Generative Adversarial Networks

Takanori Aoki

19 Sep 2019

Generative Adversarial Networks

Generative Adversarial Nets

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

Abstract

We propose a new framework for estimating generative models via an adversarial process, in which we simultaneously train two models: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G. The training procedure for G is to maximize the probability of D making a mistake. This framework corresponds to a minimax two-player game. In the space of arbitrary functions G and D, a unique solution exists, with G recovering the training data distribution and D equal to $\frac{1}{2}$ everywhere. In the case where G and D are defined by multilayer perceptrons, the entire system can be trained with backpropagation. There is no need for any Markov chains or unrolled approximate inference networks during either training or generation of samples. Experiments demonstrate the potential of the framework through qualitative and quantitative evaluation of the generated samples.

So far...

PixelCNNs define tractable density function, optimize likelihood of training data:

$$p_\theta(x) = \prod_{i=1}^{n} p_\theta(x_i|x_1, ..., x_{i-1})$$

VAEs define intractable density function with latent **z**:

$$p_\theta(x) = \int p_\theta(z)p_\theta(x|z)dz$$

Cannot optimize directly, derive and optimize lower bound on likelihood instead

What if we give up on explicitly modeling density, and just want ability to sample?

GANs: don’t work with any explicit density function!
Instead, take game-theoretic approach: learn to generate from training distribution through 2-player game

Generative Adversarial Networks

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

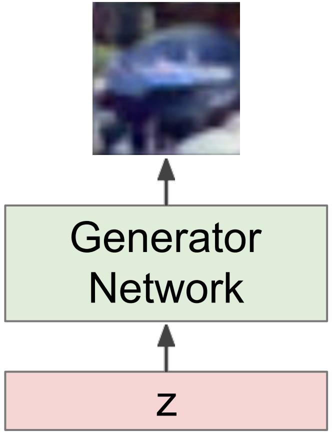
Problem: Want to sample from complex, high-dimensional training distribution. No direct way to do this!

Solution: Sample from a simple distribution, e.g. random noise. Learn transformation to training distribution.

Q: What can we use to represent this complex transformation?

A: A neural network!

Output: Sample from training distribution

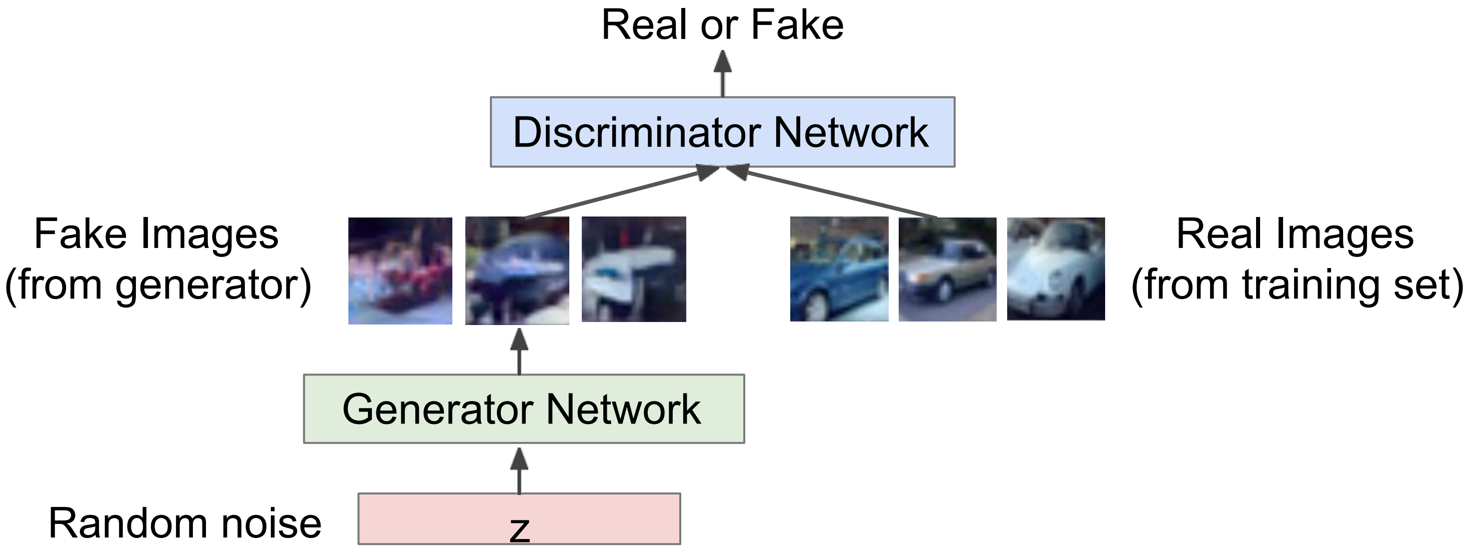


Input: Random noise

Training GANs: Two-player game

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

Generator network: try to fool the discriminator by generating real-looking images
Discriminator network: try to distinguish between real and fake images



Training GANs: Two-player game

Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

Generator network: try to fool the discriminator by generating real-looking images
Discriminator network: try to distinguish between real and fake images

Train jointly in **minimax game**

Discriminator outputs likelihood in (0,1) of real image

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x\sim p_{data}} \log \underbrace{D_{\theta_d}(x)}_{\text{Discriminator output for real data x}} + \mathbb{E}_{z\sim p(z)} \log(1 - \underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\text{Discriminator output for generated fake data G(z)}}) \right]$$

- Discriminator (θ_d) wants to **maximize objective** such that D(x) is close to 1 (real) and D(G(z)) is close to 0 (fake)
- Generator (θ_g) wants to **minimize objective** such that D(G(z)) is close to 1 (discriminator is fooled into thinking generated G(z) is real)

