

COMP547

DEEP UNSUPERVISED LEARNING

Lecture #8 - #9 – Generative Adversarial Networks



KOÇ
UNIVERSITY

Aykut Erdem // Koç University // Spring 2022

Good news, everyone!

- Assignment 2 is out!
(due April 3)
- Let us know if you
want to contribute
to COMP547
lecture notes!



Previously on COMP547

- Motivation
- Training Latent Variable Models
(including VAE and IWAE)
- Variations
- Related ideas



Image: Synthetic faces sampled from NVAE model
by Vahdat and Kautz

Lecture overview

- Motivation & Definition of Implicit Models
- Original GAN (Goodfellow et al, 2014)
- Evaluation: Parzen, Inception, Frechet
- Theory of GANs
- GAN Progression
- Conditional GANs, Cycle-Consistent Adversarial Networks
- GANs and Representations
- Applications

Disclaimer: Much of the material and slides for this lecture were borrowed from

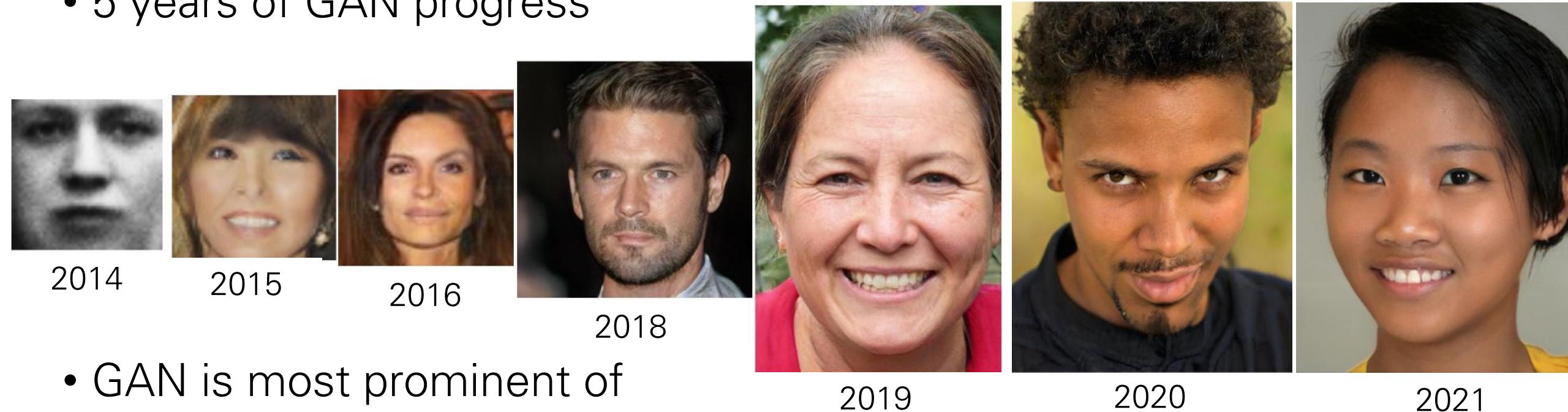
- Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas' Berkeley CS294-158 class
- Aaron Courville's IFT6135 class
- Bill Freeman, Antonio Torralba and Phillip Isola's MIT 6.869 class

Lecture overview

- Motivation and Definition of Implicit Models
- Original GAN (Goodfellow et al, 2014)
- Evaluation: Parzen, Inception, Frechet
- Theory of GANs
- GAN Progression
- Conditional GANs, Cycle-Consistent Adversarial Networks
- GANs and Representations
- Applications

Motivation: Evolution of GANs

- 5 years of GAN progress



- GAN is most prominent of Implicit Models

I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio. **Generative Adversarial Networks**. NIPS 2014.

A. Radford, L. Metz, S. Chintala. **Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks**. ICLR 2016.

M.-Y. Liu, O. Tuzel. **Coupled Generative Adversarial Networks**. NIPS 2016.

T. Karras, T. Aila, S. Laine, J. Lehtinen. **Progressive Growing of GANs for Improved Quality, Stability, and Variation**. ICLR 2018.

T. Karras, S. Laine, T. Aila. **A style-based generator architecture for generative adversarial networks**. In CVPR 2018.

T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, T. Aila. **Analyzing and Improving the Image Quality of StyleGAN**. CVPR 2020.

T. Karras, M. Aittala, S. Laine, E. Härkönen, J. Hellsten, J. Lehtinen, T. Aila. **Alias-Free Generative Adversarial Networks**. NeurIPS 2021.

Motivation: BigGAN



So far...

- Autoregressive models
 - MADE, PixelRNN/CNN, Gated PixelCNN, PixelSNAIL
- Flow models
 - Autoregressive Flows, NICE, RealNVP, Glow, Flow++
- Latent Variable Models
 - VAE, IWAE, VQ-VAE, VLAE, PixelVAE
- **Common aspect:** Likelihood-based models
 - exact (autoregressive and flows)
 - approximate (VAE)

Generative Models

- Sample
 - Evaluate likelihood
 - Train
 - Representation
- What if all we care about is sampling?

Building a sampler

- How about this sampler?

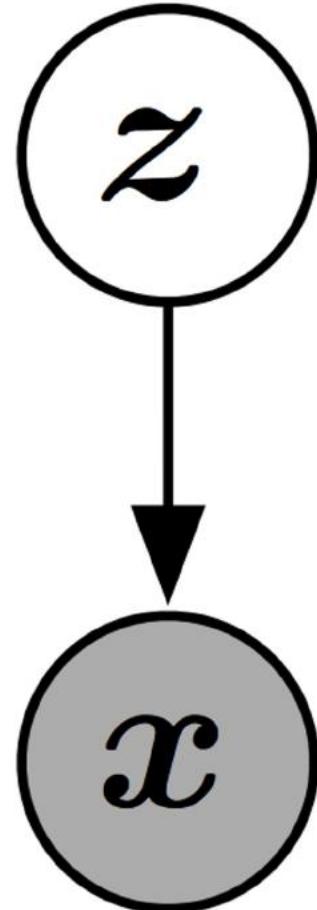
```
import glob, cv2, numpy as np
files = glob.glob('*.*jpg')
def _sample():
    idx = np.random.randint(len(files))
    return cv2.imread(files[idx])
def sample(*, n_samples):
    samples = np.array([\u0304_sample() for _ in range(n_samples)])
return samples
```

Building a sampler

- You don't just want to sample the exact data points you have.
- You want to build a generative model that can understand the underlying distribution of data points and
 - smoothly interpolate across the training samples
 - output samples similar but not the same as training data samples
 - output samples representative of the underlying factors of variation in the training distribution.
 - Example: digits with unseen strokes, faces with unseen poses, etc.

Implicit Models

- Sample z from a fixed noise source distribution (uniform or gaussian).
- Pass the noise through a deep neural network to obtain a sample x .
- Sounds familiar? Right:
 - Flow Models
 - VAE
- What's going to be different here?
 - Learning the deep neural network without explicit density estimation



Implicit Models

Given samples from data distribution $p_{data} : x_1, x_2, \dots, x_n$

Given a sampler $q_\phi(z) = \text{DNN}(z; \phi)$ where $z \sim p(z)$

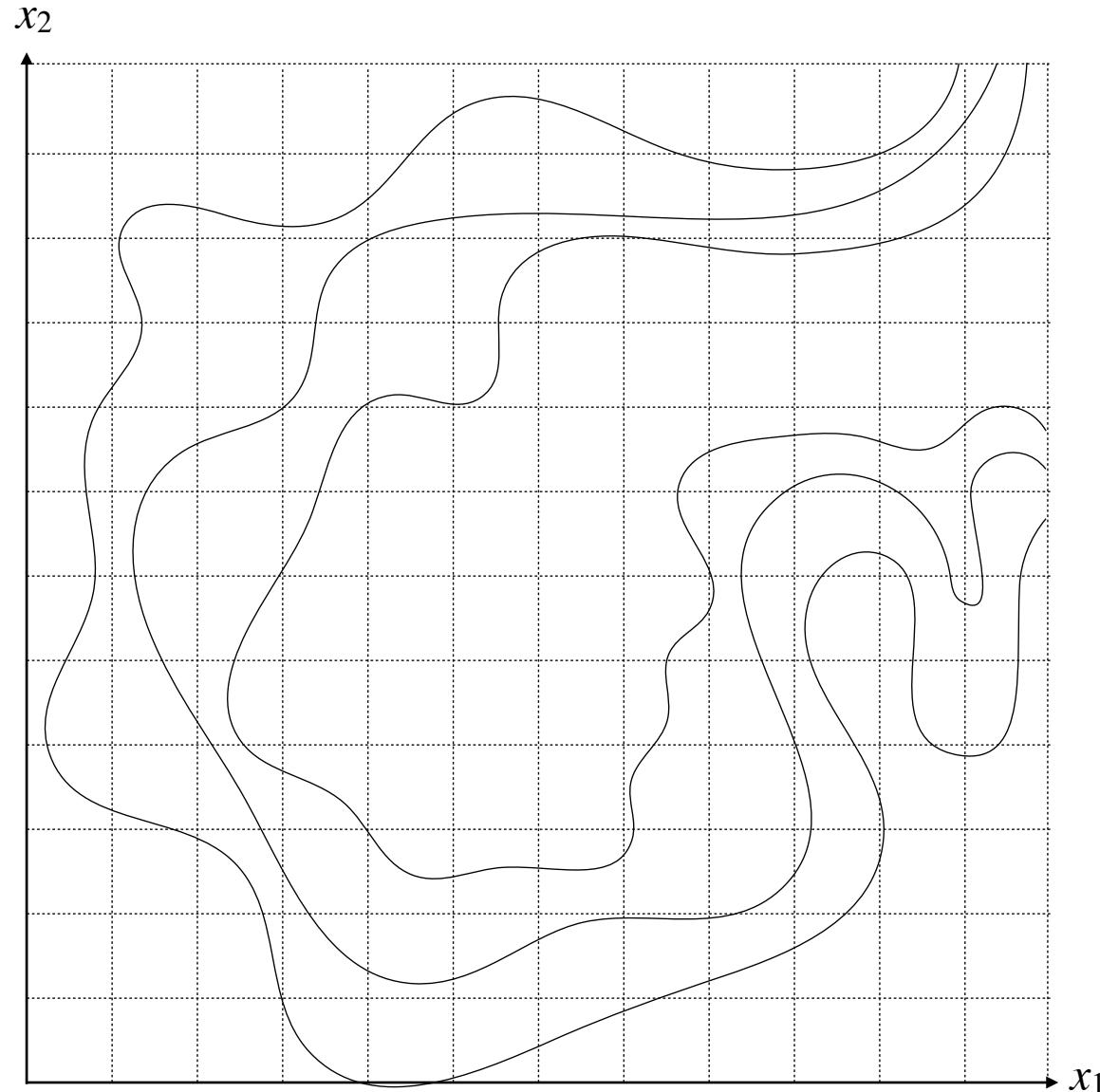
$x = q_\phi(z)$ induces a density function p_{model}

- Do not have an explicit form for p_{data} or p_{model} ; can only draw samples
- Make p_{model} as close to p_{data} as possible by learning an appropriate ϕ

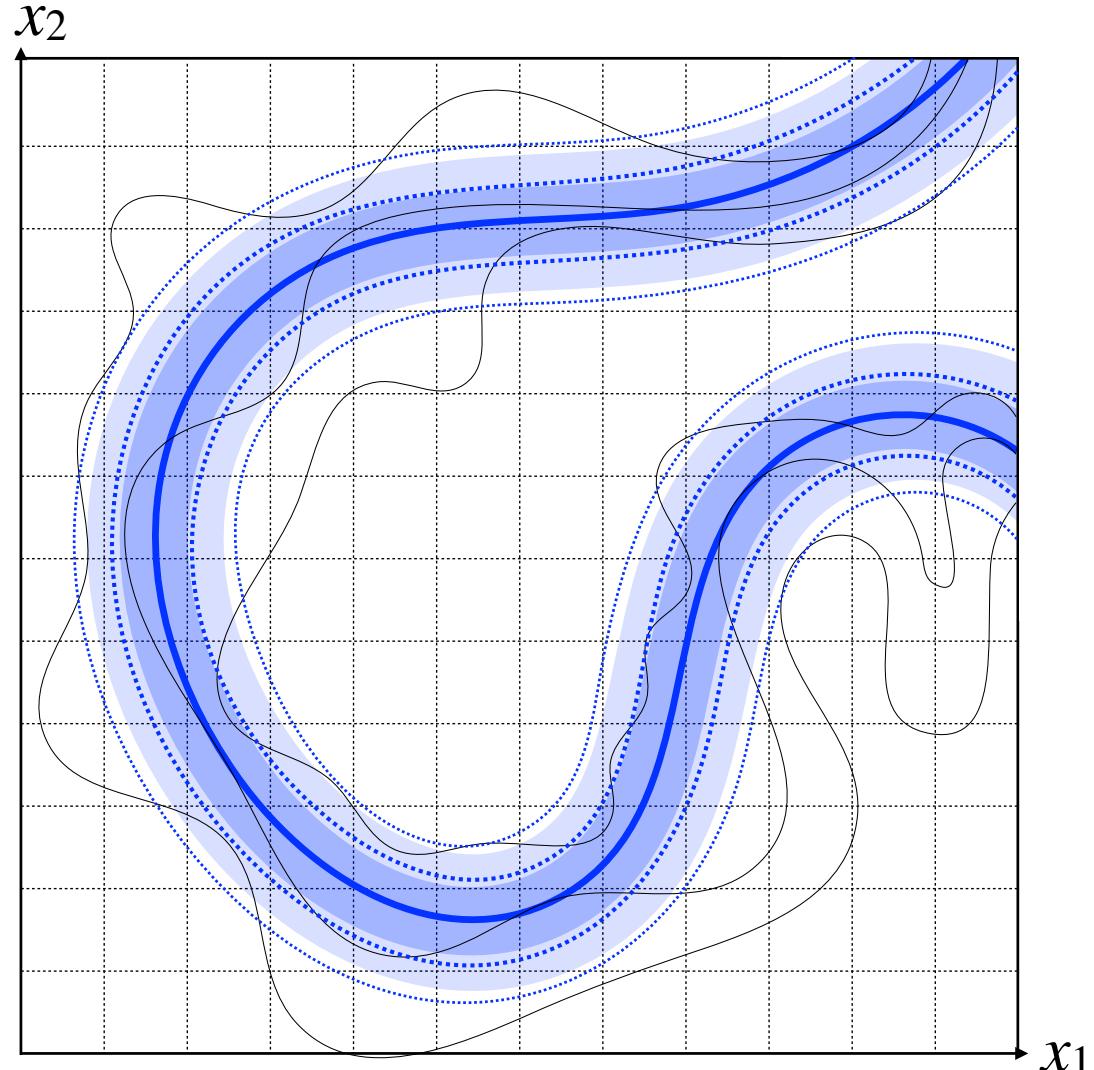
Departure from maximum likelihood

- We need some measure of how far apart p_{data} and induces p_{model} are
- With density models, we used $KL(p_{data} \parallel p_{model})$ which gave us the objective $\mathbb{E}_{x \sim p_{data}} [\log p_\theta(x)]$ (discarding the term independent of θ) where we explicitly modeled p_{model} as $p_\theta(x)$
- Not having an explicit $p_\theta(x)$ requires us to come up distance measures that potentially behave differently from maximum likelihood.
- Example: Maximum Mean Discrepancy (MMD), Jensen Shannon Divergence (JSD), Earth Mover's Distance, etc.

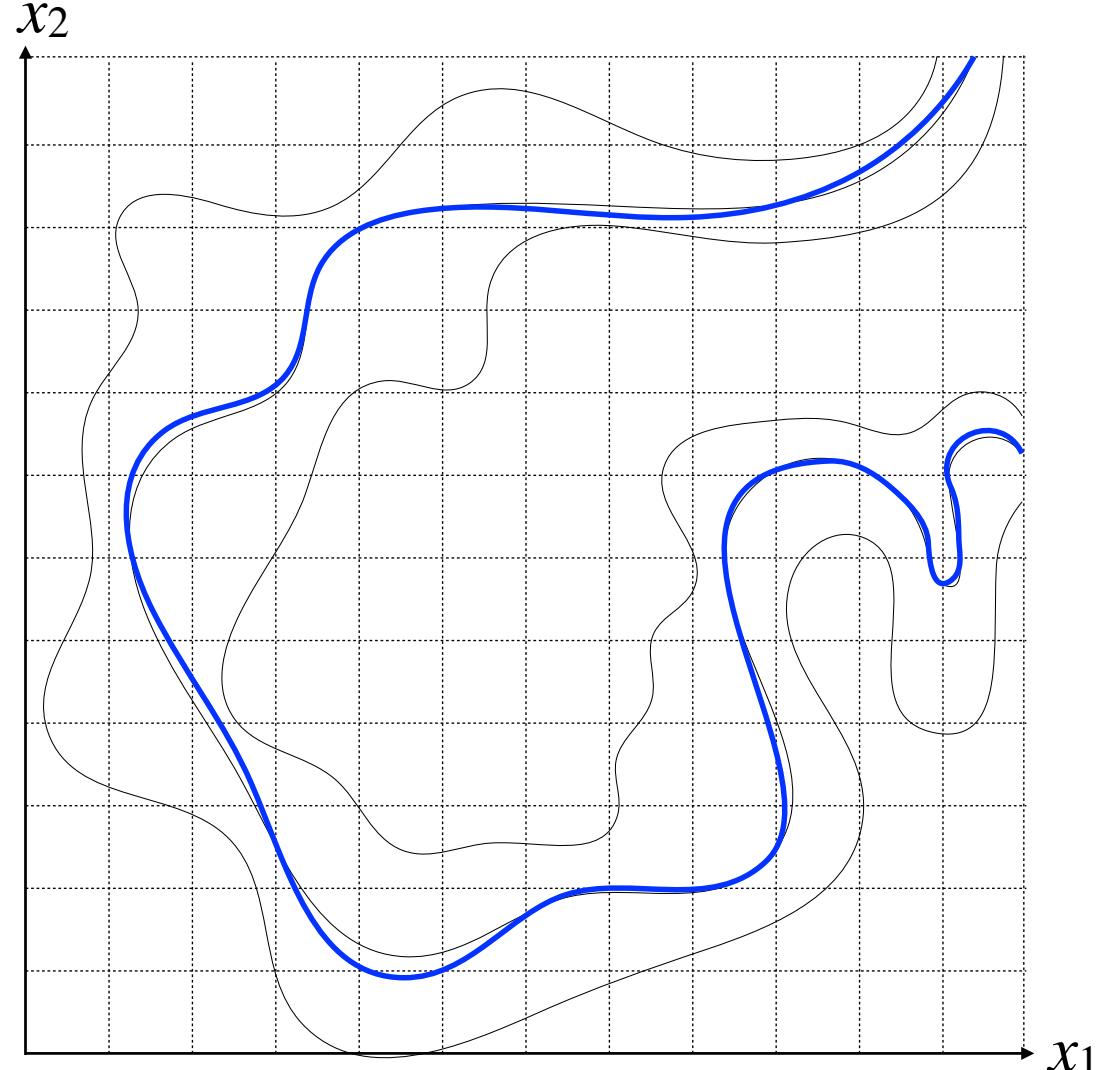
Cartoon of the Image manifold



What makes GANs special?



more traditional max-likelihood approach



GAN

Lecture overview

- Motivation and Definition of Implicit Models
- **Original GAN (Goodfellow et al, 2014)**
- Evaluation: Parzen, Inception, Frechet
- Theory of GANs
- GAN Progression
- Conditional GANs, Cycle-Consistent Adversarial Networks
- Applications

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.

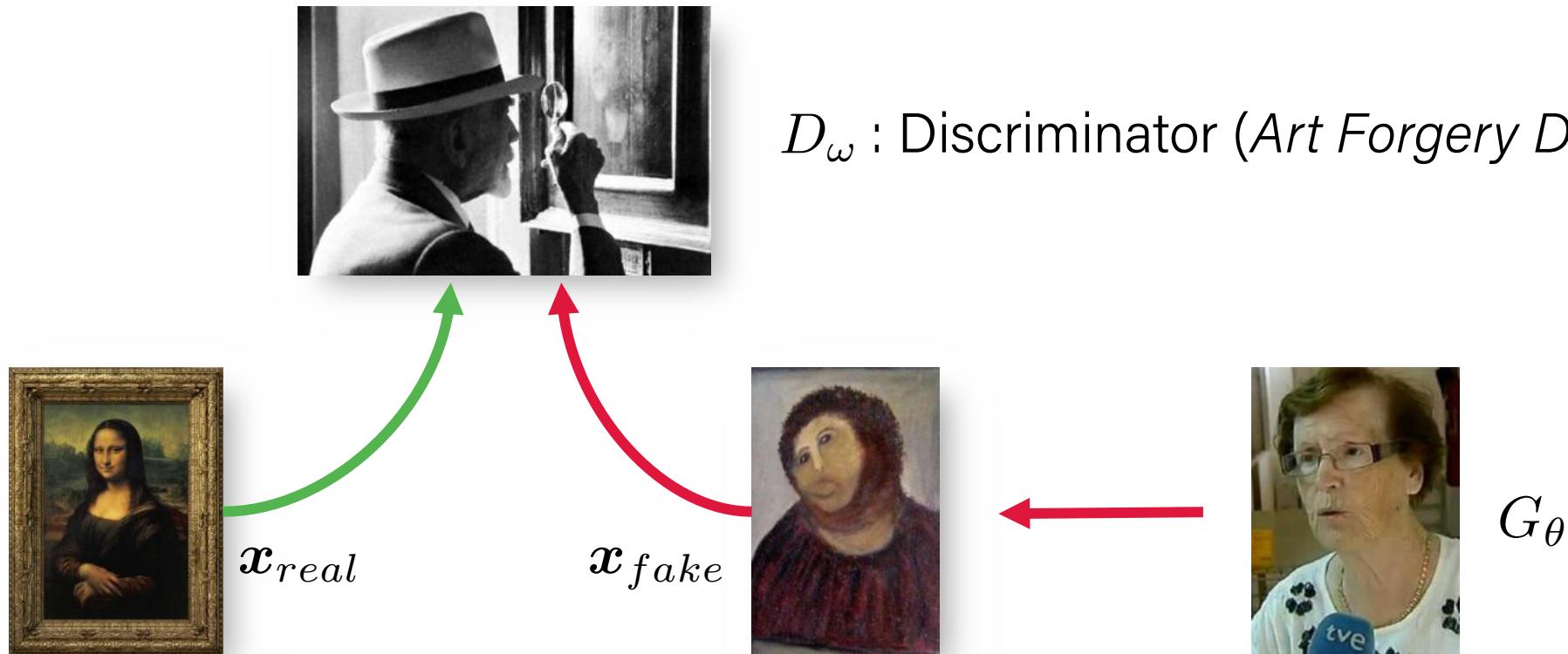
Generative Adversarial Networks

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

- 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)

Intuition behind GANs

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



D_ω : Discriminator (*Art Forgery Detective*)

Generative Adversarial Networks

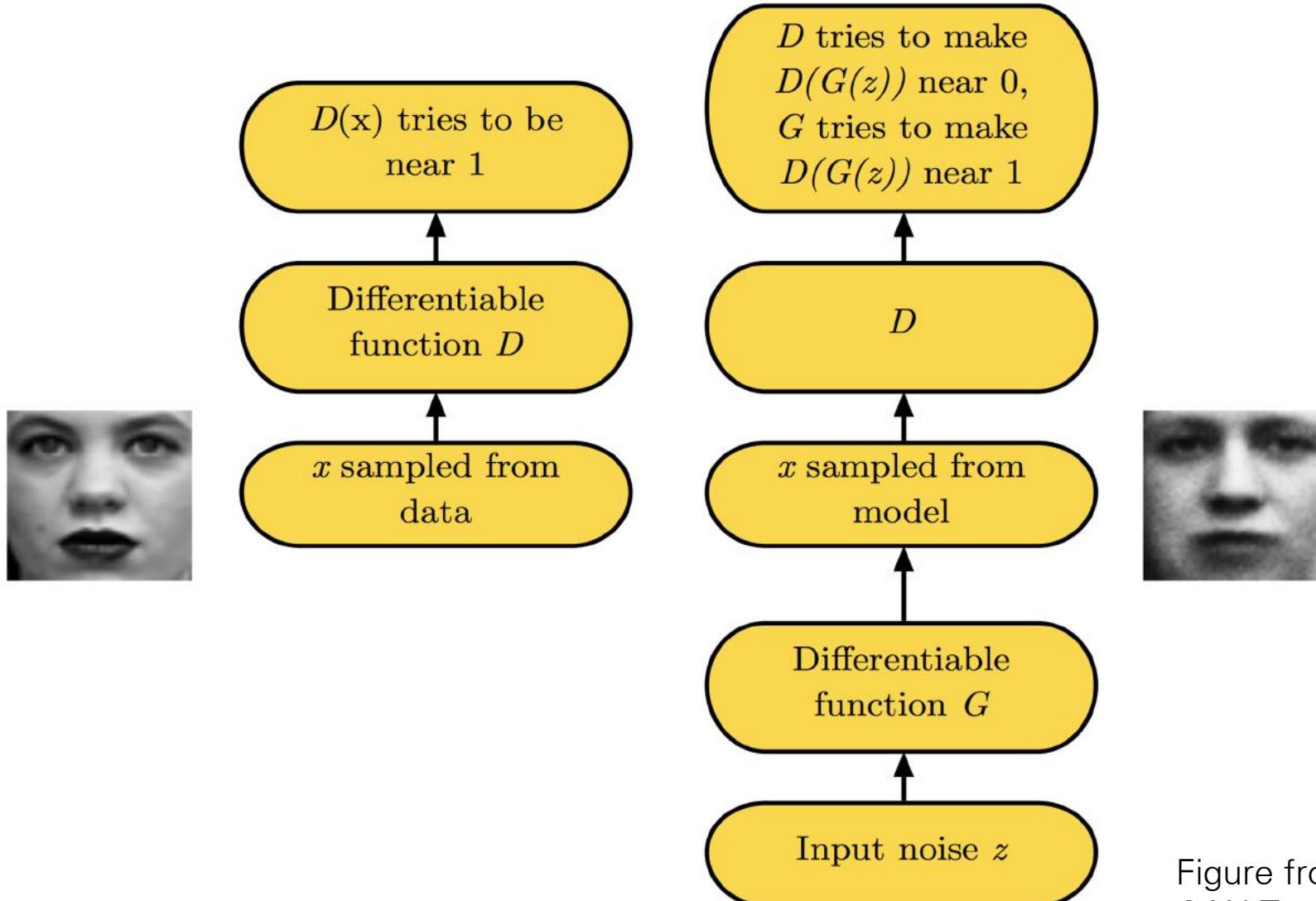


Figure from NeurIPS 2016
GAN Tutorial (Goodfellow)

GANs - Pseudocode

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

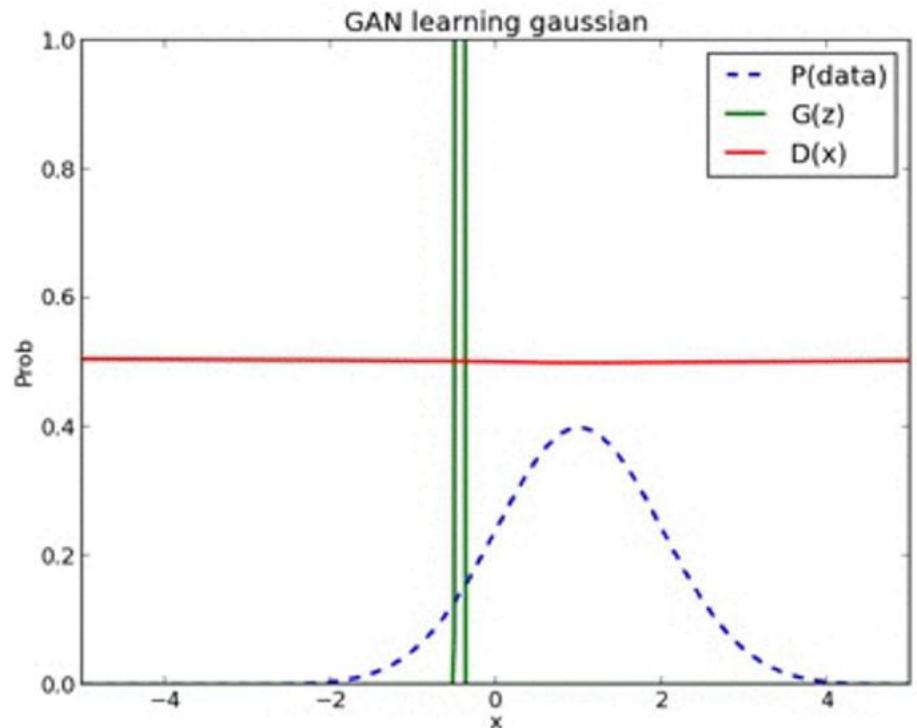
$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m \left[\log D(\mathbf{x}^{(i)}) + \log (1 - D(G(\mathbf{z}^{(i)}))) \right].$$

end for
 • Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$.
 • Update the generator by descending its stochastic gradient:

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

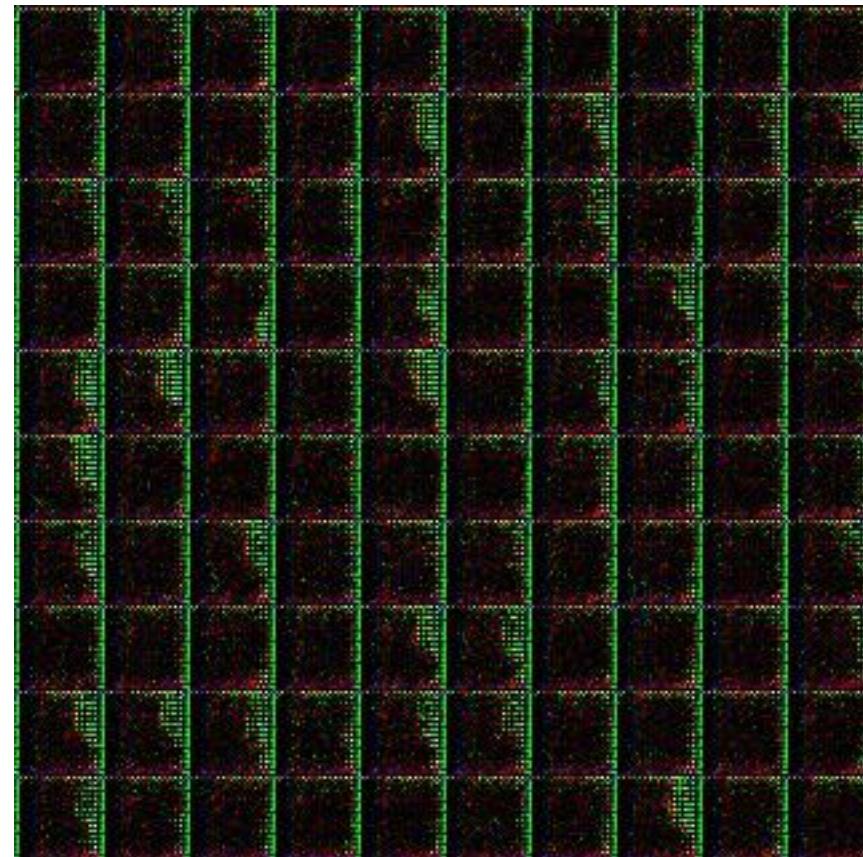
end for
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Training Procedure



Source: Alec Radford

Generating 1D points



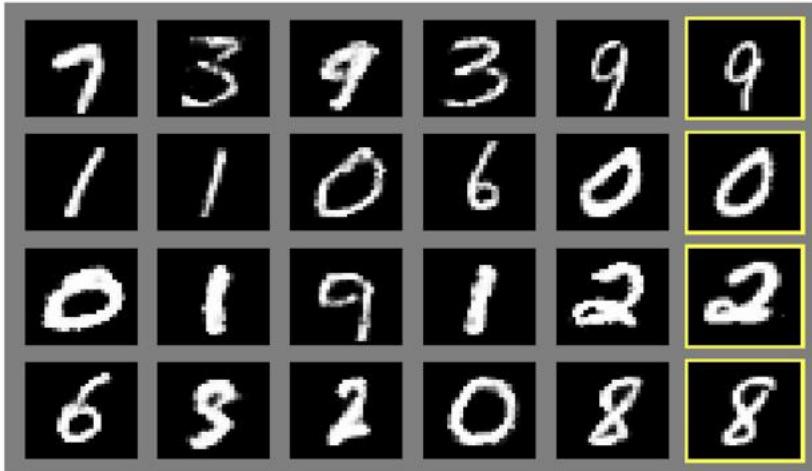
Source: OpenAI blog

Generating images

GAN in Action

<https://poloclub.github.io/ganlab/>

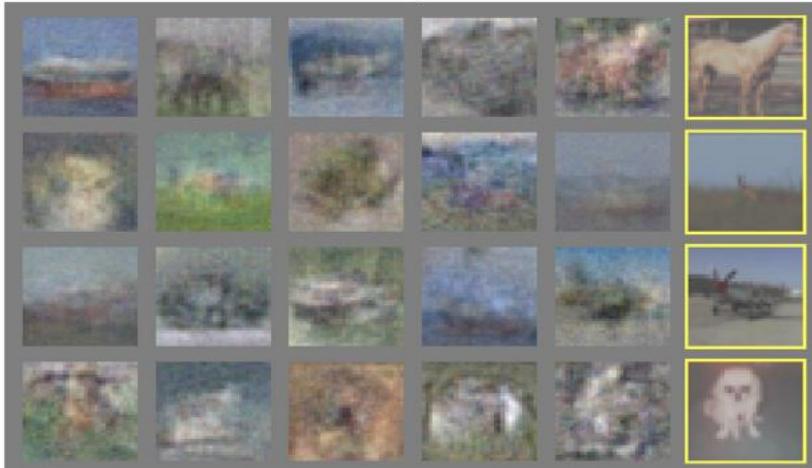
GAN samples from 2014



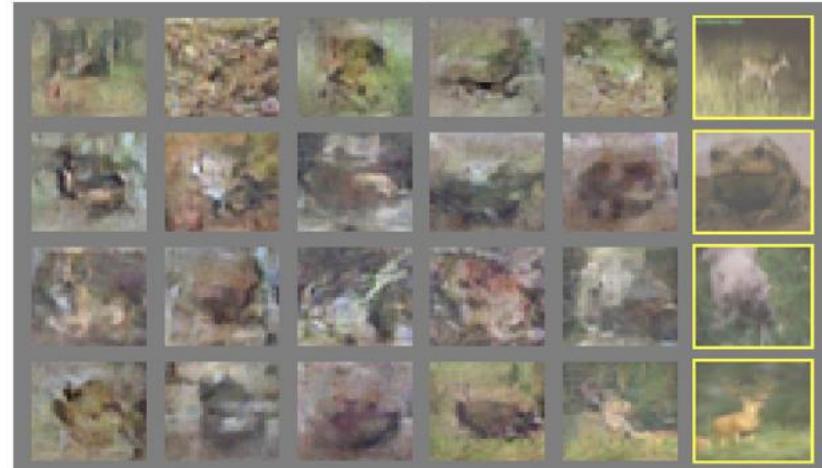
a)



b)



c)



d)

Lecture overview

- Motivation and Definition of Implicit Models
- Original GAN (Goodfellow et al, 2014)
- **Evaluation: Parzen, Inception, Frechet**
- Theory of GANs
- GAN Progression
- Conditional GANs, Cycle-Consistent Adversarial Networks
- GANs and Representations
- Applications

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

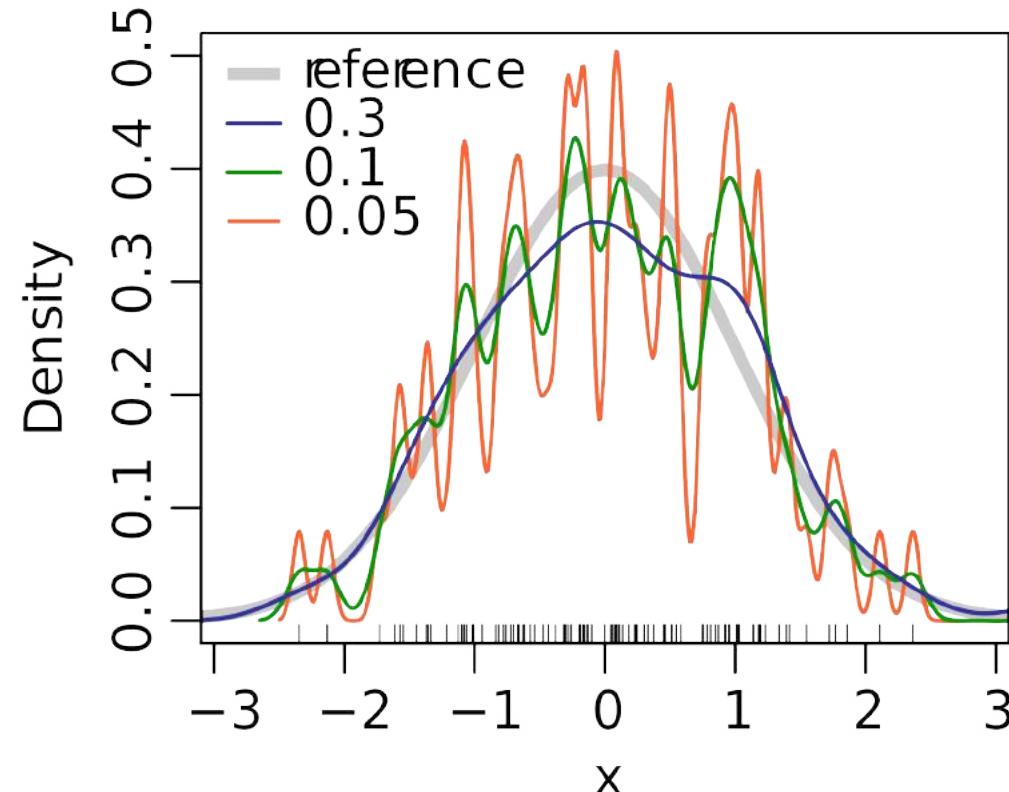
- Also known as Kernel Density Estimator (KDE)
- An estimator with kernel K and bandwidth h :

