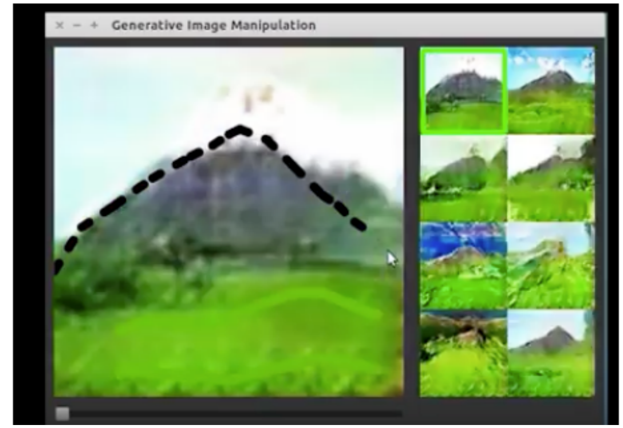
Generative Adversarial Networks (GAN)

ECE57000: Artificial Intelligence

David I. Inouye

<u>Why</u> study generative models?

- Sketching realistic photos



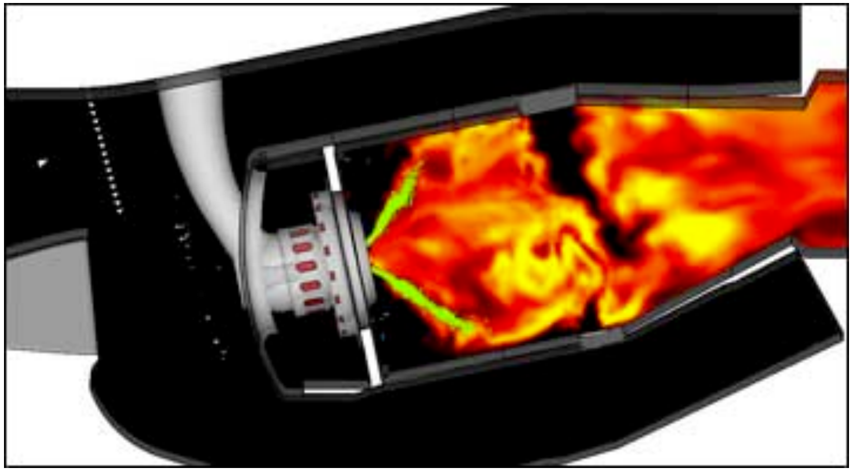
- Style transfer

- Super resolution

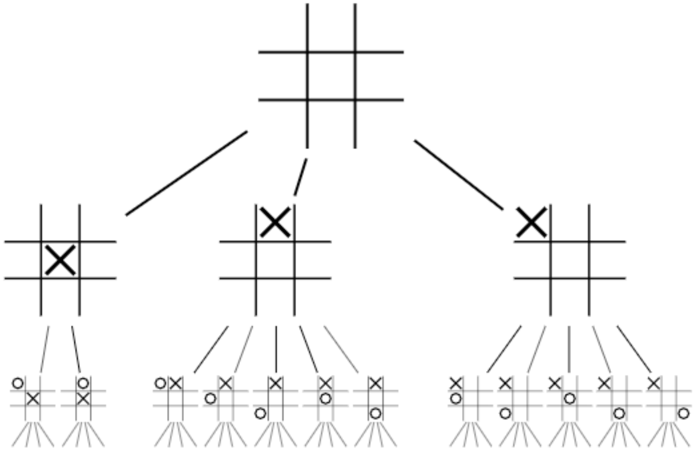
Much of material from: Goodfellow, 2012 tutorial on GANs.

Why study generative models?

- Emulate complex physics simulations to be faster



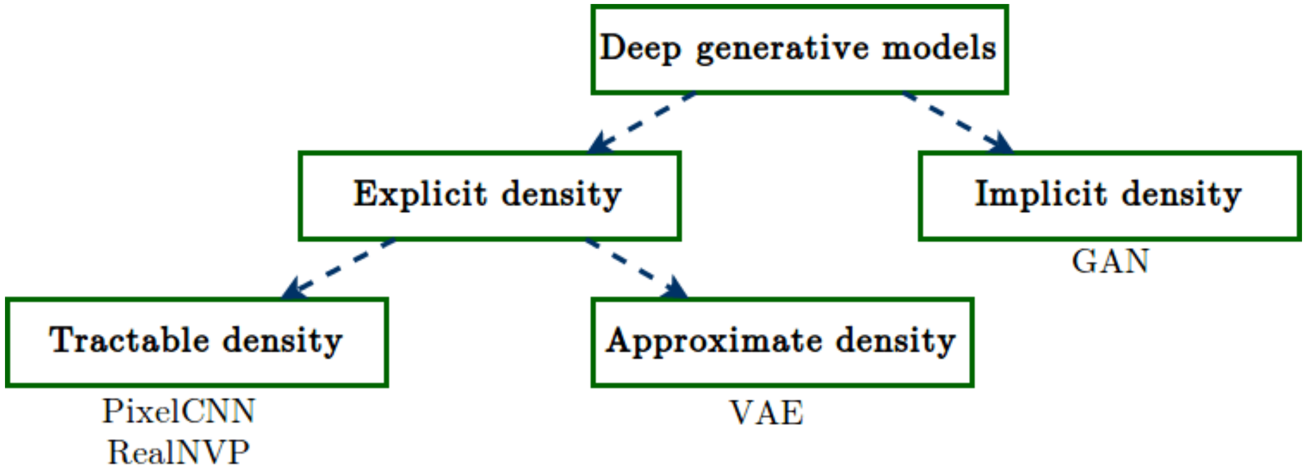
- Reinforcement learning - Attempt to model the real world so we can simulate possible futures



Much of material from: Goodfellow, 2012 tutorial on GANs.