Training GANs: Two-player game

Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:

1. **Gradient ascent** on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

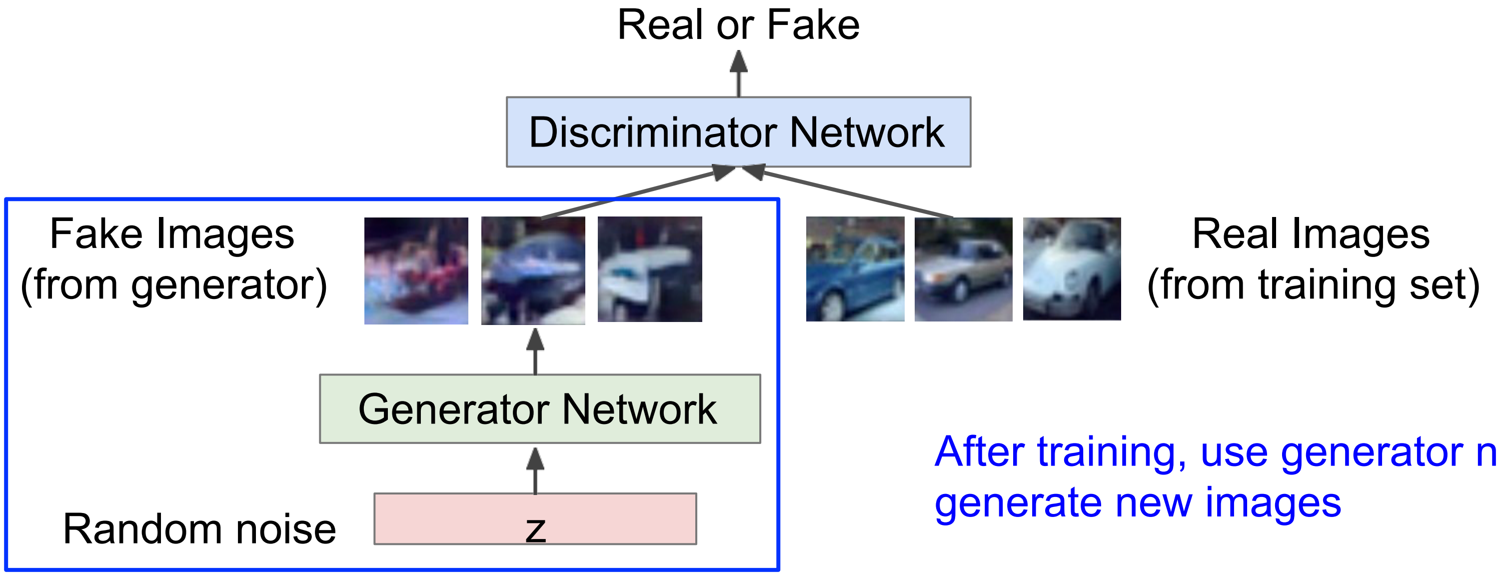
2. **Gradient descent** on generator

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

Training GANs: Two-player game

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

Generator network: try to fool the discriminator by generating real-looking images
Discriminator network: try to distinguish between real and fake images



After training, use generator network to generate new images

Generative Adversarial Networks

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

- Two player minimax game between generator (G) and discriminator (D)
- (D) tries to maximize the log-likelihood for the binary classification problem [data: real (1), generated: fake (0)]
- (G) tries to minimize the log-probability of its samples being classified as "fake" by the discriminator (D)

Generative Adversarial Networks

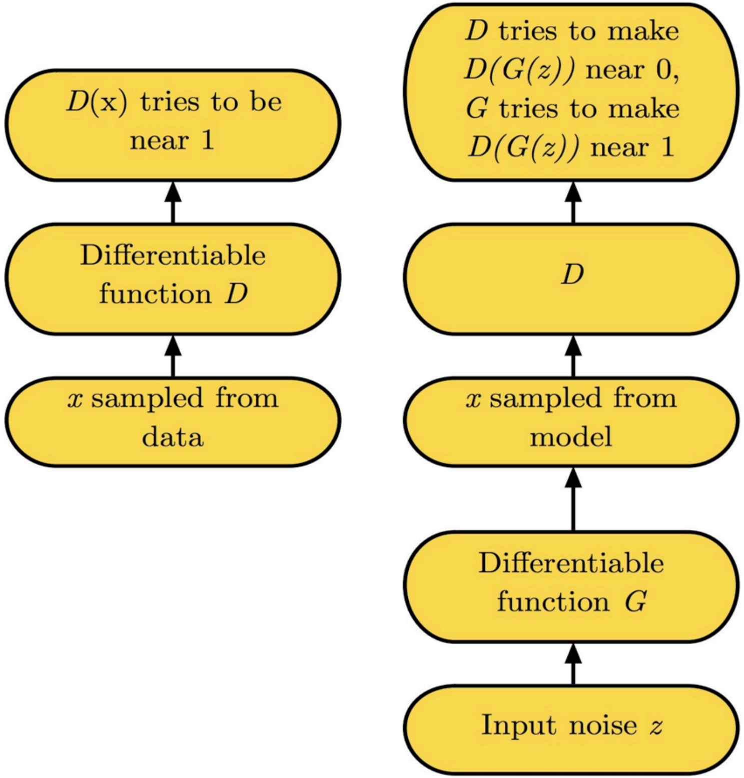
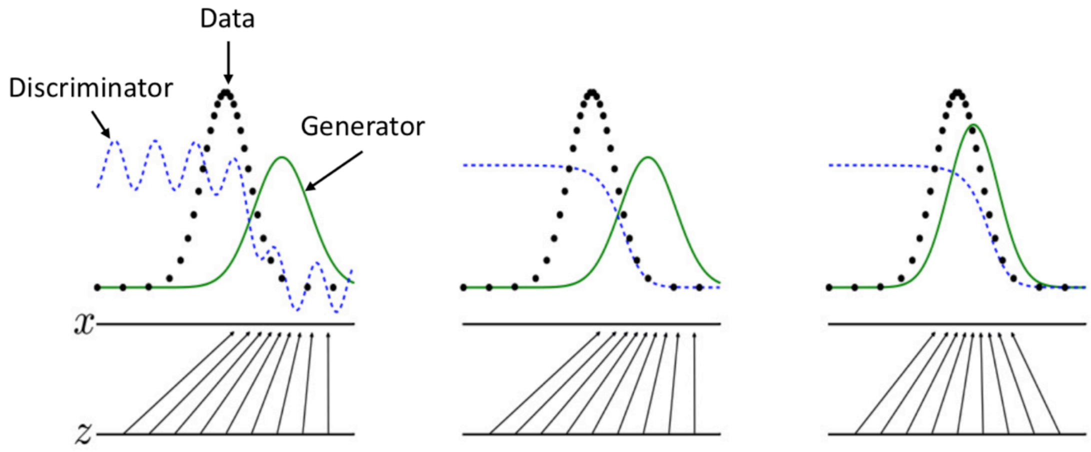


