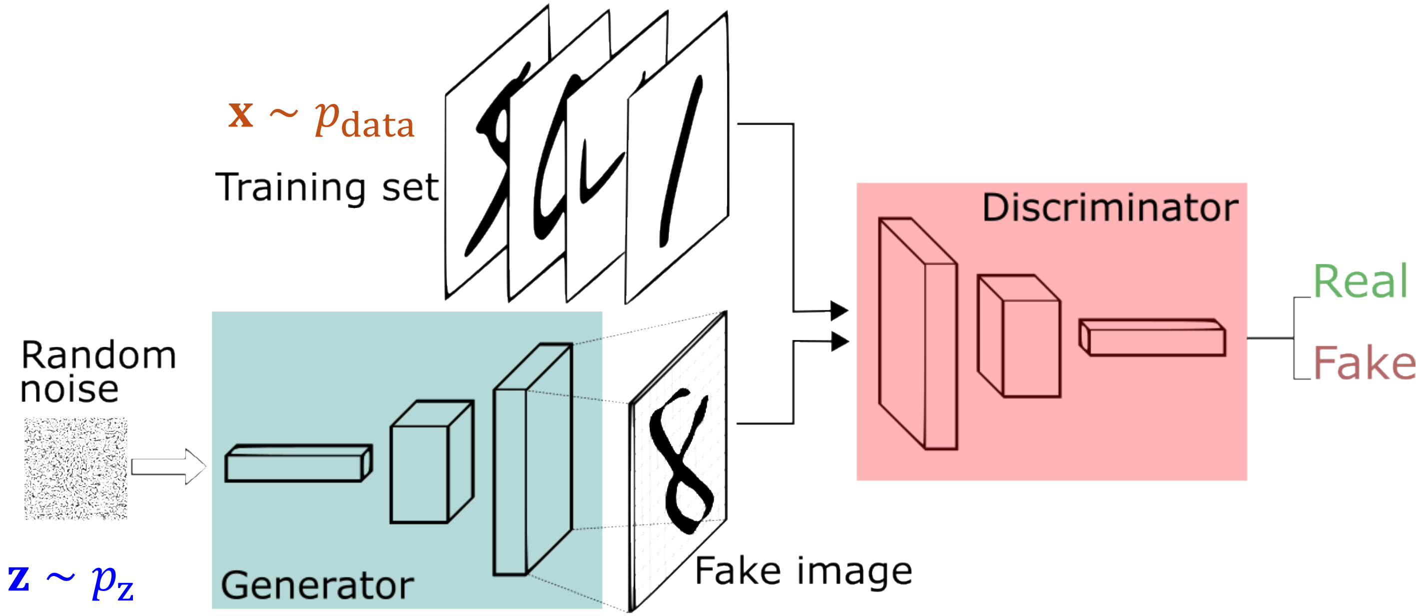
ECE 57000

Generative Adversarial Networks

Chaoyue Liu

Fall 2024

GAN architecture



G (Generator) takes a random variable **z** as input, outputs an object that is the same shape as the training sample **x**

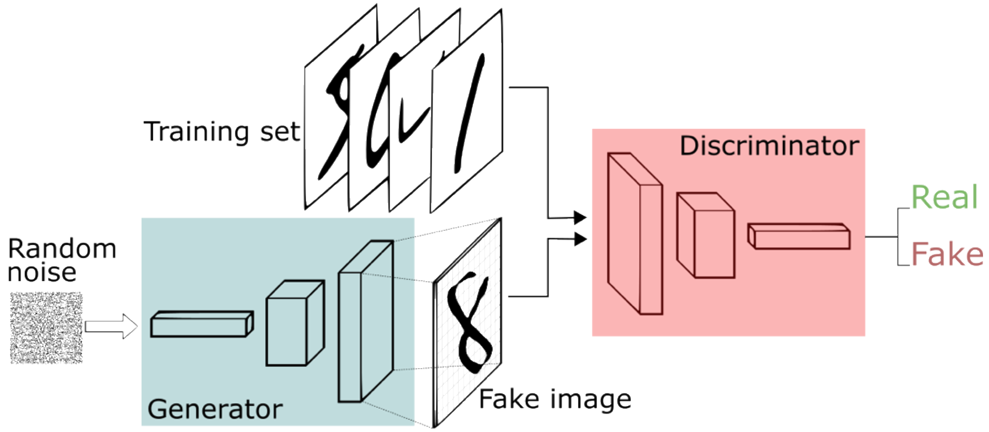
D (Discriminator) is a probabilistic binary classifier, i.e., output is probability between 0 and 1; labels = {real, fake}

