Generative Adversarial Networks (GANs)

Jie Tang

Tsinghua University

May 8, 2019

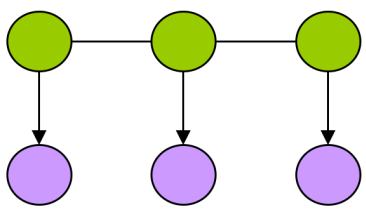
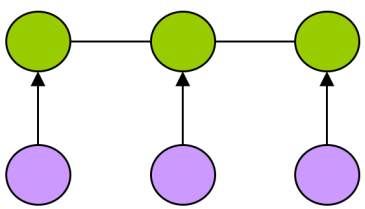
Overview

Outline

Generative Model

- In generative model, we're interested in training a model, for example, a neural network, that models a distribution, such as a distribution over natural images.
- Formally, we have a training set $\{\mathbf{x}_i\}_{i=1}^N$ where $\mathbf{x}_i \sim \mathbb{P}_r$. Generative models aim at learning a distribution $\mathbb{P}_g$, which is an estimate of $\mathbb{P}_r$, using $\{\mathbf{x}_i\}_{i=1}^N$.
- As soon as we have obtained $\mathbb{P}_g$, we can sample from $\mathbb{P}_g$. If $\mathbb{P}_g = \mathbb{P}_r$, the samples should look like $\{\mathbf{x}_i\}_{i=1}^N$.
  - For example, when $\{\mathbf{x}_i\}_{i=1}^N$ are real images, the samples should look real.
- Thus, one direct way to judge the quality of a generative model is to sample from it.

Generative vs. Discriminative

Generative	Discriminative
$P(Y\|X)$	$P(Y\|X)$
Example: HMM	Example: MaxEnt, MEMM, CRF
 States generates observations	 Observations (features) determine states
Learning = finding model generating observation (sequence) from state (sequence)	Learning = finding model mapping observation (sequence) to state (sequence)
Tagging = finding most likely state (sequence) having generated given observation (sequence)	Tagging = finding most likely state (sequence) mapped from given observation (sequence)

Generative Model

- Generative models can be roughly classified into two categories.
  - Explicit generative models are models that explicitly perform density estimation on $\{\mathbf{x}_i\}_{i=1}^N$ so that the log-likelihood objective under max-likelihood criterion can be explicitly written down. For example, mixture of Gaussian model.
  - Implicit generative models are models that do not explicitly perform density estimation through max-likelihood criterion. Rather, they focus on the generation, i.e., the sampling of data from $\mathbb{P}_g$, directly. For example, generative adversarial network (GAN).
- When real data's distribution is too complex, explicit generative models like VAE have to resort to approximation to perform density estimation, which incurs bias intrinsically.
- That's why we may favor implicit generative models like GAN, which avoid density estimation completely yet are able to generate new data.