Figure taken from NeurIPS 2016 GAN Tutorial (Goodfellow)

Generative Adversarial Networks

Illustration of GANs



Generator $\min_{G} \max_{D} J^{(D)}$ Discriminator $\max_{D} J^{(D)}$

$$J^{(D)} = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

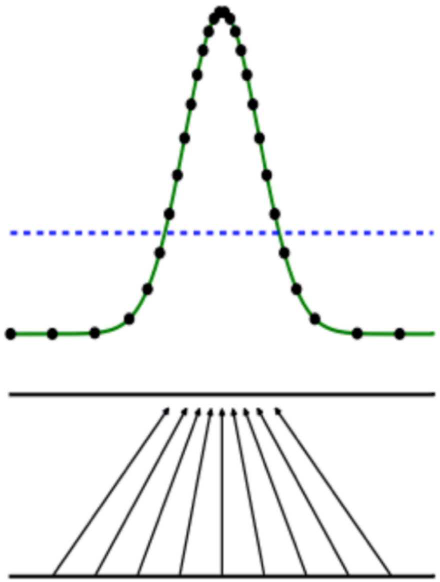
http://wnzhang.net/tutorials/sigir2018/docs/sigir18-irgan-full-tutorial.pdf

Generative Adversarial Networks

Ideal Final Equilibrium

- Generator generates perfect data distribution
- Discriminator cannot distinguish the true and generated data

. . .



http://wnzhang.net/tutorials/sigir2018/docs/sigir18-irgan-full-tutorial.pdf

Generative Adversarial Networks

- What's the optimal discriminator given generated and true distributions?

$$
\begin{aligned}
V(G, D) &= \mathbb{E}_{x \sim p_{\text{data}}} [\log D(x)] + \mathbb{E}_{z \sim p(z)} [\log(1 - D(G(z)))] \\
&= \int_x p_{\text{data}}(x) \log D(x) dx + \int_z p(z) \log(1 - D(G(z))) dz \\
&= \int_x p_{\text{data}}(x) \log D(x) dx + \int_x p_g(x) \log(1 - D(x)) dx \\
&= \int_x \left[p_{\text{data}}(x) \log D(x) + p_g(x) \log(1 - D(x)) \right] dx
\end{aligned}
$$

$$
\nabla_y \left[a \log y + b \log(1 - y) \right] = 0 \implies y^* = \frac{a}{a+b} \quad \forall \quad [a, b] \in \mathbb{R}^2 \backslash [0, 0]
$$

$$
\implies D^*(x) = \frac{p_{\text{data}}(x)}{(p_{\text{data}}(x) + p_g(x))}
$$

Generative Adversarial Networks

- What is the generator objective given the optimal discriminator?

$$
\begin{aligned}
V(G, D^*) &= \mathbb{E}_{x\sim p_{\text{data}}}[\log D^*(x)] + \mathbb{E}_{x\sim p_g}[\log(1 - D^*(x))] \\
&= \mathbb{E}_{x\sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)}\right] + \mathbb{E}_{x\sim p_g}\left[\log \frac{p_g(x)}{p_{\text{data}}(x) + p_g(x)}\right] \\
&= -\log(4) + \underbrace{KL\left(p_{\text{data}} \| \left(\frac{p_{\text{data}} + p_g}{2}\right)\right) + KL\left(p_{\text{g}} \| \left(\frac{p_{\text{data}} + p_g}{2}\right)\right)}_{\text{(Jensen-Shannon Divergence (JSD) of } p_{\text{data}} \text{ and } p_g) \geq 0}
\end{aligned}
$$

$$V(G^*, D^*) = -\log(4) \text{ when } p_g = p_{\text{data}}$$

Generative Adversarial Networks

- What is the generator objective given the optimal discriminator?