GAN architecture

GAN objective intuition: Competitive game between two players

- Intuition: Competitive game between two players
  - Counterfeiter is trying to avoid getting caught
  - Police is trying to catch counterfeiter

- Analogy with GANs
  - Counterfeiter = Generator G
  - Police = Discriminator D



GAN loss function

The rule of the game:

$$\min_{G} \max_{D} \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}\left[\log\left(1 - D\left(G(\mathbf{z})\right)\right)\right]$$

-- Connection with binary classification loss (logistic loss)

Discriminator D: trying to maximize the log-likelihood (minimize logistic loss), in order to distinguish fake objects (generated by G) from real objects (from training set).

Generator G: try to minimize the objective function, via generating more real-like objects

During this game, both G and D get better and better!

-- Analogy: prey and predator

GAN theory

$$\min_{G} \max_{D} \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}\left[\log\left(1 - D\left(G(\mathbf{z})\right)\right)\right]$$

First, let's look at the inner maximization problem (fixing G for now):

- The discriminator seeks to be an optimal classifier

$$D^* = \arg\max_{D} \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}\left[\log\left(1 - D\left(G(\mathbf{z})\right)\right)\right]$$

- **Given a fixed $\boldsymbol{G}$**, the optimal discriminator is the optimal Bayesian classifier (derive!)

$$D^*(\mathbf{x}) = p^*(y = 1|\mathbf{x}) = \frac{p_{data}(\mathbf{x})}{p_{data}(\mathbf{x}) + p_g(\mathbf{x})}$$

GAN theory

Given that the inner maximization is perfect, the outer minimization is equivalent to Jensen Shannon Divergence **for the given *G***: (derive!)

$$\mathcal{L}(G) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log D^*(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}\left[\log\left(1 - D^*\left(G(\mathbf{z})\right)\right)\right]$$

$$= 2\,JSD\left(p_{data}, p_g\right) + constant$$

- **<u>Jensen Shannon Divergence</u>** is a symmetric version of KL divergence

$$JSD\left(p(x), q(x)\right)$$
$$= \frac{1}{2}KL\left(p(x), \frac{1}{2}\left(p(x) + q(x)\right)\right) + \frac{1}{2}KL\left(q(x), \frac{1}{2}\left(p(x) + q(x)\right)\right)$$
$$= \frac{1}{2}KL(p(x), m(x)) + \frac{1}{2}KL(q(x), m(x))$$

- JSD also has the property of KL:

$$JSD\left(p_{data}, p_g\right) \geq 0, \text{and} = 0\ if\ and\ only\ if\ p_{data} = p_g$$

GAN theory

Thus, the optimal generator G^* will generate samples that perfectly mimic the true distribution:

$$\arg\min_G \mathcal{L}(G) = \arg\min_G JSD(p_{data}, p_g)$$

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}\left[\log\left(1 - D\left(G(\mathbf{z})\right)\right)\right]$$

Two optimization components:

- Inner maximization over discriminator approximates JSD
- Outer minimization minimizes this JSD approximation

In theory, we can then update our G via

$$\nabla_G \mathcal{L}(G) = \nabla_G JSD\left(p_{\mathrm{data}}, p_g\right) = \nabla_G \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}}}[\log D^*(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}\left[\log\left(1 - D^*\left(G(\mathbf{z})\right)\right)\right]$$

*However, after updating G, *the max must be solved again* (at least for this theory to hold).

GAN: training

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

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 $p_g(\boldsymbol{z})$.