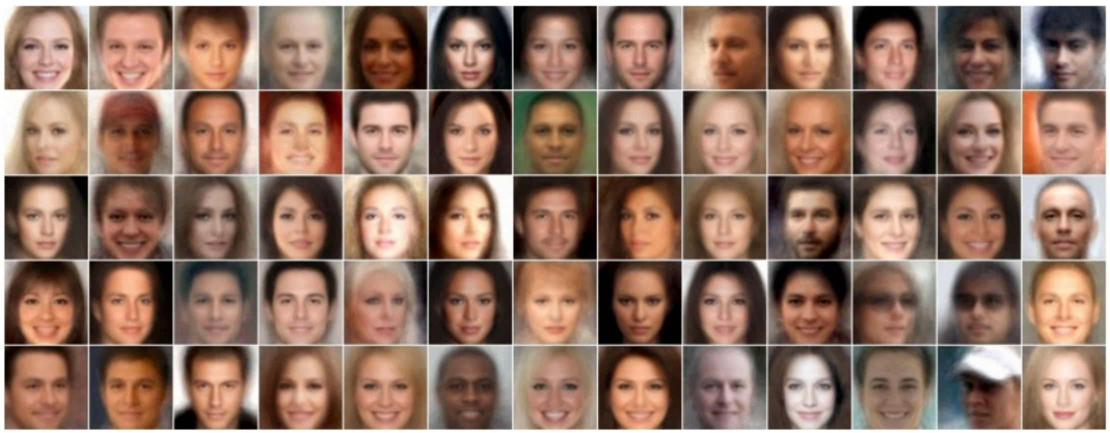
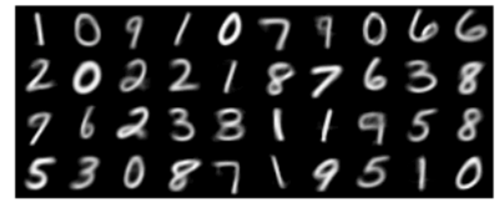
How do we learn these generative models?

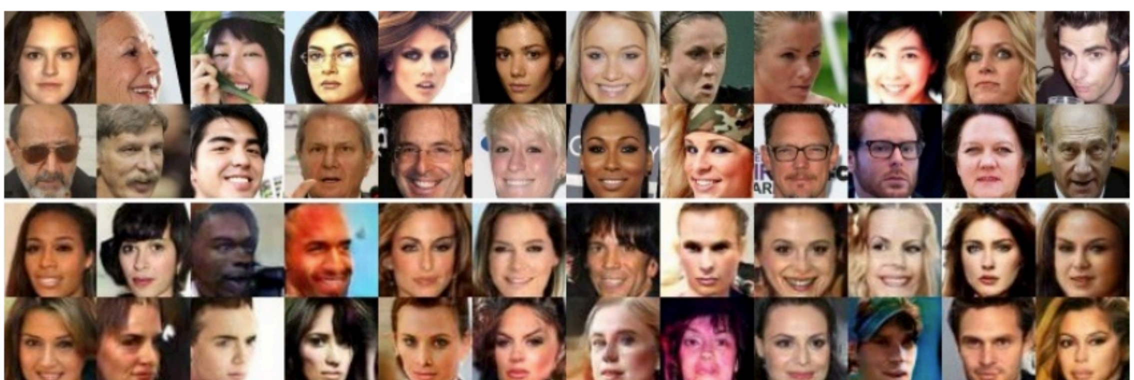
- Primary classical approach is MLE
  - Density function is explicit parameterized by θ
  - Examples: Gaussian, Mixture of Gaussians
- Problem: Classic methods cannot model very high dimensional spaces like images
  - Remember a 256x256x3 image is roughly 200k dimensions

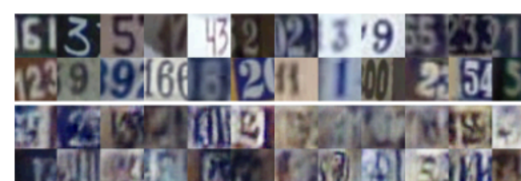


Maybe not a problem: GMMs compared to GANs
http://papers.nips.cc/paper/7826-on-gans-and-gmms.pdf