Behaviors across divergence measures

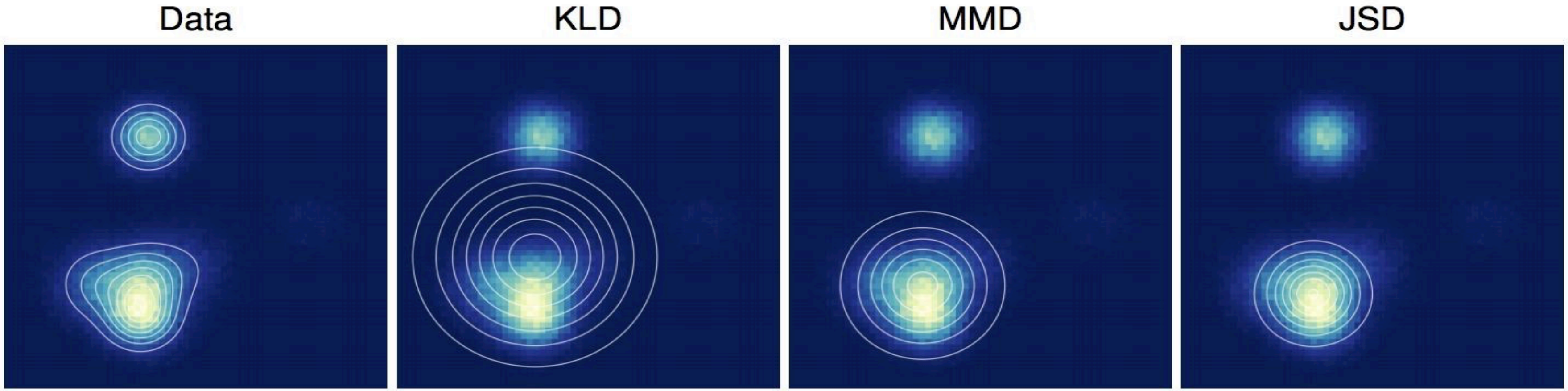
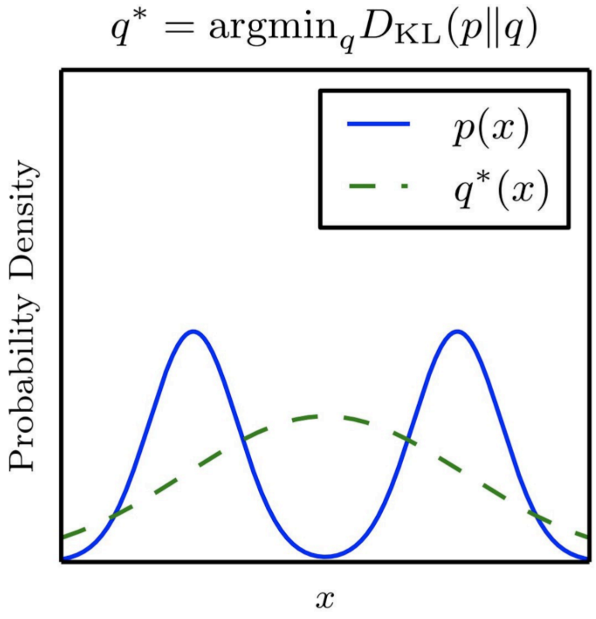


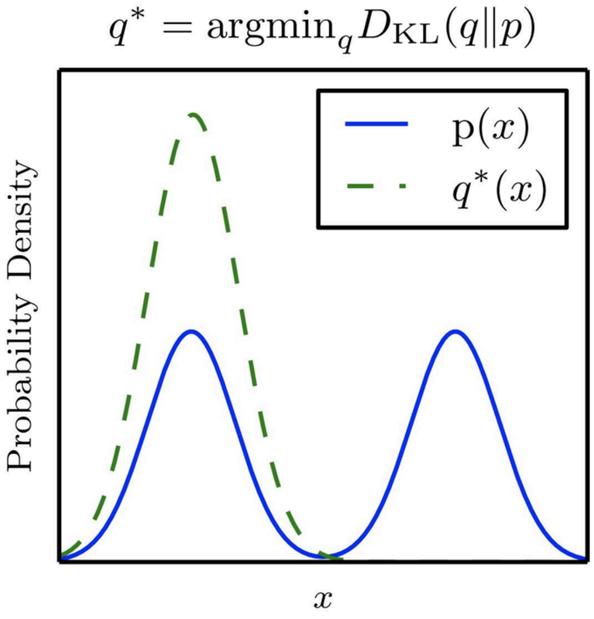
Figure 1: An isotropic Gaussian distribution was fit to data drawn from a mixture of Gaussians by either minimizing Kullback-Leibler divergence (KLD), maximum mean discrepancy (MMD), or Jensen-Shannon divergence (JSD). The different fits demonstrate different tradeoffs made by the three measures of distance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

A note on the evaluation of generative models (Theis, Van den Oord, Bethge 2015)

Direction of KL divergence



Maximum likelihood



Reverse KL

Deep Learning Textbook (Goodfellow 2016)- Chapter 3

Mode covering vs Mode seeking: Tradeoffs

- For compression, one would prefer to ensure all points in the data distribution are assigned probability mass.
- For generating good samples, blurring across modes spoils perceptual quality because regions outside the data manifold are assigned non-zero probability mass.
- Picking one mode without assigning probability mass on points outside can produce “better-looking” samples.
- Caveat: More expressive density models can place probability mass more accurately. Example: Using mixture of gaussians as opposed to a single isotropic gaussian.

Back to GANs

Recall

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

Mini-Exercise:

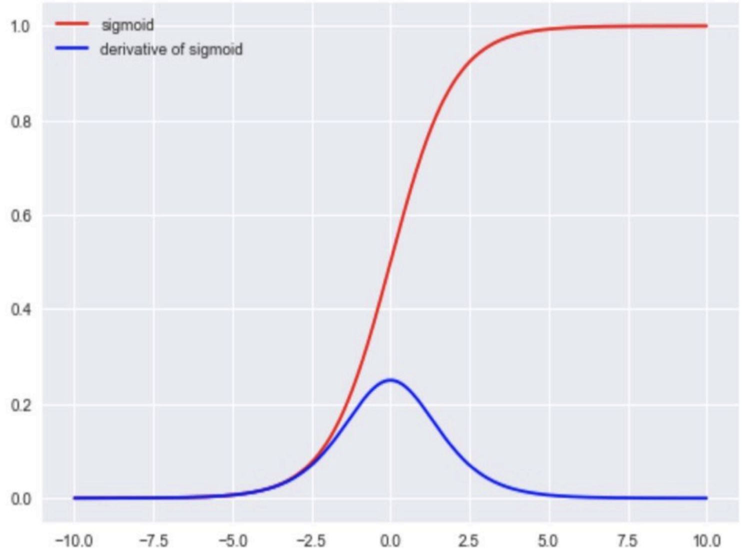
- Is it feasible to run the inner optimization to completion?
- For this specific objective, would it create problems if we were able to do so?