GAN's Place in Generative Models

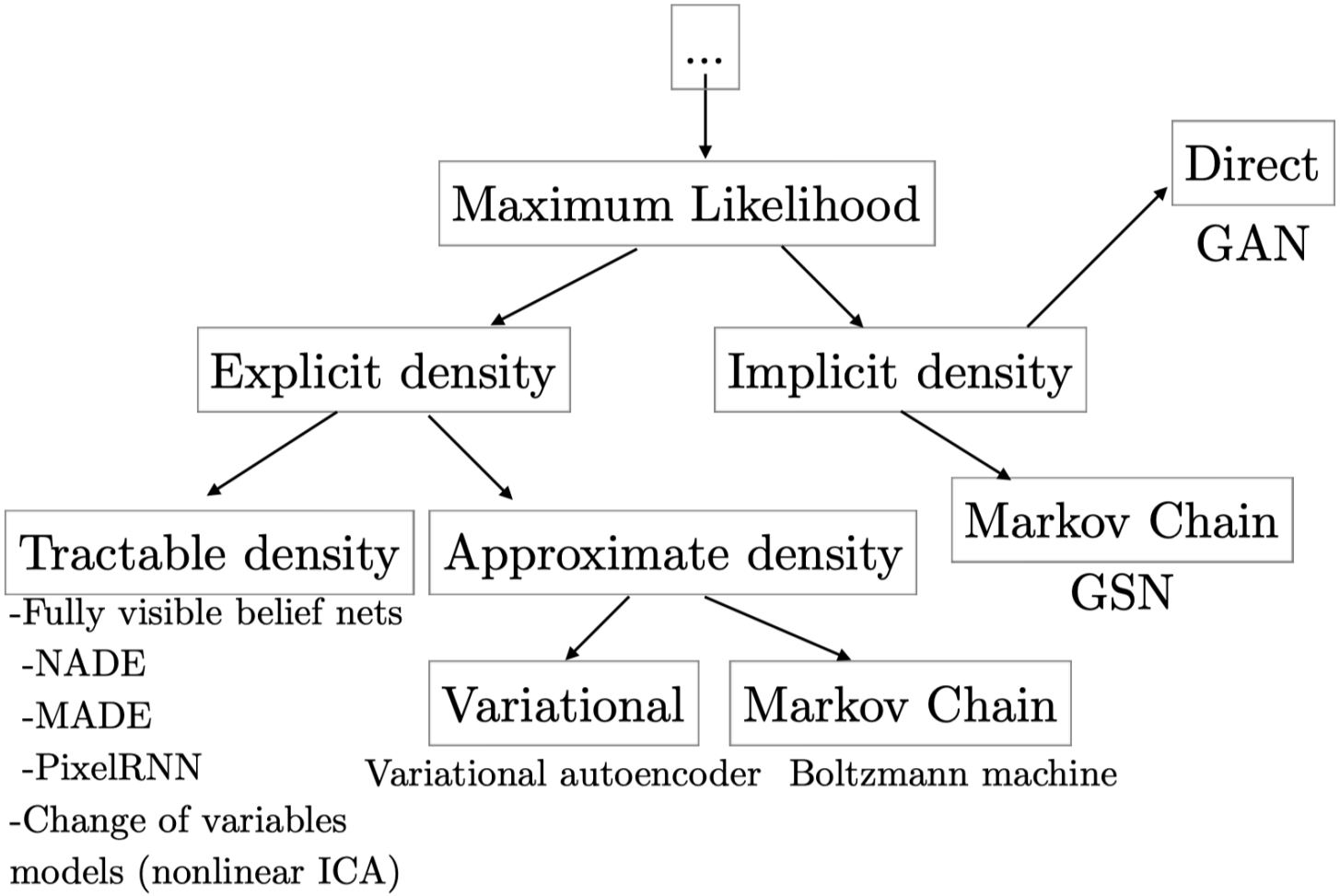


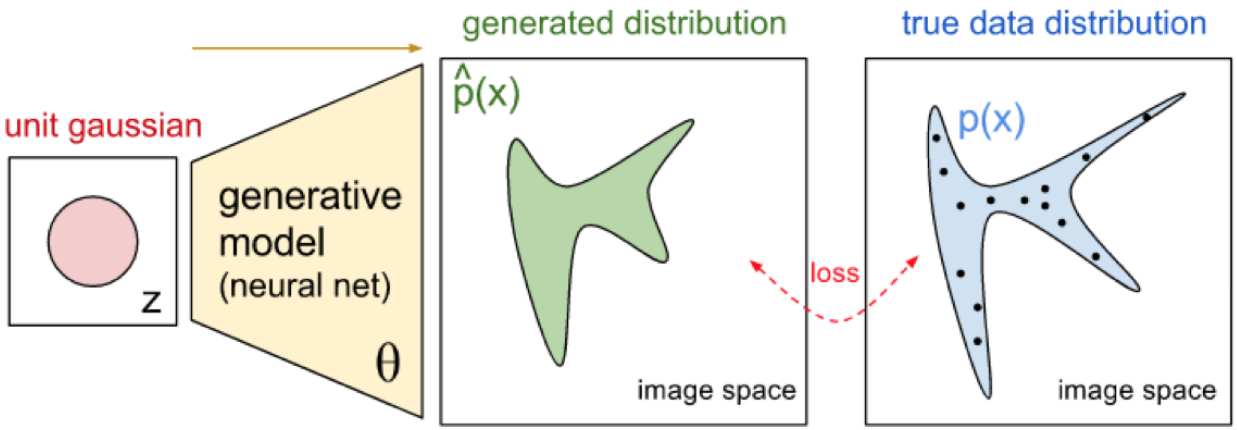
Figure 1: Deep generative models that can learn via the principle of maximum likelihood differ with respect to how they represent or approximate the likelihood (Courtesy to Ian Goodfellow).