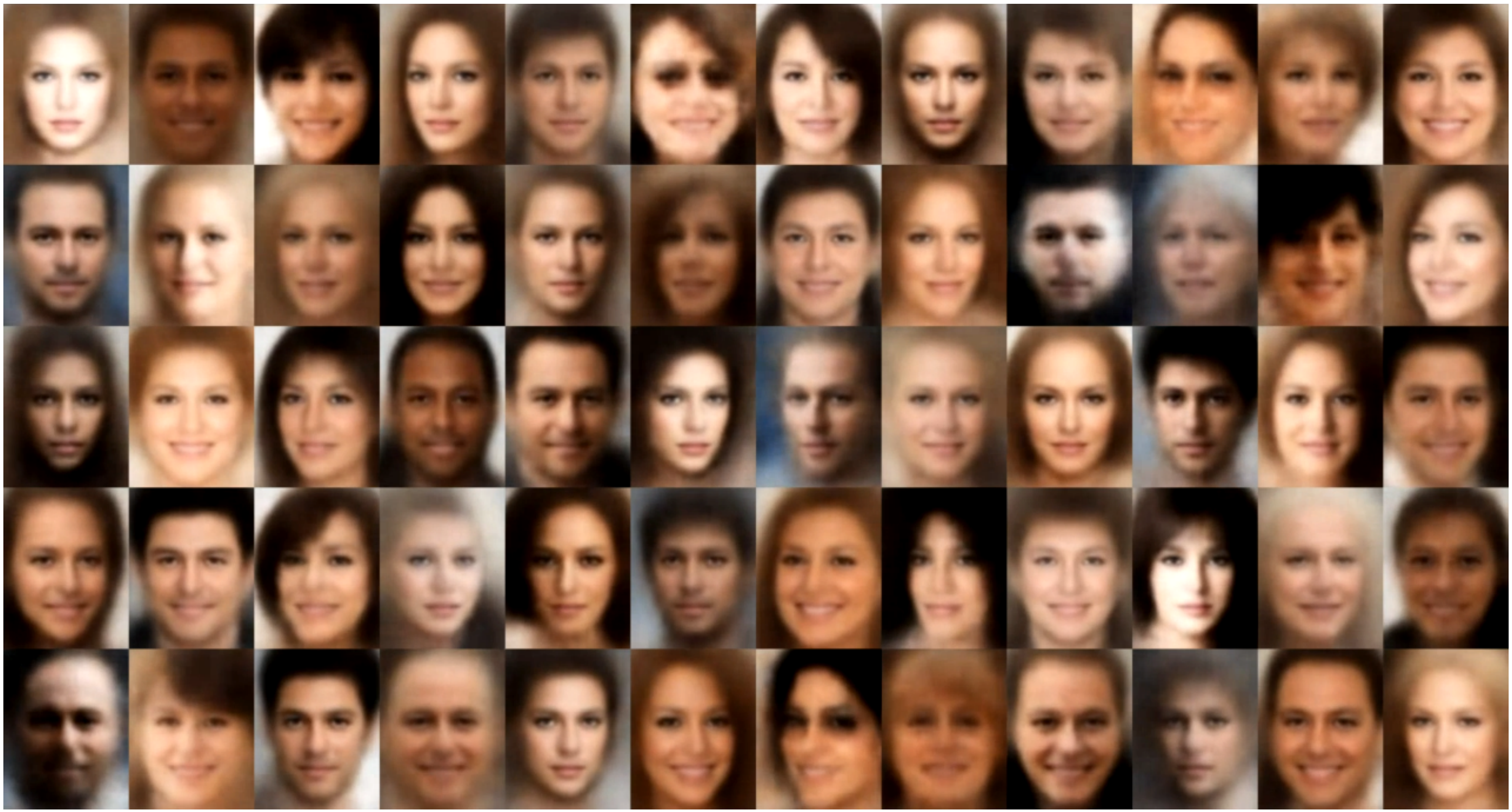
- Which one is based on GANs?



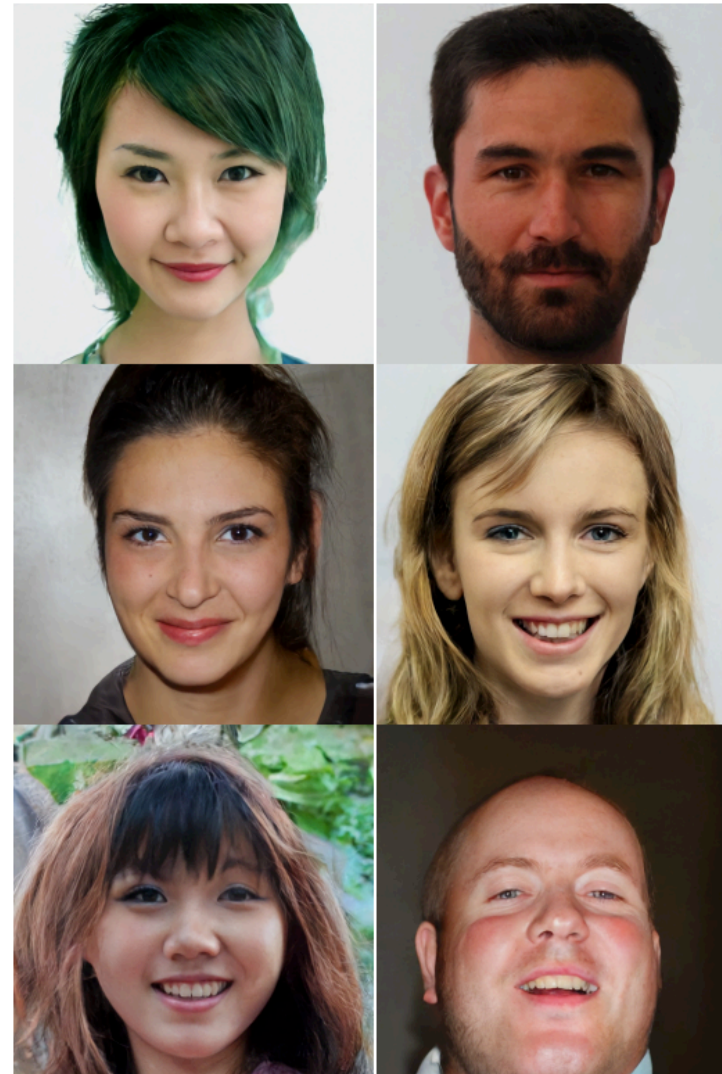
VAEs are one way to create a generative model for images though images are blurry



https://github.com/WojciechMormul/vae

Maybe not a drawback…
VQ-VAE-2 at *NeurIPS 2019*

Generated high-quality images (probably don't ask how long it takes to train this though…)



Razavi, A., van den Oord, A., & Vinyals, O. (2019). Generating diverse high-fidelity images with vq-vae-2. In *Advances in Neural Information Processing Systems* (pp. 14866-14876).