Back to GANs

- Generator samples confidently classified as fake by the discriminator receive no gradient for the generator update.
- Referred to as the 'Discriminator Saturation' problem.

$$\nabla_{G(z)} \log(1 - D(G(z))) \text{ where } D(x) = \text{sigmoid}(x;\theta) = \sigma(x;\theta)$$

$$\nabla_x \sigma(x) = \sigma(x)(1 - \sigma(x))$$



Back to GANs

- Alternate between optimizing (taking gradient descent steps on) the discriminator and generator objectives

$$L^{(D)}(\theta_D, \theta_G) = -\mathbb{E}_{x \sim p_{\text{data}}} [\log D(x; \theta_D)] - \mathbb{E}_{z \sim p(z)} [\log(1 - D(G(z; \theta_G), \theta_D))]$$

$$L^{(G)}(\theta_D, \theta_G) = \mathbb{E}_{z \sim p(z)} [\log(1 - D(G(z; \theta_G), \theta_D))]$$

$$\theta_D := \theta_D - \alpha^{(D)} \nabla_{\theta_D} L^{(D)}(\theta_D, \theta_G)$$

$$\theta_G := \theta_G - \beta^{(G)} \nabla_{\theta_G} L^{(G)}(\theta_D, \theta_G)$$

- Balancing these two updates is hard for the zero-sum game
- Goodfellow suggests modifying the generator objective to make the adversarial game non-zero sum and help address the saturation problem

Training GANs: Two-player game

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

Putting it together: GAN training algorithm

Some find k=1 more stable, others use k > 1, no best rule.

Recent work (e.g. Wasserstein GAN) alleviates this problem, better stability!

for number of training iterations **do**

  for $\boxed{k \text{ steps}}$ **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:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D_{\theta_d}(x^{(i)}) + \log(1 - D_{\theta_d}(G_{\theta_g}(z^{(i)}))) \right]$$

  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 ascending its stochastic gradient (improved objective):

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(D_{\theta_d}(G_{\theta_g}(z^{(i)})))$$

end for

GANs - Non Saturating version

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

$$L^{(G)} = -L^{D} \equiv \min_{G} \mathbb{E}_{z\sim p(z)} \log(1 - D(G(z)))$$

Not zero-sum

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

$$L^{(G)} = -\mathbb{E}_{z\sim p(z)} \log(D(G(z)) \equiv \max_{G} \mathbb{E}_{z\sim p(z)} \log(D(G(z))$$

GAN samples from 2014

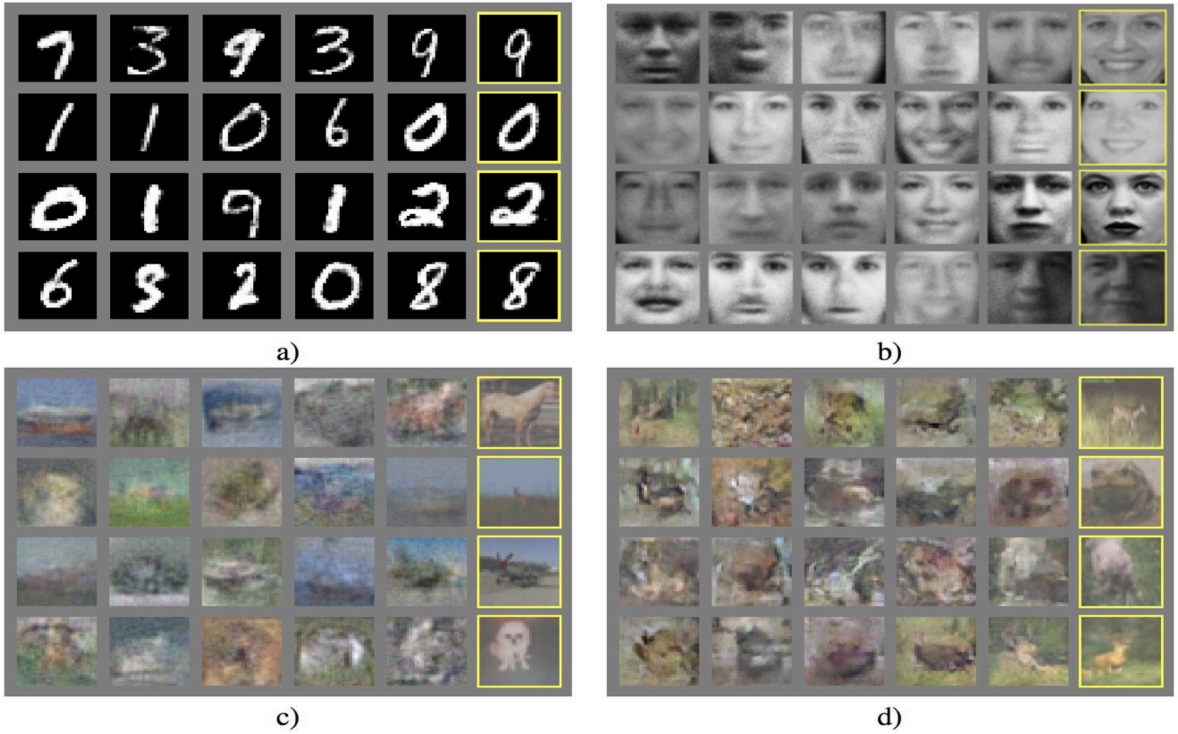
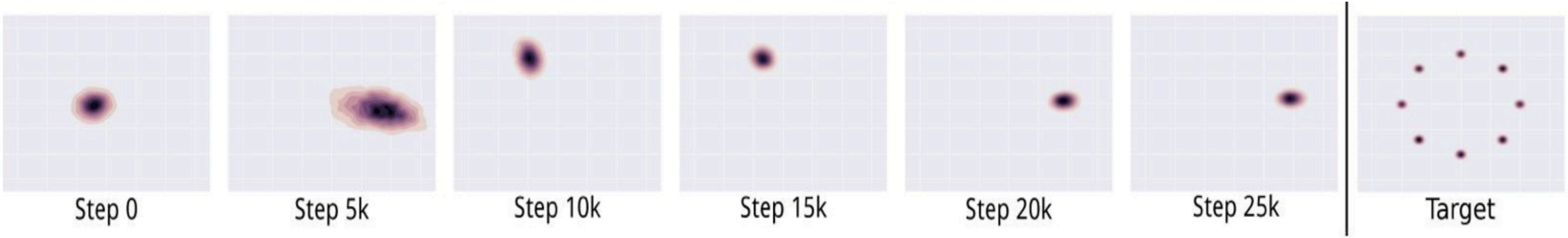