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

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

Parzen-Window density estimator

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



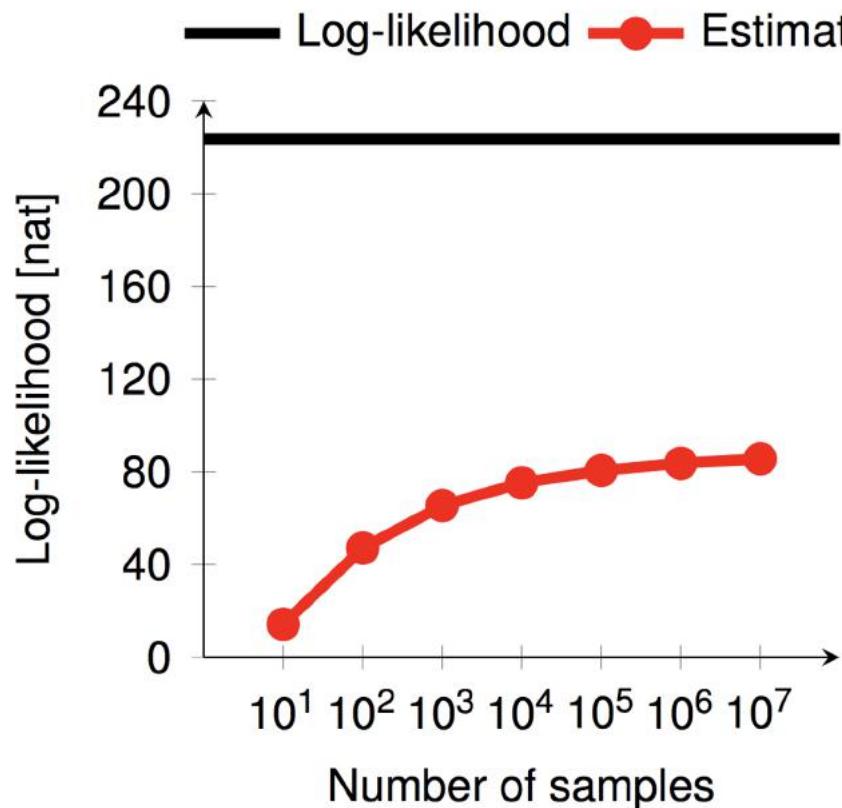
Evaluation

| Model | MNIST | TFD |
|------------------|---------------|---------------|
| DBN [3] | 138 ± 2 | 1909 ± 66 |
| Stacked CAE [3] | 121 ± 1.6 | 2110 ± 50 |
| Deep GSN [5] | 214 ± 1.1 | 1890 ± 29 |
| Adversarial nets | 225 ± 2 | 2057 ± 26 |

Parzen Window density estimates (Goodfellow et al, 2014)

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 |
| <i>k</i> -means | 313 |

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:

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

Inception Score

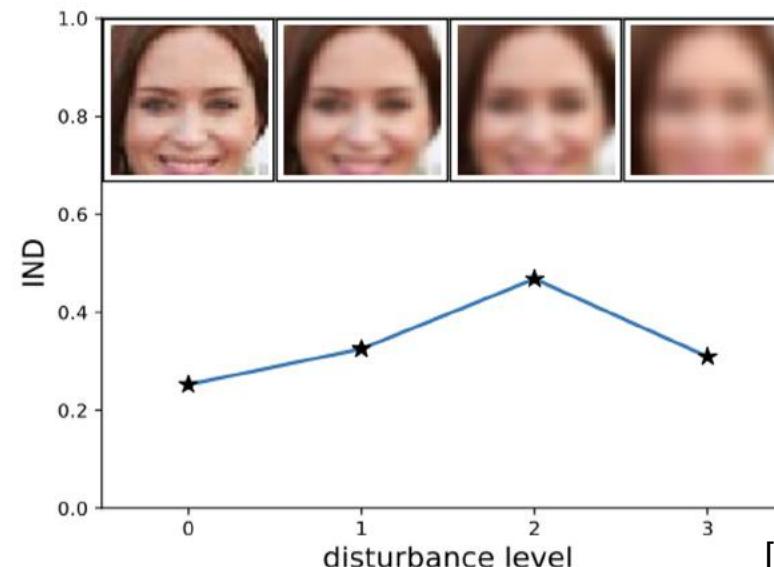
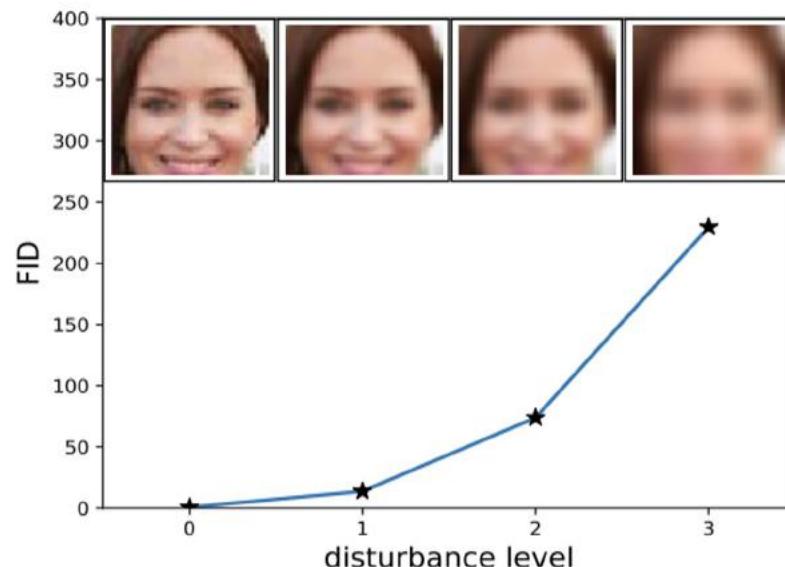
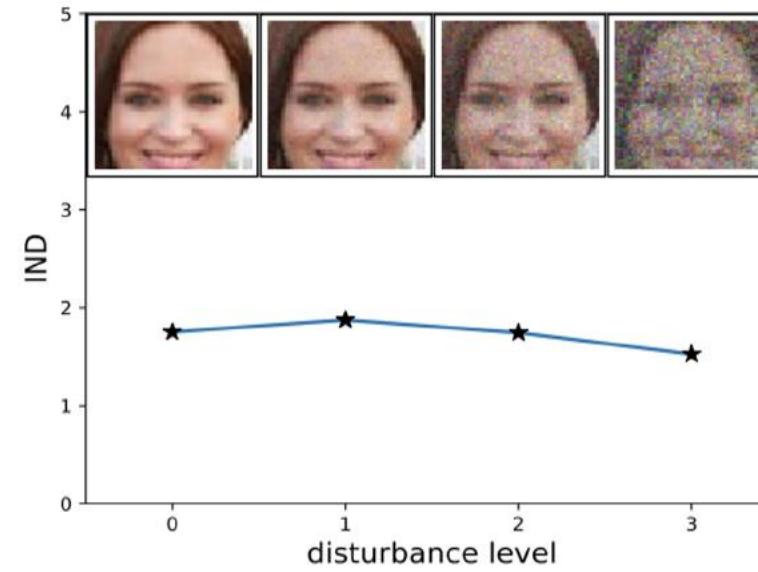
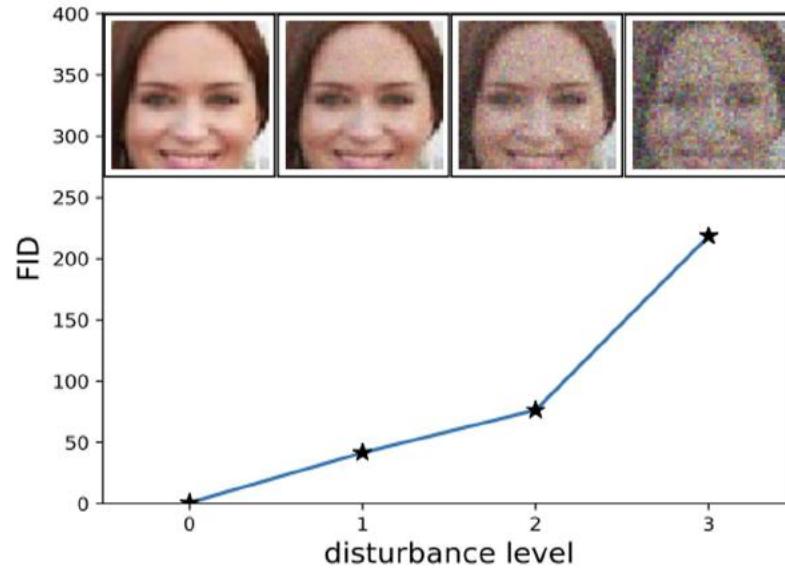
| Samples |  |  |  |  |
|------------------|--|---|---|---|
| Model | Real data | Our methods | -VBN+BN | -L+HA |
| Score \pm std. | 11.24 \pm .12 | 8.09 \pm .07 | 7.54 \pm .07 | 6.86 \pm .06 |

Fréchet Inception Distance

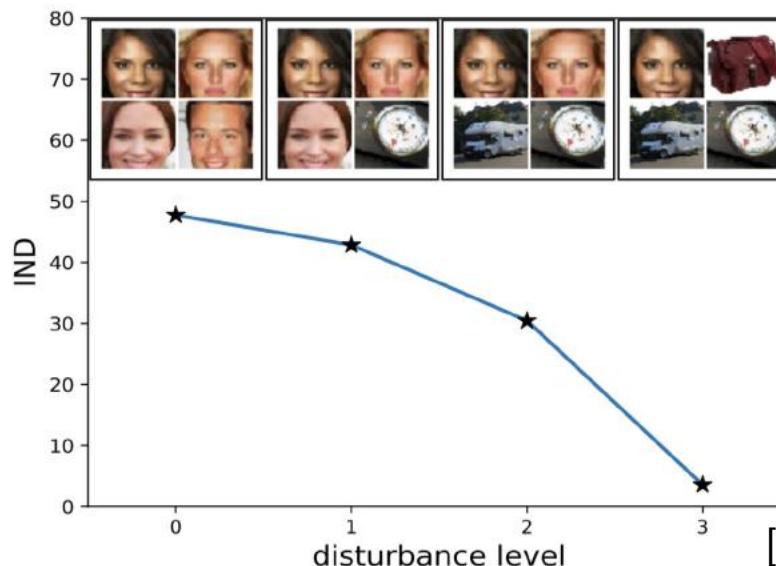
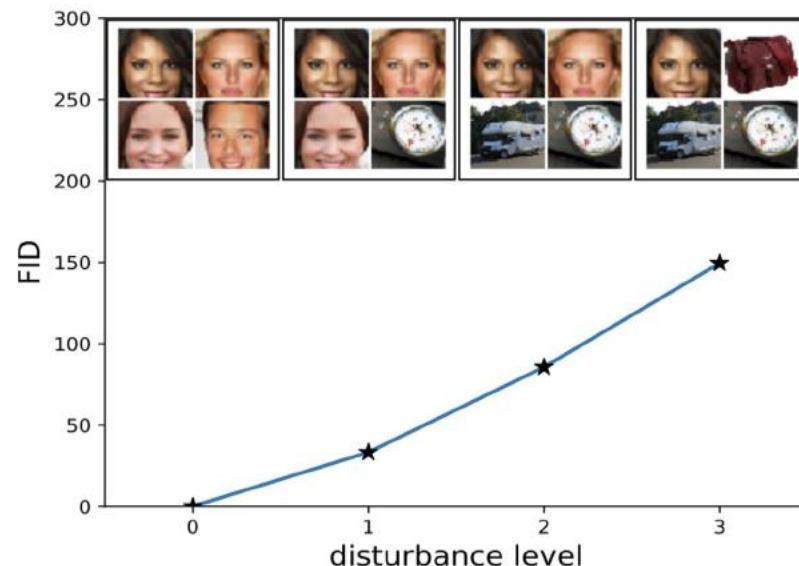
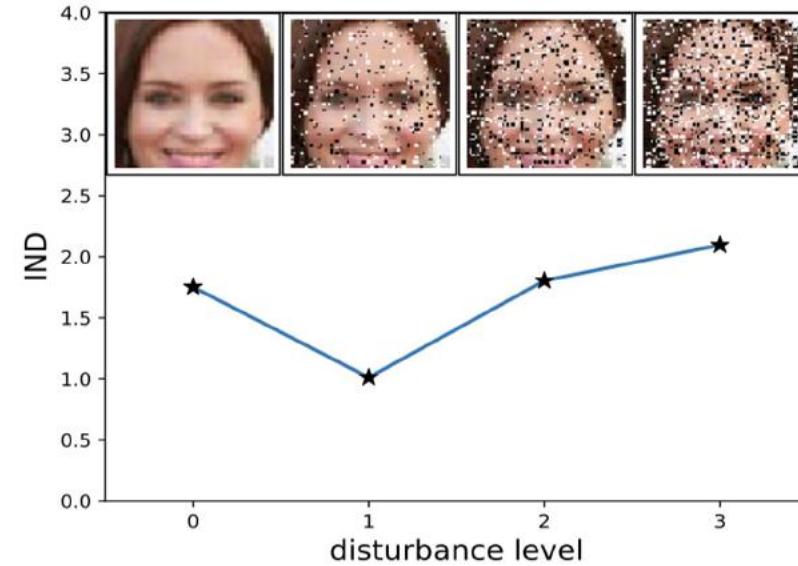
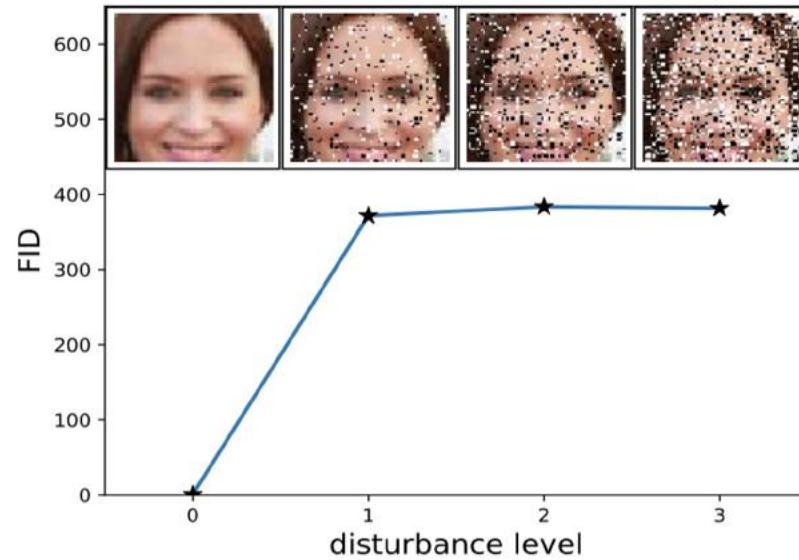
- Inception Score doesn't sufficiently measure diversity: a list of 1000 images (one of each class) can obtain perfect Inception Score
- FID was proposed to capture more nuances
- Embed image x into some feature space (2048-dimensional activations of the Inception-v3 pool3 layer), then compare mean (m)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

$$d^2((\mathbf{m}, \mathbf{C}), (\mathbf{m}_w, \mathbf{C}_w)) = \|\mathbf{m} - \mathbf{m}_w\|_2^2 + \text{Tr}(\mathbf{C} + \mathbf{C}_w - 2(\mathbf{C}\mathbf{C}_w)^{1/2})$$

Fréchet Inception Distance



Fréchet Inception Distance



Fréchet Inception Distance

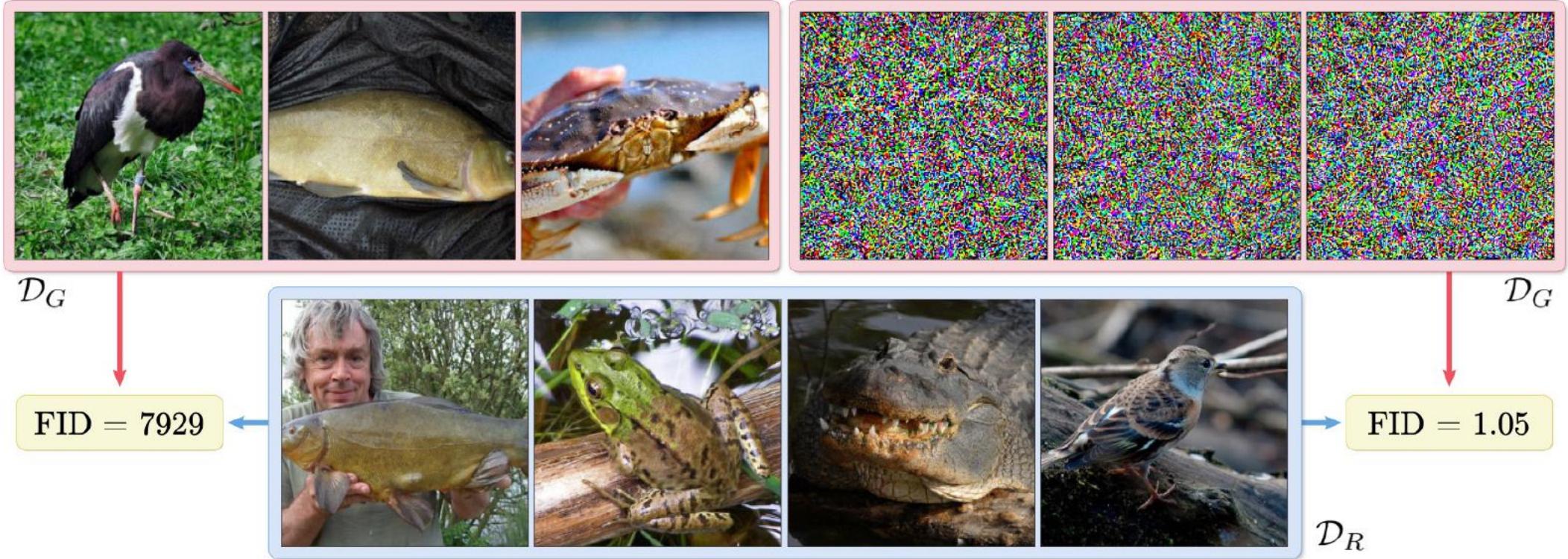


Figure 1. **Does the Fréchet Inception Distance (FID) accurately measure the distances between image distributions?** We generate datasets that demonstrate the unreliability of FID in judging perceptual (dis)similarities between image distributions. The top left box shows a sample of a dataset constructed by introducing imperceptible noise to each ImageNet image. Despite the remarkable visual similarity between this dataset and ImageNet (bottom box), an extremely large FID (almost 8000) between these two datasets showcases FID's failure to capture perceptual similarities. On the other hand, a remarkably low FID (almost 1.0) between a dataset of random noise images (samples shown in the top right box) and ImageNet illustrates FID's failure to capture perceptual dissimilarities.

One solution: Replace the Inception component of FID with a robustly trained counterpart!

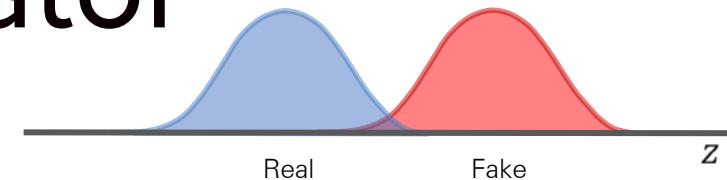
Generative Adversarial Networks

- Key pieces of GAN
 - Fast sampling
 - No inference
 - Notion of optimizing directly for what you care about – perceptual samples

Lecture overview

- Motivation and Definition of Implicit Models
- Original GAN (Goodfellow et al, 2014)
- Evaluation: Parzen, Inception, Frechet
- **Theory of GANs**
- GAN Progression
- Conditional GANs, Cycle-Consistent Adversarial Networks
- GANs and Representations
- Applications

GAN: Bayes-Optimal Discriminator



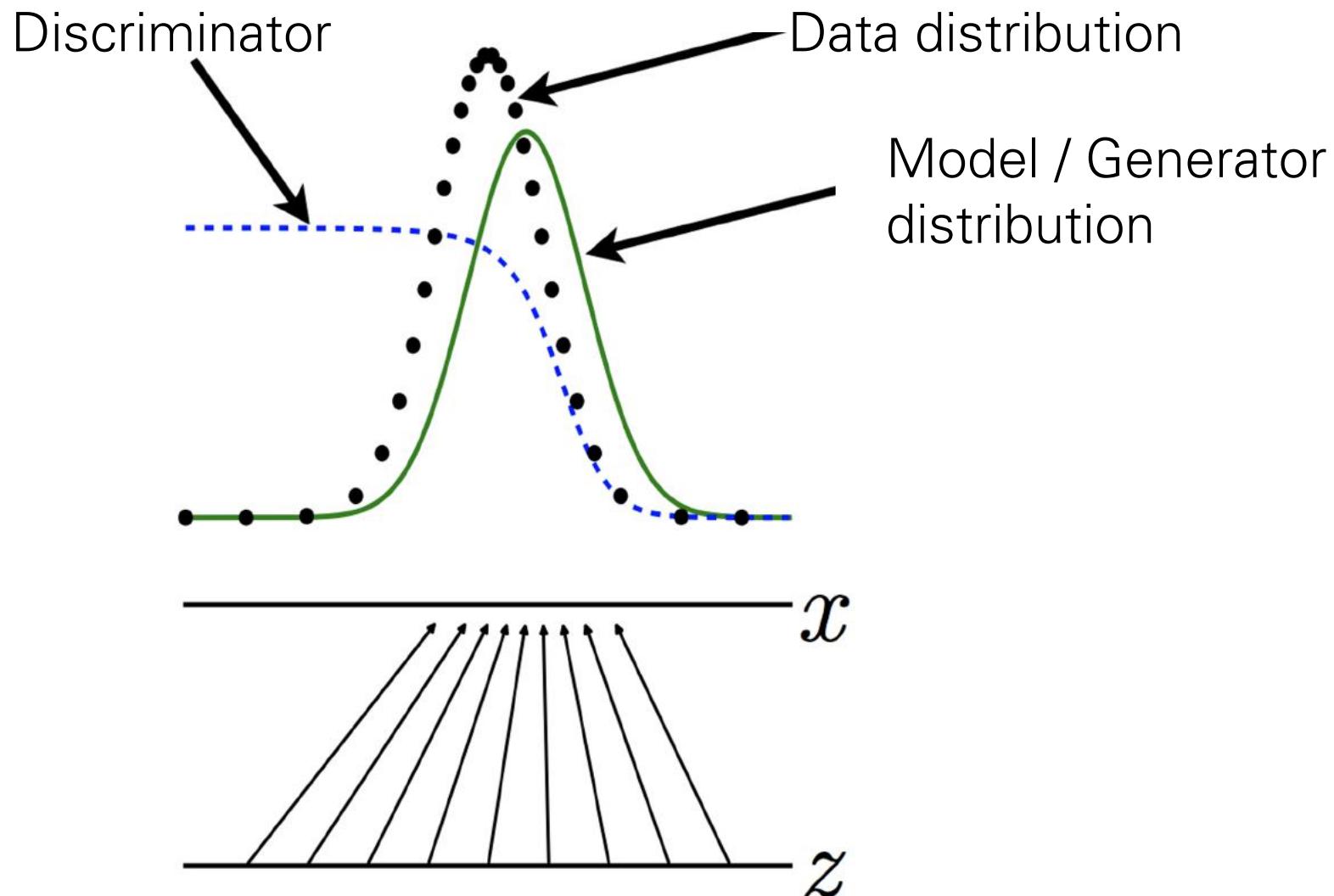
- What's the optimal discriminator given generated and true distributions?

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

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

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

GAN: Bayes-Optimal Discriminator



[Figure Source: Goodfellow
NeurIPS 2016 Tutorial on GANs]

GAN: Generator Objective under Bayes-Optimal Discriminator D*?

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

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

Compare this with ML objective: $KL(p_{\text{data}} \parallel p_g)$

Behaviors across divergence measures

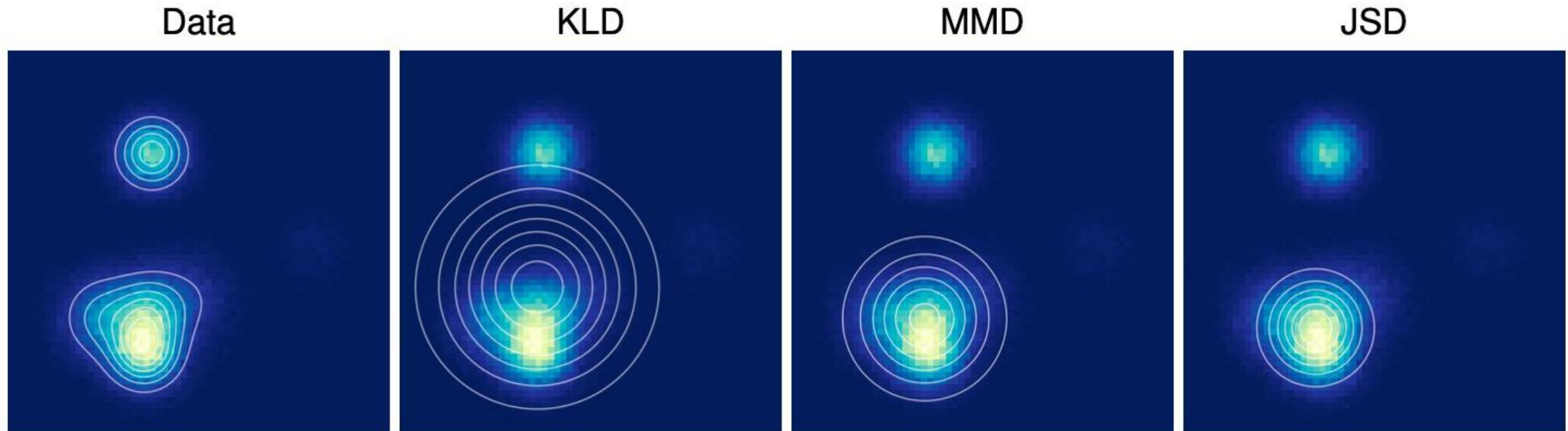
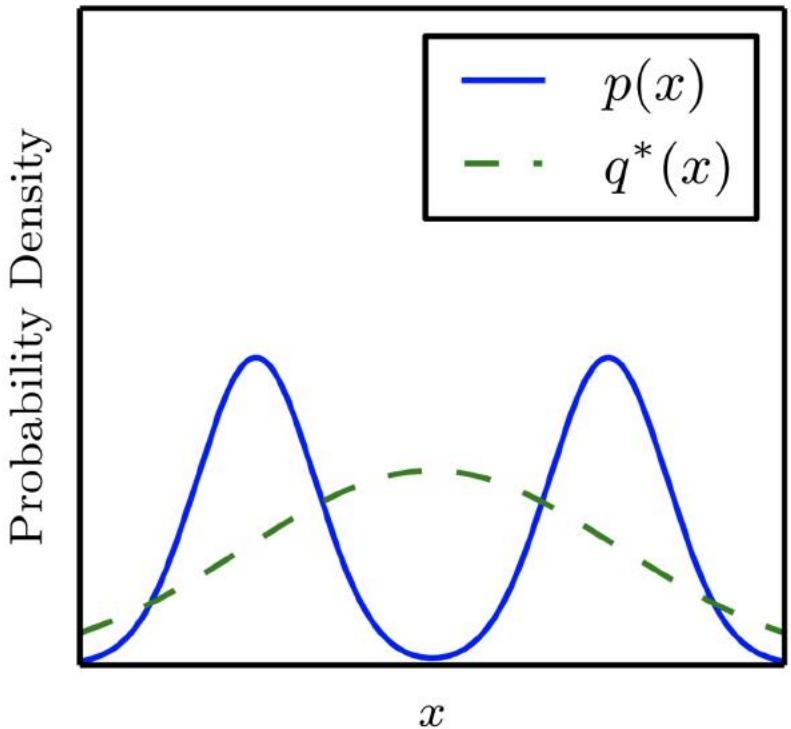


Figure 1: An isotropic Gaussian distribution was fit to data drawn from a mixture of Gaussians by either minimizing Kullback-Leibler divergence (KLD), maximum mean discrepancy (MMD), or Jensen-Shannon divergence (JSD). The different fits demonstrate different tradeoffs made by the three measures of distance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

[“A note on the evaluation of generative models” – Theis, Van den Oord, Bethge 2016]

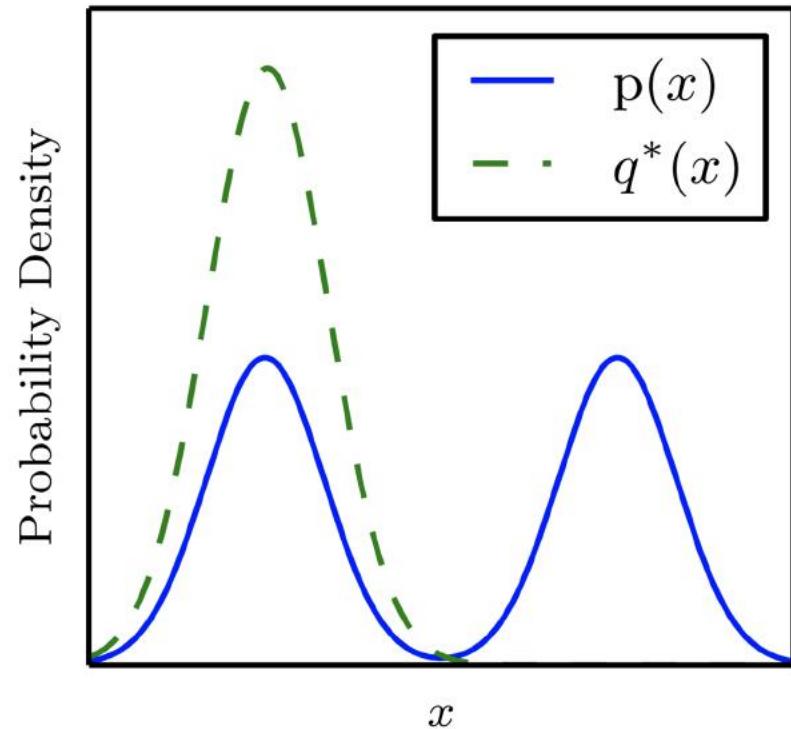
Direction of KL divergence

$$q^* = \operatorname{argmin}_q D_{\text{KL}}(p\|q)$$



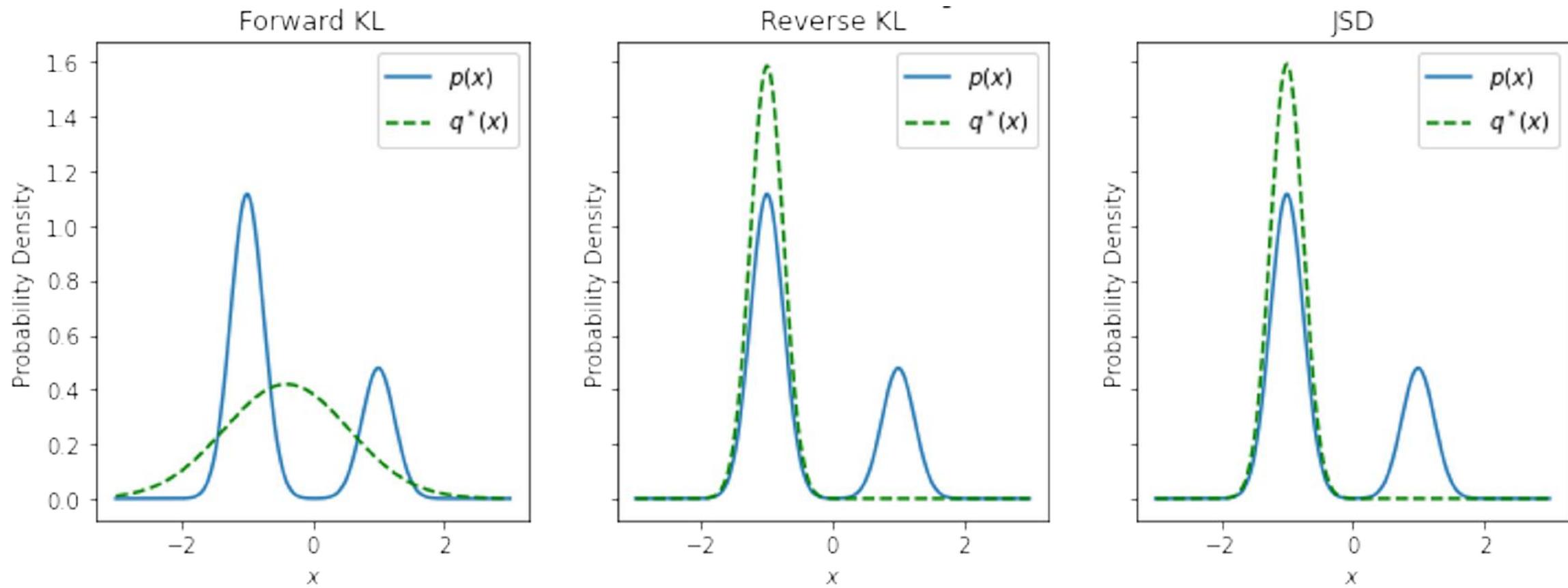
Maximum likelihood

$$q^* = \operatorname{argmin}_q D_{\text{KL}}(q\|p)$$



Reverse KL

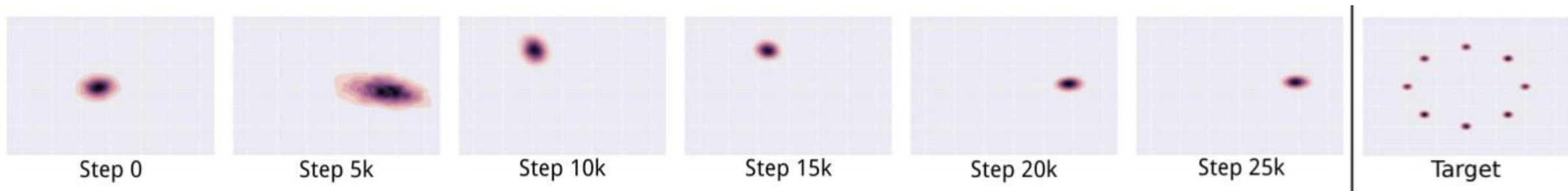
KL and JSD



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. For example, using mixture of Gaussians as opposed to a single isotropic gaussian.

Mode Collapse



- Standard GAN training collapses when the true distribution is a mixture of gaussians!

(Figure from Metz et al. 2016)

Back to GANs

Recall

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


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?