Newer (not necessarily better) approach: Train generative model <u>**without explicit density**</u>

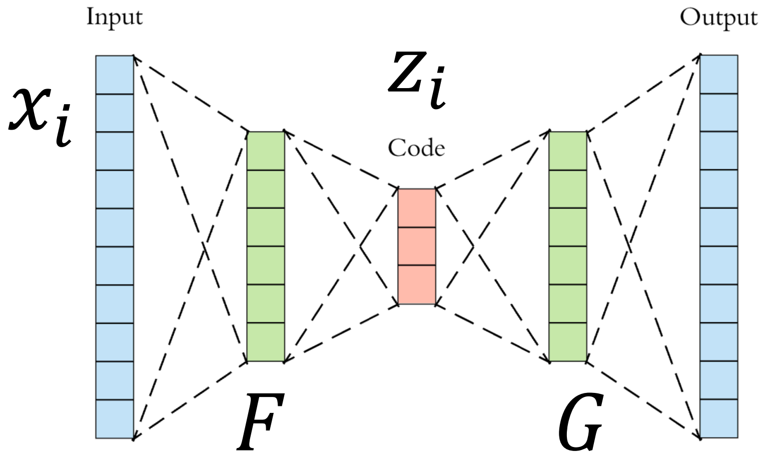
- GMMs and VAEs had **explicit** density function (i.e., mathematical formula for density $p(x;\theta)$)

- In GANs, we just try learn a sample **generator**
  - **Implicit** density ($p(x)$ exists but cannot be written down)

- Sample generation is simple
  - $z \sim p_z$, e.g., $z \sim \mathcal{N}(0, I) \in \mathbb{R}^{100}$
  - $G_\theta(z) = \hat{x} \sim \hat{p}_g(x)$
  - Where G is a deep neural network

Unlike VAEs, GANs do not (usually) have inference networks

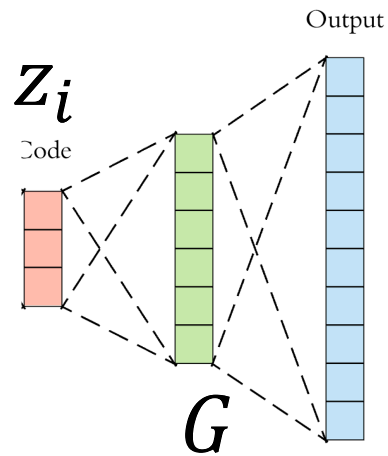
VAE



$$\tilde{x}_i \sim p(x_i|G(z_i))$$

$$L(x_i, \tilde{x}_i)$$

GAN



$$\tilde{x}_i = G(z_i)$$

$$L(x_i, \tilde{x}_i)?$$

No pair of original and reconstructed
How to train?

Key challenge: Comparing two distributions known <u>only through samples</u>

- In GANs, we cannot produce pairs of original and reconstructed samples as in VAEs

- But have samples from original data and generated distributions

$$D_{\mathrm{data}} = \{x_i\}_{i=1}^{n}, \qquad x_i \sim p_{\mathrm{data}}(x)$$
$$D_{\mathrm{g}} = \{x_i\}_{i=1}^{\infty}, \qquad x_i \sim p_{\mathrm{g}}(x|G)$$

- How do we compare two distributions only through samples?
  - Fundamental, bigger than generative models

Could we use KL divergence as in MLE training?

- We can approximate the KL term up to A constant

$$KL\left(p_{data}(x), p_g(x)\right) = \mathbb{E}_{p_{data}}\left[\log \frac{p_{data}(x)}{p_g(x)}\right]$$

$$= \mathbb{E}_{p_{data}}[-\log p_g(x)] + \mathbb{E}_{p_{data}}[\log p_{data}(x)]$$

$$\approx \widehat{\mathbb{E}}_{p_{data}}[-\log p_g(x)] + constant$$

$$= \sum_i -\log p_g(x_i) + constant$$

$$= \sum_i -\log p_g(x_i) + constant$$

Because GANs do not have an explicit density, we cannot compute this KL divergence.