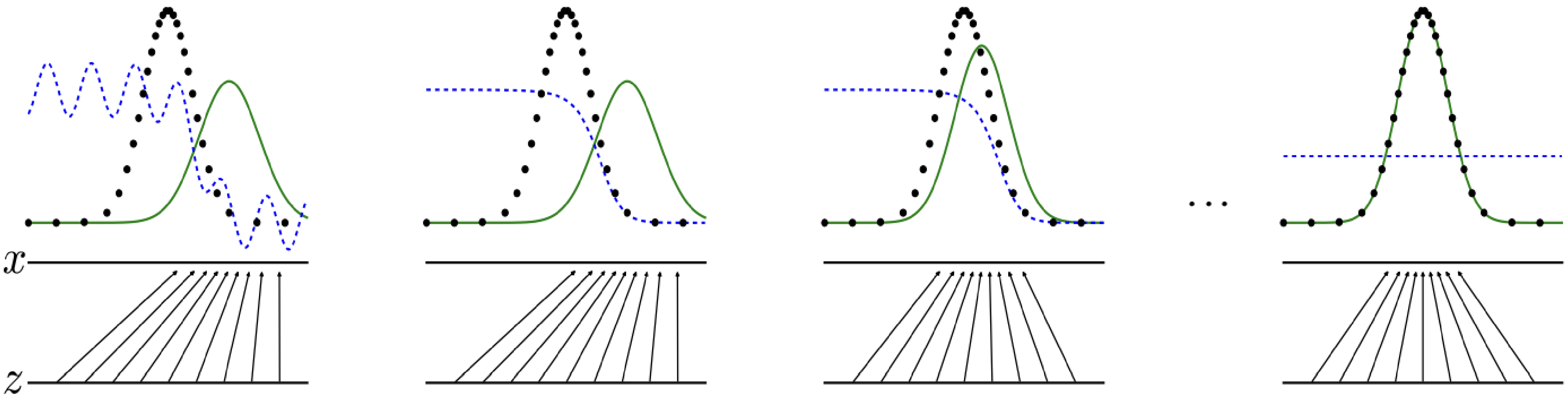
  • Update the generator by descending its stochastic gradient:

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

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Goodfellow, et al. *Generative adversarial networks*. https://arxiv.org/pdf/1406.2661



(a) (b) (c) (d)

Another perspective

$$\begin{aligned}\nabla_G \mathcal{L}(G) &= \nabla_G \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}\left[\log\left(1 - D\big(G(\mathbf{z})\big)\right)\right] \\ &= \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}\left[\frac{-1}{1 - D\big(G(\mathbf{z})\big)} \color{red}{\nabla_G D\big(G(\mathbf{z})\big)}\right]\end{aligned}$$

The gradient $\nabla_G \mathcal{L}(G)$ for generator *G* contains the gradient of discriminator *D* w.r.t. its input.

- When updating, generator *G* knows the discriminator's preference
- Then, generator *G* updates its faking "technology" according to *D*'s preference

In addition, the better performance of *G* gives more challenge for *D*, hence encouraging *D* to perform better.

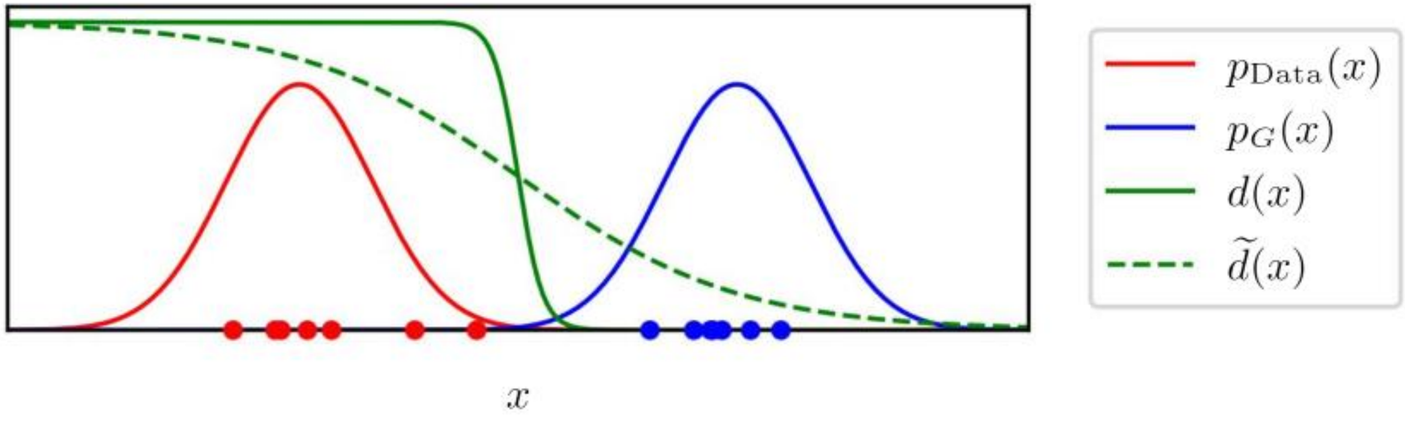
This is more like a *collaboration (or student & teacher)* instead of competing