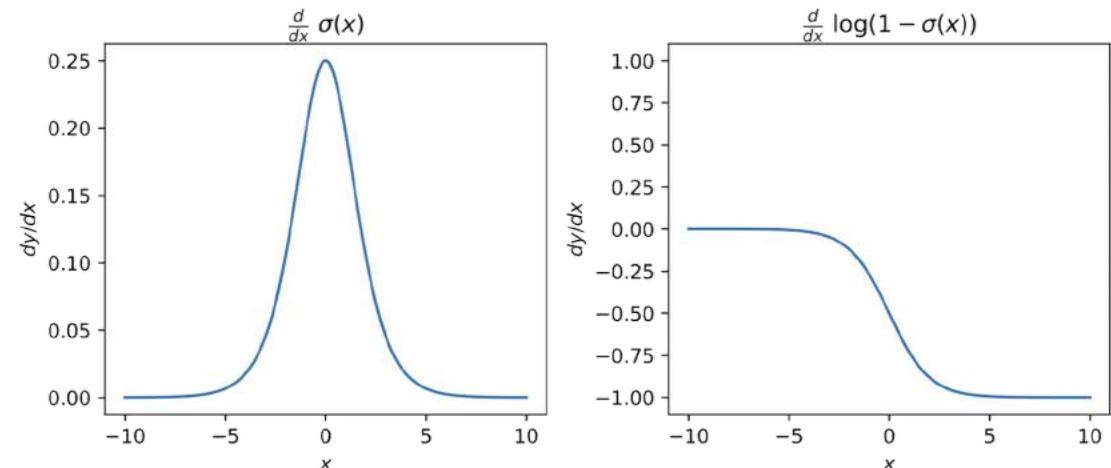
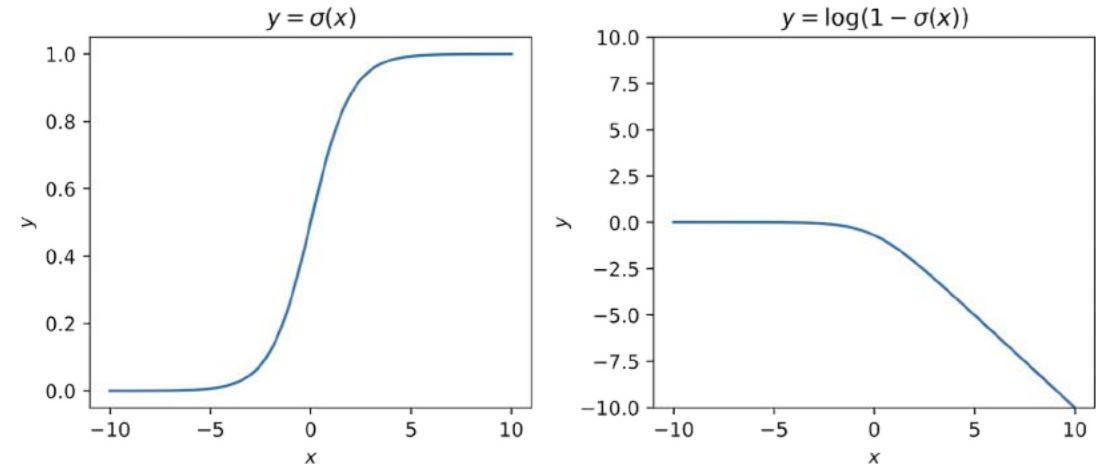
Discriminator Saturation

- Generator samples confidently classified as fake by the discriminator receive no gradient for the generator update.

$$\nabla_{G(z)} \log(1 - D(G(z))) \quad \text{where}$$

$$D(x) = \text{sigmoid}(x; \theta) = \sigma(x; \theta)$$

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



Avoiding Discriminator Saturation: (1) Alternating Optimization

- Alternate gradient steps on 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

Avoiding Discriminator Saturation: (2) Non-Saturating Formulation

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

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



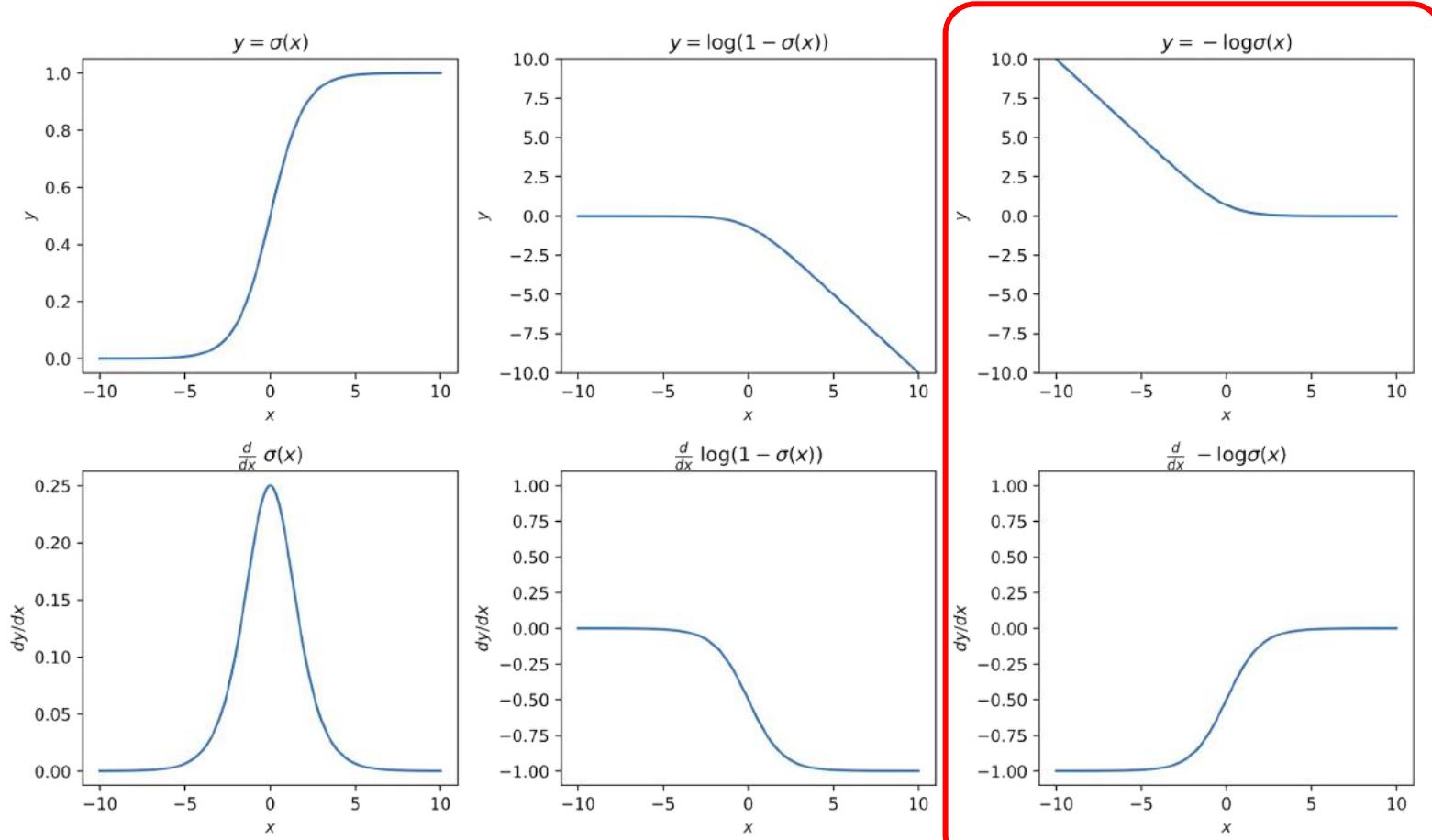
→ Not zero-sum

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

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

Avoiding Discriminator Saturation: (2) Non Saturating Formulation

- ORIGINAL ISSUE:
Generator samples confidently classified as fake by the discriminator receive no gradient for the generator update.
- FIX: non-saturating loss for when discriminator confident about fake



Lecture overview

- Motivation and Definition of Implicit Models
- Original GAN (Goodfellow et al, 2014)
- Evaluation: Parzen, Inception, Frechet
- Theory of GANs
- **GAN Progression**
 - DC GAN (Radford et al, 2016)
 - Improved Training of GANs (Salimans et al'16), Projected GAN (Sauer et al'21)
 - WGAN, WGAN-GP, Progressive GAN, SN-GAN, SAGAN
 - BigGAN, BigGAN-Deep, StyleGAN, StyleGAN2, StyleGAN3, StyleGAN-XL, VIB-GAN
- Conditional GANs, Cycle-Consistent Adversarial Networks
- GANs and Representations
- Applications

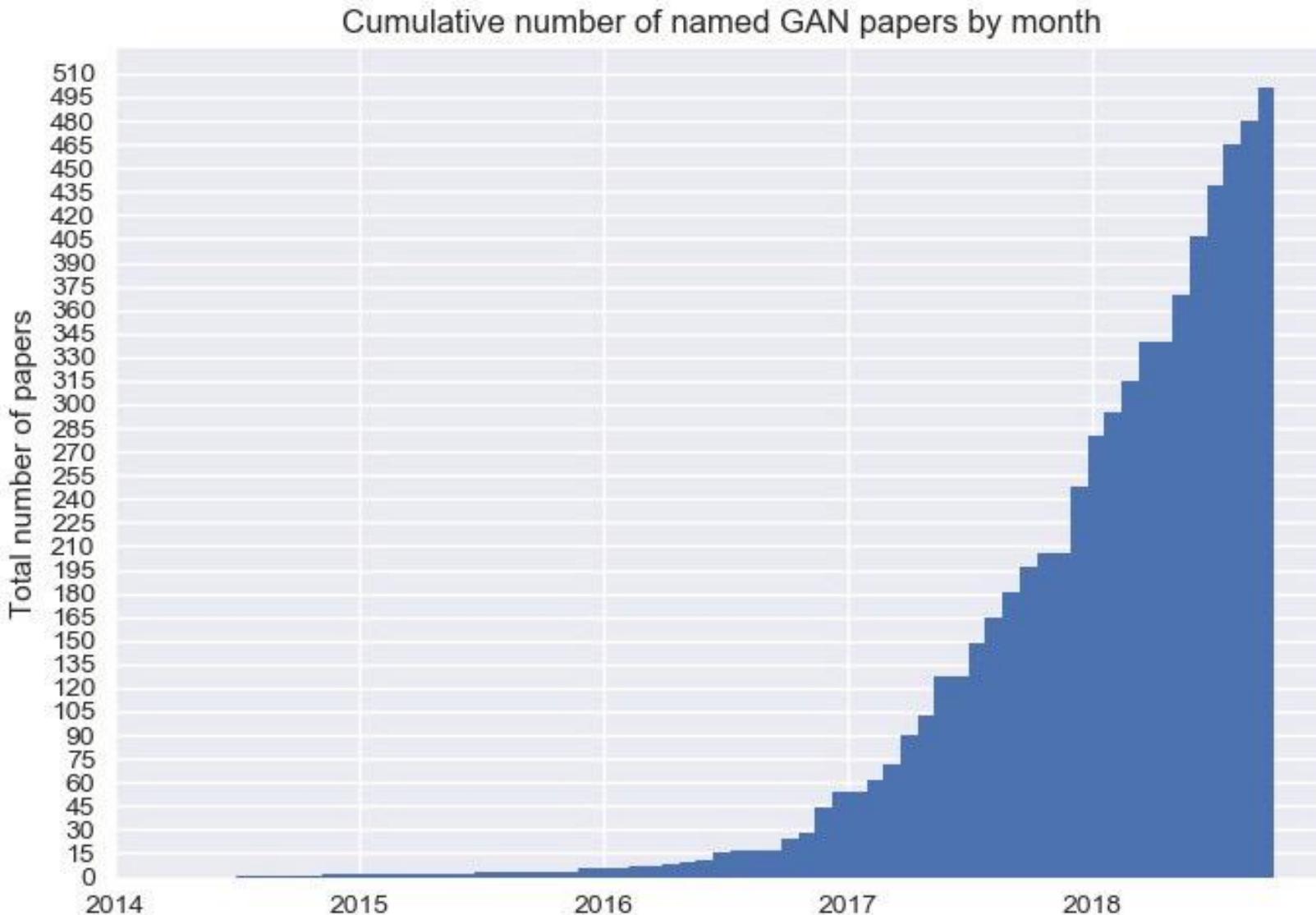
GAN Zoo

AN — Generative Adversarial Networks
3D-GAN — Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling acGAN — Face Aging With Conditional Generative Adversarial Networks
AC-GAN — Conditional Image Synthesis With Auxiliary Classifier GANs
AdaGAN — AdaGAN: Boosting Generative Models
AEGAN — Learning Inverse Mapping by Autoencoder based Generative Adversarial Nets
AffGAN — Amortised MAP Inference for Image Super-resolution
AL-CGAN — Learning to Generate Images of Outdoor Scenes from Attributes and Semantic Layouts
ALI — Adversarially Learned Inference
AMGAN — Generative Adversarial Nets with Labeled Data by Activation Maximization
AnoGAN — Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery
ArtGAN — ArtGAN: Artwork Synthesis with Conditional Categorical GANs
b-GAN — b-GAN: Unified Framework of Generative Adversarial Networks
Bayesian GAN — Deep and Hierarchical Implicit Models
BEGAN — BEGAN: Boundary Equilibrium Generative Adversarial Networks
BiGAN — Adversarial Feature Learning
BS-GAN — Boundary-Seeking Generative Adversarial Networks
CGAN — Conditional Generative Adversarial Nets
CCGAN — Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks
CatGAN — Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks
CoGAN — Coupled Generative Adversarial Networks
Context-RNN-GAN — Contextual RNN-GANs for Abstract Reasoning Diagram Generation
C-RNN-GAN — C-RNN-GAN: Continuous recurrent neural networks with adversarial training
CS-GAN — Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets
CVAE-GAN — CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training
CycleGAN — Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks
DTN — Unsupervised Cross-Domain Image Generation
DCGAN — Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks
DiscoGAN — Learning to Discover Cross-Domain Relations with Generative Adversarial Networks
DR-GAN — Disentangled Representation Learning GAN for Pose-Invariant Face Recognition
DualGAN — DualGAN: Unsupervised Dual Learning for Image-to-Image Translation
EBGAN — Energy-based Generative Adversarial Network
f-GAN — f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization
GAWWN — Learning What and Where to Draw
GoGAN — Gang of GANs: Generative Adversarial Networks with Maximum Margin Ranking
GP-GAN — GP-GAN: Towards Realistic High-Resolution Image Blending
IAN — Neural Photo Editing with Introspective Adversarial Networks
iGAN — Generative Visual Manipulation on the Natural Image Manifold
IcGAN — Invertible Conditional GANs for image editing
ID-CGAN — Image De-raining Using a Conditional Generative Adversarial Network
Improved GAN — Improved Techniques for Training GANs
InfoGAN — InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Networks
LAGAN — Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis
LAPGAN — Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks
LR-GAN — LR-GAN: Layered Recursive Generative Adversarial Networks for Image Generation
LSGAN — Least Squares Generative Adversarial Networks
LS-GAN — Loss-Sensitive Generative Adversarial Networks on Lipschitz Densities
MGAN — Precomputed Real-Time Texture Synthesis with Markovian Generative Adversarial Networks
MAGAN — MAGAN: Margin Adaptation for Generative Adversarial Networks
MAD-GAN — Multi-Agent Diverse Generative Adversarial Networks
MalGAN — Generating Adversarial Malware Examples for Black-Box Attacks Based on GAN
MaliGAN — Maximum-Likelihood Augmented Discrete Generative Adversarial Networks
MARTA-GAN — Deep Unsupervised Representation Learning for Remote Sensing Images
McGAN — McGan: Mean and Covariance Feature Matching GAN
MDGAN — Mode Regularized Generative Adversarial Networks
MedGAN — Generating Multi-label Discrete Electronic Health Records using Generative Adversarial Networks
MIX+GAN — Generalization and Equilibrium in Generative Adversarial Nets (GANs)
MPM-GAN — Message Passing Multi-Agent GANs
MV-BiGAN — Multi-view Generative Adversarial Networks
pix2pix — Image-to-Image Translation with Conditional Adversarial Networks
PPGN — Plug & Play Generative Networks: Conditional Iterative Generation of Images in Latent Space
PrGAN — 3D Shape Induction from 2D Views of Multiple Objects
RenderGAN — RenderGAN: Generating Realistic Labeled Data
RTT-GAN — Recurrent Topic-Transition GAN for Visual Paragraph Generation
SGAN — Stacked Generative Adversarial Networks
SGAN — Texture Synthesis with Spatial Generative Adversarial Networks
SAD-GAN — SAD-GAN: Synthetic Autonomous Driving using Generative Adversarial Networks
SalGAN — SalGAN: Visual Saliency Prediction with Generative Adversarial Networks
SEGAN — SEGAN: Speech Enhancement Generative Adversarial Network
SegGAN — SegGAN: Segmenting and Generating the Invisible
SeqGAN — SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient
SimGAN — Learning from Simulated and Unsupervised Images through Adversarial Training
SketchGAN — Adversarial Training For Sketch Retrieval
SL-GAN — Semi-Latent GAN: Learning to generate and modify facial images from attributes
Softmax-GAN — Softmax GAN
SRGAN — Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network
S2GAN — Generative Image Modeling using Style and Structure Adversarial Networks
SSL-GAN — Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks
StackGAN — StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks
TGAN — Temporal Generative Adversarial Nets
TAC-GAN — TAC-GAN — Text Conditioned Auxiliary Classifier Generative Adversarial Network
TP-GAN — Beyond Face Rotation: Global and Local Perception GAN for Photorealistic and Identity Preserving Frontal View Synthesis
Triple-GAN — Triple Generative Adversarial Nets
Unrolled GAN — Unrolled Generative Adversarial Networks
VGAN — Generating Videos with Scene Dynamics
VGAN — Generative Adversarial Networks as Variational Training of Energy Based Models
VAE-GAN — Autoencoding beyond pixels using a learned similarity metric
VariGAN — Multi-View Image Generation from a Single-View
VIGAN — Image Generation and Editing with Variational Info Generative Adversarial Networks
WGAN — Wasserstein GAN
WGAN-GP — Improved Training of Wasserstein GANs
WaterGAN — WaterGAN: Unsupervised Generative Network to Enable Real-time Color Correction of Monocular Underwater Images

Deep Hunt, blog by Avinash Hindupur
<https://deephunt.in/the-gan-zoo-79597dc8c347>

GAN Zoo

An explo-GAN of papers



Explosive growth — All the named GAN variants cumulatively since 2014.

Credit: Bruno Gavranović

Deep Hunt, blog by Avinash Hindupur

Lecture overview

- Motivation and Definition of Implicit Models
- Original GAN (Goodfellow et al, 2014)
- Evaluation: Parzen, Inception, Frechet
- Theory of GANs
- **GAN Progression**
 - DC GAN (Radford et al, 2016)
 - Improved Training of GANs (Salimans et al'16), Projected GAN (Sauer et al'21)
 - WGAN, WGAN-GP, Progressive GAN, SN-GAN, SAGAN
 - BigGAN, BigGAN-Deep, StyleGAN, StyleGAN-v2, VIB-GAN
- Conditional GANs, Cycle-Consistent Adversarial Networks
- GANs and Representations
- Applications

Deep Convolutional GAN (DCGAN)

UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS

Alec Radford & Luke Metz

indico Research

Boston, MA

{alec, luke}@indico.io

Soumith Chintala

Facebook AI Research

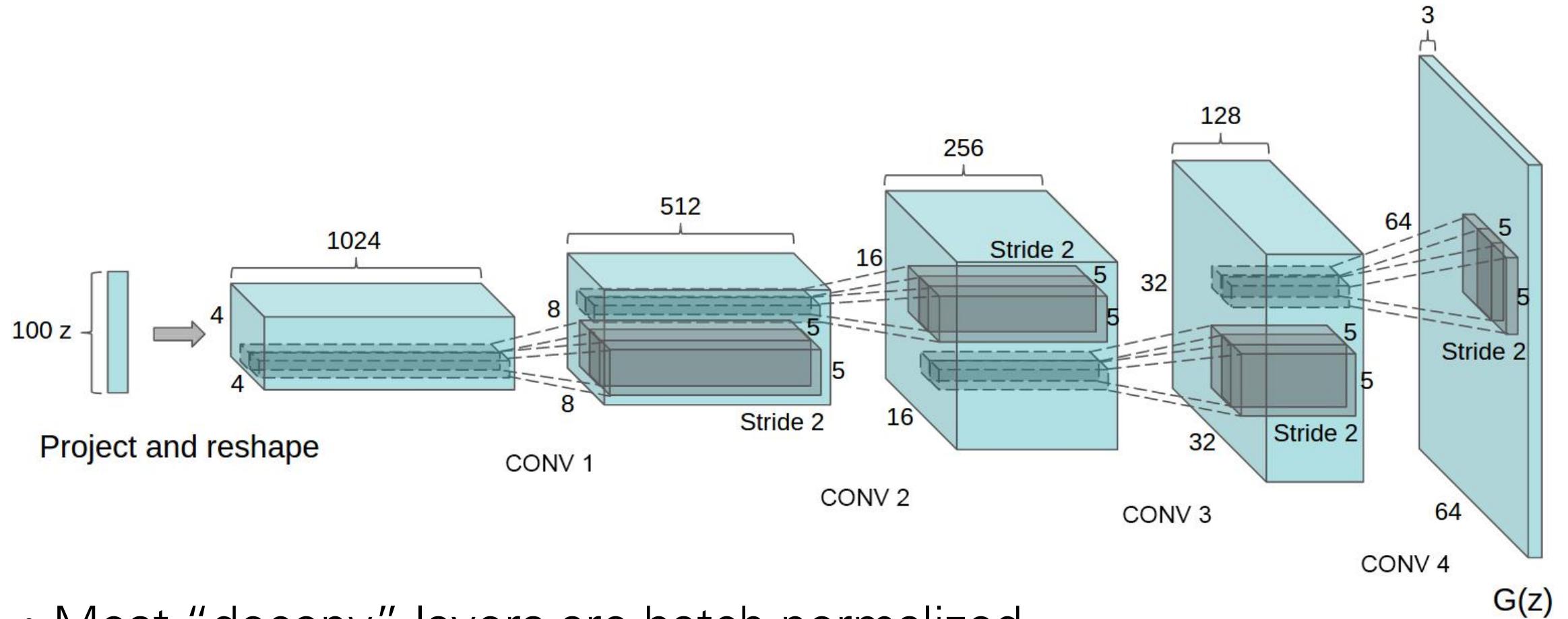
New York, NY

soumith@fb.com

ABSTRACT

In recent years, supervised learning with convolutional networks (CNNs) has seen huge adoption in computer vision applications. Comparatively, unsupervised learning with CNNs has received less attention. In this work we hope to help bridge the gap between the success of CNNs for supervised learning and unsupervised learning. We introduce a class of CNNs called deep convolutional generative adversarial networks (DCGANs), that have certain architectural constraints, and demonstrate that they are a strong candidate for unsupervised learning. Training on various image datasets, we show convincing evidence that our deep convolutional adversarial pair learns a hierarchy of representations from object parts to scenes in both the generator and discriminator. Additionally, we use the learned features for novel tasks - demonstrating their applicability as general image representations.

Deep Convolutional GAN (DCGAN)

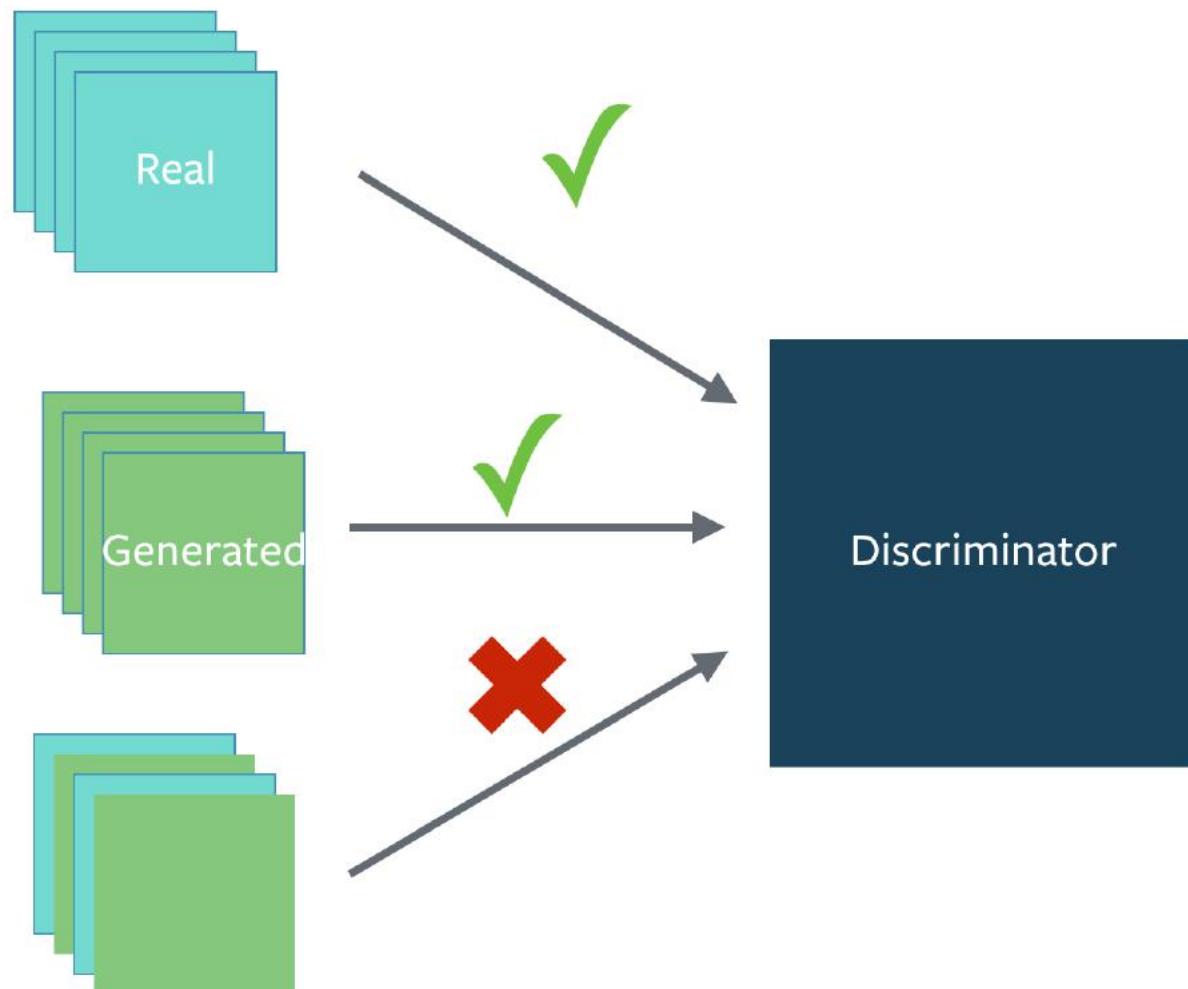


- Most “deconv” layers are batch normalized

DCGAN - Architecture Design

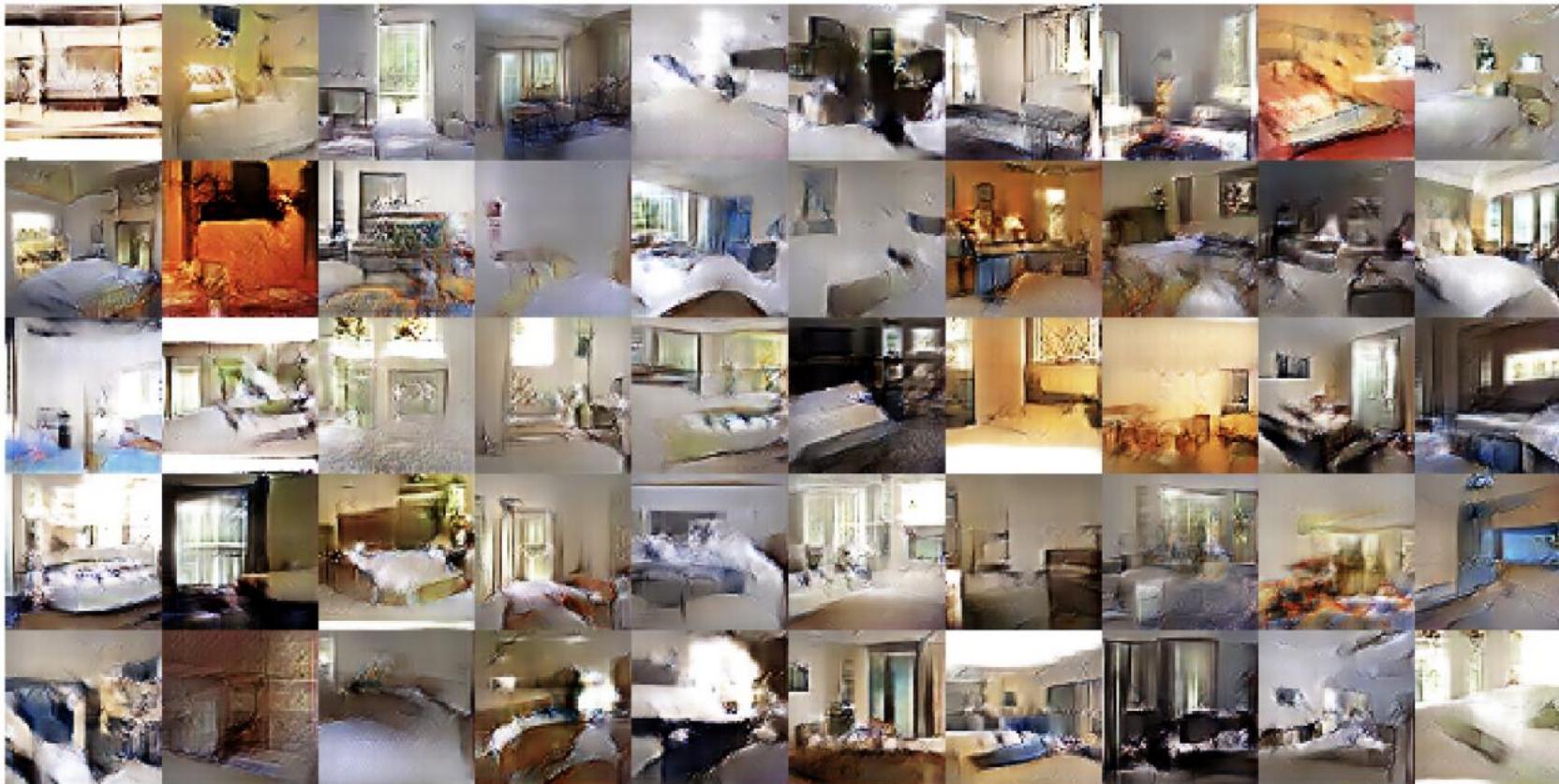
- Supervised Learning CNNs not directly usable
 - Remove max-pooling and mean-pooling
 - Upsample using transposed convolutions in the generator
 - Downsample with strided convolutions and average pooling
 - Non-Linearity: ReLU for generator, Leaky-ReLU (0.2) for discriminator
 - Output Non-Linearity: tanh for Generator, sigmoid for discriminator
 - Batch Normalization used to prevent mode collapse
 - Batch Normalization is not applied at the output of G and input of D
- Optimization details
 - Adam: small LR - 2e-4; small momentum: 0.5, batch-size: 128

DCGAN Batch Norm



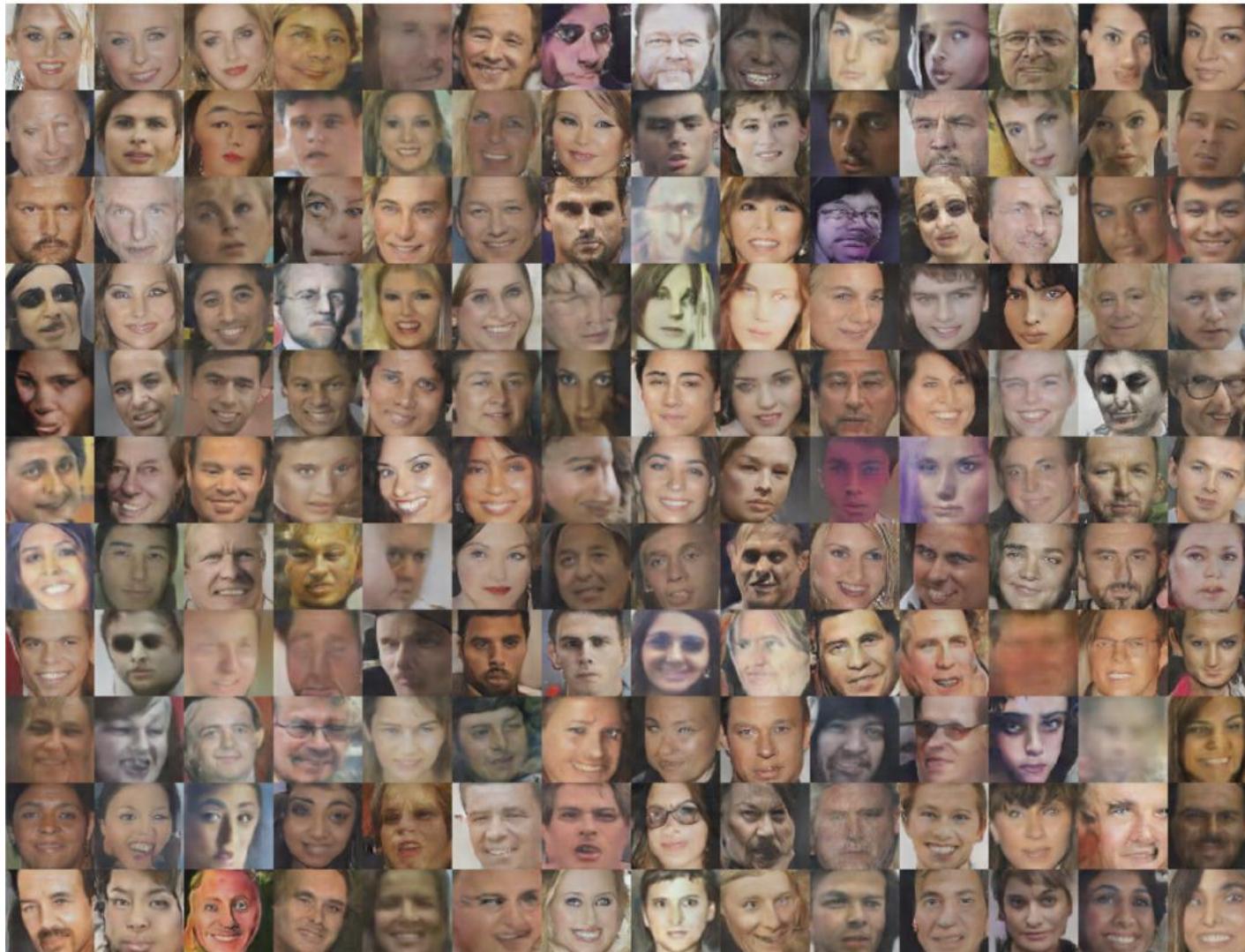
DCGAN - Key Results

- Good samples on datasets with 3M images (Faces, Bedrooms) for the first time



[Radford et al 2016]

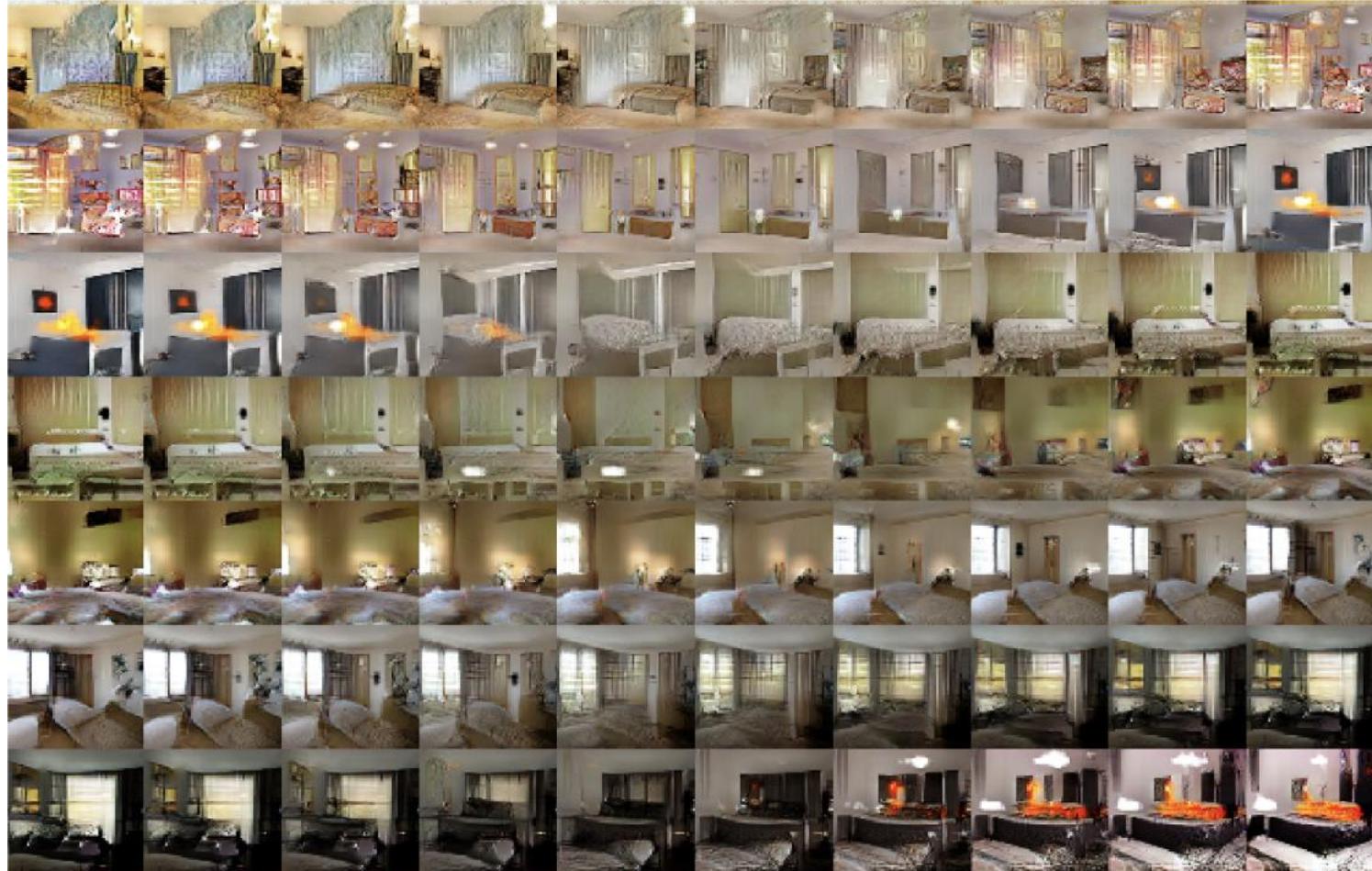
DCGAN - Key Results



[Radford et al 2016]