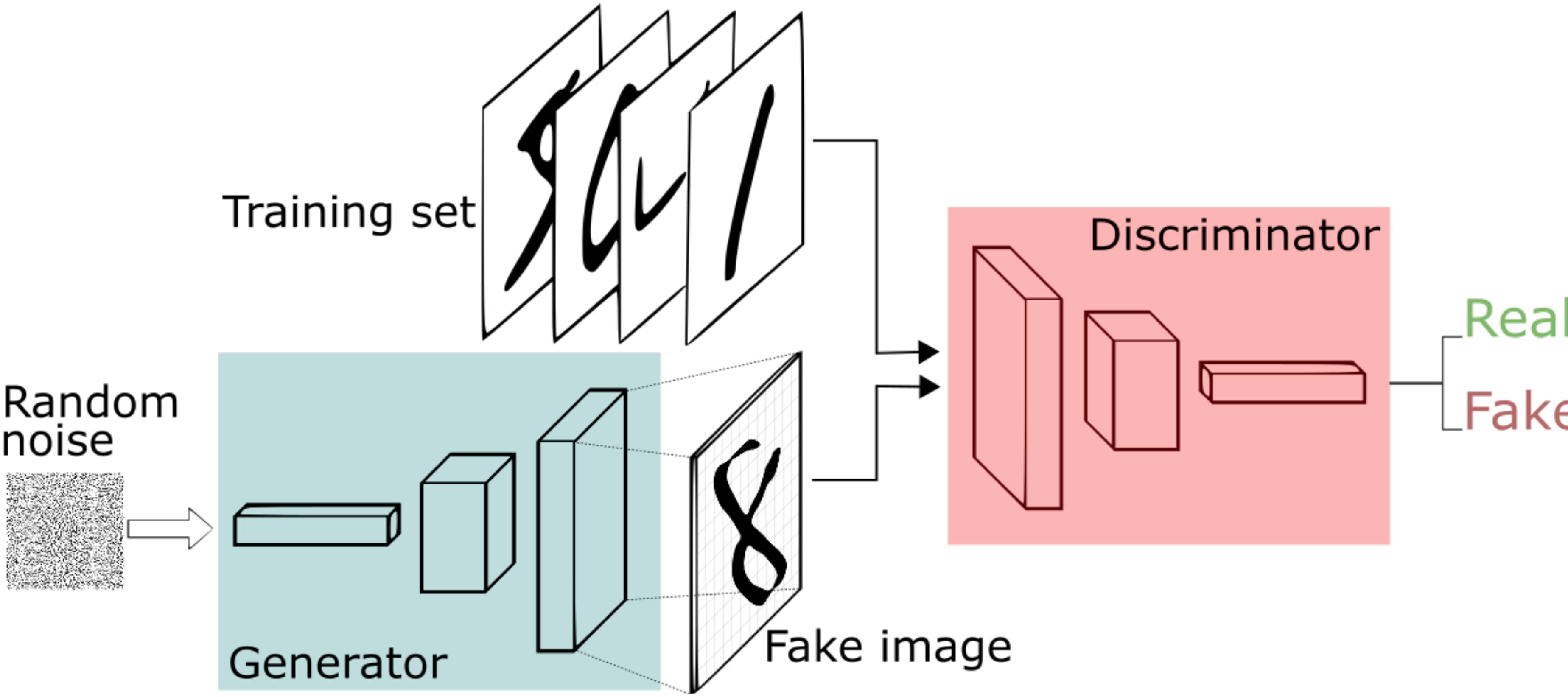
GANs introduce the idea of <u>adversarial training</u> for estimating the distance between two distributions

- GANs approximate the Jensen-Shannon Divergence (JSD) closely related to KL divergence
- GANs optimize both the JSD approximation and the generative model simultaneously
  - A different type of two network setup
- Broadly applicable for comparing distributions only through samples

How do we learn this implicit generative model? 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 denoted G
  - Police = Discriminator denoted D

How do we learn this implicit generative model?
Train two deep networks simultaneously



https://www.freecodecamp.org/news/an-intuitive-introduction-to-generative-adversarial-networks-gans-7a2264a81394/

How do we learn this implicit generative model? Intuition: Competitive game between two players

- Minimax: "Minimize the ***worst case*** (max) loss"
  - Counterfeiter goal: "Minimize chance of getting caught assuming the best possible police."

- Abstract formulation as minimax game

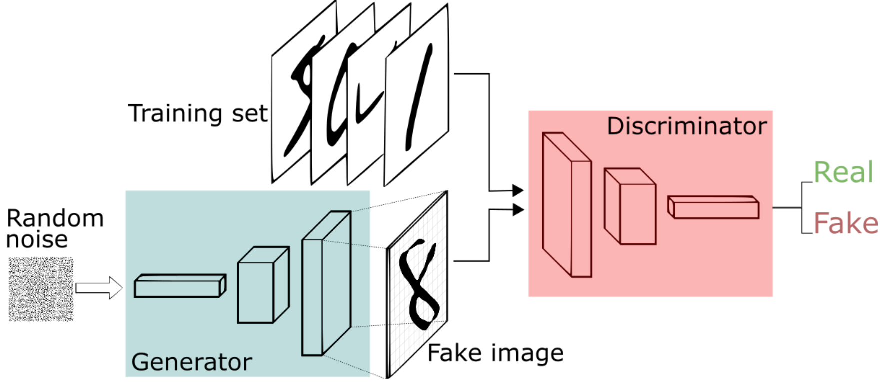
$$\min_G \max_D \mathbb{E}_{x\sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z\sim p_z}\left[\log\left(1 - D\big(G(z)\big)\right)\right]$$

- The value function is

$$\mathrm{V(D, G)} = \mathbb{E}_{x\sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z\sim p_z}\left[\log\left(1 - D\big(G(z)\big)\right)\right]$$

- Key feature: No restrictions on the networks D and G

The discriminator seeks to be optimal classifier



- Let's look at the inner maximization problem

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

- **Given a fixed *G***, the optimal discriminator is the optimal Bayesian classifier

$$D^*(\tilde{x}) = p^*(\tilde{y} = 1|\tilde{x}) = \frac{p_{data}(\tilde{x})}{p_{data}(\tilde{x}) + \hat{p}_g(\tilde{x})}$$

Derivation for the optimal discriminator

- **Given a fixed G**, the optimal discriminator is the optimal classifier between images
- $C(G) = \max_D \mathbb{E}_{x\sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z\sim p_z}\left[\log\left(1 - D\left(G(z)\right)\right)\right]$
- $= \max_D \mathbb{E}_{x\sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{x\sim \hat{p}_g}\left[\log\left(1 - D(x)\right)\right]$