GAN: training

Common problems with GANs: **Vanishing gradients** for generator caused by a discriminator that is "too good"

- Vanishing gradient means $\nabla_G \mathcal{L}(D, G) \approx 0$.
  - Gradient updates do not improve G

$$\nabla_G \mathcal{L}(G) = \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}\left[\frac{1}{1 - D\big(G(\mathbf{z})\big)} \nabla_G D\big(G(\mathbf{z})\big)\right]$$



GAN: training

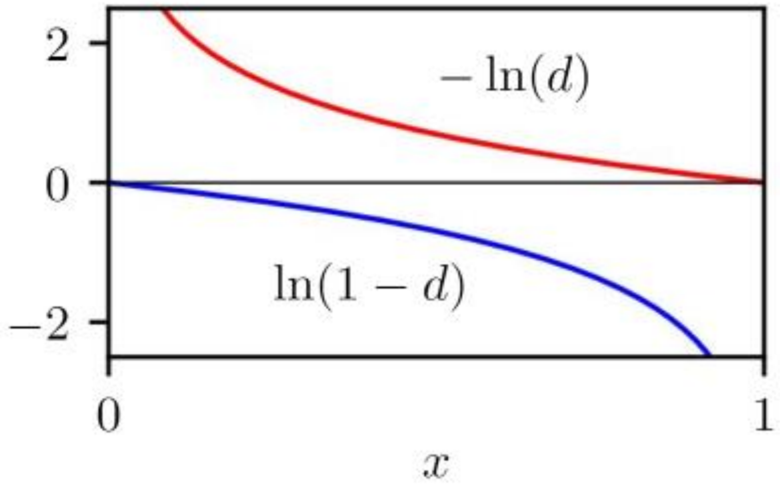
Common problems with GANs: **Vanishing gradients** for generator caused by a discriminator that is "too good"

- Vanishing gradient means $\nabla_G \mathcal{L}(D, G) \approx 0$.
  - Gradient updates do not improve G

Modified minimax loss for generator (derive!)

$$\min_G \mathbb{E}_{p_g}\left[\log\left(1 - D\big(G(z)\big)\right)\right]$$

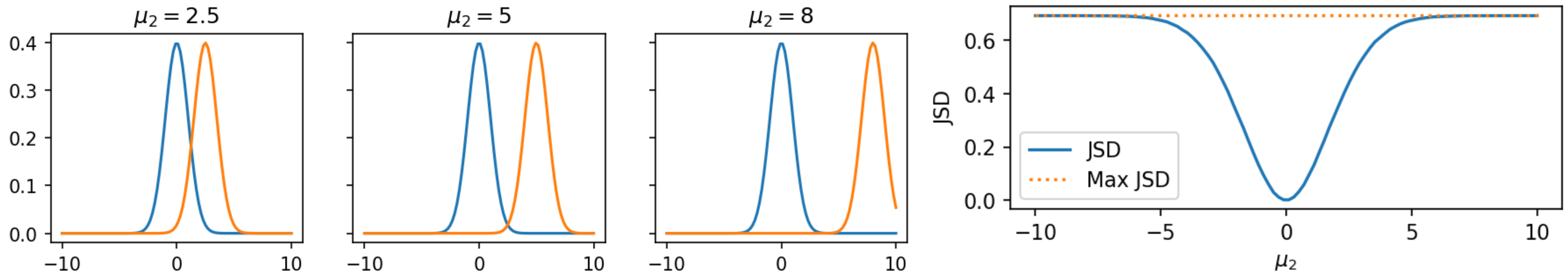
$$\Rightarrow \quad \min_G \mathbb{E}_{p_z}\left[-\log D\big(G(z)\big)\right]$$