DCGAN - Key Results

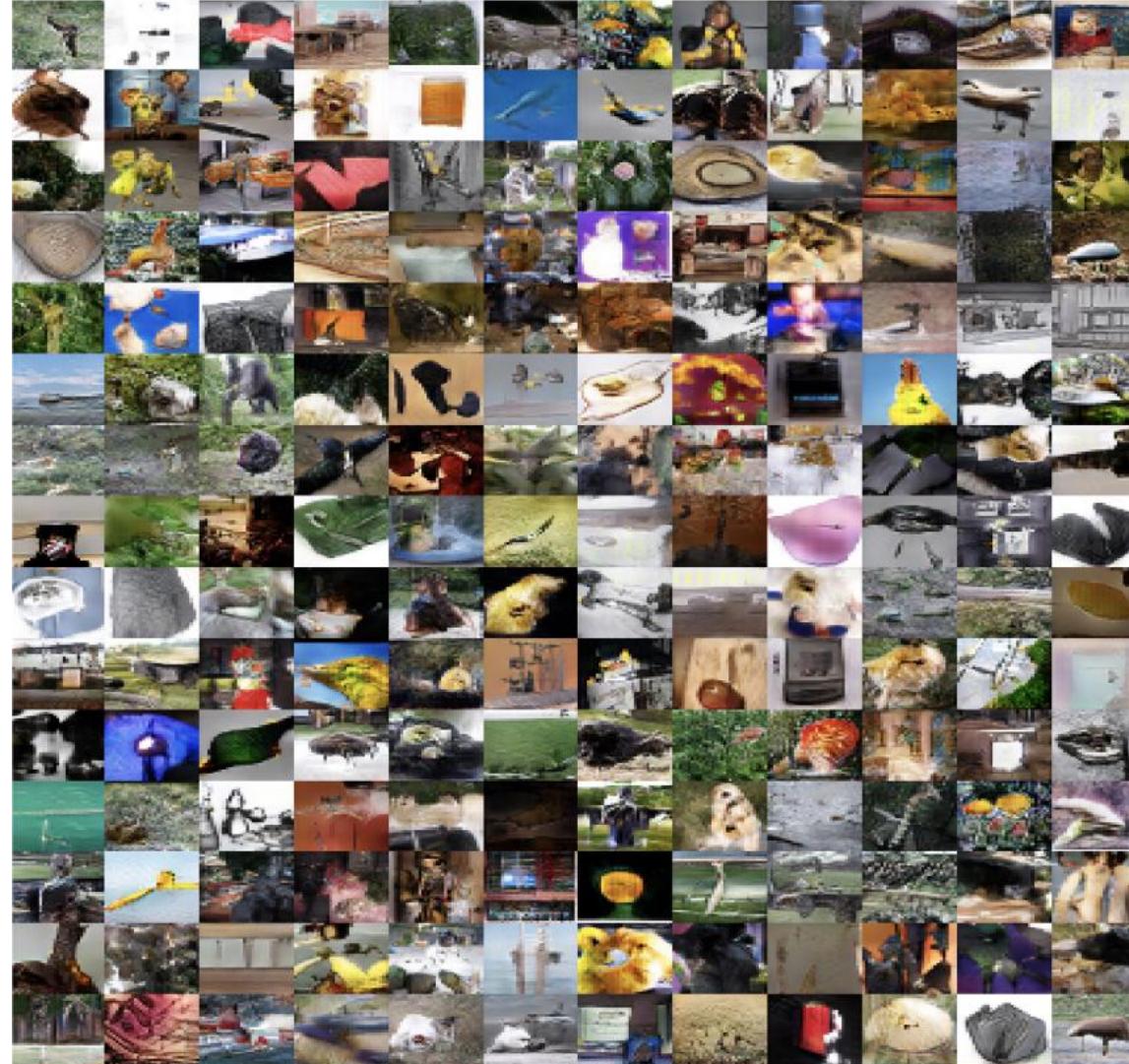
- Smooth interpolations in high dimensions



[Radford et al 2016]

DCGAN - Key Results

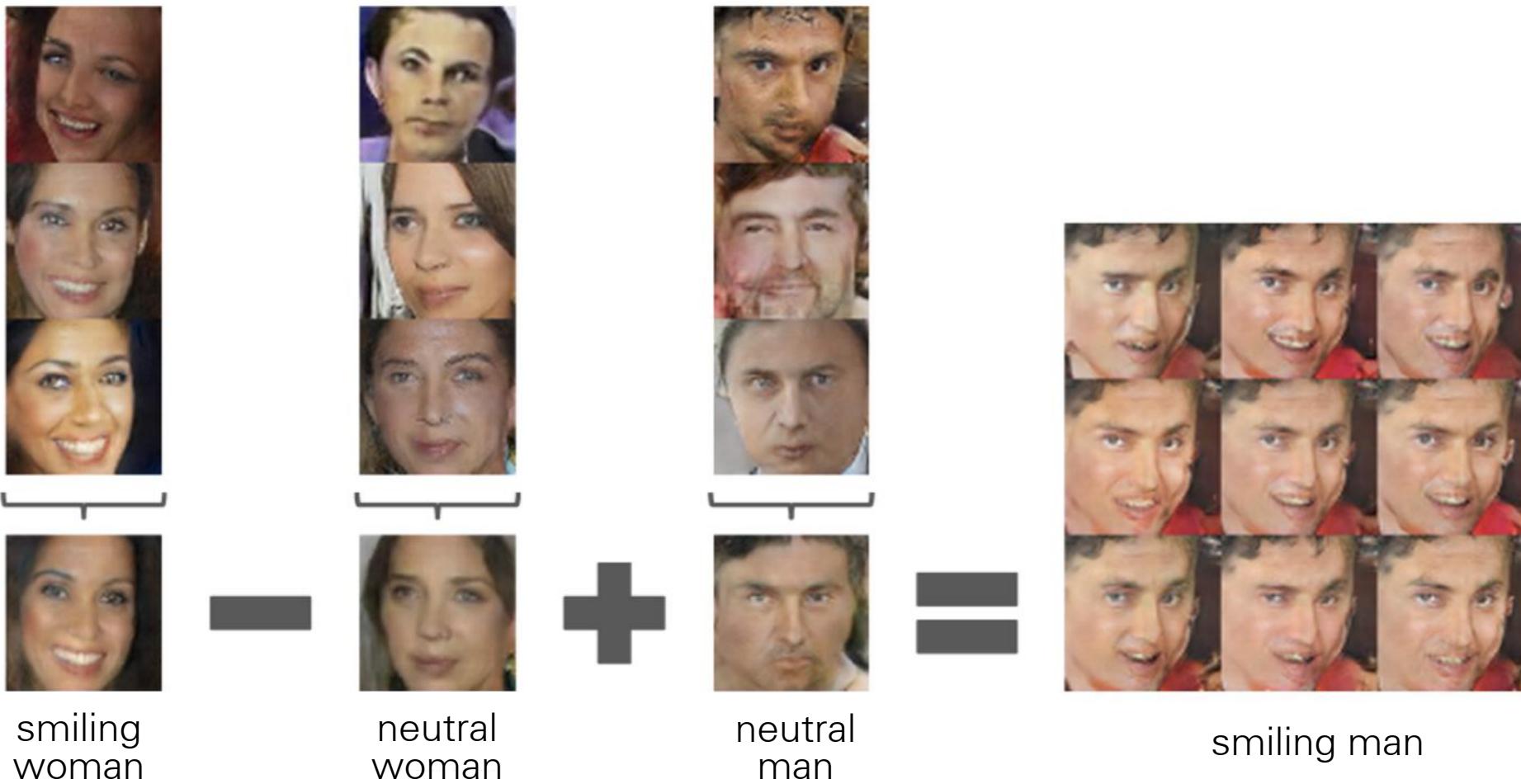
- Imagenet samples



[Radford et al 2016]

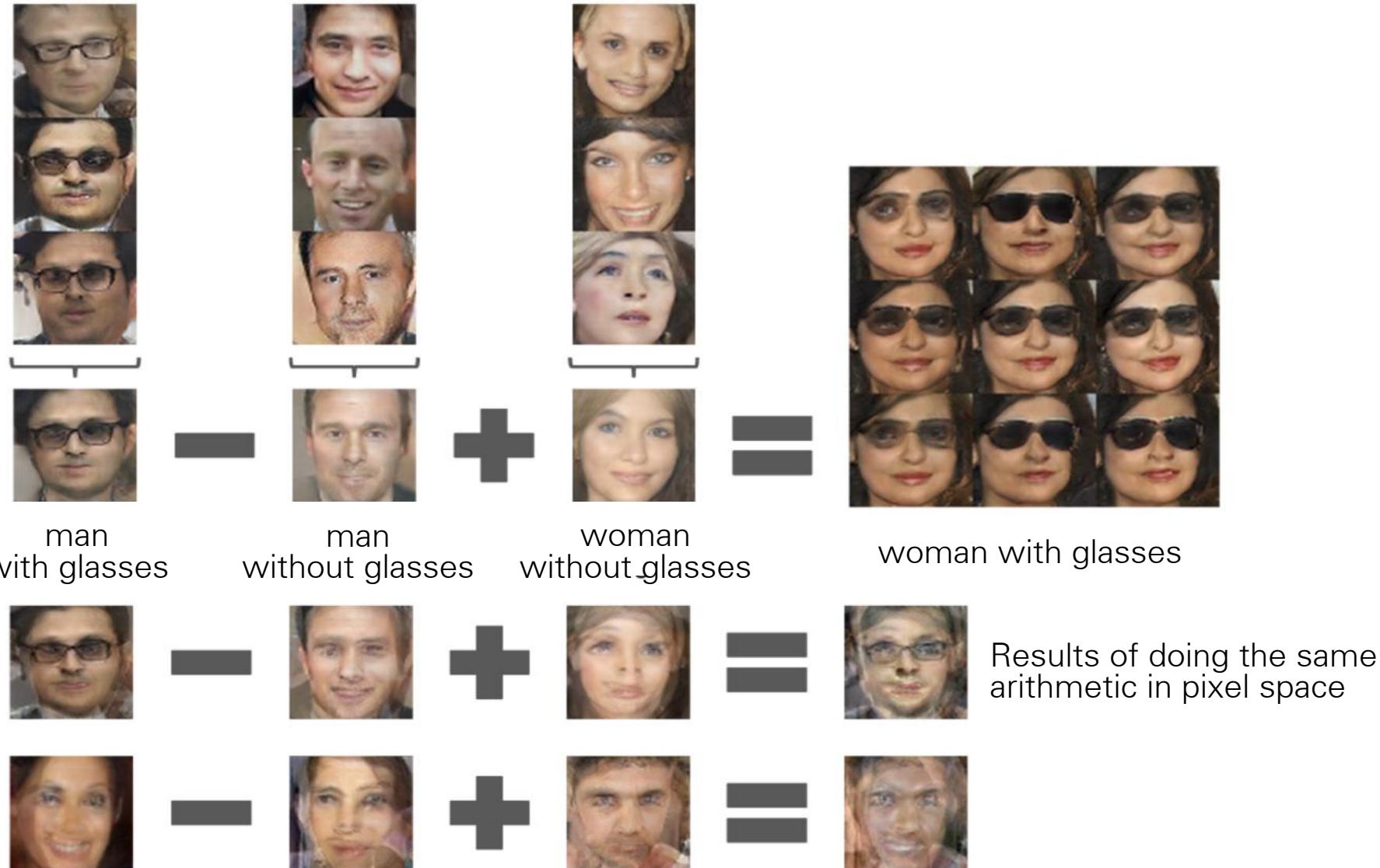
DCGAN - Key Results

- Vector Arithmetic

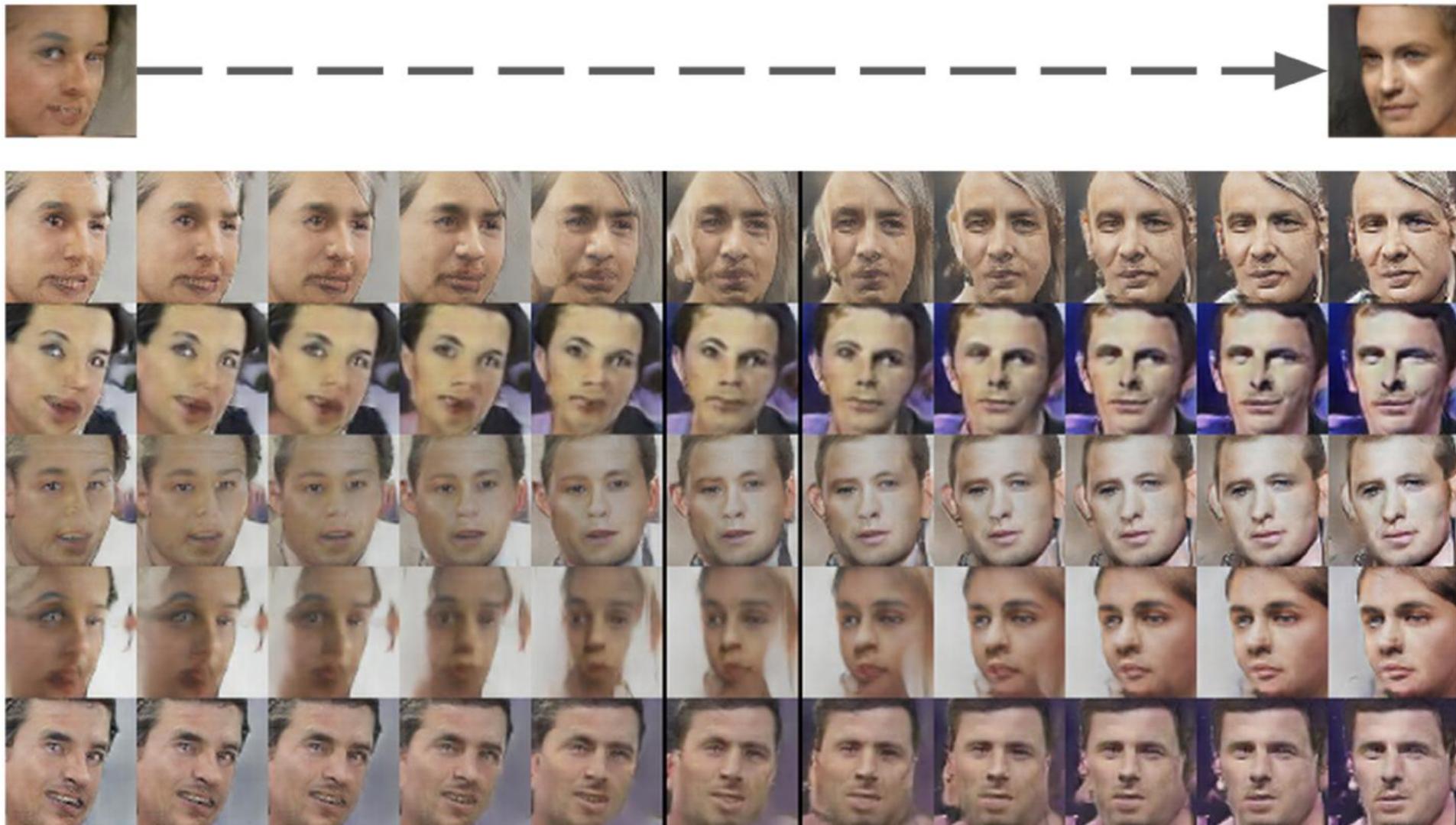


[Radford et al 2016]

DCGAN - Key Results



DCGAN - Key Results



[Radford et al 2016]

DCGAN - Key Results

- Representation Learning

| Model | Accuracy | Accuracy (400 per class) | max # of features units |
|----------------------------|----------|--------------------------|-------------------------|
| 1 Layer K-means | 80.6% | 63.7% ($\pm 0.7\%$) | 4800 |
| 3 Layer K-means Learned RF | 82.0% | 70.7% ($\pm 0.7\%$) | 3200 |
| View Invariant K-means | 81.9% | 72.6% ($\pm 0.7\%$) | 6400 |
| Exemplar CNN | 84.3% | 77.4% ($\pm 0.2\%$) | 1024 |
| DCGAN (ours) + L2-SVM | 82.8% | 73.8% ($\pm 0.4\%$) | 512 |

DCGAN - Conclusions

- Incredible samples for any generative model
- GANs could be made to work well with architecture details
- Perceptually good samples and interpolations
- Representation Learning
- **Problems to address:**
 - Unstable training
 - Brittle architecture / hyperparameters

Lecture overview

- Motivation and Definition of Implicit Models
- Original GAN (Goodfellow et al, 2014)
- Evaluation: Parzen, Inception, Frechet
- Theory of GANs
- **GAN Progression**
 - DC GAN (Radford et al, 2016)
 - **Improved Training of GANs (Salimans et al'16)**, Projected GAN (Sauer et al'21)
 - WGAN, WGAN-GP, Progressive GAN, SN-GAN, SAGAN
 - BigGAN, BigGAN-Deep, StyleGAN, StyleGAN2, StyleGAN3, StyleGAN-XL, VIB-GAN
- Conditional GANs, Cycle-Consistent Adversarial Networks
- GANs and Representations
- Applications

Improved training of GANs

- Feature Matching
- Minibatch discrimination
- Historical Averaging
- Virtual batch normalization
- One-sided label smoothing

Improved Techniques for Training GANs

Tim Salimans
tim@openai.com

Ian Goodfellow
ian@openai.com

Wojciech Zaremba
woj@openai.com

Vicki Cheung
vicki@openai.com

Alec Radford
alec.radford@gmail.com

Xi Chen
peter@openai.com

[Salimans 2016]

Improved training of GANs

- Feature Matching

$$||\mathbb{E}_{x \sim p_{\text{data}}} f(x) - \mathbb{E}_{z \sim p(z)} f(G(z))||^2$$

Generator objective

[Salimans 2016]

Improved training of GANs

- Minibatch discrimination

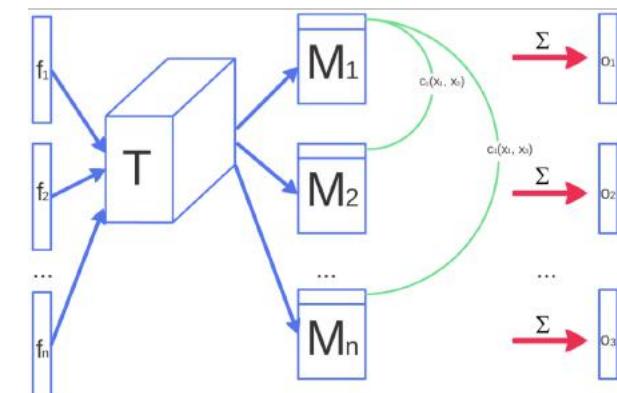
$$\mathbf{f}(\mathbf{x}_i) \in \mathbb{R}^A \quad T \in \mathbb{R}^{A \times B \times C} \quad M_i \in \mathbb{R}^{B \times C}$$

$$c_b(\mathbf{x}_i, \mathbf{x}_j) = \exp(-||M_{i,b} - M_{j,b}||_{L_1}) \in \mathbb{R}$$

$$o(\mathbf{x}_i)_b = \sum_{j=1}^n c_b(\mathbf{x}_i, \mathbf{x}_j) \in \mathbb{R}$$

$$o(\mathbf{x}_i) = [o(\mathbf{x}_i)_1, o(\mathbf{x}_i)_2, \dots, o(\mathbf{x}_i)_B] \in \mathbb{R}^B$$

$$o(\mathbf{X}) \in \mathbb{R}^{n \times B}$$



[Salimans 2016]

Allows to incorporate side information from other samples and is superior to feature matching in the unconditional setting.
Helps addressing mode collapse by allowing discriminator to detect if the generated samples are too close to each other.

Improved training of GANs

- Historical Averaging

$$\|\boldsymbol{\theta} - \frac{1}{t} \sum_{i=1}^t \boldsymbol{\theta}[i]\|^2$$

Improved training of GANs

- One-sided label smoothing

Default discriminator cost:

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



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

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

Improved training of GANs

- Why one-sided?

Reinforces current generator behavior

$$D(\mathbf{x}) = \frac{(1 - \alpha)p_{\text{data}}(\mathbf{x}) + \beta p_{\text{model}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_{\text{model}}(\mathbf{x})}$$

Improved training of GANs

- Virtual Batch Normalization
 - Use a reference batch (fixed) to compute normalization statistics
 - Construct a batch containing the sample and reference batch

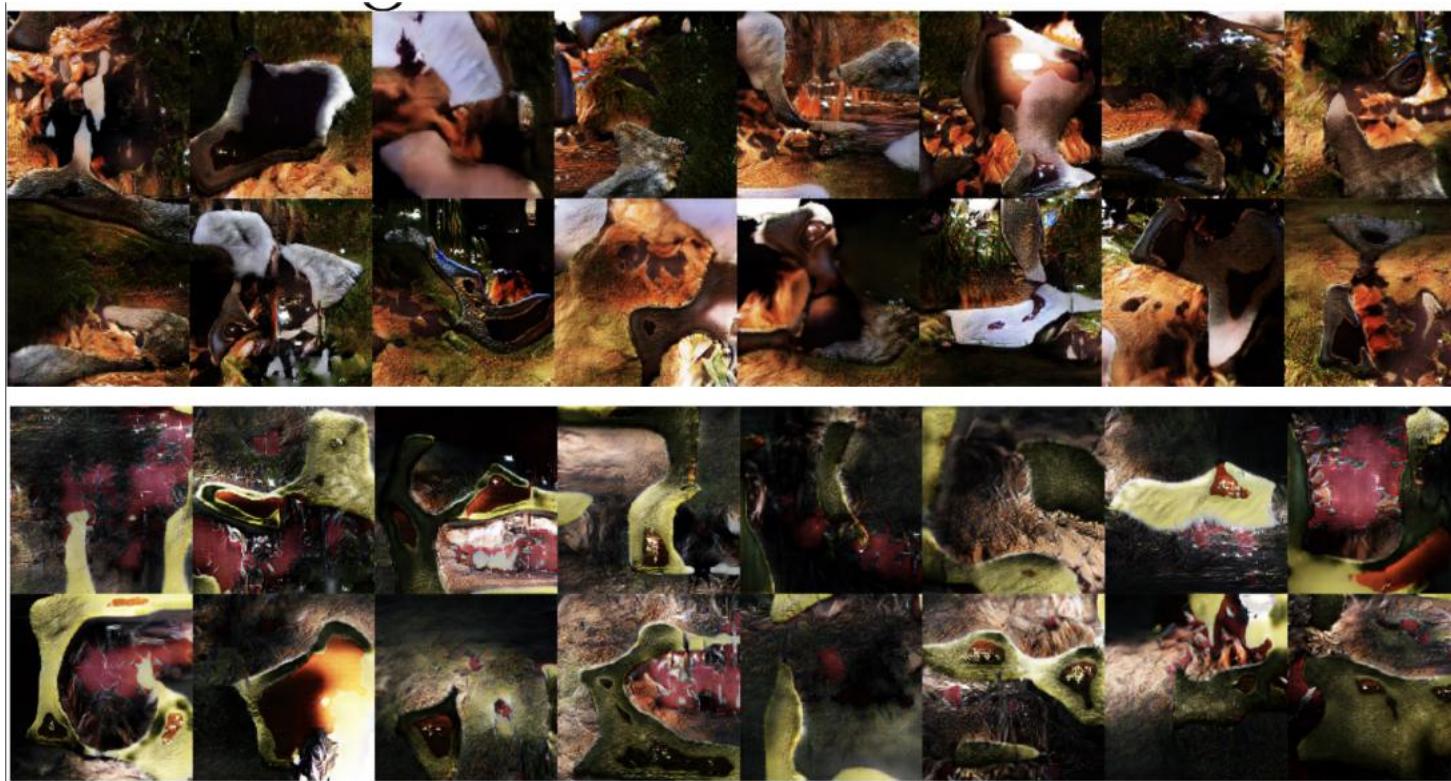


Figure source:
NeurIPS tutorial
Goodfellow 2016

Improved training of GANs

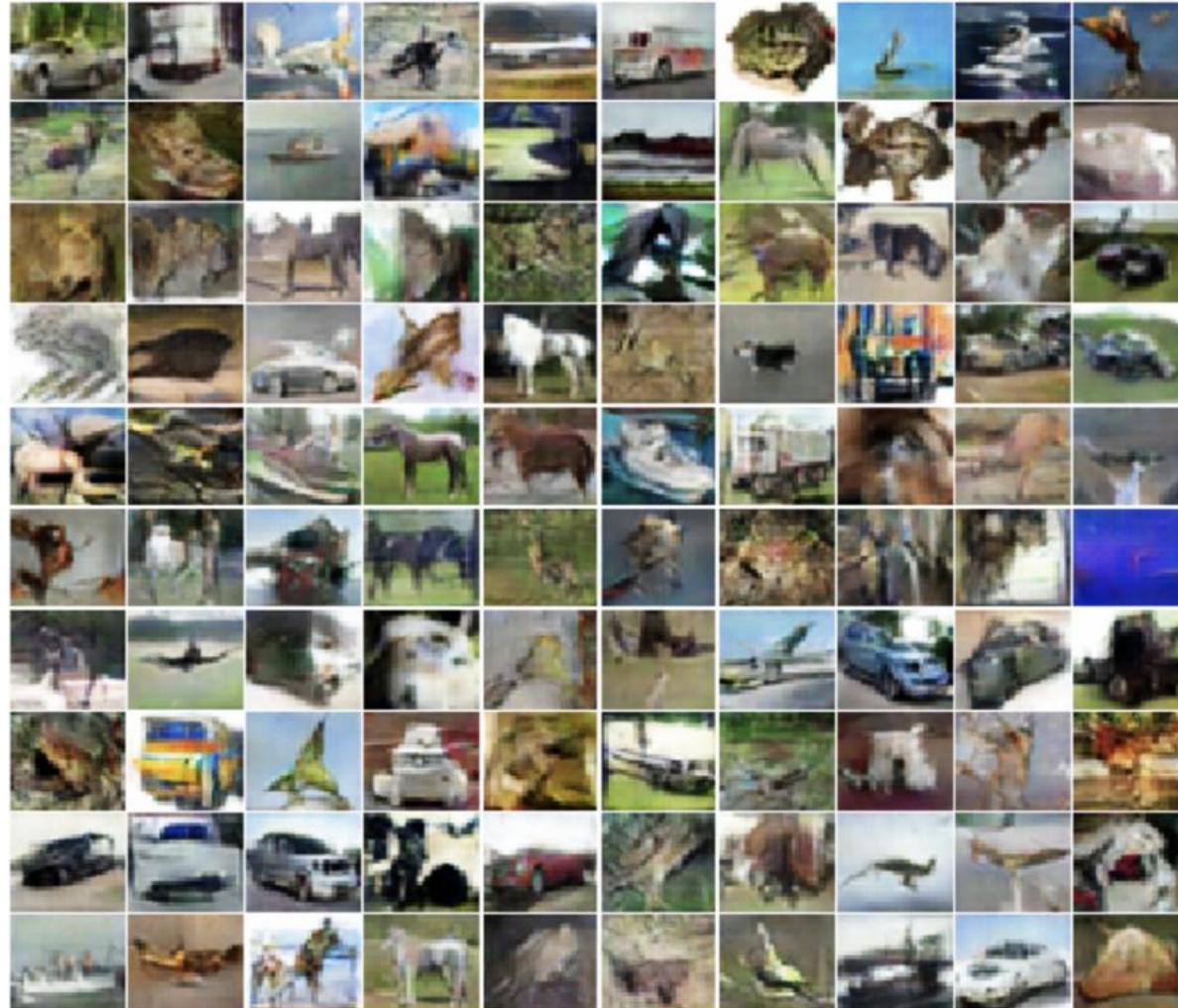
- Semi-Supervised Learning
 - Predict labels in addition to fake/real in the discriminator
 - Approximate way of modeling $p(x,y)$
 - Generator doesn't have to be made conditional $p(x|y)$
 - Use a deeper architecture for the discriminator compared to generator

$$\begin{aligned} L &= -\mathbb{E}_{\mathbf{x}, y \sim p_{\text{data}}(\mathbf{x}, y)} [\log p_{\text{model}}(y|\mathbf{x})] - \mathbb{E}_{\mathbf{x} \sim G} [\log p_{\text{model}}(y = K+1|\mathbf{x})] \\ &= L_{\text{supervised}} + L_{\text{unsupervised}}, \text{ where} \end{aligned}$$

$$L_{\text{supervised}} = -\mathbb{E}_{\mathbf{x}, y \sim p_{\text{data}}(\mathbf{x}, y)} \log p_{\text{model}}(y|\mathbf{x}, y < K+1)$$

$$L_{\text{unsupervised}} = -\{\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} \log[1 - p_{\text{model}}(y = K+1|\mathbf{x})] + \mathbb{E}_{\mathbf{x} \sim G} \log[p_{\text{model}}(y = K+1|\mathbf{x})]\}$$

Improved training of GANs



Salimans 2016

Lecture overview

- Motivation and Definition of Implicit Models
- Original GAN (Goodfellow et al, 2014)
- Evaluation: Parzen, Inception, Frechet
- Theory of GANs
- **GAN Progression**
 - DC GAN (Radford et al, 2016)
 - Improved Training of GANs (Salimans et al'16), **Projected GAN** (Sauer et al'21)
 - WGAN, WGAN-GP, Progressive GAN, SN-GAN, SAGAN
 - BigGAN, BigGAN-Deep, StyleGAN, StyleGAN2, StyleGAN3, StyleGAN-XL, VIB-GAN
- Conditional GANs, Cycle-Consistent Adversarial Networks
- GANs and Representations
- Applications

Projected GAN

- Training GANs in pretrained feature spaces improves image quality, training speed, and sample efficiency.

$$\min_G \max_{\{D_l\}} \sum_{l \in \mathcal{L}} (\mathbb{E}_{\mathbf{x}} [\log D_l (P_l(\mathbf{x}))] + \mathbb{E}_{\mathbf{z}} [\log (1 - D_l (P_l(G(\mathbf{z}))))])$$

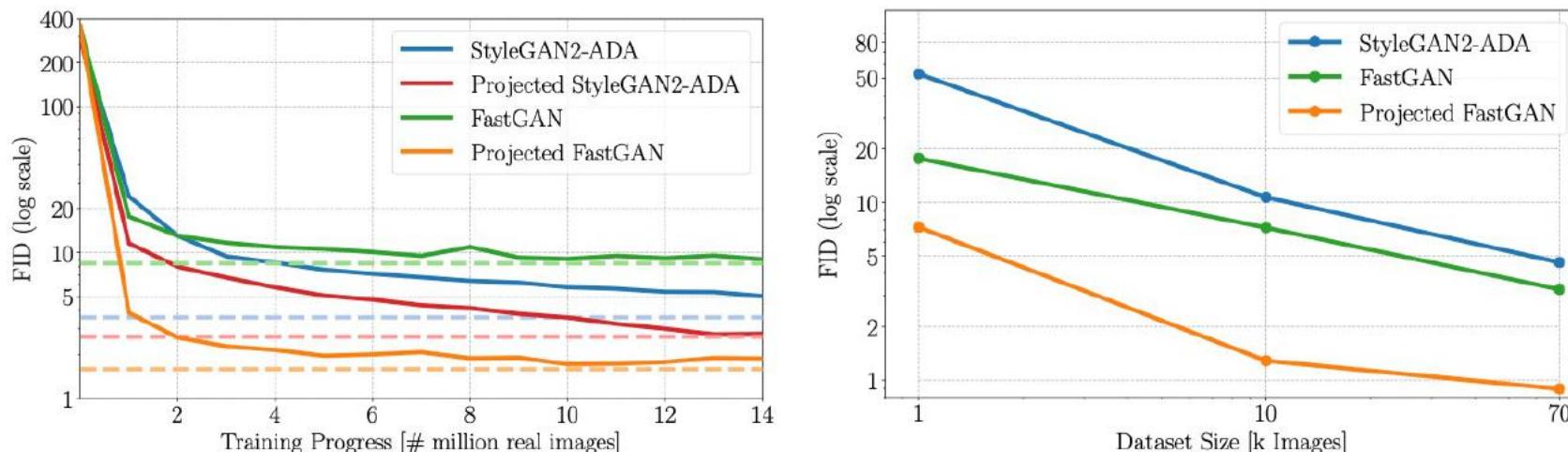


Figure 4: Training Properties. Left: Projected FastGAN surpasses the best FID of StyleGAN2 (at 88 M images) after just 1.1 M images on LSUN-Church. Right: Projected FastGAN yields significantly improved FID scores, even when using subsets of CLEVR with 1k and 10k samples.

Projected GAN

- Training GANs in pretrained feature spaces improves image quality, training speed, and sample efficiency.

$$\min_G \max_{\{D_l\}} \sum_{l \in \mathcal{L}} (\mathbb{E}_{\mathbf{x}} [\log D_l (P_l(\mathbf{x}))] + \mathbb{E}_{\mathbf{z}} [\log (1 - D_l (P_l(G(\mathbf{z}))))])$$

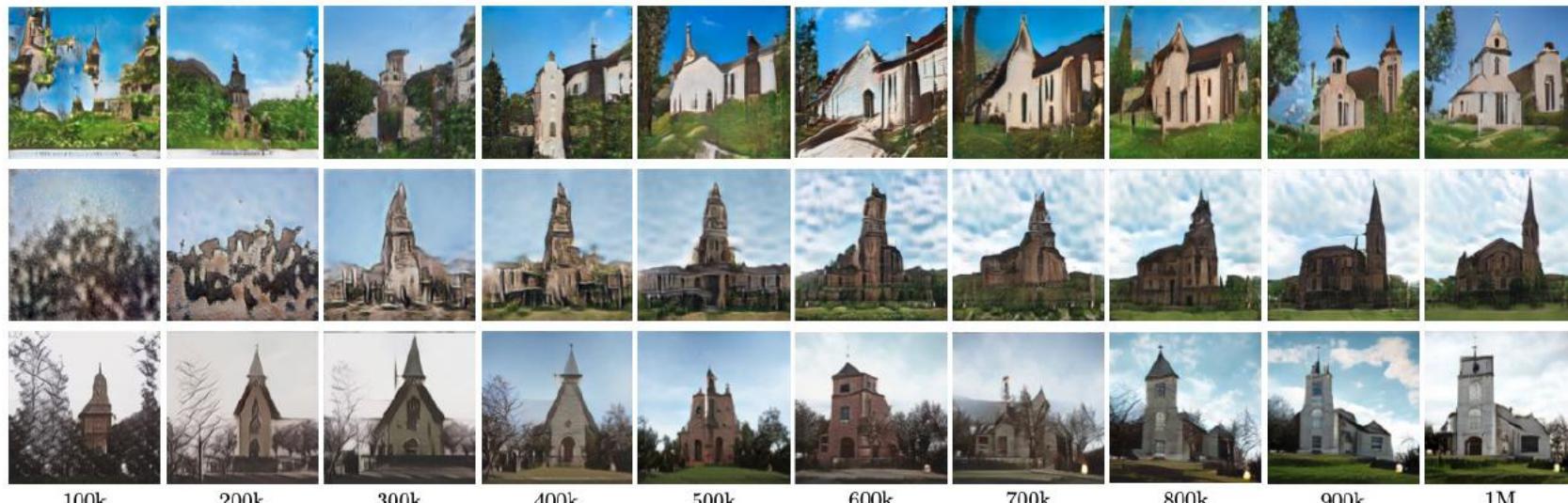


Figure 5: **Training progress on LSUN church at 256^2 pixels.** Shown are samples for a fixed noise vector \mathbf{z} over k images. From top to bottom: FastGAN, StyleGAN2-ADA, Projected GAN.

Projected GAN

- The choice of pretrained network is important!

| | EfficientNet | | | | | ResNet | | Transformer | | |
|--------------|--------------|-------|-------|-------|-------|--------|-------|-------------|-------|--------|
| | lite0 | lite1 | lite2 | lite3 | lite4 | R18 | R50 | R50-CLIP | DeiT | ViT |
| Params (M) ↓ | 2.96 | 3.72 | 4.36 | 6.42 | 11.15 | 11.18 | 23.51 | 23.53 | 92.36 | 317.52 |
| IN top-1 ↑ | 75.48 | 76.64 | 77.47 | 79.82 | 81.54 | 69.75 | 79.04 | N/A | 85.42 | 85.16 |
| FID ↓ | 2.53 | 1.65 | 1.69 | 1.79 | 2.35 | 4.16 | 4.40 | 3.80 | 2.46 | 12.38 |

Table 2: **Pretrained Feature Networks Study.** We train the projected GAN with different pretrained feature networks. We find that compact EfficientNets outperform both ResNets and Transformers.

- More details:
 - Multi-Scale Discriminators
 - Random Projections
 - Cross-Channel Mixing (CCM)
 - Cross-Scale Mixing (CSM)

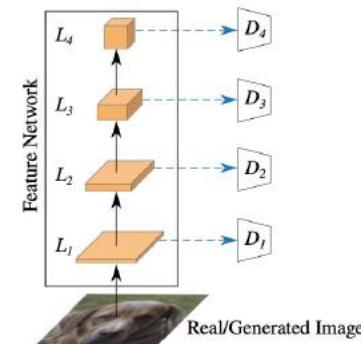


Figure 2: **CCM** (dashed blue arrows) employs 1×1 convolutions with random weights.