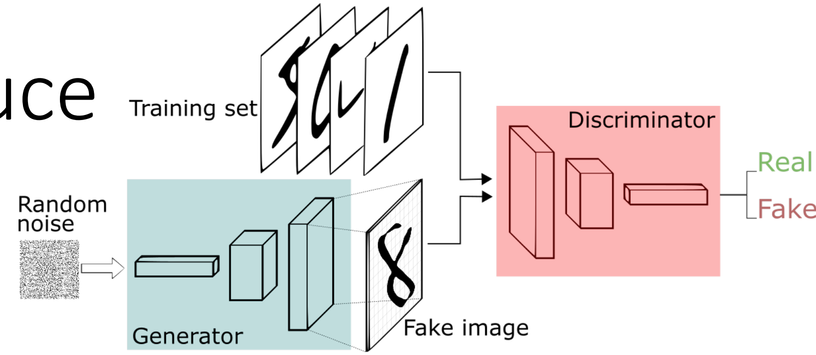
Opposite of reparametrization trick! ☺

- $= \max_D \mathbb{E}_{\tilde{x},\tilde{y}}\left[\tilde{y}\log D(\tilde{x}) + (1-\tilde{y})\log\left(1 - D(\tilde{x})\right)\right]$

$$\text{where } p(\tilde{x},\tilde{y}) = p(\tilde{y})p(\tilde{x}|\tilde{y}), \qquad p(\tilde{y}) = \frac{1}{2},$$
$$p(\tilde{x}|y=0) = p_g(x), \qquad p(\tilde{x}|y=1) = p_{data}(x)$$

- $= \max_D \mathbb{E}_{\tilde{x},\tilde{y}}[\log p_D(\tilde{y}|\tilde{x})]$
- $D^*(\tilde{x}) = p^*(\tilde{y}=1|\tilde{x}) = \frac{p(\tilde{x},\tilde{y}=1)}{p(\tilde{x})} = \frac{\frac{1}{2}p_{\text{data}}(\tilde{x})}{\frac{1}{2}p_{\text{data}}(\tilde{x})+\frac{1}{2}\hat{p}_g(\tilde{x})} = \frac{p_{\text{data}}(\tilde{x})}{p_{\text{data}}(\tilde{x})+\hat{p}_g(\tilde{x})}$

The generator seeks to produce data that is like real data



- **Given that the inner maximization is perfect**, the inner minimization is equivalent to Jensen Shannon Divergence:

$$C(G) = \max_D V(D,G)$$
$$= 2\, JSD\left(p_{data}, \hat{p}_g\right) + constant$$

- **Jensen Shannon Divergence** 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:

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

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

$$\arg\min_G C(G) = \arg\min_G JSD\left(p_{data}, \hat{p}_g\right)$$

Derivation of inner maximization being equivalent to JSD

- $C(G) = \max_D \mathbb{E}_{x\sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z\sim p_z}\left[\log\left(1 - D\big(G(z)\big)\right)\right]$
- $= \max_D \mathbb{E}_{x\sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{x\sim \hat{p}_g}\left[\log\big(1 - D(x)\big)\right]$
- $= \mathbb{E}_{x\sim p_{\text{data}}}[\log D^*(x)] + \mathbb{E}_{x\sim \hat{p}_g}\left[\log\big(1 - D^*(x)\big)\right]$
- $= \mathbb{E}_{\tilde{x}\sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\tilde{x})}{p_{\text{data}}(\tilde{x})+\hat{p}_g(\tilde{x})}\right] + \mathbb{E}_{\tilde{x}\sim \hat{p}_g}\left[\log\left(1 - \frac{p_{\text{data}}(\tilde{x})}{p_{\text{data}}(\tilde{x})+\hat{p}_g(\tilde{x})}\right)\right]$
- $= \mathbb{E}_{\tilde{x}\sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\tilde{x})}{p_{\text{data}}(\tilde{x})+\hat{p}_g(\tilde{x})}\right] + \mathbb{E}_{\tilde{x}\sim \hat{p}_g}\left[\log\left(\frac{\hat{p}_g(\tilde{x})}{p_{\text{data}}(\tilde{x})+\hat{p}_g(\tilde{x})}\right)\right]$
- $= \mathbb{E}_{\tilde{x}\sim p_{\text{data}}}\left[\log \frac{\frac{1}{2}p_{\text{data}}(\tilde{x})}{\frac{1}{2}\big(p_{\text{data}}(\tilde{x})+\hat{p}_g(\tilde{x})\big)}\right] + \mathbb{E}_{\tilde{x}\sim \hat{p}_g}\left[\log\left(\frac{\frac{1}{2}\hat{p}_g(\tilde{x})}{\frac{1}{2}\big(p_{\text{data}}(\tilde{x})+\hat{p}_g(\tilde{x})\big)}\right)\right]$
- $= \mathbb{E}_{\tilde{x}\sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\tilde{x})}{\frac{1}{2}\big(p_{\text{data}}(\tilde{x})+\hat{p}_g(\tilde{x})\big)}\right] + \mathbb{E}_{\tilde{x}\sim \hat{p}_g}\left[\log\left(\frac{\hat{p}_g(\tilde{x})}{\frac{1}{2}\big(p_{\text{data}}(\tilde{x})+\hat{p}_g(\tilde{x})\big)}\right)\right] - \log 4$
- $= 2\, JSD\big(p_{\text{data}}, \hat{p}_g\big) - \log 4$

https://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

What if inner maximization is not perfect?