Figure from Goodfellow et al 2014

Mode Collapse



Standard GAN training collapses when the true distribution is a mixture of gaussians (Figure from Metz et al 2016)

How to evaluate?

- Evaluation for GANs is still an open problem
- Unlike density models, you cannot report explicit likelihood estimates on test sets.

Parzen-Window density estimator

- Also known as Kernel Density Estimator (KDE)
- An estimator with kernel K and bandwidth h:

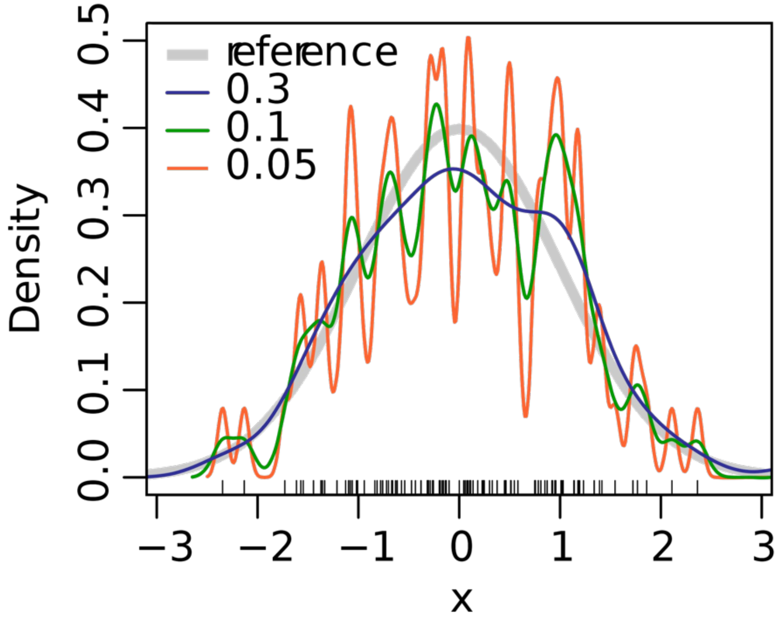
$$\hat{p_h}(x) = \frac{1}{nh} \sum_i K\left(\frac{x - x_i}{h}\right)$$

- In generative model evaluation, K is usually density function of standard Normal distribution

Bishop 2006

Parzen-Window density estimator

- Bandwidth h matters
- Bandwidth h chosen according to validation set



Bishop 2006

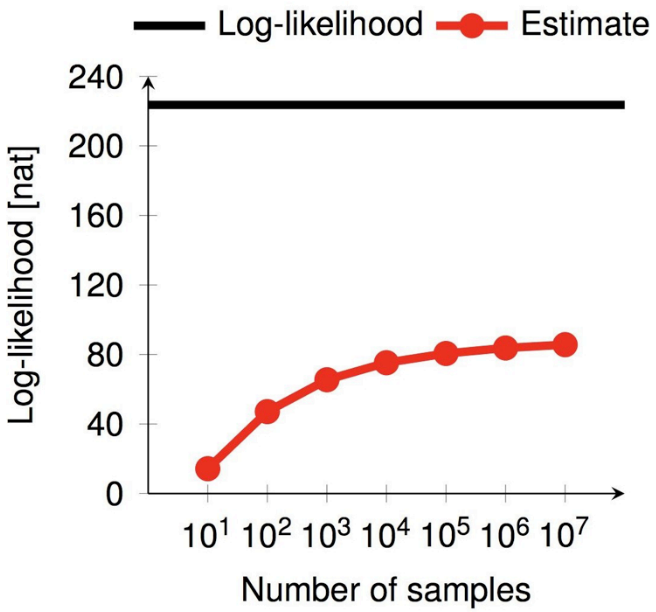
Evaluation

Model	MNIST	TFD
DBN [3]	138 ± 2	1909 ± 66
Stacked CAE [3]	121 ± 1.6	$\mathbf{2110 \pm 50}$
Deep GSN [5]	214 ± 1.1	1890 ± 29
Adversarial nets	$\mathbf{225 \pm 2}$	$\mathbf{2057 \pm 26}$

Parzen Window density estimates (Goodfellow et al)