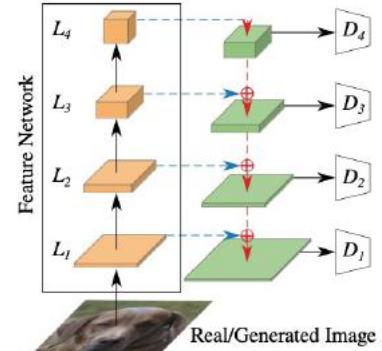
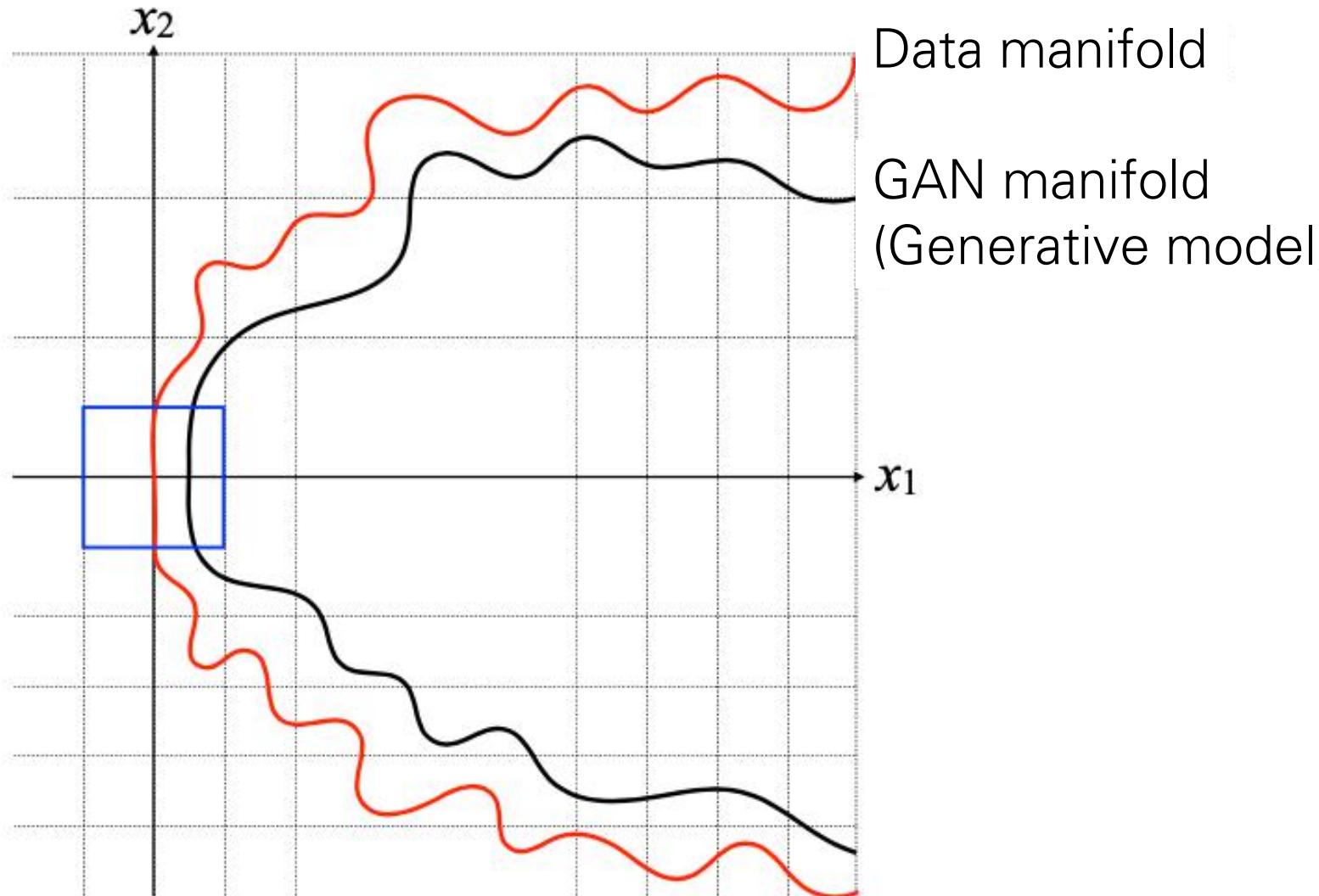


Figure 3: **CSM** (dashed red arrows) adds random 3×3 convolutions and bilinear upsampling, yielding a U-Network.

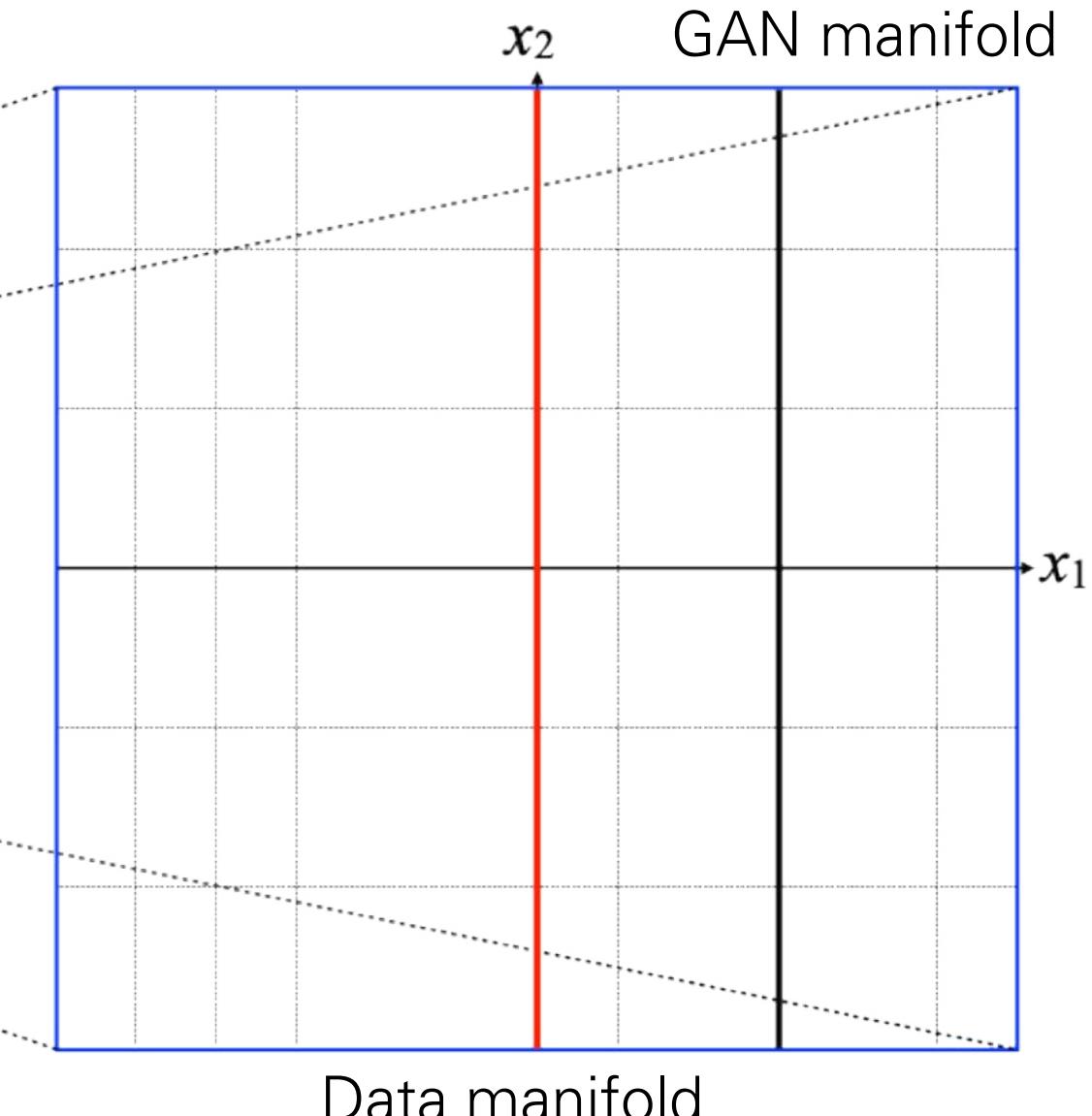
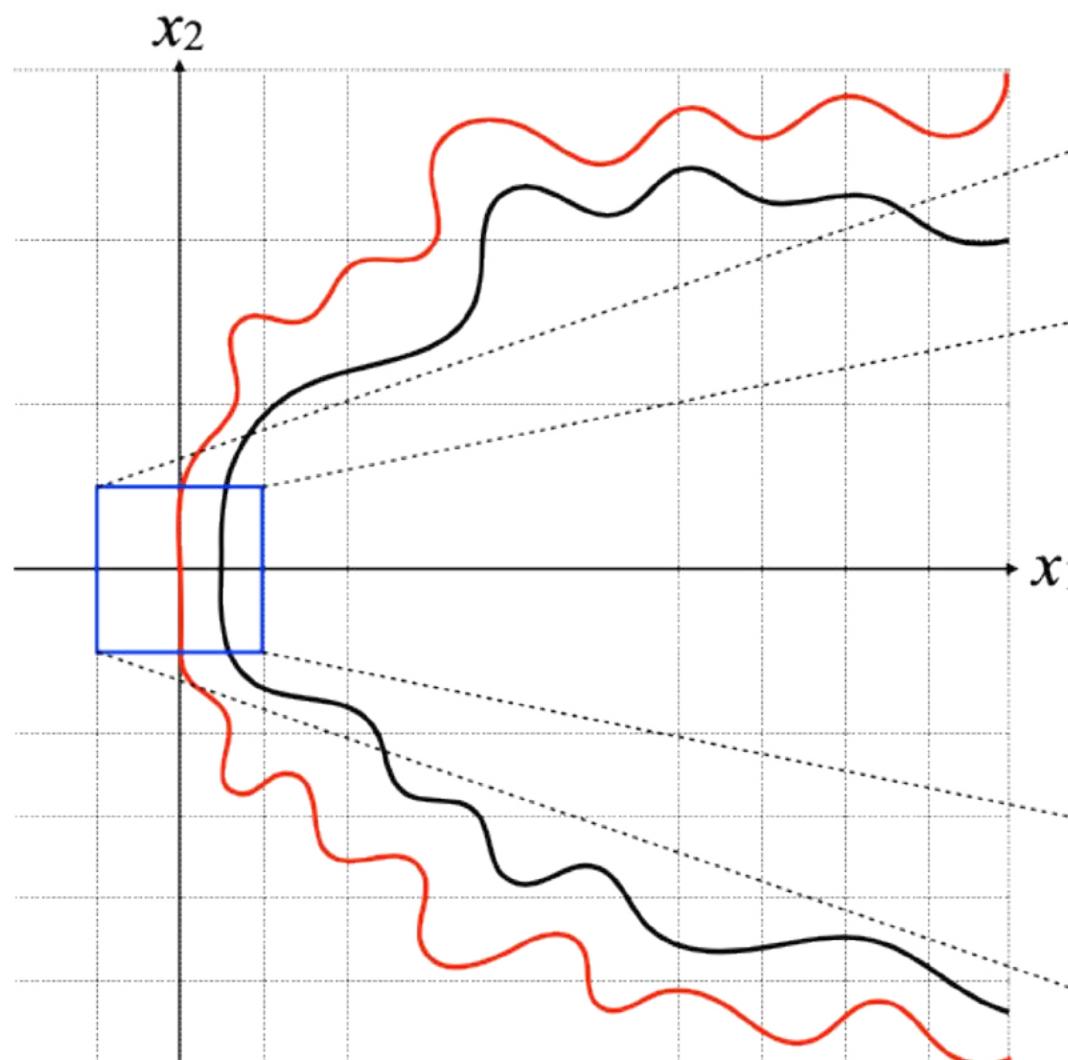
Lecture overview

- Motivation and Definition of Implicit Models
- Original GAN (Goodfellow et al, 2014)
- Evaluation: Parzen, Inception, Frechet
- Theory of GANs
- **GAN Progression**
 - DC GAN (Radford et al, 2016)
 - Improved Training of GANs (Salimans et al'16), Projected GAN (Sauer et al'21)
 - **WGAN**, WGAN-GP, Progressive GAN, SN-GAN, SAGAN, Projected GAN
 - BigGAN, BigGAN-Deep, StyleGAN, StyleGAN2, StyleGAN3, StyleGAN-XL, VIB-GAN
- Conditional GANs, Cycle-Consistent Adversarial Networks
- GANs and Representations
- Applications

Training a GAN: Distances between Manifolds



Training a GAN: Distances between Manifolds

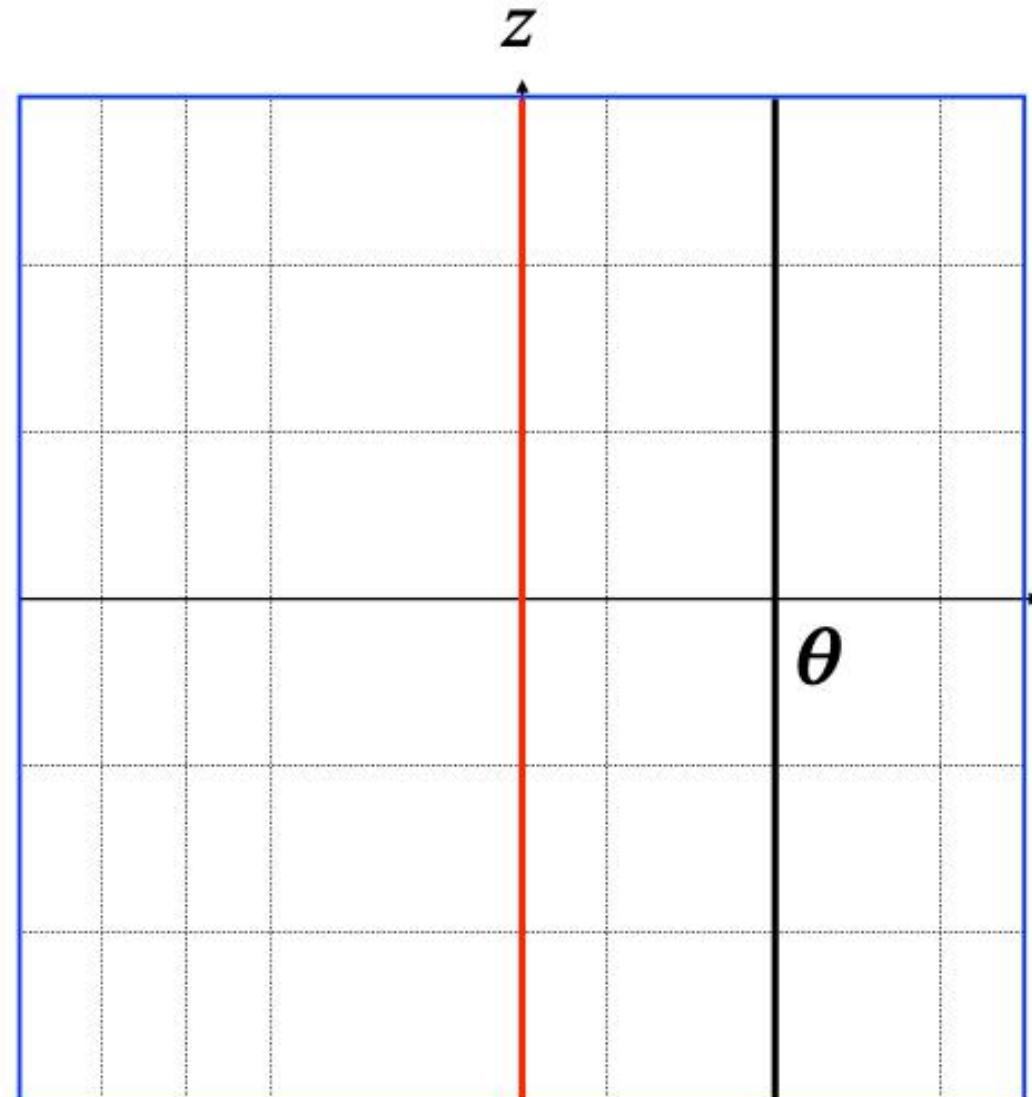


Jensen-Shannon Divergence

$$\text{JS}(\mathbb{P}_r \parallel \mathbb{P}_g) = \text{KL}\left(\mathbb{P}_r \parallel \frac{\mathbb{P}_r + \mathbb{P}_g}{2}\right) + \text{KL}\left(\mathbb{P}_g \parallel \frac{\mathbb{P}_r + \mathbb{P}_g}{2}\right)$$

- What is the JS divergence in this simple case?

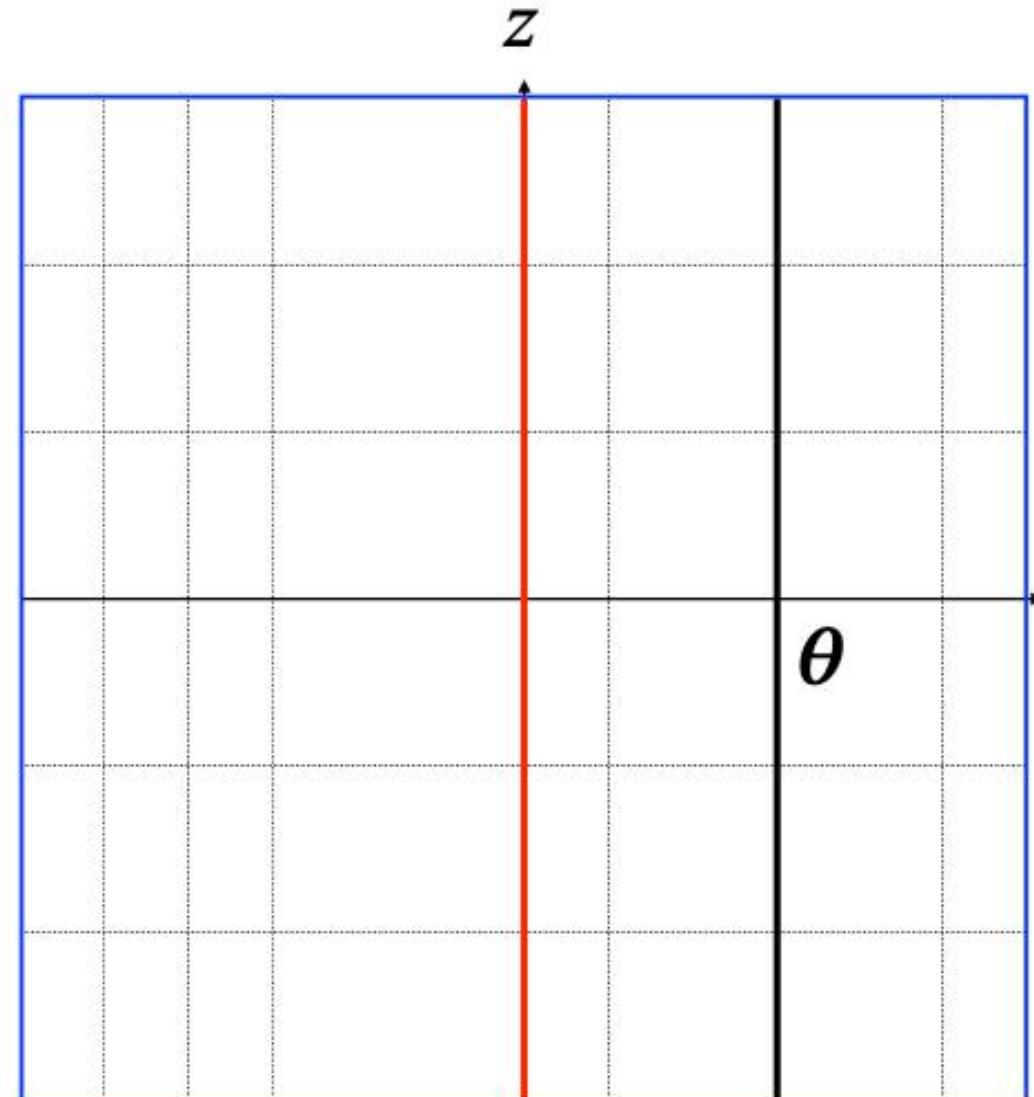
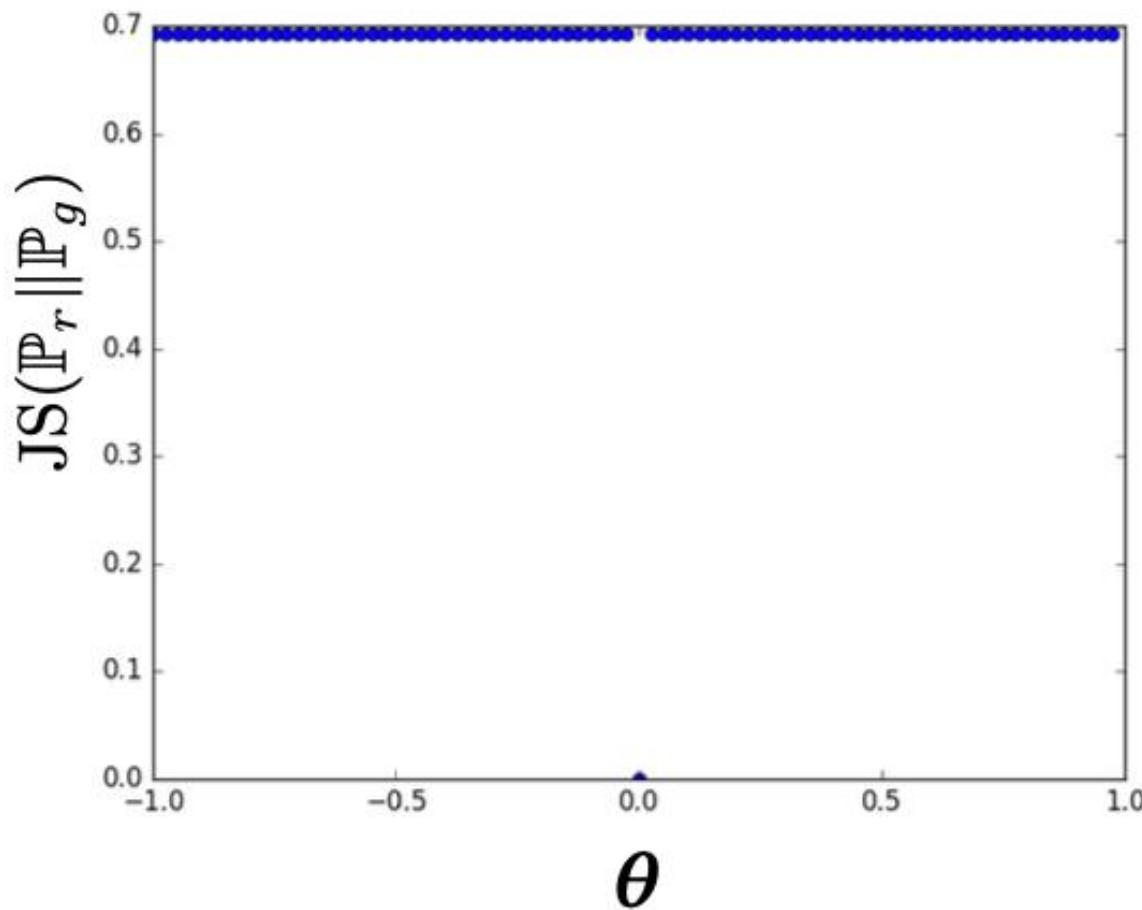
$$\text{JS}(\mathbb{P}_r \parallel \mathbb{P}_g) = \begin{cases} \log 2 & \text{if } \theta \neq 0 \\ 0 & \text{if } \theta = 0 \end{cases}$$



Example from (Arjovsky et al. 2017)

Jensen-Shannon Divergence

$$\text{JS}(\mathbb{P}_r \parallel \mathbb{P}_g) = \begin{cases} \log 2 & \text{if } \theta \neq 0 \\ 0 & \text{if } \theta = 0 \end{cases}$$



Example from (Arjovsky et al. 2017)

Wasserstein Distance

- JS divergence is not a useful learning signal to train GANs.
- Another distance measure inspired from Optimal Transport is the Earth Mover (EM) (also called Wasserstein-1 Distance) distance

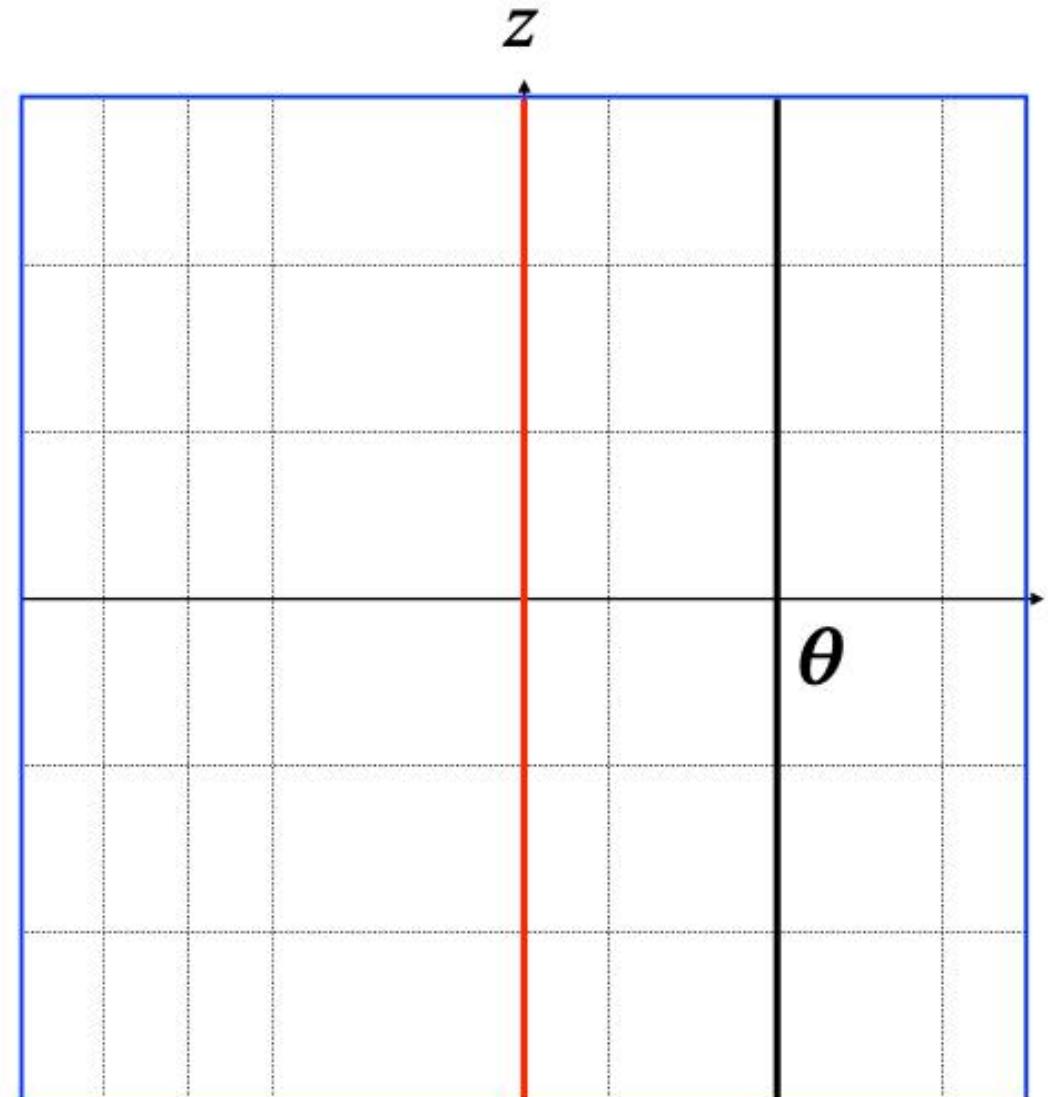
$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y) \sim \gamma} [\|x - y\|]$$

- The EM distance is continuous everywhere and differentiable almost everywhere (under mild assumptions).

Wasserstein Distance

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y) \sim \gamma} [\|x - y\|]$$

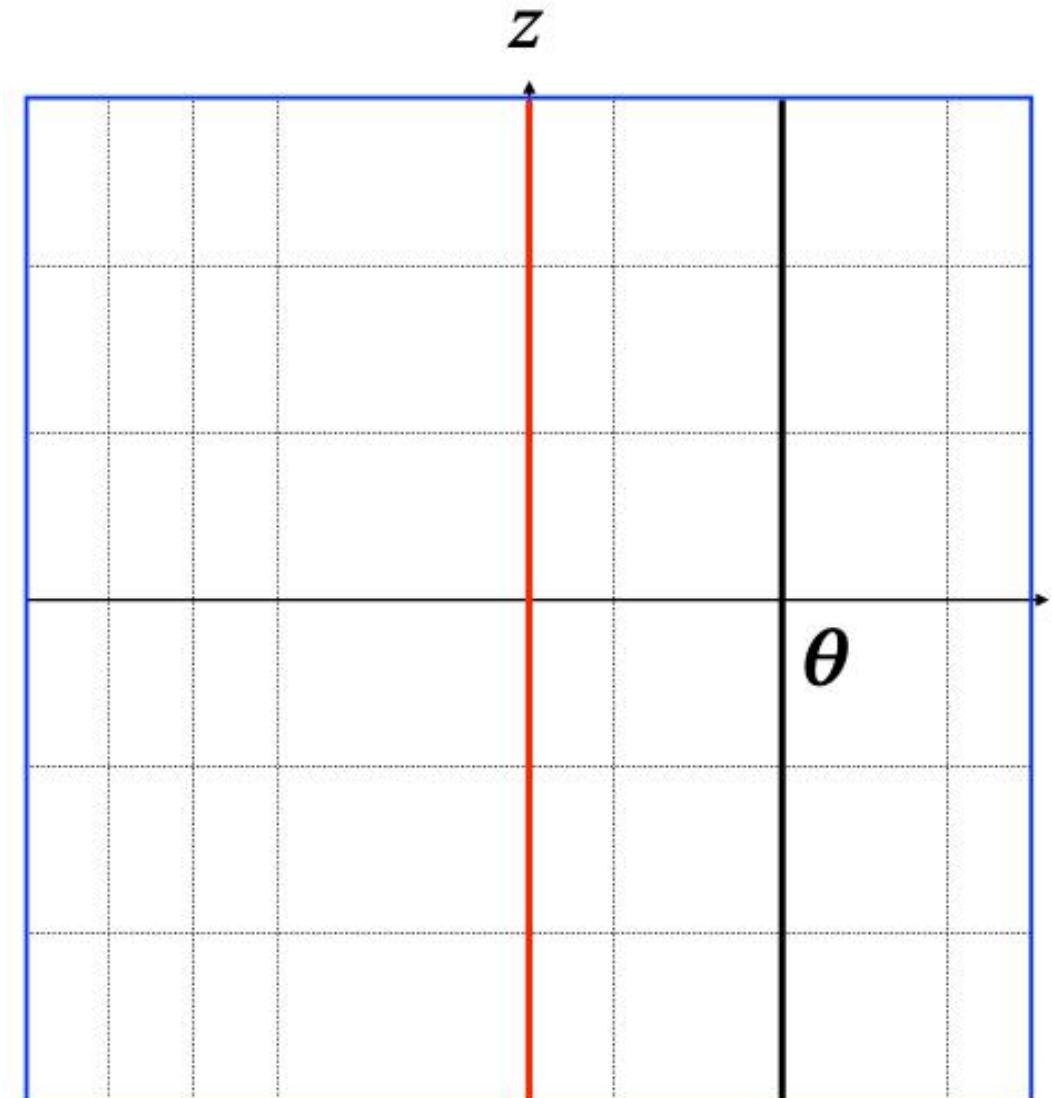
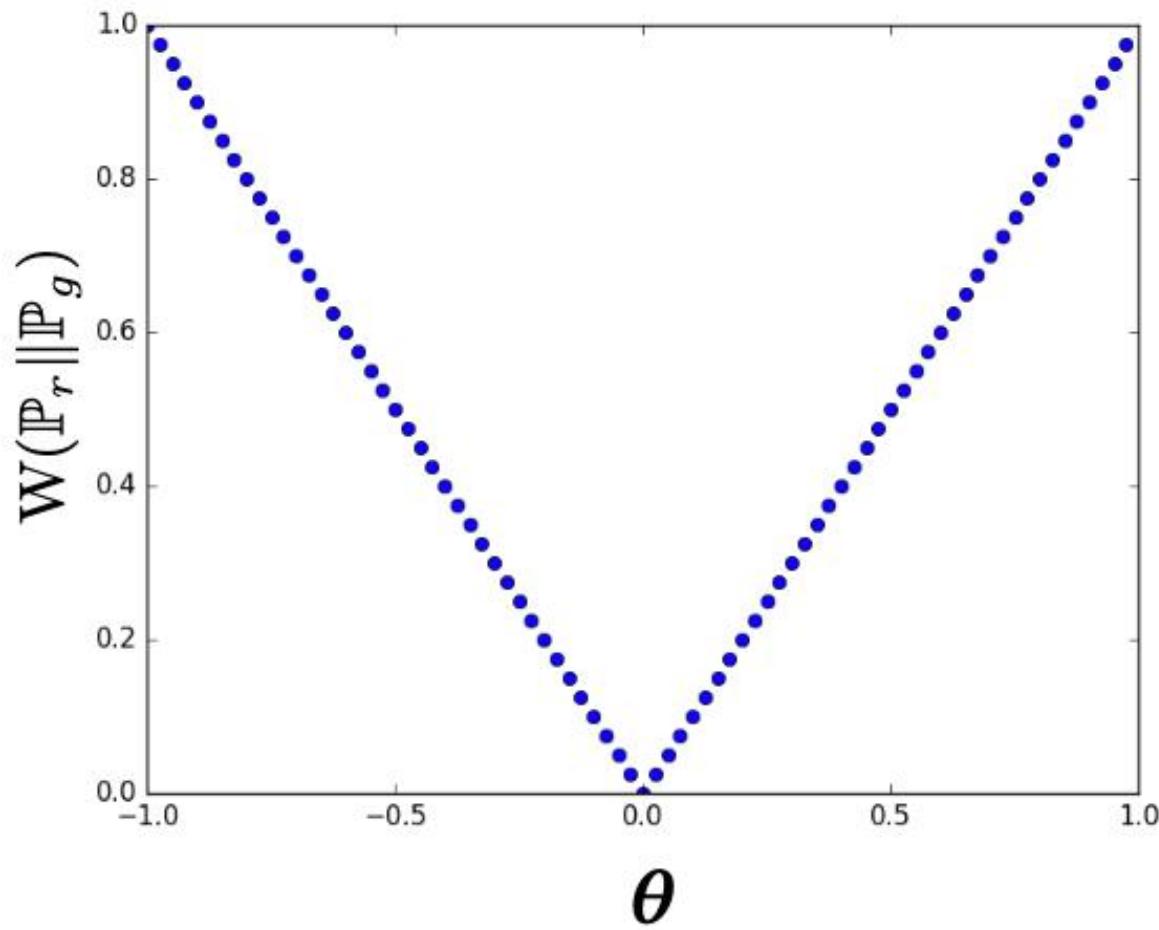
- What is the EM (or Wasserstein) distance in this simple case?