Implicit Generative Model

- Implicit generative model implicitly defines a probability distribution through a mapping.
- The input is a vector $\mathbf{z}$ sampled from some fixed, simple distribution, for example, a standard Gaussian.
- The model denoted by G could be any mapping, for example, a mapping given by a neural network.
- The output $G(\mathbf{z})$ then obeys some other distribution. We are interested when the mapping G maps $\mathbf{z}$ into the real data space $\mathbf{x}$.

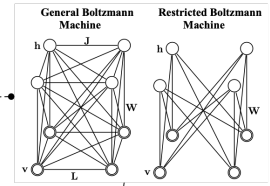
Implicit Generative Model

- The following figure gives an illustration about how an implicit generative model works.



https://blog.openai.com/generative-models/

DBM(2009)

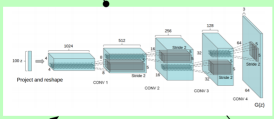


Alec Radford
Indico Research
deep adversarial learning

Jun-Yan Zhu
UC Berkeley(now@MIT)

Hinton
Deep Learning

DCGAN(2015)

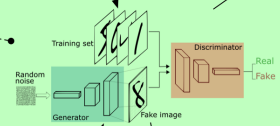


video2video(2018)

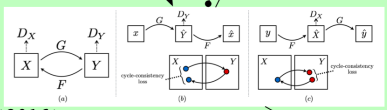
Ting-Chun Wang
NVIDIA

Tero Karras
NVIDIA

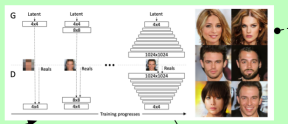
GAN(2014)



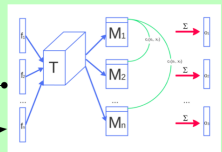
CycleGAN(2017)



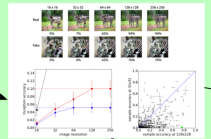
PGGAN(2017)



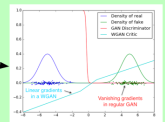
ImprovedGAN(2016)



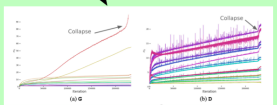
PACGAN(2016)

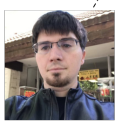


WGAN(2017)



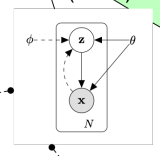
BigGAN(2018)





Ian Goodfellow
Google Brain
Deep Adversarial Learning

VAE(2013)



Andrew Brock
Heriot-Watt University&DeepMind

Max Welling
University of Amsterdam

Durk Kingma

Tim Salimans
OpenAI(now@Google)

Zinan Lin
CMU

Martin Arjovsky
NYC

Zoo of GANs

GAN — 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 Categorial 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 Nets
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
SeGAN — SeGAN: 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
S^2GAN — 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 AdversarialNetworks
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

Figure 2: See https://github.com/hindupuravinash/the-gan-zoo

An Explo-GAN of Papers

- The number of papers about GAN variants has seen explosive growth since the publication of the renowned paper "Generative Adversarial Nets" in 2014 by Ian J. Goodfellow. The paper has a citation of 8136 as of 4.10.2019 according to Google Scholar.
- See the figure.