GAN: training

Common problems with GANs: **Vanishing gradients** for generator caused by a discriminator that is "too good"

- Vanishing gradient means $\nabla_G \mathcal{L}(D, G) \approx 0$.
  - Gradient updates do not improve G
- Theoretically, this is an issue of JSD

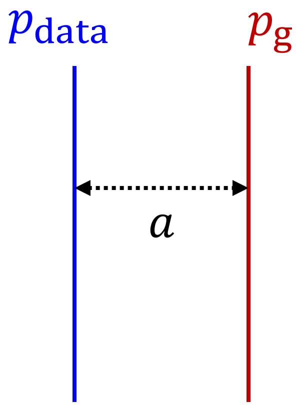


- Practically, careful balance during training required:
  - Optimizing D too much leads to vanishing gradient
  - **But** training too little means it is not close to JSD

GAN: training

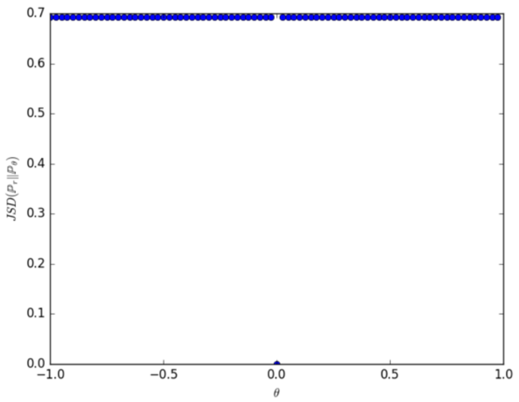
Observation: In many practical cases, p_{data} and p_g lie on low dimensional manifolds, and have disjoint supports

Example:



$$KL(p_{data}, p_g) = \begin{cases} +\infty, & if\ a \neq 0 \\ 0 & if\ a = 0 \end{cases}$$

$$JS(p_{data}, p_g) = \begin{cases} \log 2\,, & if\ a \neq 0 \\ 0 & if\ a = 0 \end{cases}$$



KL divergence:

$$KL(p, q) = \mathbb{E}_p[\log p - \log q]$$

JS divergence:

$$JS(p, q) = \frac{1}{2} KL(p, m) + \frac{1}{2} KL(q, m)$$

$$m = (p + q)/2$$

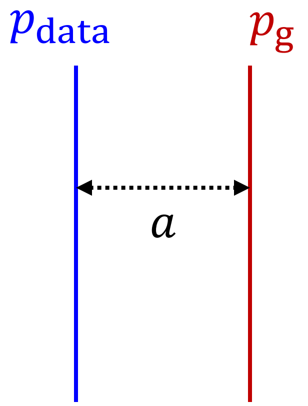
Wasserstein distance

Wasserstein-1 distance (or *Earth-mover* distance):

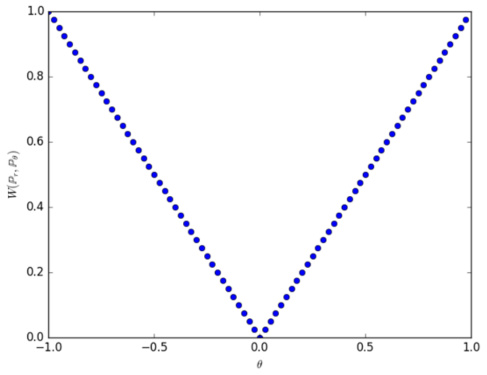
$$W(p,q) = \inf_{\gamma \in \Pi(p,q)} \mathbb{E}_{(x,z)\sim\gamma}[||x - z||]$$

$\Pi(p,q)$ denotes the set of joint distribution $\gamma(x,z)$ whose marginals are respectively p and q.