Example from (Arjovsky et al. 2017)

Wasserstein Distance

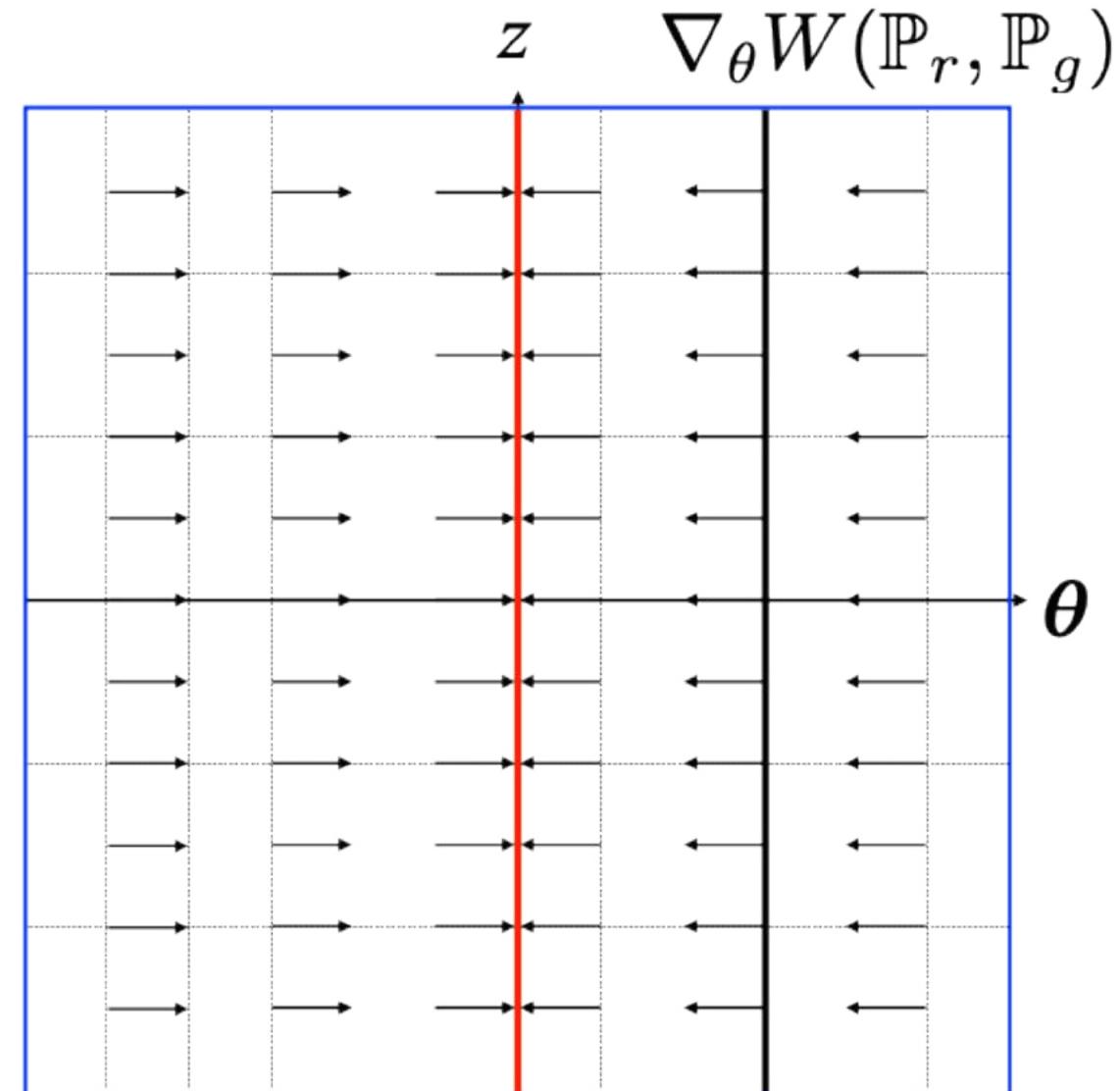
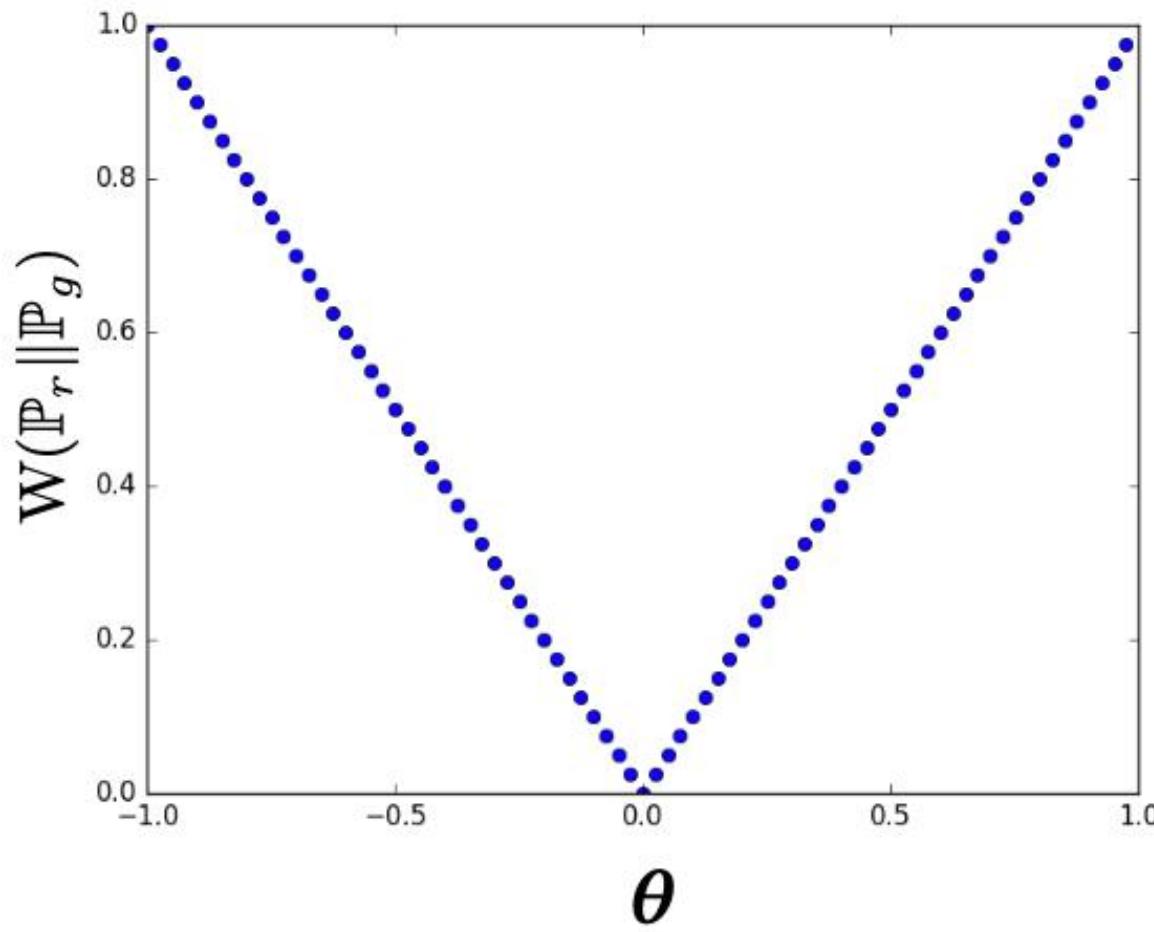
$$W(\mathbb{P}_r \parallel \mathbb{P}_g) = |\theta|$$



Example from (Arjovsky et al. 2017)

Wasserstein Distance

$$W(\mathbb{P}_r \parallel \mathbb{P}_g) = |\theta|$$



Example from (Arjovsky et al. 2017)

Wasserstein GAN

A real-valued function $f: X \rightarrow Y$ is called K -Lipschitz continuous if there exists a real constant $K \geq 0$ such that, for all x_1, x_2 .
 $|f(x_1) - f(x_2)| \leq K |x_1 - x_2|$
Here K is known as a Lipschitz constant for function $f(\cdot)$

- $W(\mathbb{P}_r \parallel \mathbb{P}_g)$ might have nice properties compared to $\text{JS}(\mathbb{P}_r \parallel \mathbb{P}_g)$
- However, the infimum is intractable in:

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y) \sim \gamma} [\|x - y\|]$$

- Can exploit Kantorovich-Rubinstein duality:

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

where the supremum is over all the 1-Lipschitz functions $f: \mathcal{X} \rightarrow \mathbb{R}$

Wasserstein GAN

- The WGAN Objective function:

$$\min_G \max_{D \in \mathcal{D}} \mathbb{E}_{x \sim \mathbb{P}_r} [D(x)] - \mathbb{E}_{\tilde{x} \sim \mathbb{P}_g} [D(\tilde{x})]$$

where \mathcal{D} is the set of 1-Lipschitz functions.

- Open question: how to effectively enforce the Lipschitz constraint on the critic D ?
 - Arjovsky et al. (2017) propose to clip the weights of the critic to lie within a compact space $[-c, c]$.
 - Results in a subset of the k -Lipschitz functions (k is a function of c).

Wasserstein GAN - Pseudocode

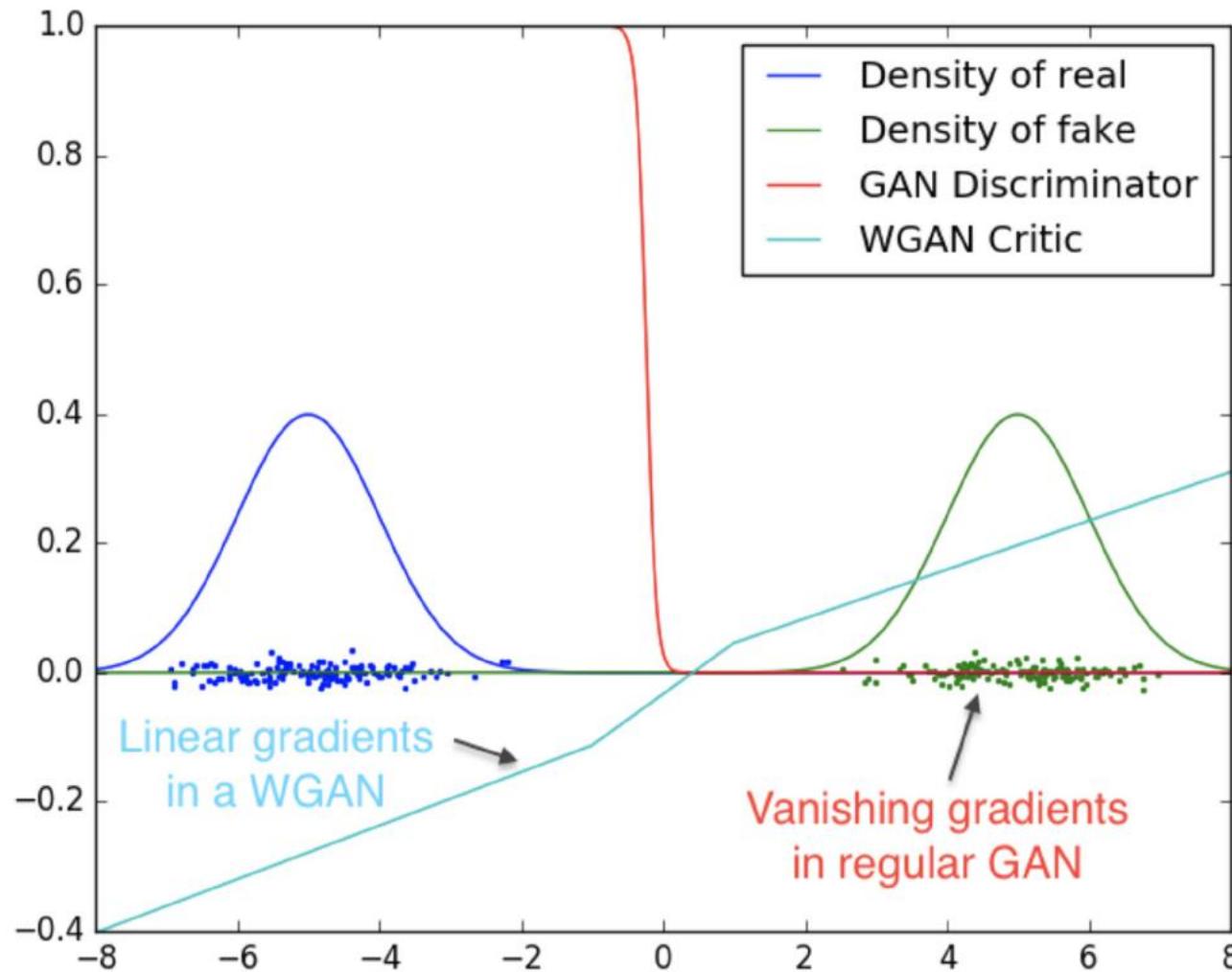
Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

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.

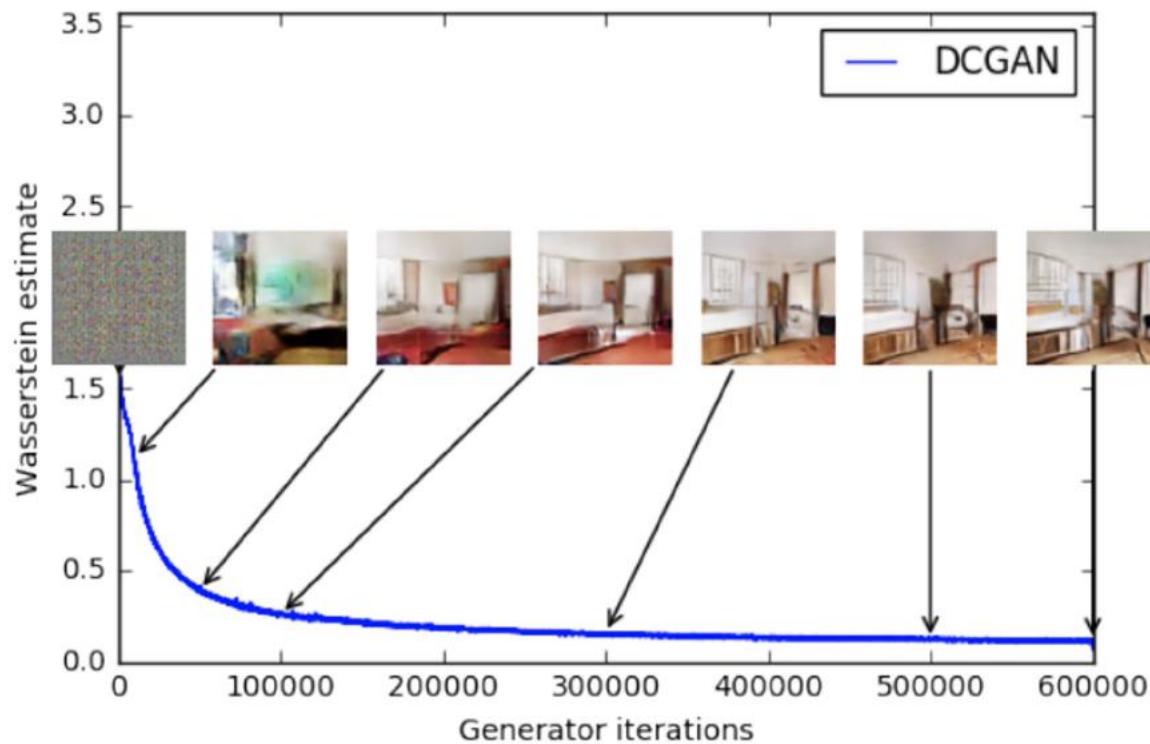
```
1: while  $\theta$  has not converged do
2:   for  $t = 0, \dots, n_{\text{critic}}$  do
3:     Sample  $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$  a batch from the real data.
4:     Sample  $\{z^{(i)}\}_{i=1}^m \sim p(z)$  a batch of prior samples.
5:      $g_w \leftarrow \nabla_w \left[ \frac{1}{m}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

Wasserstein GAN - Training critic to converge

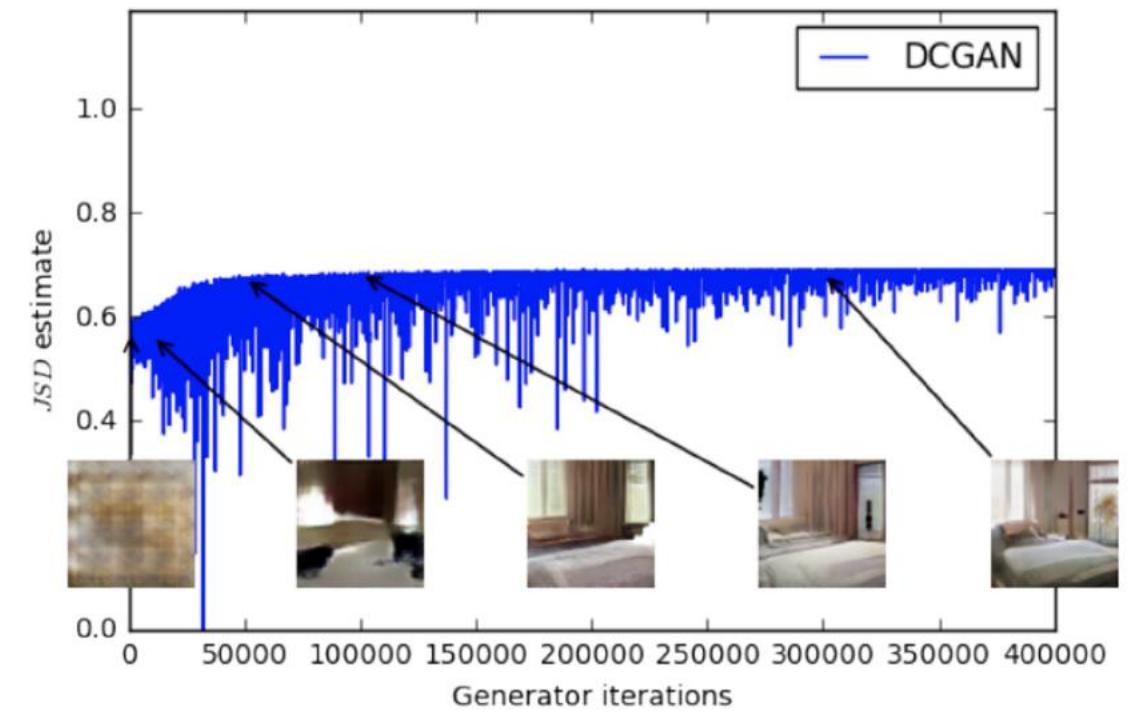


Wasserstein distance correlates with sample quality

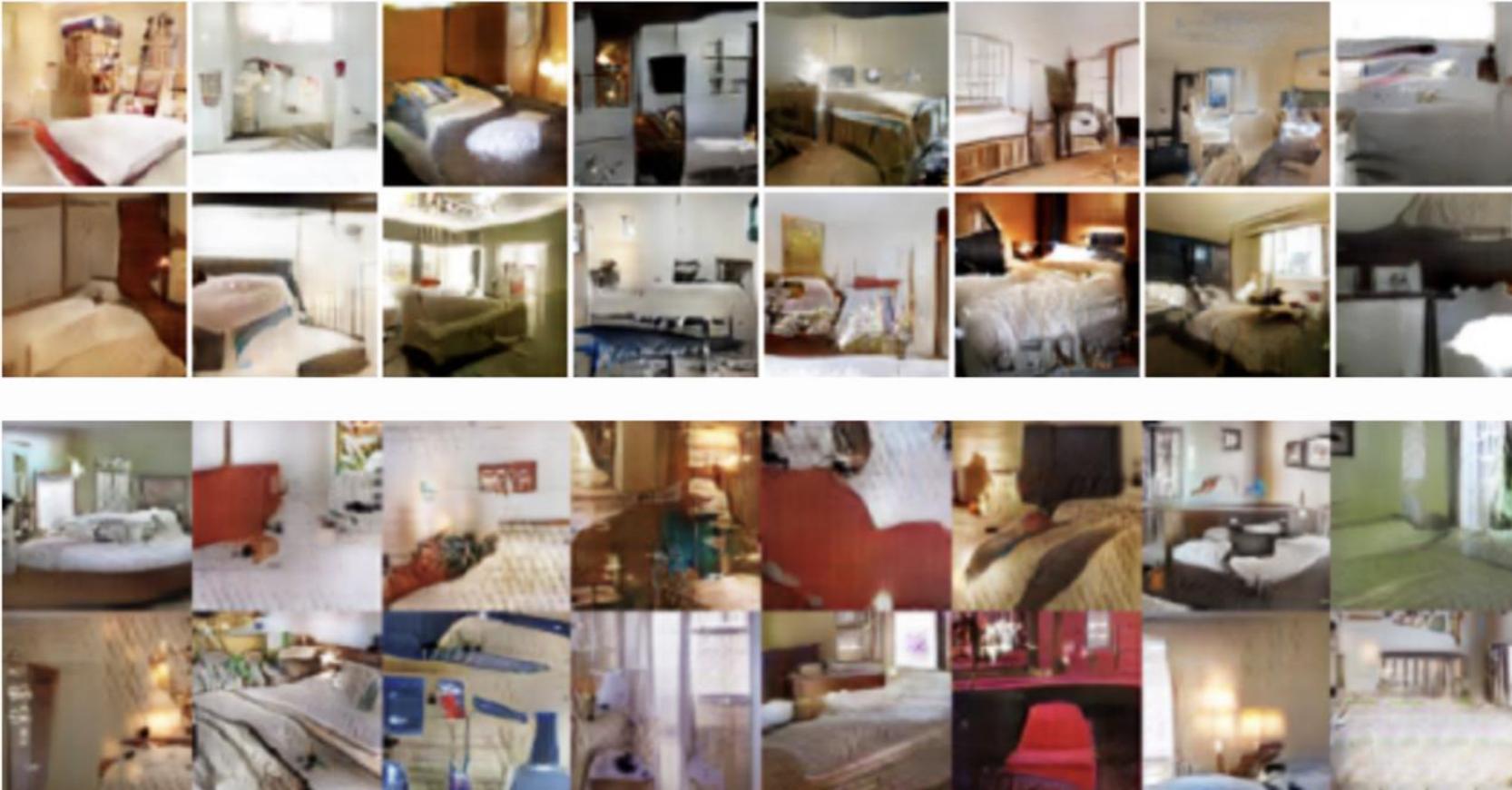
Wasserstein Estimate



JSD Estimate

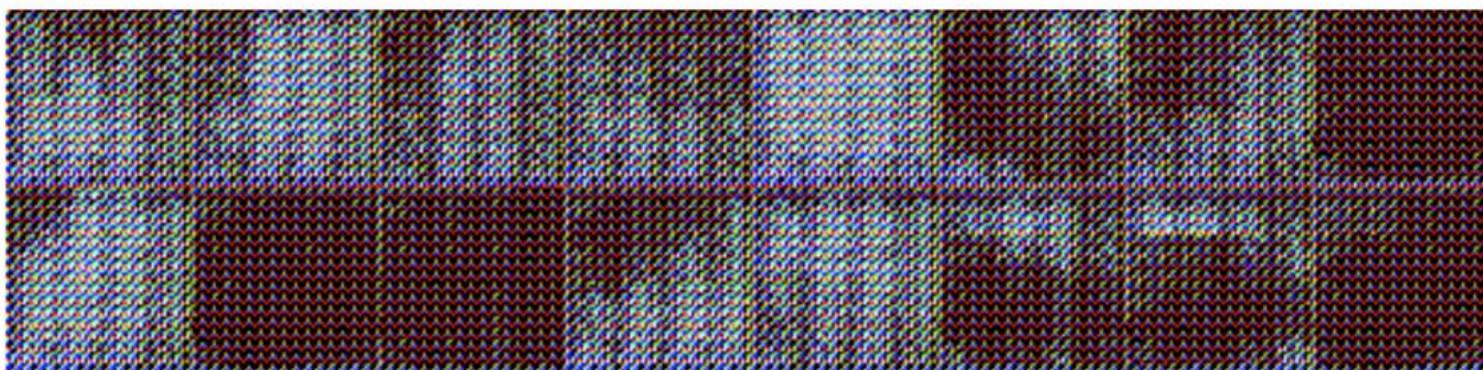


WGAN Samples on par with DCGAN



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

WGAN robust to architecture choices



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

WGAN robust to architecture choices



Top: WGAN with MLP architecture. Bottom: Standard GAN, same architecture

WGAN Summary

Standard GAN

$$\min_G \max_D \mathbb{E}_{x \sim P_r} [\log D(x)] + \mathbb{E}_{\tilde{x} \sim P_g} [\log(1 - D(\tilde{x}))]$$



Wasserstein GAN

$$\min_G \max_{D \in \mathcal{D}} \mathbb{E}_{x \sim P_r} [D(x)] - \mathbb{E}_{\tilde{x} \sim P_g} [D(\tilde{x})]$$

WGAN Summary

- New divergence measure for optimizing the generator
- Addresses instabilities with JSD version (sigmoid cross entropy)
- Robust to architectural choices
- Progress on mode collapse and stability of derivative wrt input
- Introduces the idea of using lipschitzness to stabilize GAN training
- **Negative:**

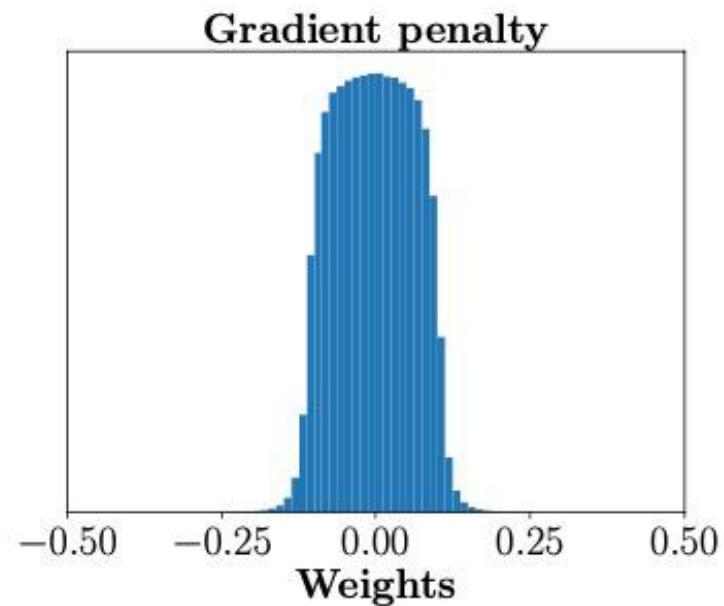
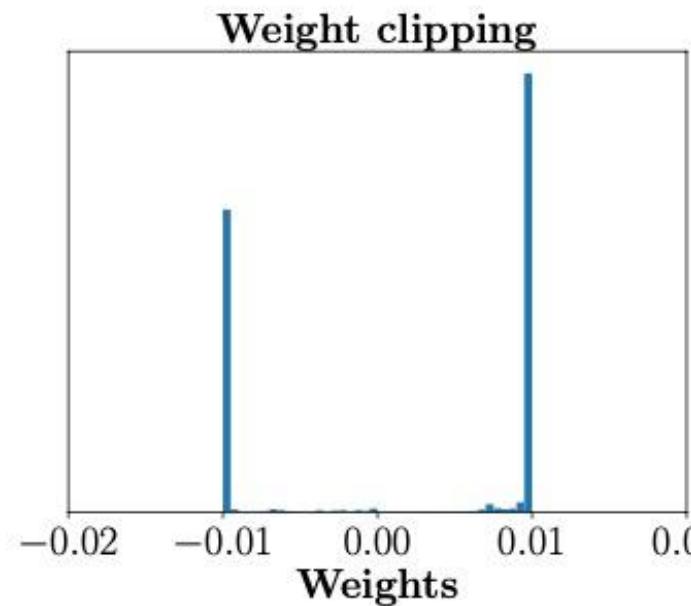
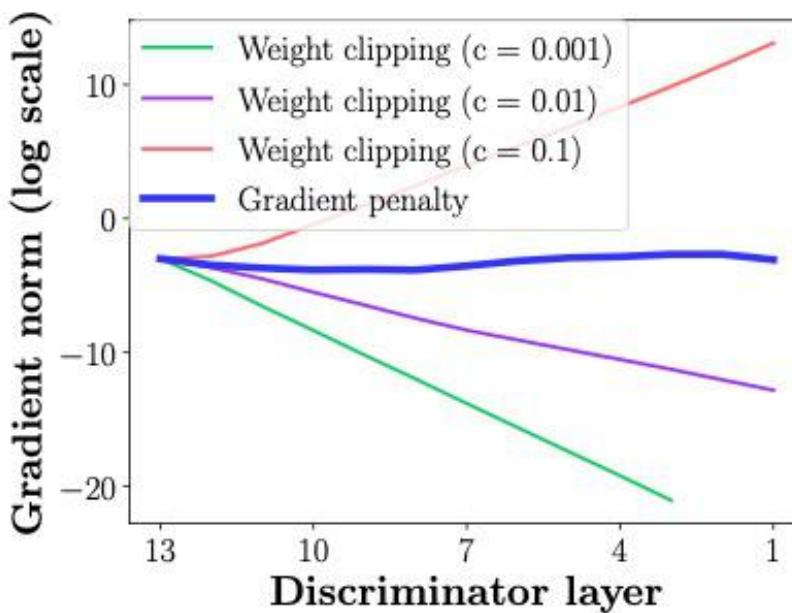
Weight clipping is a clearly terrible way to enforce a Lipschitz constraint. If the clipping parameter is large, then it can take a long time for any weights to reach their limit, thereby making it harder to train the critic till optimality. If the clipping is small, this can easily lead to vanishing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

Lecture overview

- Motivation and Definition of Implicit Models
- Original GAN (Goodfellow et al, 2014)
- Evaluation: Parzen, Inception, Frechet
- Theory of GANs
- **GAN Progression**
 - DC GAN (Radford et al, 2016)
 - Improved Training of GANs (Salimans et al'16), Projected GAN (Sauer et al'21)
WGAN, **WGAN-GP**, Progressive GAN, SN-GAN, SAGAN
 - BigGAN, BigGAN-Deep, StyleGAN, StyleGAN2, StyleGAN3, StyleGAN-XL, VIB-GAN
- Conditional GANs, Cycle-Consistent Adversarial Networks
- GANs and Representations
- Applications

Issues with Weight Clipping

1. Underuse capacity
2. Exploding and vanishing gradients



WGAN-GP: Gradient Penalty Approach

Improved Training of Wasserstein GANs

Ishaan Gulrajani^{1,*}, Faruk Ahmed¹, Martin Arjovsky², Vincent Dumoulin¹, Aaron Courville^{1,3}

¹ Montreal Institute for Learning Algorithms

² Courant Institute of Mathematical Sciences

³ CIFAR Fellow

igul222@gmail.com

{faruk.ahmed,vincent.dumoulin,aaron.courville}@umontreal.ca

ma4371@nyu.edu

Abstract

Generative Adversarial Networks (GANs) are powerful generative models, but suffer from training instability. The recently proposed Wasserstein GAN (WGAN) makes progress toward stable training of GANs, but sometimes can still generate only poor samples or fail to converge. We find that these problems are often due to the use of weight clipping in WGAN to enforce a Lipschitz constraint on the critic, which can lead to undesired behavior. We propose an alternative to clipping weights: penalize the norm of gradient of the critic with respect to its input. Our proposed method performs better than standard WGAN and enables stable training of a wide variety of GAN architectures with almost no hyperparameter tuning, including 101-layer ResNets and language models with continuous generators. We also achieve high quality generations on CIFAR-10 and LSUN bedrooms. [†]

WGAN-GP: Gradient Penalty Approach

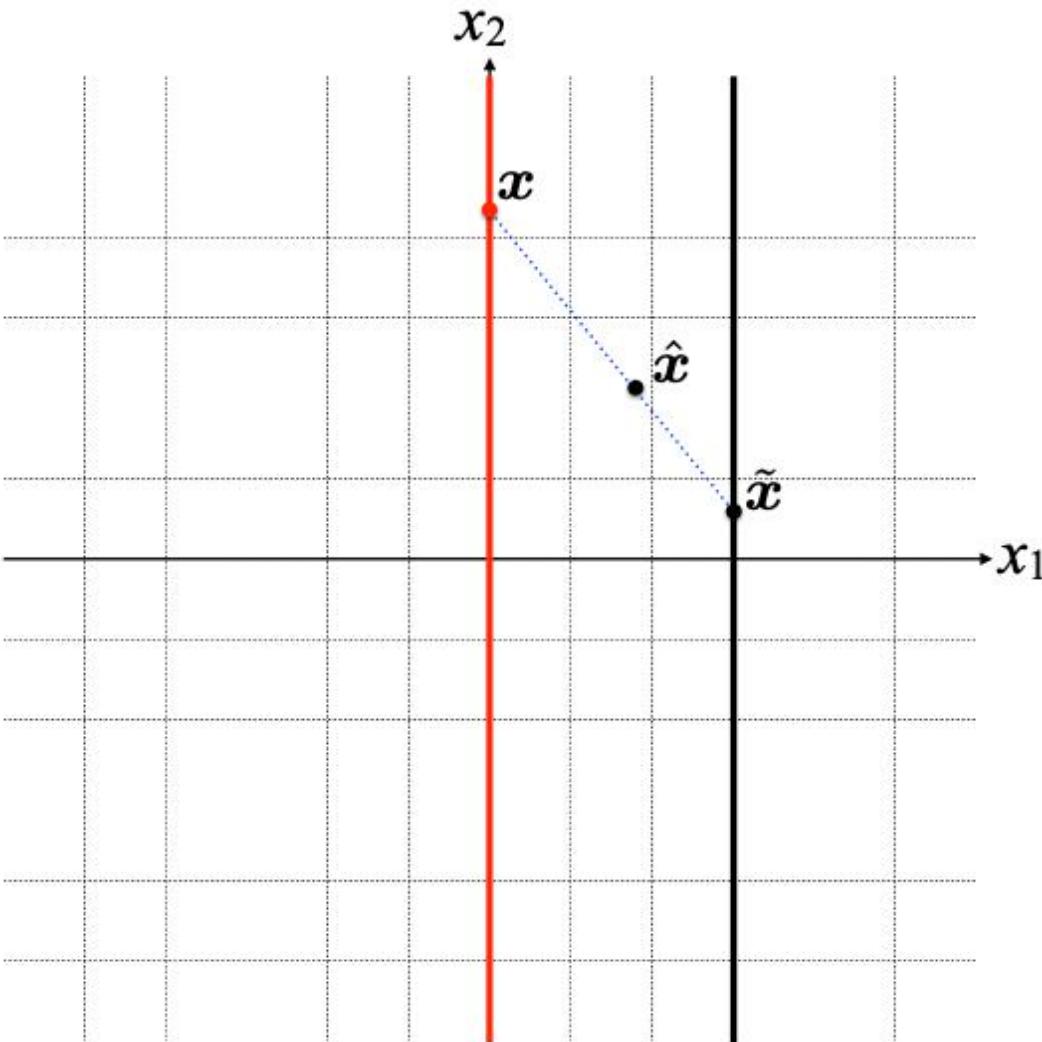
- A property of the optimal WGAN critic: If $\tilde{\mathbf{x}} \sim \mathbb{P}_g$ then there is a point $\mathbf{x} \sim \mathbb{P}_r$, such that for all points $\mathbf{x}_t = t\mathbf{x} + (1 - t)\tilde{\mathbf{x}}$ (on a straight line between \mathbf{x} and $\tilde{\mathbf{x}}$) then:

$$\nabla D^*(\mathbf{x}_t) = \frac{\mathbf{x} - \mathbf{x}_t}{\|\mathbf{x} - \mathbf{x}_t\|}$$

- This implies the optimal WGAN critic has gradient norm 1 at \mathbf{x}_t
- Gradient Penalty version of WGAN (i.e. WGAN-GP) objective

$$L = \underbrace{\mathbb{E}_{\tilde{\mathbf{x}} \sim \mathbb{P}_g} [D(\tilde{\mathbf{x}})] - \mathbb{E}_{\mathbf{x} \sim \mathbb{P}_r} [D(\mathbf{x})]}_{\text{Original critic loss}} + \lambda \underbrace{\mathbb{E}_{\hat{\mathbf{x}} \sim \mathbb{P}_{\hat{\mathbf{x}}}} \left[(\|\nabla_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}})\|_2 - 1)^2 \right]}_{\text{Our gradient penalty}}$$

WGAN-GP: Gradient Penalty Approach



- Gradient penalty:

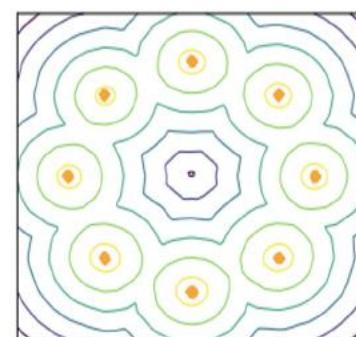
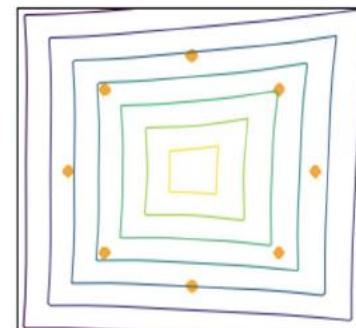
$$\mathbb{E}_{\hat{x} \sim \mathbb{P}_{\hat{x}}} \left[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2 \right]$$

Sample along straight lines:

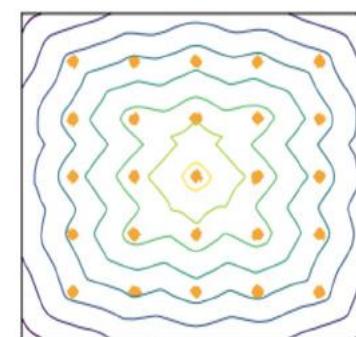
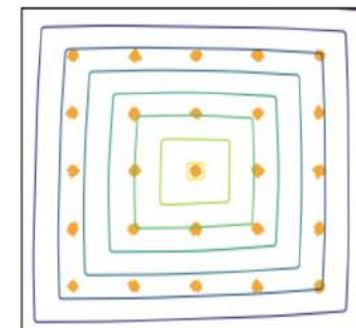
$$\begin{aligned}\epsilon &\sim U[0, 1], \mathbf{x} \sim \mathbb{P}_r, \tilde{\mathbf{x}} \sim \mathbb{P}_g \\ \hat{\mathbf{x}} &= \epsilon \mathbf{x} + (1 - \epsilon) \tilde{\mathbf{x}}\end{aligned}$$

WGAN-GP: Gradient Penalty for Lipschitzness

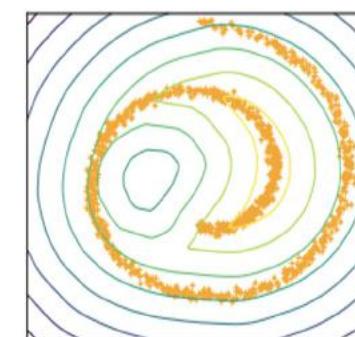
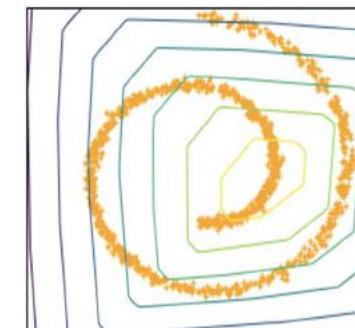
8 Gaussian



25 Gaussian



Swiss Roll



$$\max_D \underbrace{\mathbb{E}_{x \sim P_r} [D(x)] - \mathbb{E}_{\tilde{x} \sim P_g} [D(\tilde{x})]}_{\text{Wasserstein critic objective}} + \lambda \underbrace{\mathbb{E}_{\hat{x} \sim P_{\hat{x}}} \left[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2 \right]}_{\text{Gradient Penalty for Lipschitzness}}$$

$$\hat{x} \leftarrow \epsilon x + (1 - \epsilon) \tilde{x}$$

[Gulrajani et al 2017]

WGAN-GP: Pseudocode

Algorithm 1 WGAN with gradient penalty. We use default values of $\lambda = 10$, $n_{\text{critic}} = 5$, $\alpha = 0.0001$, $\beta_1 = 0$, $\beta_2 = 0.9$.

Require: The gradient penalty coefficient λ , the number of critic iterations per generator iteration n_{critic} , the batch size m , Adam hyperparameters α, β_1, β_2 .

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