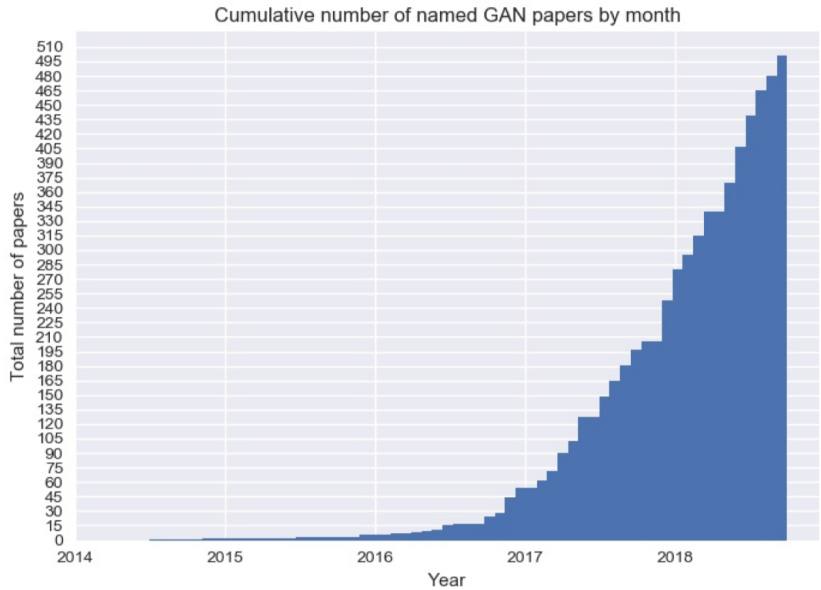


Figure 3: From Deep Hunt, blog by Avinash Hindupur.

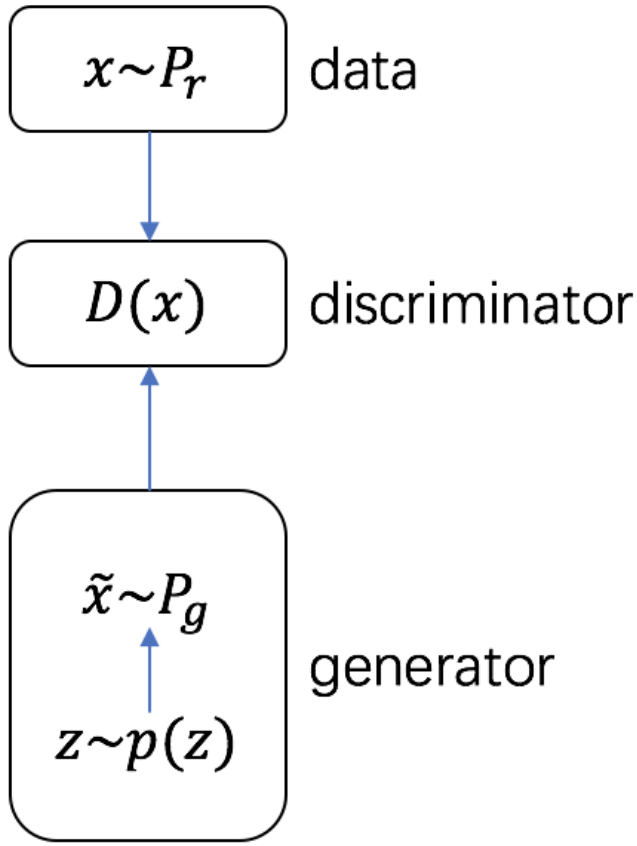
Outline

What is GAN

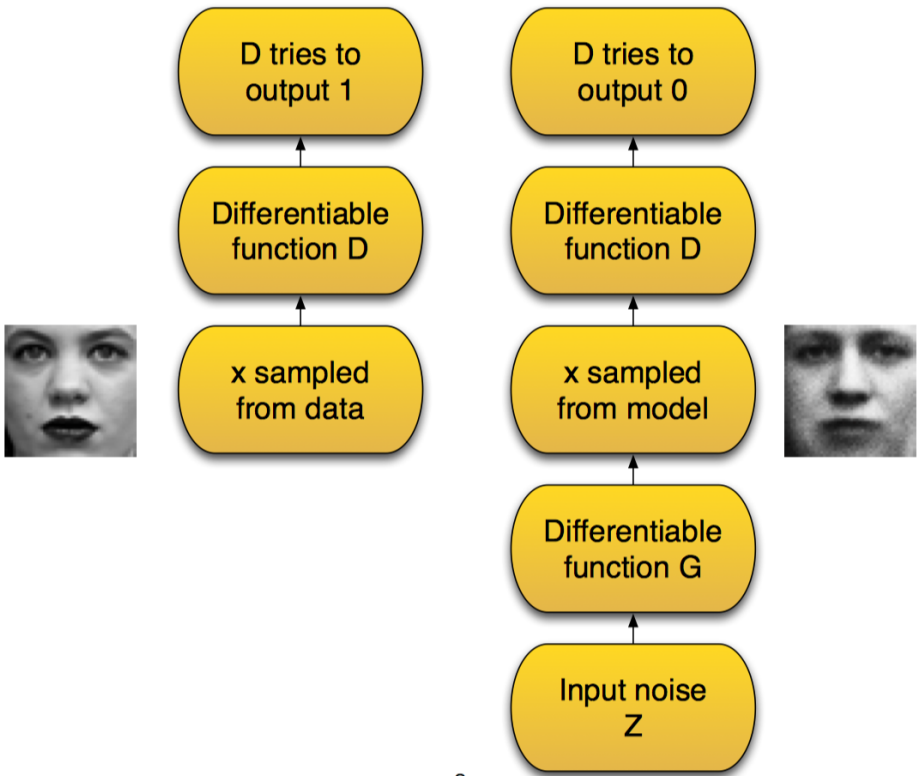
- GAN is short for generative adversarial network.
  - Generative: GAN is a generative model. Thus, it defines a probability distribution $\mathbb{P}_g$ to sample from.
  - Adversarial: GAN does not explicitly perform density estimation over $\{\mathbf{x}_i\}_{i=1}^N (\mathbf{x}_i \sim \mathbb{P}_r)$ to get $\mathbb{P}_g$. Rather, it depends on a discriminator D to distinguish $\mathbb{P}_g$ from $\mathbb{P}_r$. Thus D and G are adversaries to each other.
    - The intuition is when a good D is unable to tell $\mathbb{P}_g$ and $\mathbb{P}_r$ apart, $\mathbb{P}_g$ and $\mathbb{P}_r$ shall be the same.
  - Network: GAN uses neural networks, which are universal function approximators, to model both the generation and discrimination part.
- GAN is designed to be unbiased. Infinite data leads to $\mathbb{P}_g = \mathbb{P}_r$.
  - VAE is not. Even infinite data is given, approximation through variational inference results in a gap between $\mathbb{P}_g$ and $\mathbb{P}_r$.