Intuition: minimal "cost" to transport "mass" from distribution p to q.



$$W\left(p_{data}, p_g\right) = |a|$$



Wasserstein GAN

Kantorovich-Rubinstein duality:

$$W(p,q) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim p}[f(x)] - \mathbb{E}_{x \sim q}[f(x)]$$

Wasserstein GANs

$$\min_G \max_D \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}}[f_D(\mathbf{x})] - \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}}\left[f_D\left(G(\mathbf{z})\right)\right]$$

where D is 1-Lipschitz (special smoothness property).

Compare to the original GAN:

$$\min_G \max_D \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}}\left[\log\left(1 - D\left(G(\mathbf{z})\right)\right)\right]$$

Arjovsky, et al. *Wasserstein generative adversarial networks*. ICML 2017.

Wasserstein GAN

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

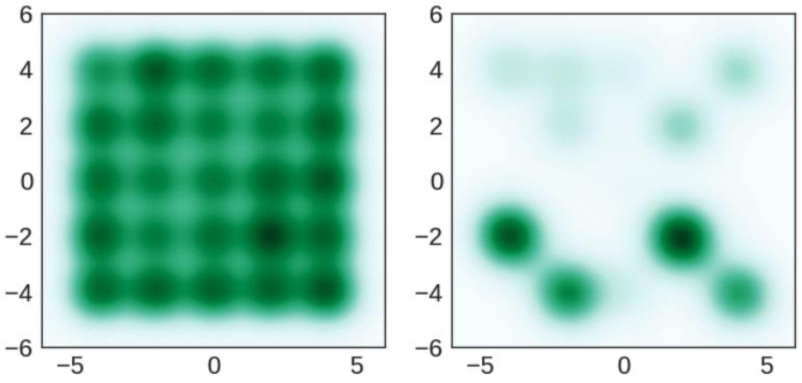
Maintaining smoothness

Improvement:
Gradient penelty

GAN: training

Common problems with GANs: **Mode collapse** hinders diversity of samples

- Wasserstein GANs
- Unrolled GANs
  - Trained on multiple discriminators simultaneously



(f) True Data (g) GAN

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

http://papers.nips.cc/paper/6923-veegan-reducing-mode-collapse-in-gans-using-implicit-variational-learning.pdf

GAN: training

Common problems with GANs: **Failure to converge** because of minimax and other instabilities

- Loss function may oscillate or never converge

- Disjoint support of distributions
  - Optimal JSD is constant value (i.e., no gradient information)
  - Add noise to discriminator inputs (similar to VAEs)