```
1: while  $\theta$  has not converged do
2:   for  $t = 1, \dots, n_{\text{critic}}$  do
3:     for  $i = 1, \dots, m$  do
4:       Sample real data  $\mathbf{x} \sim \mathbb{P}_r$ , latent variable  $\mathbf{z} \sim p(\mathbf{z})$ , a random number  $\epsilon \sim U[0, 1]$ .
5:        $\tilde{\mathbf{x}} \leftarrow G_\theta(\mathbf{z})$ 
6:        $\hat{\mathbf{x}} \leftarrow \epsilon \mathbf{x} + (1 - \epsilon) \tilde{\mathbf{x}}$ 
7:        $L^{(i)} \leftarrow D_w(\tilde{\mathbf{x}}) - D_w(\mathbf{x}) + \lambda(\|\nabla_{\hat{\mathbf{x}}} D_w(\hat{\mathbf{x}})\|_2 - 1)^2$ 
8:     end for
9:      $w \leftarrow \text{Adam}(\nabla_w \frac{1}{m} \sum_{i=1}^m L^{(i)}, w, \alpha, \beta_1, \beta_2)$ 
10:   end for
11:   Sample a batch of latent variables  $\{\mathbf{z}^{(i)}\}_{i=1}^m \sim p(\mathbf{z})$ .
12:    $\theta \leftarrow \text{Adam}(\nabla_\theta \frac{1}{m} \sum_{i=1}^m -D_w(G_\theta(\mathbf{z})), \theta, \alpha, \beta_1, \beta_2)$ 
13: end while
```

WGAN-GP: BatchNorm

No critic batch normalization Most prior GAN implementations [22, 23, 2] use batch normalization in both the generator and the discriminator to help stabilize training, but batch normalization changes the form of the discriminator’s problem from mapping a single input to a single output to mapping from an entire batch of inputs to a batch of outputs [23]. Our penalized training objective is no longer valid in this setting, since we penalize the norm of the critic’s gradient with respect to each input independently, and not the entire batch. To resolve this, we simply omit batch normalization in the critic in our models, finding that they perform well without it. Our method works with normalization schemes which don’t introduce correlations between examples. In particular, we recommend layer normalization [3] as a drop-in replacement for batch normalization.

WGAN-GP: Robustness to architectures

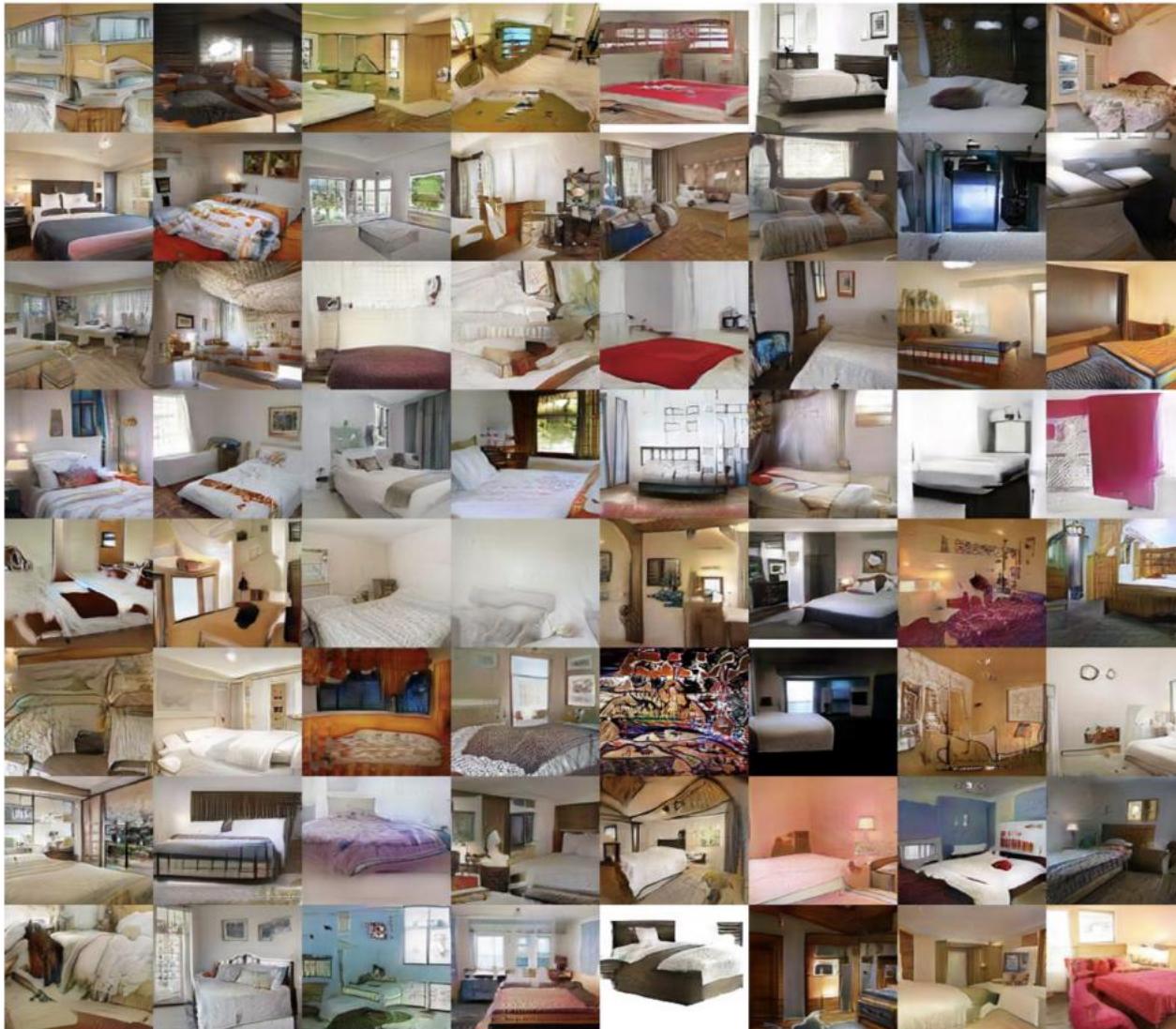
| | |
|--|---|
| Nonlinearity (G) | [ReLU, LeakyReLU, $\frac{\text{softplus}(2x+2)}{2} - 1$, tanh] |
| Nonlinearity (D) | [ReLU, LeakyReLU, $\frac{\text{softplus}(2x+2)}{2} - 1$, tanh] |
| Depth (G) | [4, 8, 12, 20] |
| Depth (D) | [4, 8, 12, 20] |
| Batch norm (G) | [True, False] |
| Batch norm (D ; layer norm for WGAN-GP) | [True, False] |
| Base filter count (G) | [32, 64, 128] |
| Base filter count (D) | [32, 64, 128] |

| Min. score | Only GAN | Only WGAN-GP | Both succeeded | Both failed |
|------------|----------|--------------|----------------|-------------|
| 1.0 | 0 | 8 | 192 | 0 |
| 3.0 | 1 | 88 | 110 | 1 |
| 5.0 | 0 | 147 | 42 | 11 |
| 7.0 | 1 | 104 | 5 | 90 |
| 9.0 | 0 | 0 | 0 | 200 |

WGAN-GP: Robustness to architectures

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

WGAN-GP: High quality samples



[Gulrajani et al 2017]

WGAN-GP: High quality samples

Table 3: Inception scores on CIFAR-10. Our unsupervised model achieves state-of-the-art performance, and our conditional model outperforms all others except SGAN.

| Unsupervised | | Supervised | |
|------------------------------|----------------|------------------------------|----------------|
| Method | Score | Method | Score |
| ALI [8] (in [27]) | $5.34 \pm .05$ | SteinGAN [26] | 6.35 |
| BEGAN [4] | 5.62 | DCGAN (with labels, in [26]) | 6.58 |
| DCGAN [22] (in [11]) | $6.16 \pm .07$ | Improved GAN [23] | $8.09 \pm .07$ |
| Improved GAN (-L+HA) [23] | $6.86 \pm .06$ | AC-GAN [20] | $8.25 \pm .07$ |
| EGAN-Ent-VI [7] | $7.07 \pm .10$ | SGAN-no-joint [11] | $8.37 \pm .08$ |
| DFM [27] | $7.72 \pm .13$ | WGAN-GP ResNet (ours) | $8.42 \pm .10$ |
| WGAN-GP ResNet (ours) | $7.86 \pm .07$ | SGAN [11] | $8.59 \pm .12$ |

WGAN-GP: Summary

- Robustness to architectural choices
- Became a very popular GAN model - 2000+ citations, has been used in NVIDIA's Progressive GANs, StyleGAN, etc - biggest GAN successes
- Residual architecture widely adopted.
- Possible negative- slow wall clock time due to gradient penalty.
- Gradient penalty applied on a heuristic distribution of samples from current generator. Could be unstable when learning rates are high.

Lecture overview

- Motivation and Definition of Implicit Models
- Original GAN (Goodfellow et al, 2014)
- Evaluation: Parzen, Inception, Frechet
- Theory of GANs
- **GAN Progression**
 - DC GAN (Radford et al, 2016)
 - Improved Training of GANs (Salimans et al'16), Projected GAN (Sauer et al'21)
WGAN, WGAN-GP, **Progressive GAN**, SN-GAN, SAGAN
 - BigGAN, BigGAN-Deep, StyleGAN, StyleGAN2, StyleGAN3, StyleGAN-XL, VIB-GAN
- Conditional GANs, Cycle-Consistent Adversarial Networks
- GANs and Representations
- Applications

Progressive growing of GANs

PROGRESSIVE GROWING OF GANs FOR IMPROVED QUALITY, STABILITY, AND VARIATION

Tero Karras

NVIDIA

Timo Aila

NVIDIA

Samuli Laine

NVIDIA

Jaakko Lehtinen

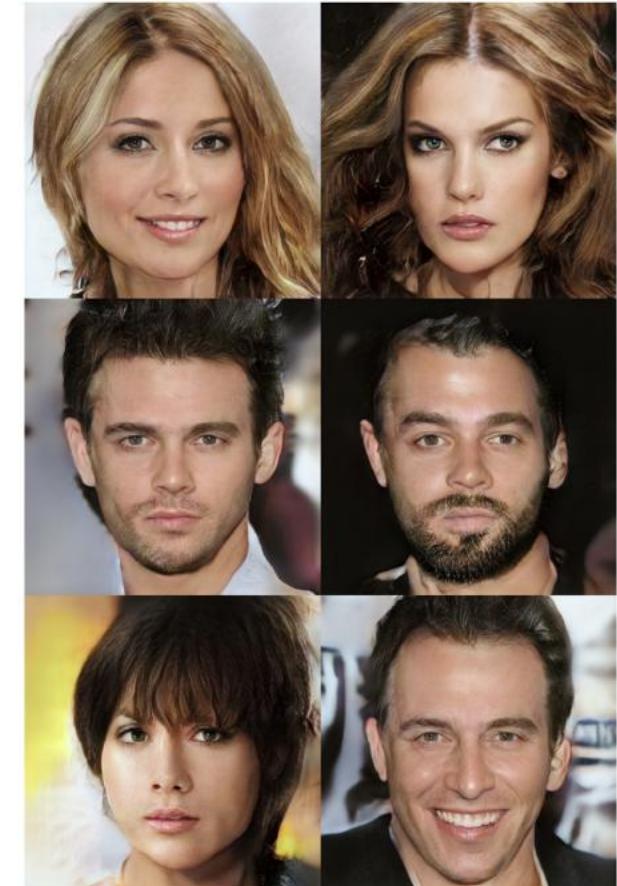
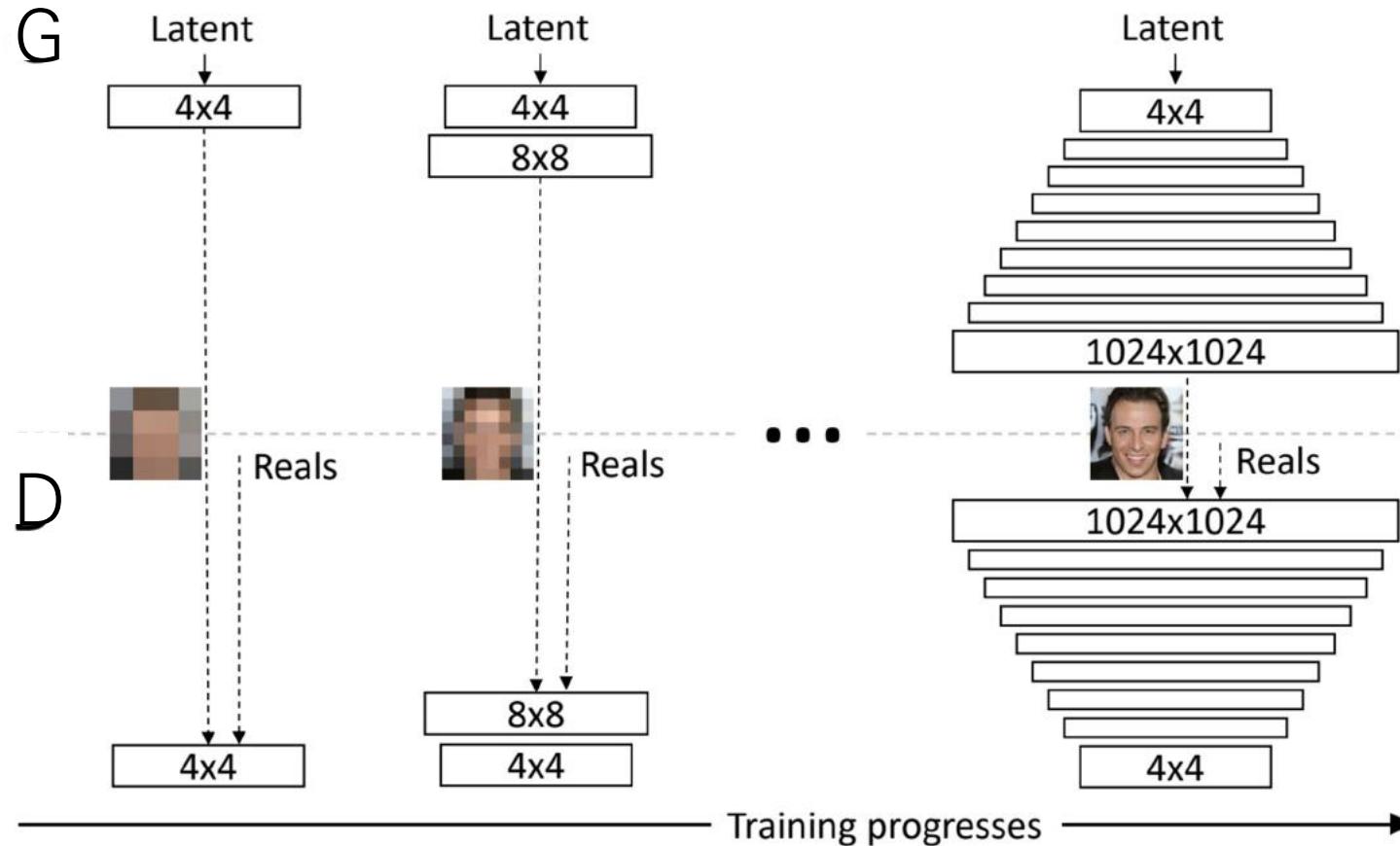
NVIDIA and Aalto University

{tkarras, taila, slaine, jlehtinen}@nvidia.com

ABSTRACT

We describe a new training methodology for generative adversarial networks. The key idea is to grow both the generator and discriminator progressively: starting from a low resolution, we add new layers that model increasingly fine details as training progresses. This both speeds the training up and greatly stabilizes it, allowing us to produce images of unprecedented quality, e.g., CELEBA images at 1024^2 . We also propose a simple way to increase the variation in generated images, and achieve a record inception score of 8.80 in unsupervised CIFAR10. Additionally, we describe several implementation details that are important for discouraging unhealthy competition between the generator and discriminator. Finally, we suggest a new metric for evaluating GAN results, both in terms of image quality and variation. As an additional contribution, we construct a higher-quality version of the CELEBA dataset.

Progressive growing of GANs



Progressive growing of GANs



Progressive growing of GANs



Mao et al. (2016b) (128×128)

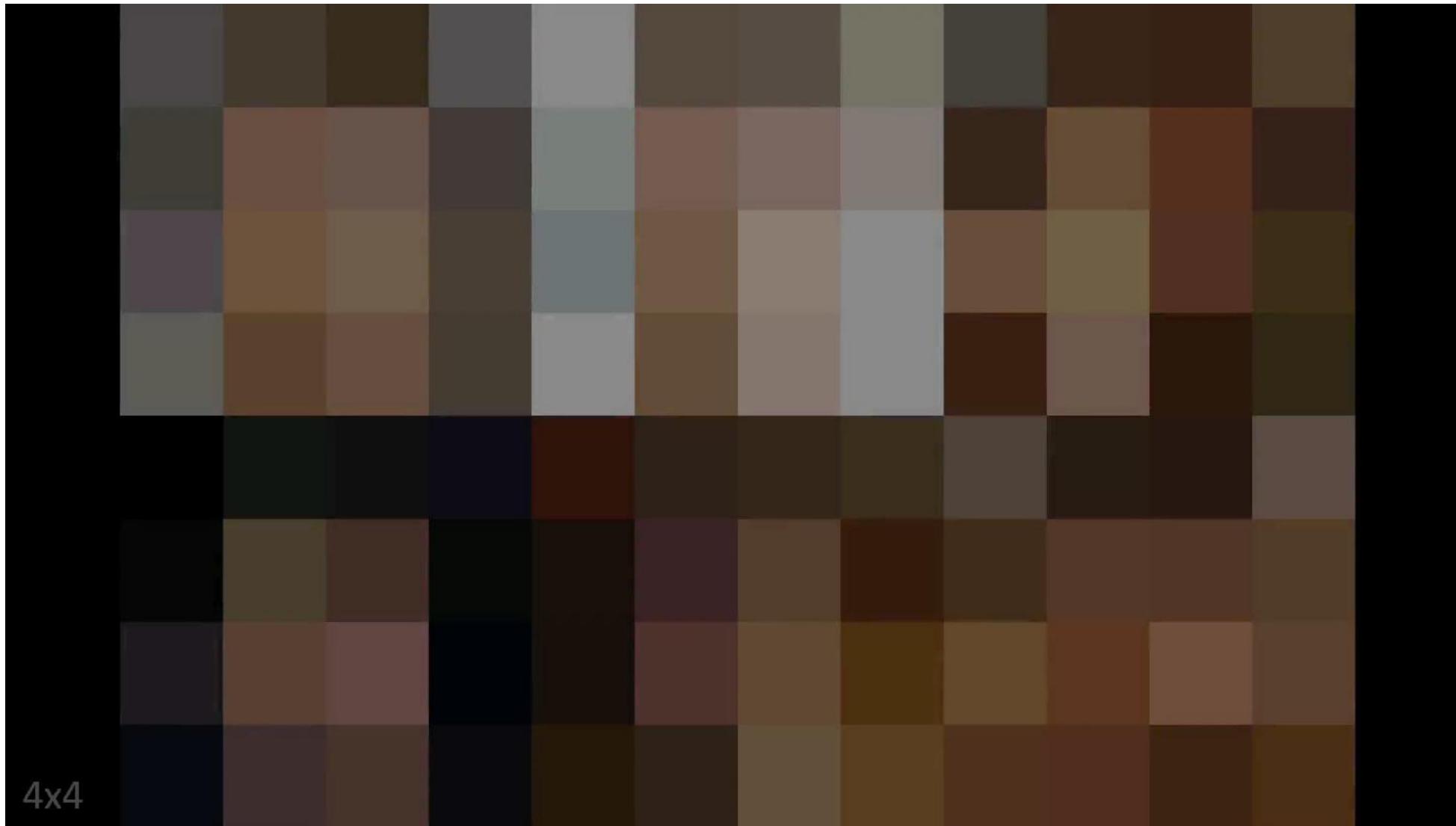
Gulrajani et al. (2017) (128×128)

Our (256×256)

Progressive growing of GANs



Progressive growing of GANs



4x4

Progressive growing of GANs



CelebA-HQ
random interpolations

[Karras et al. 2017]

Lecture overview

- Motivation and Definition of Implicit Models
- Original GAN (Goodfellow et al, 2014)
- Evaluation: Parzen, Inception, Frechet
- Theory of GANs
- **GAN Progression**
 - DC GAN (Radford et al, 2016)
 - Improved Training of GANs (Salimans et al'16), Projected GAN (Sauer et al'21)
WGAN, WGAN-GP, Progressive GAN, **SN-GAN**, SAGAN
 - BigGAN, BigGAN-Deep, StyleGAN, StyleGAN2, StyleGAN3, StyleGAN-XL, VIB-GAN
- Conditional GANs, Cycle-Consistent Adversarial Networks
- GANs and Representations
- Applications

Spectral Normalization GAN (SNGAN)

SPECTRAL NORMALIZATION FOR GENERATIVE ADVERSARIAL NETWORKS

Takeru Miyato¹, Toshiki Kataoka¹, Masanori Koyama², Yuichi Yoshida³

{miyato, kataoka}@preferred.jp

koyama.masanori@gmail.com

yyoshida@nii.ac.jp

¹Preferred Networks, Inc. ²Ritsumeikan University ³National Institute of Informatics

ABSTRACT

One of the challenges in the study of generative adversarial networks is the instability of its training. In this paper, we propose a novel weight normalization technique called spectral normalization to stabilize the training of the discriminator. Our new normalization technique is computationally light and easy to incorporate into existing implementations. We tested the efficacy of spectral normalization on CIFAR10, STL-10, and ILSVRC2012 dataset, and we experimentally confirmed that spectrally normalized GANs (SN-GANs) is capable of generating images of better or equal quality relative to the previous training stabilization techniques. The code with Chainer (Tokui et al., 2015), generated images and pretrained models are available at https://github.com/pfnet-research/sngan_projection.

[Miyato et al. 2017]

Spectral Normalization GAN (SNGAN)

(original) GAN formulation: $\min_G \max_D V(G, D)$

where $V(G, D) = \mathbb{E}_{\mathbf{x} \sim q_{\text{data}}} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{x}' \sim p_G} [\log(1 - D(\mathbf{x}'))]$

WGAN formulation: $\min_G \left[\arg \max_{\|f\|_{\text{Lip}} \leq K} V(G, D) \right]$

where $\|f\|_{\text{Lip}} \leq K \Rightarrow \|f(\mathbf{x}) - f(\mathbf{x}')\| / \|\mathbf{x} - \mathbf{x}'\| \leq K$

- Idea: Use spectral normalization to enforce the Lipschitz constraint

Spectral Normalization GAN (SNGAN)

- **Spectral Normalization strategy:** enforce the Lipschitz constraint by constraining the spectral norm of each layer of the neural network.

spectral norm of the matrix A : $\sigma(A) := \max_{\mathbf{h}: \mathbf{h} \neq \mathbf{0}} \frac{\|A\mathbf{h}\|_2}{\|\mathbf{h}\|_2} = \max_{\|\mathbf{h}\|_2 \leq 1} \|A\mathbf{h}\|_2$

- Let g be a layer of a network: $g : \mathbf{h}_{in} \mapsto \mathbf{h}_{out}$
for a linear layer $g(\mathbf{h}) = W\mathbf{h}$: $\|g\|_{\text{Lip}} = \sup_{\mathbf{h}} \sigma(\nabla g(\mathbf{h})) = \sup_{\mathbf{h}} \sigma(W) = \sigma(W)$
- For the network f , we assume the Lipschitz norm of the activation function (a) equals 1 (typically ok) and use the inequality:

$$\|g_1 \circ g_2\|_{\text{Lip}} \leq \|g_1\|_{\text{Lip}} \cdot \|g_2\|_{\text{Lip}}$$

Spectral Normalization GAN (SNGAN)

- The Lipschitz norm for the network is:

$$\|f\|_{\text{Lip}} \leq \|(\mathbf{h}_L \mapsto W^{L+1}\mathbf{h}_L)\|_{\text{Lip}} \cdot \|a_L\|_{\text{Lip}} \cdot \|(\mathbf{h}_{L-1} \mapsto W^L\mathbf{h}_{L-1})\|_{\text{Lip}}$$

activation function for layer L

$$\cdots \|a_1\|_{\text{Lip}} \cdot \|(\mathbf{h}_0 \mapsto W^1\mathbf{h}_0)\|_{\text{Lip}} = \prod_{l=1}^{L+1} \|(\mathbf{h}_{l-1} \mapsto W^l\mathbf{h}_{l-1})\|_{\text{Lip}} = \prod_{l=1}^{L+1} \sigma(W^l)$$

- Spectral Normalize the weights at each layer: $\bar{W}_{\text{SN}}(W) := W/\sigma(W)$

where $\sigma(W)$ is efficiently approximated using the power method.

(as described on the next slide)

Spectral Normalization GAN (SNGAN)

Algorithm 1 SGD with spectral normalization

- Initialize $\tilde{\mathbf{u}}_l \in \mathcal{R}^{d_l}$ for $l = 1, \dots, L$ with a random vector (sampled from isotropic distribution).
- For each update and each layer l : (warm start $\tilde{\mathbf{u}}_l$ and $\tilde{\mathbf{v}}_l$ from previous iteration)
 1. Apply power iteration method to a unnormalized weight W^l : (single iteration seems to work)

$$\tilde{\mathbf{v}}_l \leftarrow (W^l)^T \tilde{\mathbf{u}}_l / \| (W^l)^T \tilde{\mathbf{u}}_l \|_2 \quad (20)$$

$$\tilde{\mathbf{u}}_l \leftarrow W^l \tilde{\mathbf{v}}_l / \| W^l \tilde{\mathbf{v}}_l \|_2 \quad (21)$$

2. Calculate \bar{W}_{SN} with the spectral norm:

$$\bar{W}_{\text{SN}}^l(W^l) = W^l / \sigma(W^l), \text{ where } \sigma(W^l) = \tilde{\mathbf{u}}_l^T W^l \tilde{\mathbf{v}}_l \quad (22)$$

3. Update W^l with SGD on mini-batch dataset \mathcal{D}_M with a learning rate α :

$$W^l \leftarrow W^l - \alpha \nabla_{W^l} \ell(\bar{W}_{\text{SN}}^l(W^l), \mathcal{D}_M) \quad (23)$$

Spectral Normalization GAN (SNGAN)

$$V_D(\hat{G}, D) = \mathbb{E}_{\mathbf{x} \sim q_{\text{data}}(\mathbf{x})} [\min(0, -1 + D(\mathbf{x}))] + \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} [\min(0, -1 - D(\hat{G}(\mathbf{z})))]$$

$$V_G(G, \hat{D}) = - \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} [\hat{D}(G(\mathbf{z}))],$$

Geometric GAN

Jae Hyun Lim¹, Jong Chul Ye^{2,3}

¹ ETRI, South Korea

jaehyun.lim@etri.re.kr

² Dept. of Bio and Brain engineering, KAIST, South Korea

³ Dept. of Mathematical Sciences, KAIST, South Korea

jong.ye@kaist.ac.kr

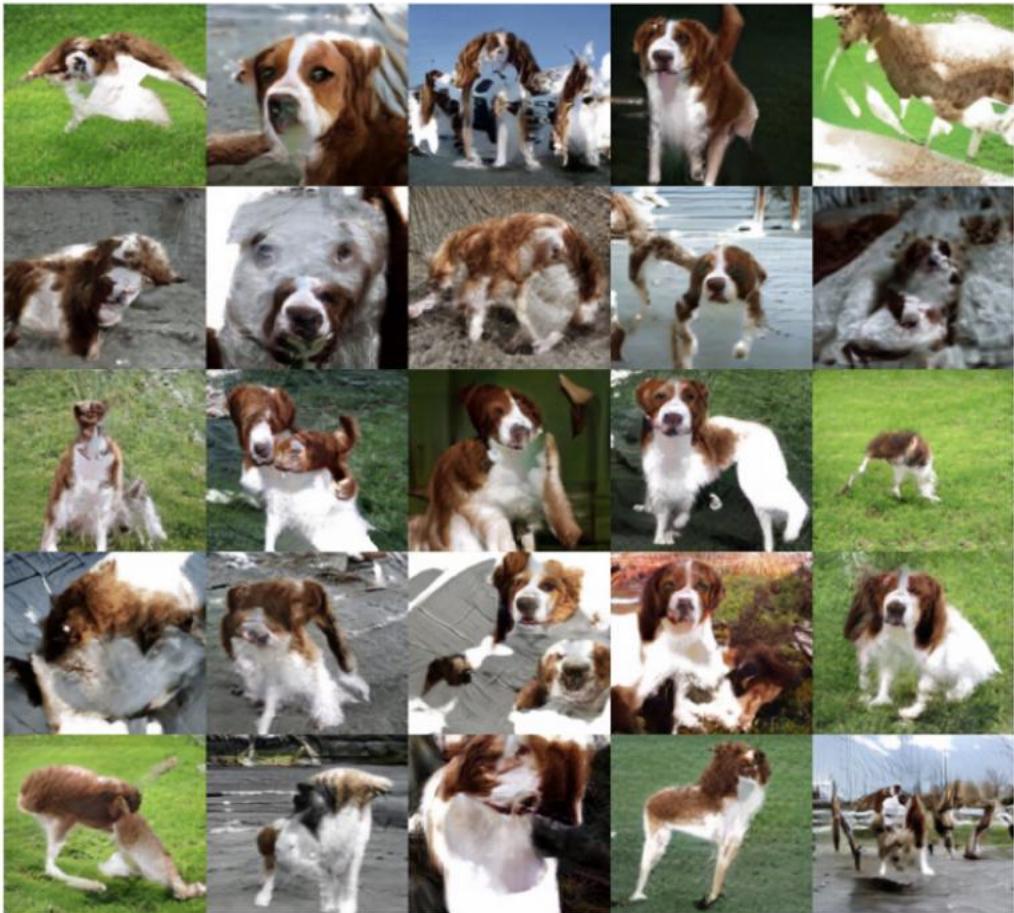
Spectral Normalization GAN (SNGAN)

| | | |
|---|--|--|