- Suppose the true maximum is not attained

$$\hat{C}(G) = \widehat{\max_D} \mathbb{E}_{x\sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z\sim p_z}\left[\log\left(1 - D\big(G(z)\big)\right)\right]$$

- Then, $\hat{C}(G)$ becomes a **lower bound** on JSD

$$\hat{C}(G) < C(G) = JSD\big(p_{data}(x), p_{g(x)}\big)$$

- However, the outer optimization is a **minimization**

$$\min_G \max_D V(D, G) \approx \min_G \hat{C}(G)$$

- Ideally, we would want an **upper bound** like in VAEs
- This can lead to significant training instability

Great! But wait… This theoretical analysis depends on critical assumptions

1. Assumptions on possible D and G
   1. Theory – All possible D and G
   2. Reality – Only functions defined by a neural network
2. Assumptions on optimality
   1. Theory – Both optimizations are solved perfectly
   2. Reality – The inner maximization is only solved approximately, and this interacts with outer minimization
3. Assumption on expectations
   1. Theory – Expectations over true distribution
   2. Reality – Empirical expectations over finite sample; for images, much of the high-dimensional space does not have samples

- **GANs can be very difficult/finicky to train**

Excellent online visualization and demo of GANs

- https://poloclub.github.io/ganlab/

Common problems with GANs: <u>Vanishing gradients</u> for generator caused by a discriminator that is "too good"

From: https://developers.google.com/machine-learning/gan/problems

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

- Modified minimax loss for generator (original GAN)

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

- Wasserstein GANs

$$V(D, G) = \mathbb{E}_{p_{data}}[D(x)] - \mathbb{E}_{p_z}\left[D\left(G(z)\right)\right]$$

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

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., & Courville, A. C. (2017). Improved training of wasserstein gans. In *Advances in neural information processing systems* (pp. 5767-5777).

Common problems with GANs: <u>Failure to converge</u> because of minimax and other instabilities

From: https://developers.google.com/machine-learning/gan/problems

- 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

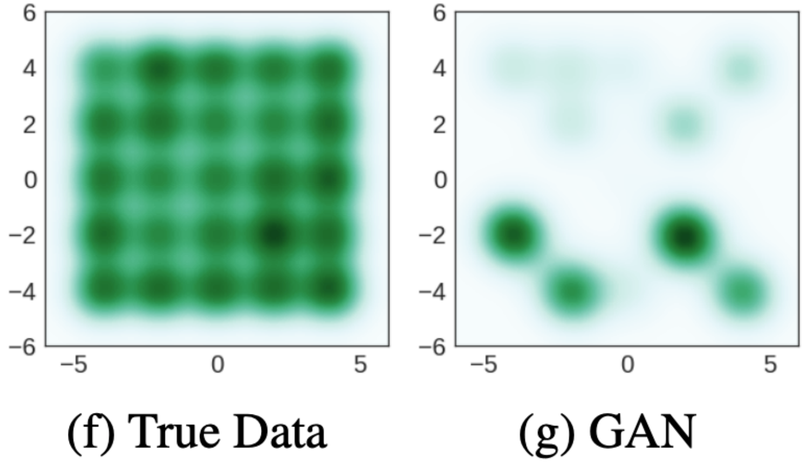
Arjovsky, M., & Bottou, L. (2017). Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*.
https://machinelearningmastery.com/practical-guide-to-gan-failure-modes/

Common problems with GANs: <u>Mode collapse</u> hinders diversity of samples

From: https://developers.google.com/machine-learning/gan/problems

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

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

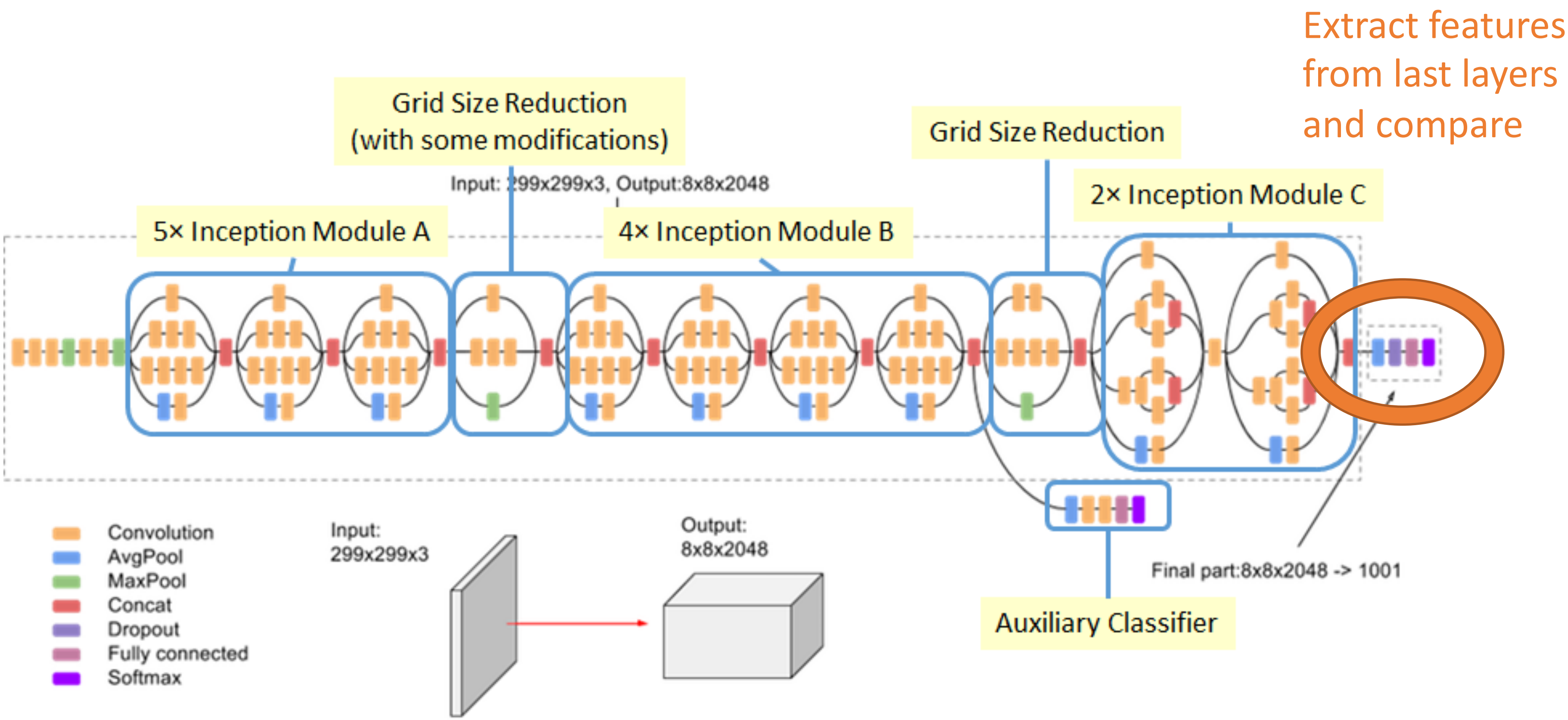
https://software.intel.com/en-us/blogs/2017/08/21/mode-collapse-in-gans