- Regularization of parameter weights
  - https://arxiv.org/pdf/1705.09367

GAN: evaluation

Evaluation of GANs is quite challenging

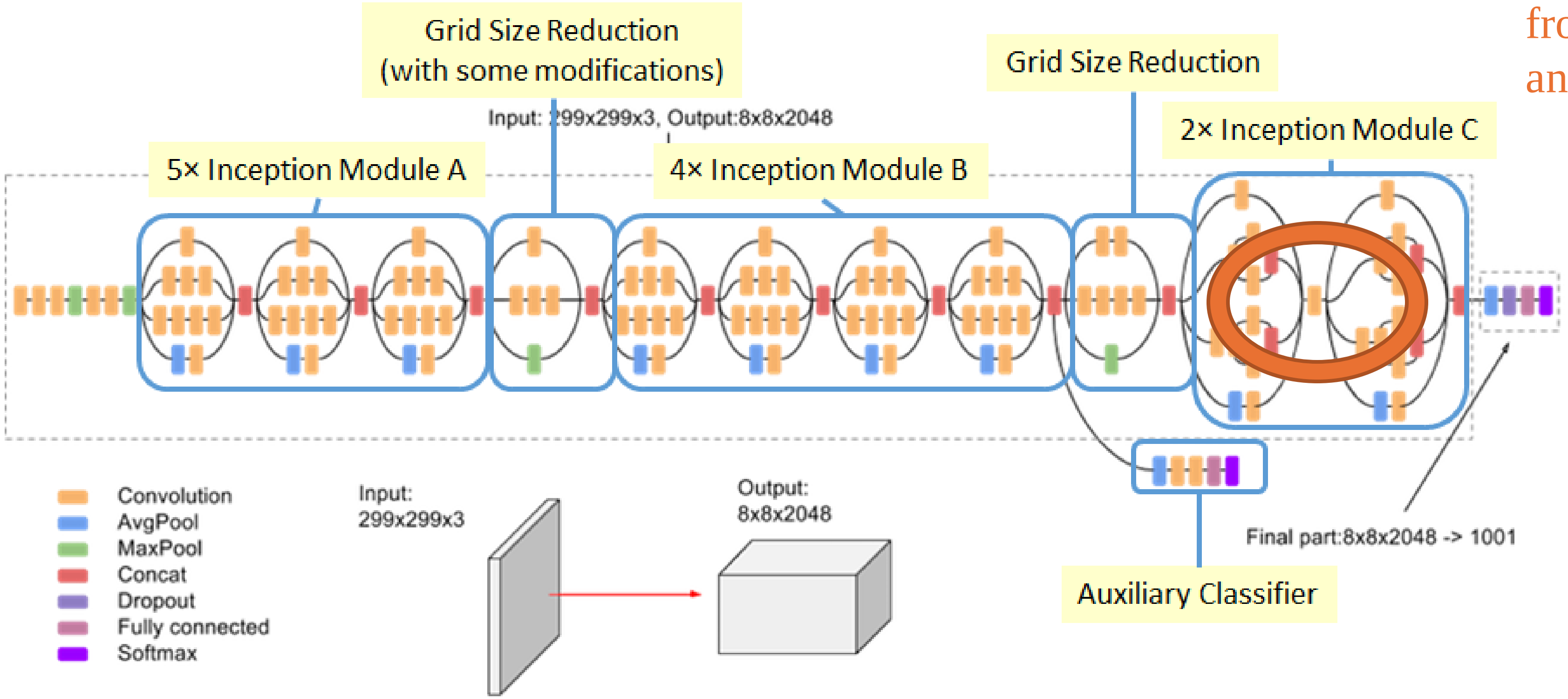
- In autoencoder models, we could use test log likelihood to evaluate

- There is no objective loss function used to train the generator of a GAN, and therefore, no way to objectively assess the progress of the training and the relative or absolute quality of the model from loss alone.

- Visually inspect image samples
  - Qualitative and biased
  - Hard to compare between methods

GAN: evaluation

Common GAN metrics compare latent representations of pre-trained InceptionV3 network



Extract features from last layers and compare

https://medium.com/@sh.tsang/review-inception-v3-1st-runner-up-image-classification-in-ilsvrc-2015-17915421f77c

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 2818-2826).

Inception score (IS) considers both clarity of images and diversity of images

- Extract Inception-V3 distribution of predicted labels, $p_{inceptionV3}(y|x_i), \forall x_i$
- Images should have "meaningful objects", i.e., $p(y|x_i)$ has **low entropy**
- The average over all generated images should be diverse, i.e., $p(y) = \frac{1}{n}\sum_i p(y|x_i)$ should have **high entropy**
- Combining these two (higher is better):

$$IS = \exp\left(\mathbb{E}_{p_g}\left[KL\left(p(y|x), p(y)\right)\right]\right)$$

  - Consider if $p(y|x) = p(y)$, i.e., all images give the same distribution over images
  - Either, all images are indistinct (e.g., they don't look like images so predictions are random)
  - Or, all images are the same (e.g., all images are dog)

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

<u>Frechet inception distance (FID)</u> compares latent features from generated and real images

- Problem: Inception score ignores real images
  - Generated images may look nothing like real images

- Extract latent representation at last pooling layer of Inception-V3 network ($d = 2048$)

