to know exact K (absorbed into learning rate tuning) ▸ Link

Wasserstein GAN

- want to train $P_\theta = g_\theta(Z)$ to match P_r
  - Z : random variable with distribution $p(z)$
  - g_θ : parametric function ($\mathcal{Z} \mapsto \mathcal{X}$; typically a neural net)

- Wasserstein distance and its derivative:

$$W(P_r, P_\theta) = \max_{w \in \mathcal{W}} \mathbb{E}_{x \sim P_r}[f_w(x)] - \mathbb{E}_{z \sim p(z)}[f_w(g_\theta(z))]$$

$$\nabla_\theta W(P_r, P_\theta) = -\mathbb{E}_{z \sim p(z)}[\nabla_\theta f_w(g_\theta(z))]$$

- training procedure:[40]

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: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.

Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$
    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
    $w \leftarrow \text{clip}(w, -c, c)$
  end for
  Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$
  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while

[40] Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223

Enforcing K-Lipschitz continuity

how to enforce K-Lipschitz continuity of f_w during training:

- original WGAN: simple but very practical trick
  - after every gradient update, clamp w to a small window (*e.g.* [-0.01,0.01])

  $\Rightarrow$ results in a compact parameter space $\mathcal{W}$

- WGAN-GP (WGAN with gradient penalty)[41]
  - penalizes gradient norm moving away from its target value 1[42]

$$L = \underbrace{\mathbb{E}_{\tilde{x}\sim P_g}[D(\tilde{x})] - \mathbb{E}_{x\sim P_r}[D(x)]}_{\text{original critic loss}} + \lambda \underbrace{\mathbb{E}_{\hat{x}\sim P_{\hat{x}}}[(||\nabla_{\hat{x}} D(\hat{x})||_2 - 1)^2]}_{\text{gradient penalty}}$$

where $\hat{x}$ is randomly sampled from $\tilde{x}$ and x

$$\hat{x} = t\tilde{x} + (1-t)x \qquad \text{with } 0 \le t \le 1$$

[41] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017). Improved training of wasserstein gans. In *Advances in neural information processing systems*, pages 5767–5777

[42] a differentiable function is 1-Lipschitz iff it has gradients with norm at most 1 everywhere