Evaluation of GANs is quite challenging

- In explicit density models, we could use test log likelihood to evaluate
- Without a density model, how do we evaluate?

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

Common GAN metrics compare latent representations of InceptionV3 network



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 image 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):

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

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).

Frechet inception distance (FID) 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

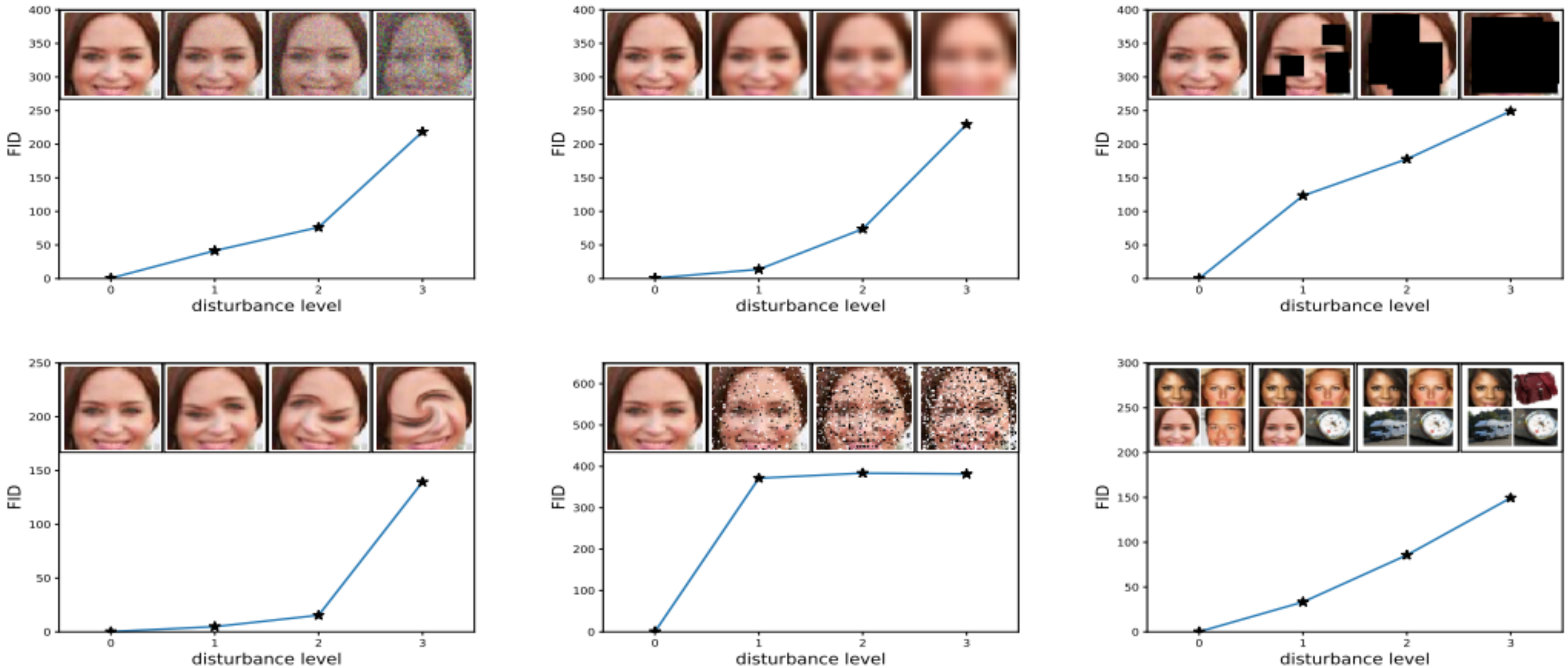
$$\mu_{data}, \Sigma_{data} \text{ and } \mu_g, \Sigma_g$$

- FID score:

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

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).

FID 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 is challenging and unsolved
- Much open research on this topic