Parzen-Window density estimator

- Parzen Window estimator can be unreliable



Model	Parzen est. [nat]
Stacked CAE	121
DBN	138
GMMN	147
Deep GSN	214
Diffusion	220
GAN	225
True distribution	**243**
GMMN + AE	282
k-means	313

A note on the evaluation of generative models (Theis, Van den Oord, Bethge 2015)

Inception Score

- Can we side-step high-dim density estimation?
- One idea: good generators generate samples that are semantically diverse
- Semantics predictor: trained Inception Network v3
  - p(y|x), y is one of the 1000 ImageNet classes
- Considerations:
  - each image x should have distinctly recognizable object -> p(y|x) should have low entropy
  - there should be as many classes generated as possible -> p(y) should have high entropy

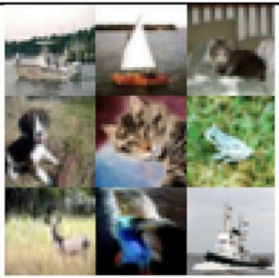
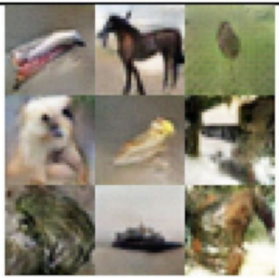
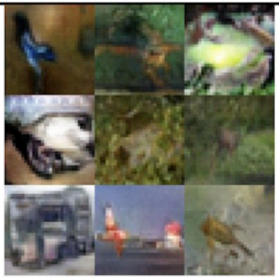
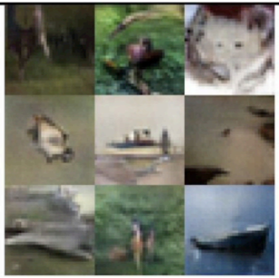
Inception Score

- Inception model: $p(y|x)$
- Marginal label distribution: $p(y) = \int_x p(y|x)p_g(x)$
- Inception Score:

$$\begin{aligned}
\mathrm{IS}(x) &= \exp(\mathbb{E}_{x \sim p_g}\left[D_{\mathrm{KL}}\left[p(y|x) \parallel p(y)\right]\right]) \\
&= \exp(\mathbb{E}_{x \sim p_g, y \sim p(y|x)}\left[\log p(y|x) - \log p(y)\right]) \\
&= \exp(H(y) - H(y|x))
\end{aligned}$$

Improved Gan (Salimans et al 2016)

Inception Score

Samples				
Model	Real data	Our methods	-VBN+BN	-L+HA
Score $\pm$ std.	$11.24 \pm .12$	$8.09 \pm .07$	$7.54 \pm .07$	$6.86 \pm .06$