- Compute empirical mean and covariance for real and generated from latent representation

$$\mu_{data}, \Sigma_{data} \text{ and } \mu_g, \Sigma_g$$

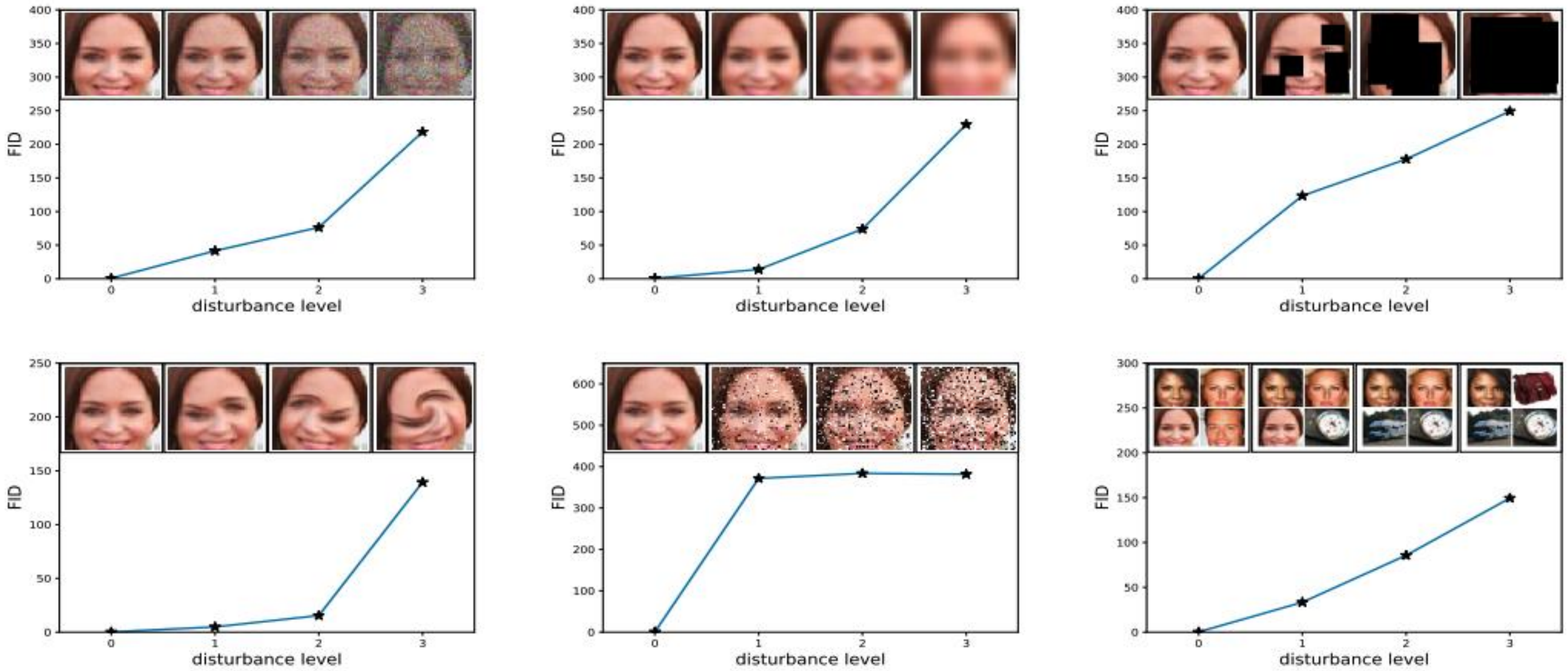
- FID score:

$$FID = \left\|\mu_{data} - \mu_g\right\|_2^2 + \mathrm{Tr}\left(\Sigma_{data} + \Sigma_g - 2\left(\Sigma_{\mathrm{data}}\Sigma_g\right)^{-\frac{1}{2}}\right)$$

  - Considers both mean **and covariance** of latent distribution

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in neural information processing systems* (pp. 6626-6637).

<u>FID</u> correlates with common distortions and corruptions



Randomly add ImageNet
images unlike celebrity dataset

Figure from Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in neural information processing systems* (pp. 6626-6637).

GAN Summary:
Impressive innovation with strong empirical results but hard to train

- Good empirical results on generating sharp images

- Training is challenging in practice

- Evaluation of generative models is challenging (and still unsolved in my opinion)