| $z \in \mathbb{R}^{128} \sim \mathcal{N}(0, I)$ | RGB image $x \in \mathbb{R}^{128 \times 128 \times 3}$ | RGB image $x \in \mathbb{R}^{128 \times 128 \times 3}$ |
| dense, $4 \times 4 \times 1024$ | ResBlock down 64 | ResBlock down 64 |
| ResBlock up 1024 | ResBlock down 128 | ResBlock down 128 |
| ResBlock up 512 | ResBlock down 256 | ResBlock down 256 |
| ResBlock up 256 | ResBlock down 512 | Concat(Embed(y), \mathbf{h}) |
| ResBlock up 128 | ResBlock down 1024 | ResBlock down 512 |
| ResBlock up 64 | ResBlock 1024 | ResBlock down 1024 |
| BN, ReLU, 3×3 conv 3 | ReLU | ResBlock 1024 |
| Tanh | Global sum pooling | ReLU |
| (a) Generator | dense $\rightarrow 1$ | Global sum pooling |
| | (b) Discriminator for unconditional GANs. | dense $\rightarrow 1$ |

(c) Discriminator for conditional GANs. For computational ease, we embedded the integer label $y \in \{0, \dots, 1000\}$ into 128 dimension before concatenating the vector to the output of the intermediate layer.

Spectral Normalization GAN (SNGAN)

Welsh springer spaniel



Pizza

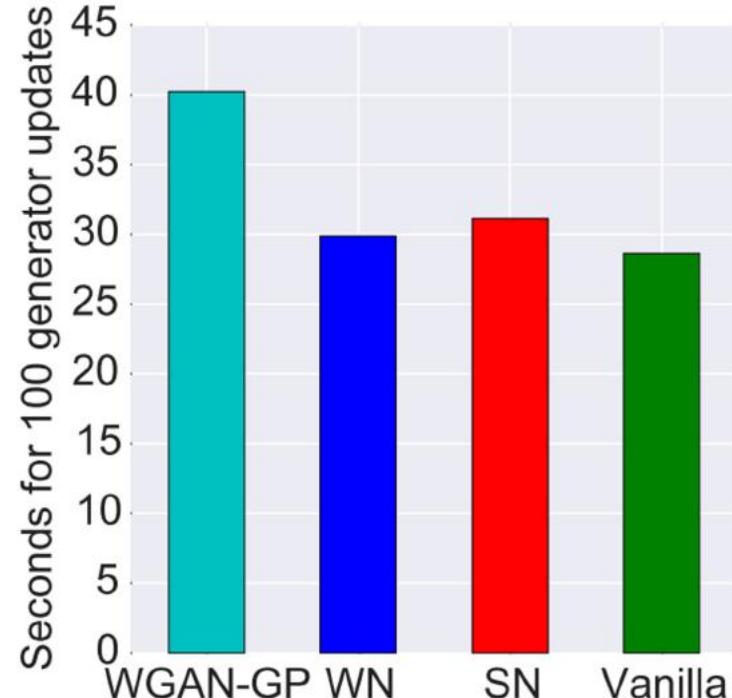


[Miyato et al. 2017]

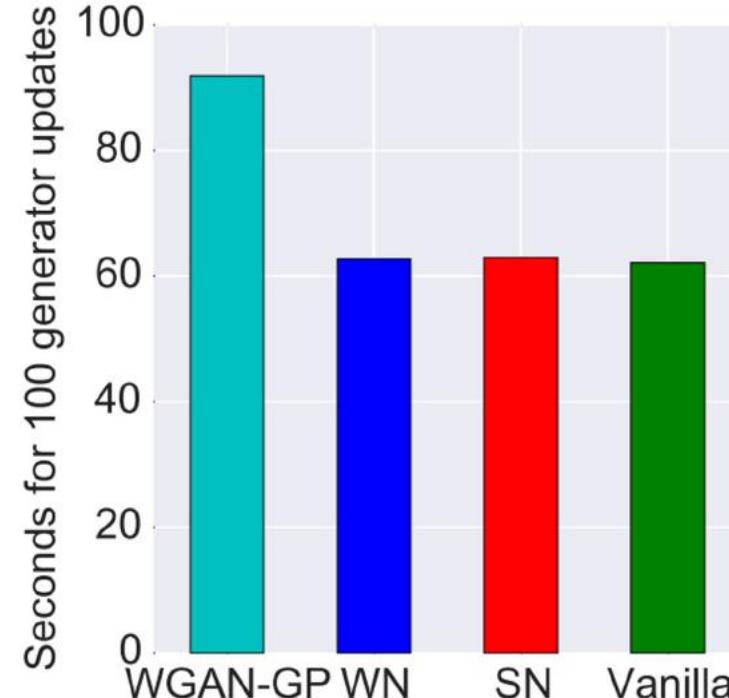
SNGAN: Summary

- High quality class conditional samples at Imagenet scale
- First GAN to work on full Imagenet (million image dataset)
- Computational benefits over WGAN-GP (single power iteration and no need of a backward pass)

SNGAN: Computational Benefits



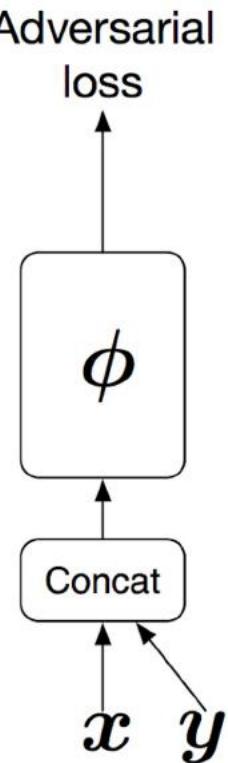
(a) CIFAR-10 (image size: $32 \times 32 \times 3$)



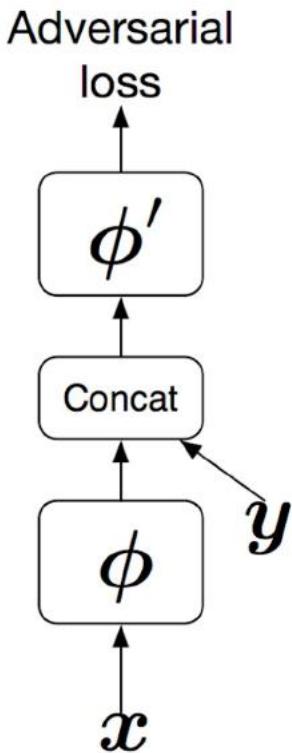
(b) STL-10 (images size: $48 \times 48 \times 3$)

Projection Discriminator

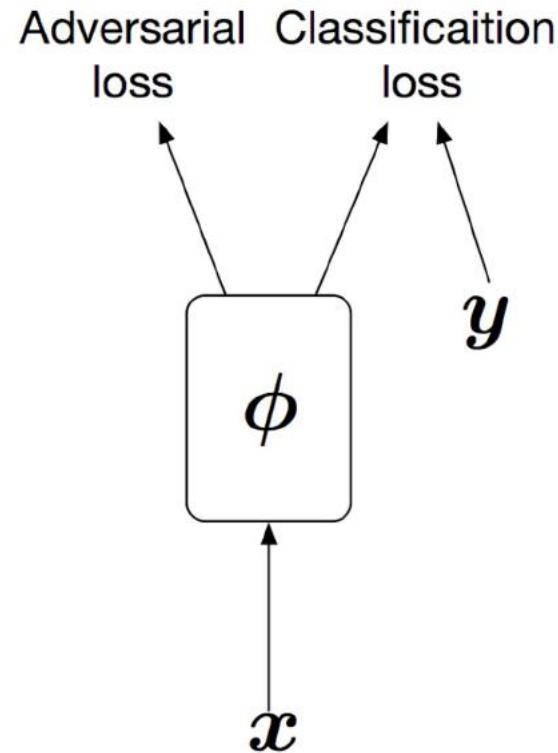
**(a) cGANs,
input concat**
(Mirza & Osindero, 2014)



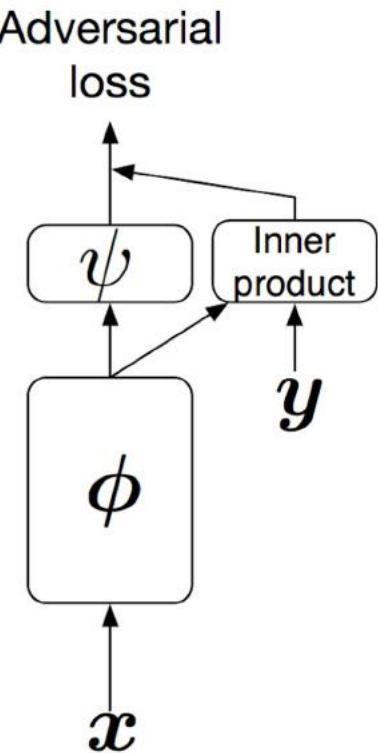
**(b) cGANs,
hidden concat**
(Reed et al., 2016)



(c) AC-GANs
(Odena et al., 2017)



(d) (ours) Projection



Lecture overview

- Motivation and Definition of Implicit Models
- Original GAN (Goodfellow et al, 2014)
- Evaluation: Parzen, Inception, Frechet
- Theory of GANs
- **GAN Progression**
 - DC GAN (Radford et al, 2016)
 - Improved Training of GANs (Salimans et al'16), Projected GAN (Sauer et al'21)
WGAN, WGAN-GP, Progressive GAN, SN-GAN, **SAGAN**
 - BigGAN,BigGAN-Deep,StyleGAN,StyleGAN2,StyleGAN3,StyleGAN-XL,VIB-GAN
- Conditional GANs, Cycle-Consistent Adversarial Networks
- GANs and Representations
- Applications

Self Attention GAN (SAGAN)

Self-Attention Generative Adversarial Networks

Han Zhang*

Rutgers University

Ian Goodfellow

Google Brain

Dimitris Metaxas

Rutgers University

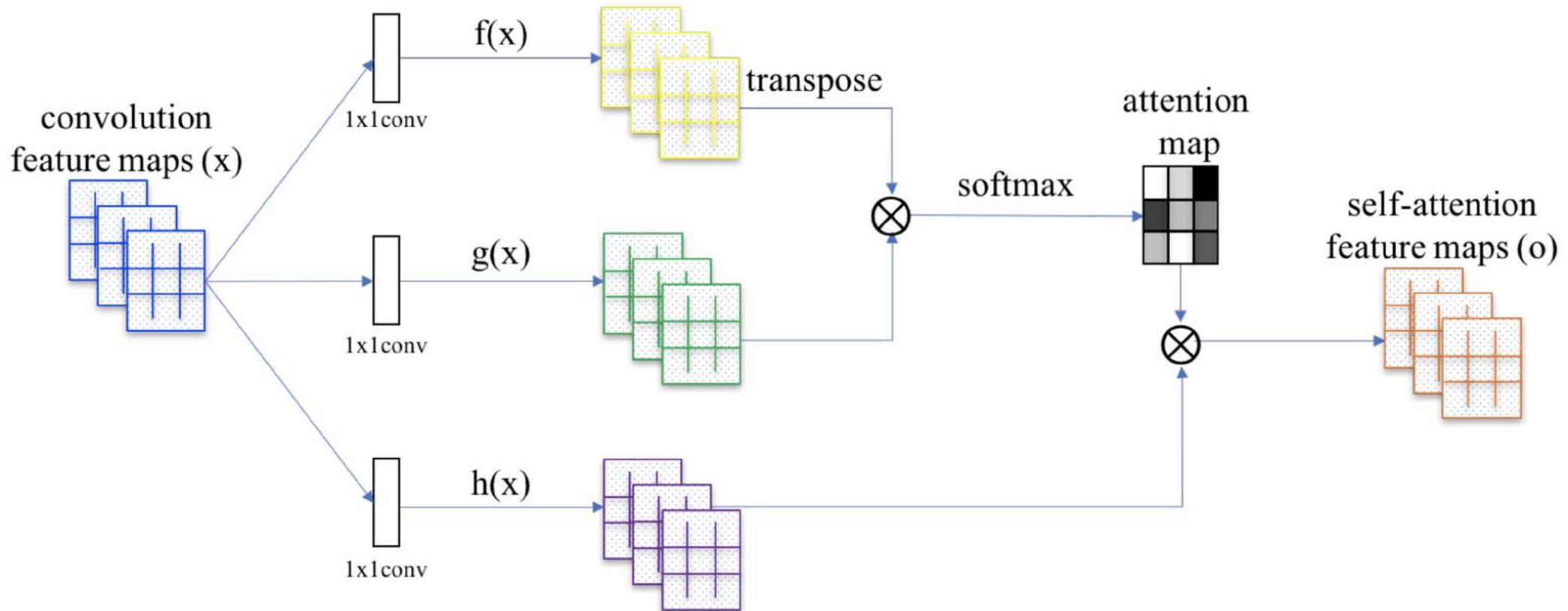
Augustus Odena

Google Brain

Abstract

In this paper, we propose the Self-Attention Generative Adversarial Network (SAGAN) which allows attention-driven, long-range dependency modeling for image generation tasks. Traditional convolutional GANs generate high-resolution details as a function of only spatially local points in lower-resolution feature maps. In SAGAN, details can be generated using cues from all feature locations. Moreover, the discriminator can check that highly detailed features in distant portions of the image are consistent with each other. Furthermore, recent work has shown that generator conditioning affects GAN performance. Leveraging this insight, we apply spectral normalization to the GAN generator and find that this improves training dynamics. The proposed SAGAN achieves the state-of-the-art results, boosting the best published Inception score from 36.8 to 52.52 and reducing Fréchet Inception distance from 27.62 to 18.65 on the challenging ImageNet dataset. Visualization of the attention layers shows that the generator leverages neighborhoods that correspond to object shapes rather than local regions of fixed shape.

Self Attention GAN (SAGAN)



Self Attention GAN (SAGAN)

$$f(x) = W_f x, \ g(x) = W_g x$$

$$\beta_{j,i} = \frac{\exp(s_{ij})}{\sum_{i=1}^N \exp(s_{ij})}$$

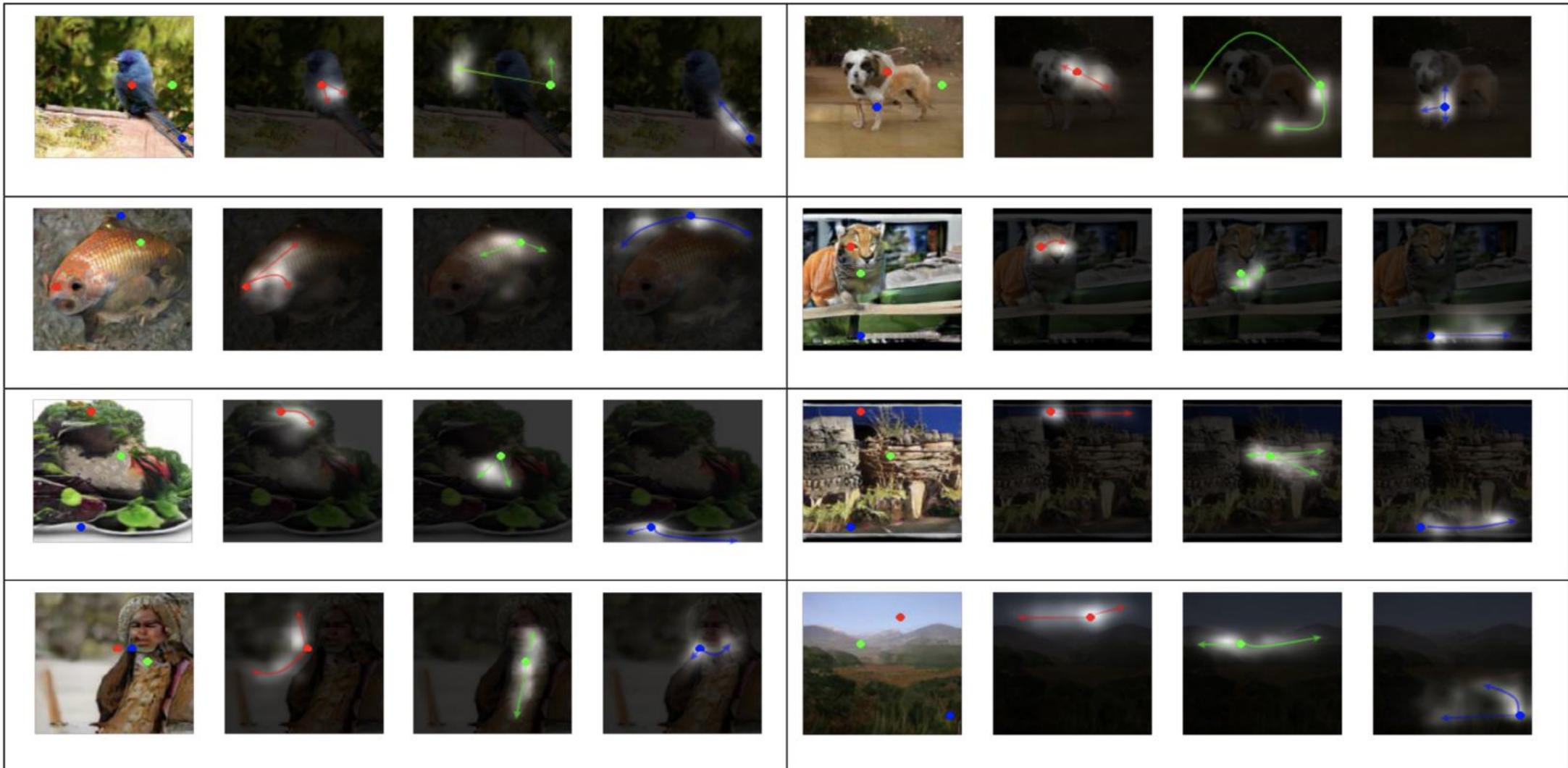
$$s_{ij} = \mathbf{f}(\mathbf{x}_i)^T \mathbf{g}(\mathbf{x}_j)$$

$$\mathbf{y}_i = \gamma \mathbf{o}_i + \mathbf{x}_i$$

Self Attention GAN (SAGAN)

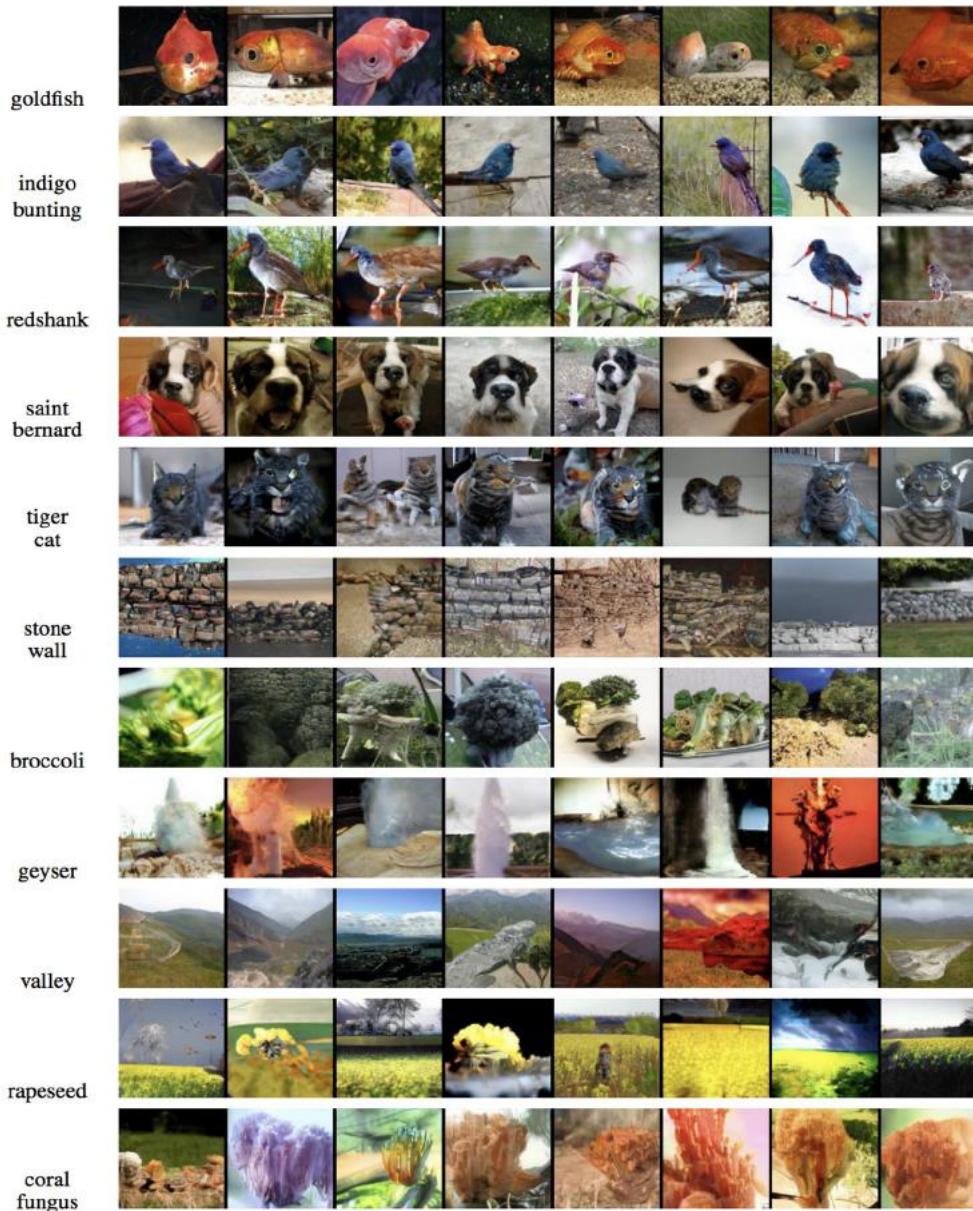
- Applies spectral normalization to both the generator and discriminator weight matrices
 - This is counter-intuitive to popular belief that you only have to mathematically condition the discriminator
- Uses self-attention in both the generator and discriminator
- Hinge Loss
- First GAN to produce “good” unconditional full ImageNet samples
- Conditional models
 - Conditional BN for G, Projection Discriminator for D

Self Attention GAN (SAGAN)



[Zhang et al. 2018]

Self Attention GAN (SAGAN)



[Zhang et al. 2018]

Self Attention GAN (SAGAN)

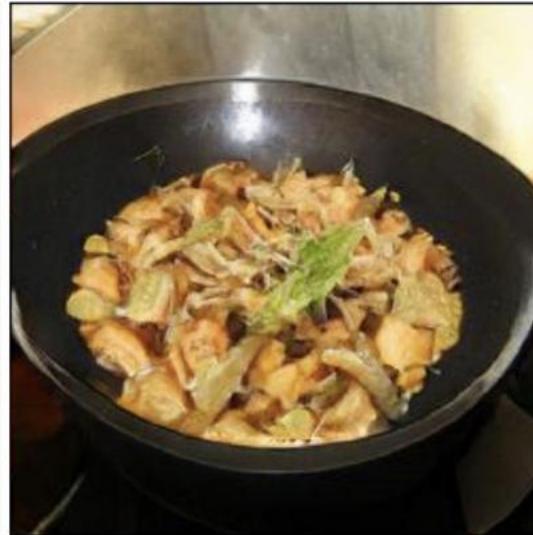
| Model | Inception Score | FID |
|-----------------------|-----------------|--------------|
| AC-GAN [31] | 28.5 | / |
| SNGAN-projection [17] | 36.8 | 27.62* |
| SAGAN | 52.52 | 18.65 |

Table 2: Comparison of the proposed SAGAN with state-of-the-art GAN models [19, 17] for class conditional image generation on ImageNet. FID of SNGAN-projection is calculated from officially released weights.

Lecture overview

- Motivation and Definition of Implicit Models
- Original GAN (Goodfellow et al, 2014)
- Evaluation: Parzen, Inception, Frechet
- Theory of GANs
- **GAN Progression**
 - DC GAN (Radford et al, 2016)
 - Improved Training of GANs (Salimans et al'16), Projected GAN (Sauer et al'21)
WGAN, WGAN-GP, Progressive GAN, SN-GAN, SAGAN
 - BigGAN, BigGAN-Deep, StyleGAN, StyleGAN2, StyleGAN3, StyleGAN-XL, VIB-GAN
- Conditional GANs, Cycle-Consistent Adversarial Networks
- GANs and Representations
- Applications

BigGAN



BigGAN



BigGAN

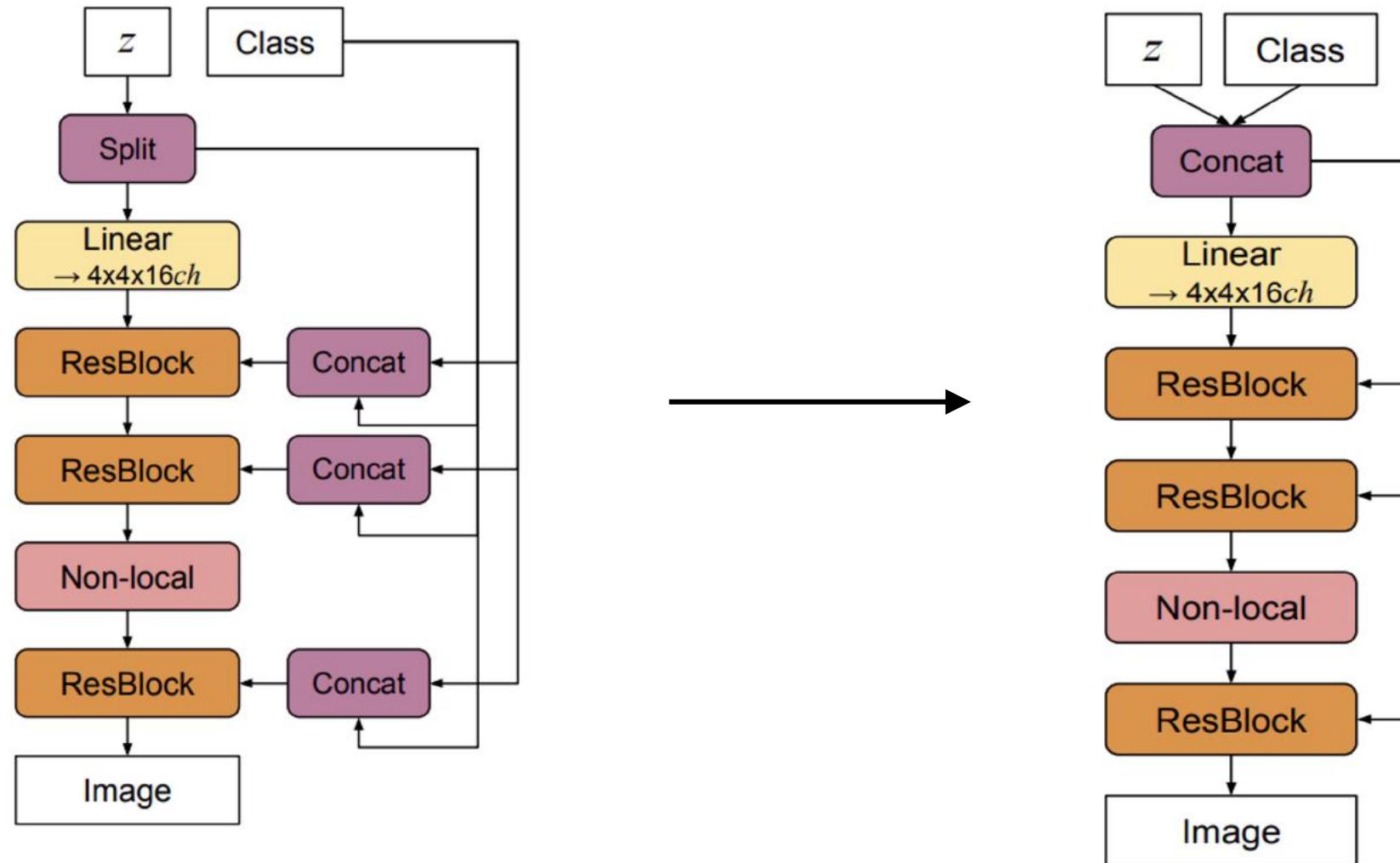
$$R_\beta(W) = \beta \|W^\top W - I\|_F^2$$



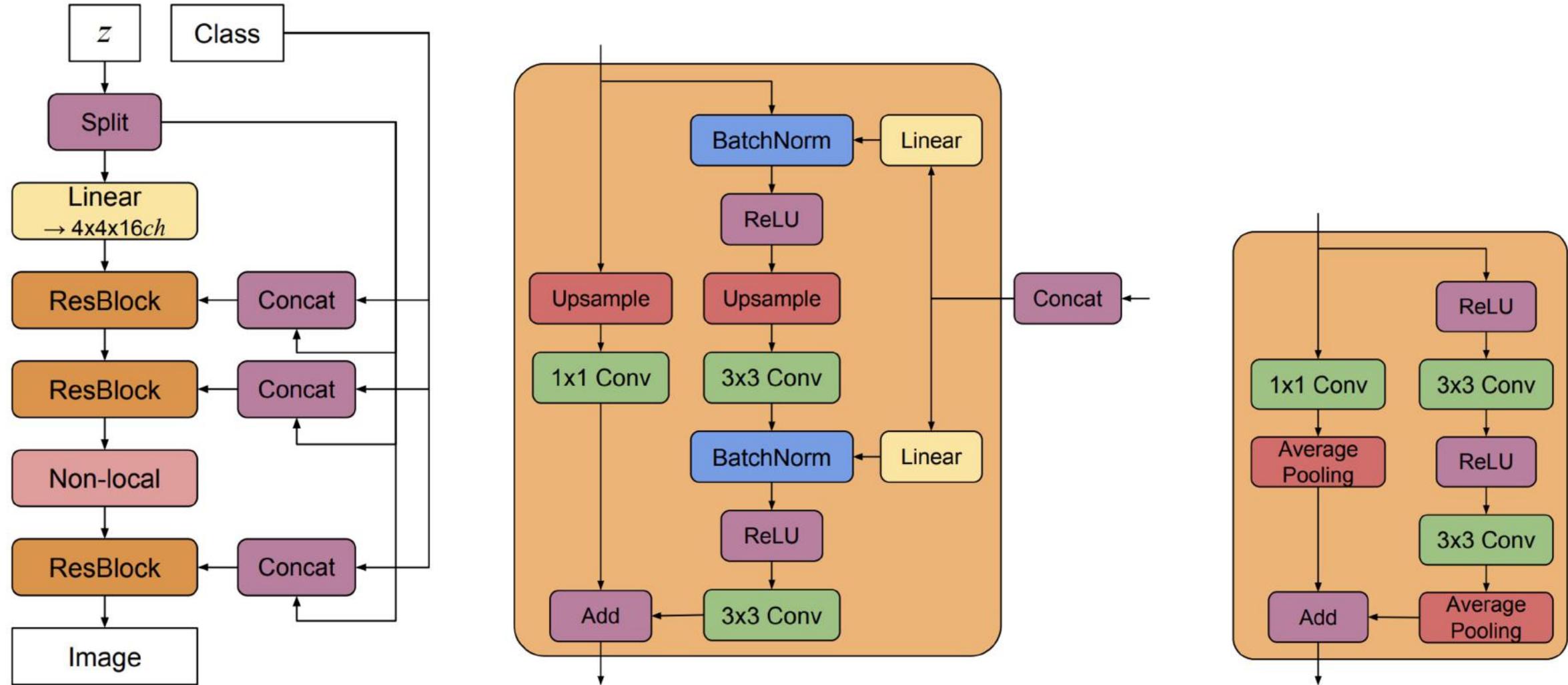
$$R_\beta(W) = \beta \|W^\top W \odot (1 - I)\|_F^2,$$

Orthogonal Regularization

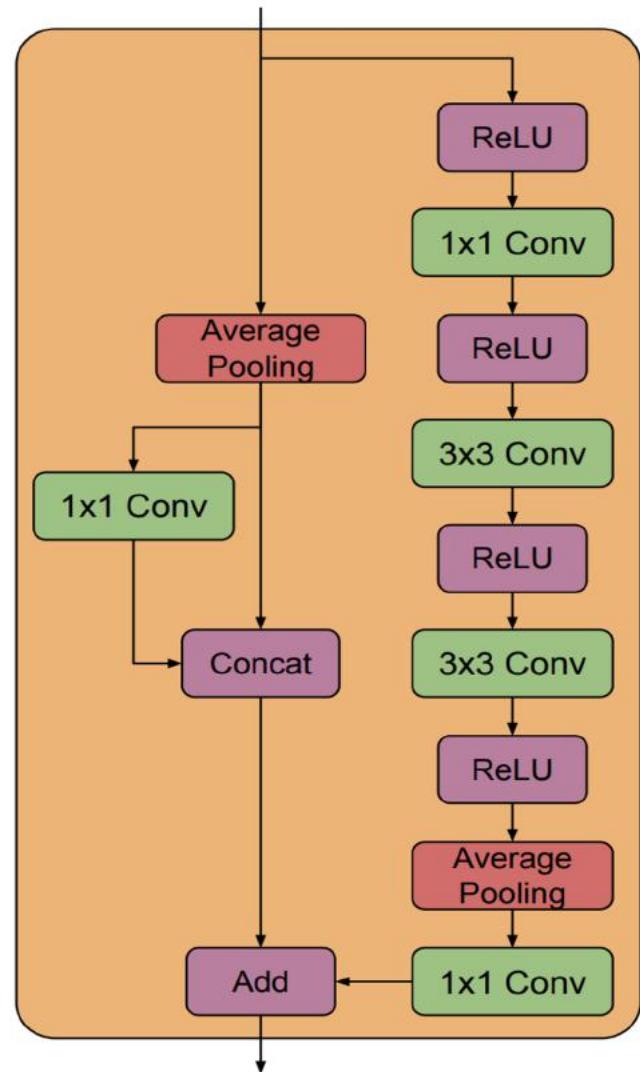
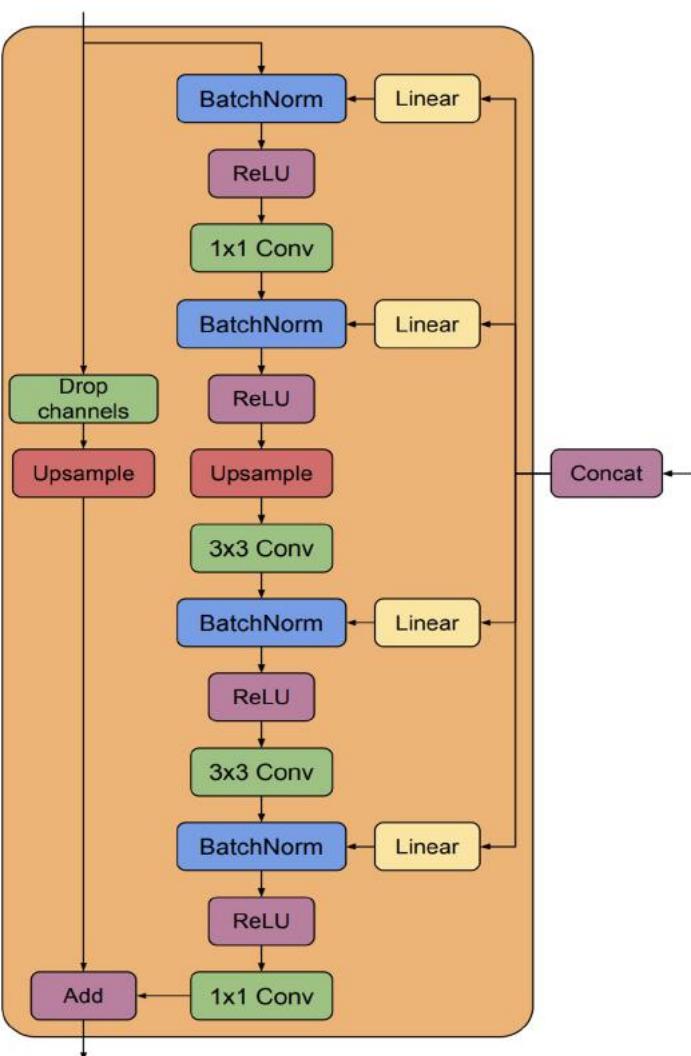
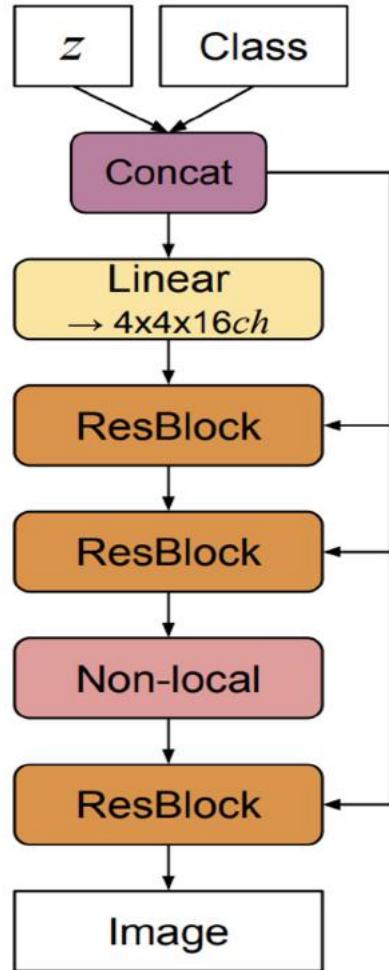
BigGAN and BigGAN-deep



BigGAN



BigGAN-deep



BigGAN

Table 6: BigGAN architecture for 512×512 images. Relative to the 256×256 architecture, we add an additional ResBlock at the 512×512 resolution. Memory constraints force us to move the non-local block in both networks back to 64×64 resolution as in the 128×128 pixel setting.

| |
|--|
| $z \in \mathbb{R}^{160} \sim \mathcal{N}(0, I)$ |
| Embed(y) $\in \mathbb{R}^{128}$ |
| Linear $(20 + 128) \rightarrow 4 \times 4 \times 16ch$ |
| ResBlock up $16ch \rightarrow 16ch$ |
| ResBlock up $16ch \rightarrow 8ch$ |
| ResBlock up $8ch \rightarrow 8ch$ |
| ResBlock up $8ch \rightarrow 4ch$ |
| Non-Local Block (64×64) |
| ResBlock up $4ch \rightarrow 2ch$ |
| ResBlock up $2ch \rightarrow ch$ |
| ResBlock up $ch \rightarrow ch$ |
| BN, ReLU, 3×3 Conv $ch \rightarrow 3$ |
| Tanh |

(a) Generator

| |
|--|
| RGB image $x \in \mathbb{R}^{512 \times 512 \times 3}$ |
| ResBlock down $ch \rightarrow ch$ |
| ResBlock down $ch \rightarrow 2ch$ |
| ResBlock down $2ch \rightarrow 4ch$ |
| Non-Local Block (64×64) |
| ResBlock down $4ch \rightarrow 8ch$ |
| ResBlock down $8ch \rightarrow 8ch$ |
| ResBlock down $8ch \rightarrow 16ch$ |
| ResBlock down $16ch \rightarrow 16ch$ |
| ResBlock $16ch \rightarrow 16ch$ |
| ReLU, Global sum pooling |
| Embed(y) $\cdot h$ + (linear $\rightarrow 1$) |

(b) Discriminator

BigGAN

- Increase your batch size (as much as you can)
- Use Cross-Replica (Sync) Batch Norm
- Increase your model size
- Wider helps as much as deeper
- Fuse class information at all levels
- Hinge Loss
- Orthonormal regularization & Truncation Trick

BigGAN

| Batch | Ch. | Param (M) | Shared | Skip- z | Ortho. | Itr $\times 10^3$ | FID | IS |
|-------|-----|-----------|-----------------|-----------|--------|-------------------|---------------------|---------------------|
| 256 | 64 | 81.5 | SA-GAN Baseline | | | 1000 | 18.65 | 52.52 |
| 512 | 64 | 81.5 | ✗ | ✗ | ✗ | 1000 | 15.30 | 58.77(± 1.18) |
| 1024 | 64 | 81.5 | ✗ | ✗ | ✗ | 1000 | 14.88 | 63.03(± 1.42) |
| 2048 | 64 | 81.5 | ✗ | ✗ | ✗ | 732 | 12.39 | 76.85(± 3.83) |
| 2048 | 96 | 173.5 | ✗ | ✗ | ✗ | 295(± 18) | 9.54(± 0.62) | 92.98(± 4.27) |
| 2048 | 96 | 160.6 | ✓ | ✗ | ✗ | 185(± 11) | 9.18(± 0.13) | 94.94(± 1.32) |
| 2048 | 96 | 158.3 | ✓ | ✓ | ✗ | 152(± 7) | 8.73(± 0.45) | 98.76(± 2.84) |
| 2048 | 96 | 158.3 | ✓ | ✓ | ✓ | 165(± 13) | 8.51(± 0.32) | 99.31(± 2.10) |
| 2048 | 64 | 71.3 | ✓ | ✓ | ✓ | 371(± 7) | 10.48(± 0.10) | 86.90(± 0.61) |

BigGAN

| Model | Res. | FID/IS | (min FID) / IS | FID / (valid IS) | FID / (max IS) |
|-------------|------|------------------------------|------------------------------|------------------------------|---------------------------|
| SN-GAN | 128 | 27.62/36.80 | N/A | N/A | N/A |
| SA-GAN | 128 | 18.65/52.52 | N/A | N/A | N/A |
| BigGAN | 128 | $8.7 \pm .6$ / 98.8 ± 3 | $7.7 \pm .2$ / 126.5 ± 0 | $9.6 \pm .4$ / 166.3 ± 1 | 25 ± 2 / 206 ± 2 |
| BigGAN | 256 | $8.7 \pm .1$ / 142.3 ± 2 | $7.7 \pm .1$ / 178.0 ± 5 | $9.3 \pm .3$ / 233.1 ± 1 | 25 ± 5 / 291 ± 4 |
| BigGAN | 512 | 8.1/144.2 | 7.6/170.3 | 11.8/241.4 | 27.0/275 |
| BigGAN-deep | 128 | $5.7 \pm .3$ / 124.5 ± 2 | $6.3 \pm .3$ / 148.1 ± 4 | $7.4 \pm .6$ / 166.5 ± 1 | 25 ± 2 / 253 ± 11 |
| BigGAN-deep | 256 | $6.9 \pm .2$ / 171.4 ± 2 | $7.0 \pm .1$ / 202.6 ± 2 | $8.1 \pm .1$ / 232.5 ± 2 | 27 ± 8 / 317 ± 6 |
| BigGAN-deep | 512 | 7.5/152.8 | 7.7/181.4 | 11.5/241.5 | 39.7/298 |

BigGAN - Truncation Trick



(a)

(b)

Remarkably, our best results come from using a different latent distribution for sampling than was used in training. Taking a model trained with $z \sim \mathcal{N}(0, I)$ and sampling z from a *truncated normal* (where values which fall outside a range are resampled to fall inside that range) immediately provides a boost to IS and FID. We call this the *Truncation Trick*: truncating a z vector by resampling the values with magnitude above a chosen threshold leads to improvement in individual sample quality at the cost of reduction in overall sample variety. Figure 2(a) demonstrates this: as the threshold is reduced, and elements of z are truncated towards zero (the mode of the latent distribution), individual samples approach the mode of \mathbf{G} 's output distribution. Related observations about this trade-off were made in (Marchesi, 2016; Pieters & Wiering, 2014).

BigGAN - Sampling

The default behavior with batch normalized classifier networks is to use a running average of the activation moments at test time. Previous works (Radford et al., 2016) have instead used batch statistics when sampling images. While this is not technically an invalid way to sample, it means that results are dependent on the test batch size (and how many devices it is split across), and further complicates reproducibility.

We find that this detail is extremely important, with changes in test batch size producing drastic changes in performance. This is further exacerbated when one uses exponential moving averages of \mathbf{G} 's weights for sampling, as the BatchNorm running averages are computed with non-averaged weights and are poor estimates of the activation statistics for the averaged weights.

To counteract both these issues, we employ “standing statistics,” where we compute activation statistics at sampling time by running the \mathbf{G} through multiple forward passes (typically 100) each with different batches of random noise, and storing means and variances aggregated across all forward passes. Analogous to using running statistics, this results in \mathbf{G} 's outputs becoming invariant to batch size and the number of devices, even when producing a single sample.

BigGAN



(a) 128×128



(b) 256×256



(c) 512×512



(d)

BigGAN