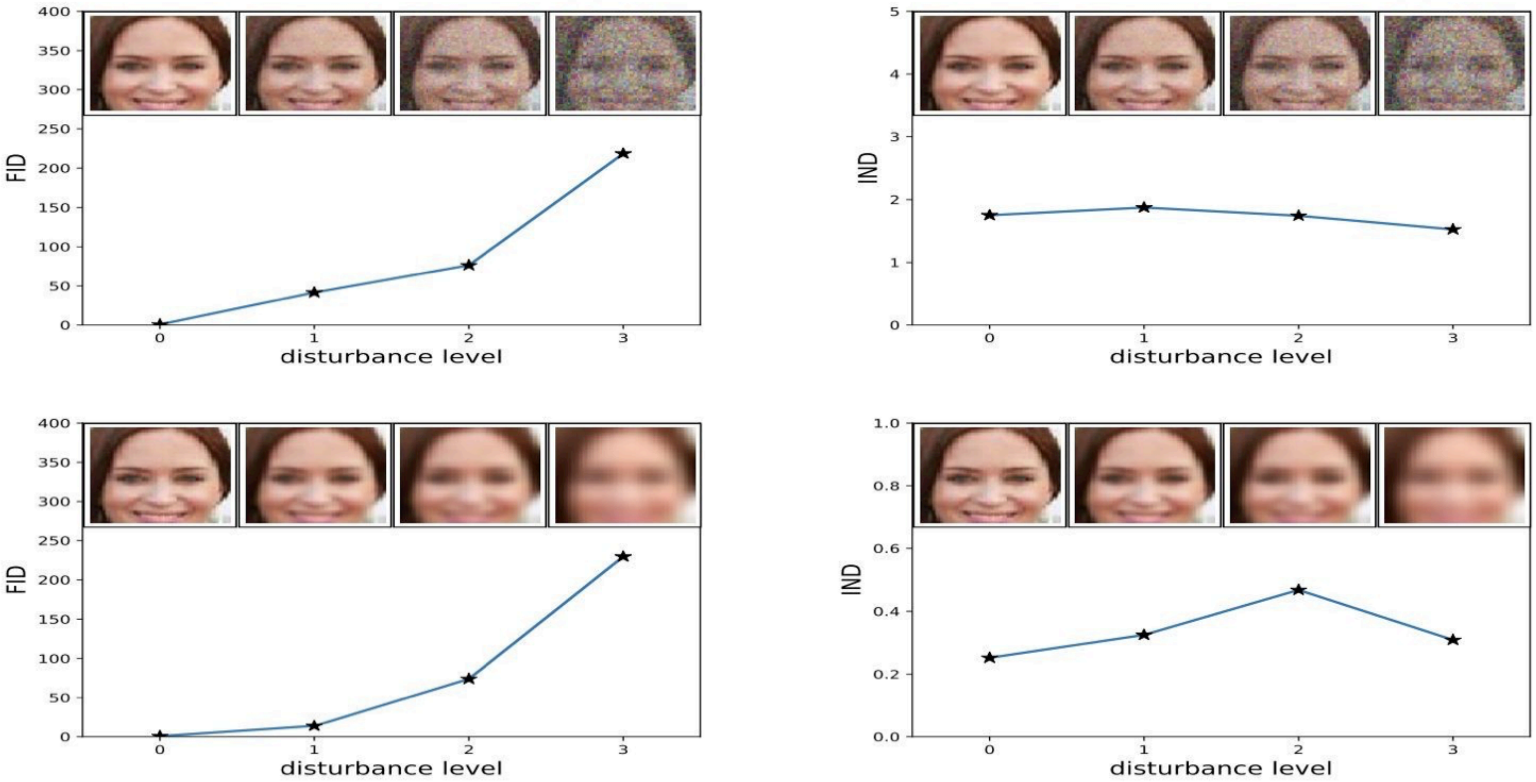
Fréchet Inception Distance

- Inception Score doesn't sufficiently measure diversity: a list of 1000 images (one of each class) can obtain perfect Inception Score
- FID was proposed to capture more nuances
- Embed image x into some feature space (2048-dimensional activations of the Inception-v3 pool3 layer), then compare mean (m) & covariance (C) of those random features

$$d^2((\boldsymbol{m}, \boldsymbol{C}), (\boldsymbol{m}_w, \boldsymbol{C}_w)) = \|\boldsymbol{m} - \boldsymbol{m}_w\|_2^2 + \mathrm{Tr}\left(\boldsymbol{C} + \boldsymbol{C}_w - 2(\boldsymbol{C}\boldsymbol{C}_w)^{1/2}\right)$$

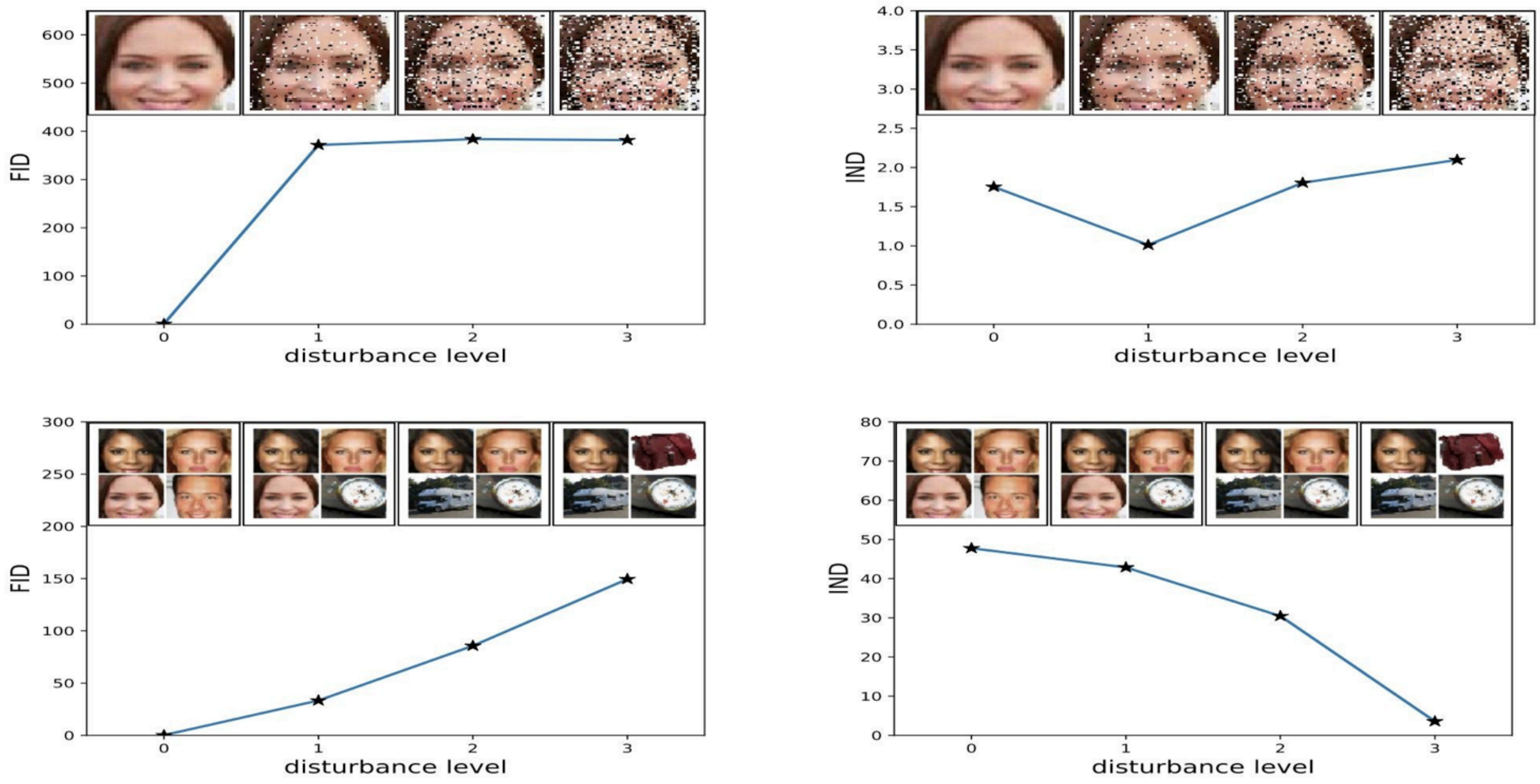
(Heusel et al, 2017)

Fréchet Inception Distance



017)

Fréchet Inception Distance



17)

Generative Adversarial Networks

- Key pieces of GAN
  - Fast sampling
  - No inference
  - Goodfellow suggests building inference by reversing / inverting a GAN - not really shown to work so far
  - Notion of optimizing directly for what you care about - perceptual samples