Architecture of GAN

- GAN is a game between two players
  - A discriminator D.
  - A generator G.
- D tries to discriminate between:
  - A sample from the data distribution.
  - A sample given by G.
- G tries to fool D by generating samples that are hard for D to distinguish from real data.



An Example


D tries to output 1
Differentiable function D
x sampled from data
D tries to output 0
Differentiable function D
x sampled from model
Differentiable function G
Input noise Z

GAN's Objective

- Formally, we can express the game between G and D with the minimax objective

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_r}[\log D(\mathbf{x})] + \mathbb{E}_{\tilde{\mathbf{x}}\sim\mathbb{P}_g}[\log(1 - D(\tilde{\mathbf{x}}))], \quad (1)$$

  where
  - $\mathbb{P}_r$ is the distribution of real data.
  - $\mathbb{P}_g$ is the distribution implicitly modeled by

$$\mathbf{z} \sim p(\mathbf{z}), \tilde{\mathbf{x}} \sim G(\mathbf{z}),$$

    where G is the generative network, also called generator.
  - The generator G's input $\mathbf{z}$ is sampled from some simple noise distribution (e.g., uniform or Gaussian).

Optimal Condition

- One can show that for any $\mathbb{P}_g$, the optimal non-parametric discriminator is (See proof in next slide)

$$D^*(\mathbf{x}) = \frac{\mathbb{P}_r(\mathbf{x})}{\mathbb{P}_g(\mathbf{x}) + \mathbb{P}_r(\mathbf{x})}. \tag{2}$$

- Under an ideal discriminator, the generator minimizes the Jensen-Shannon divergence between $\mathbb{P}_r$ and $\mathbb{P}_g$, i.e.,

$$JS(\mathbb{P}_r||\mathbb{P}_g) = KL(\mathbb{P}_r||\frac{\mathbb{P}_r + \mathbb{P}_g}{2}) + KL(\mathbb{P}_g||\frac{\mathbb{P}_r + \mathbb{P}_g}{2}), \tag{3}$$

  where KL denote the KullbackLeibler divergence.
- Therefore, when infinite data and infinite model capacity is assumed, the optimum of G corresponds to the real data's distribution, i.e., $\mathbb{P}_g^* = \mathbb{P}_r$.

Global Optimality of $\mathbb{P}_g = \mathbb{P}_r$

The optimal D for any given G is

$$D^*_G(\mathbf{x}) = \frac{\mathbb{P}_r(\mathbf{x})}{\mathbb{P}_r(\mathbf{x}) + \mathbb{P}_g(\mathbf{x})}.$$

Proof: For given G, D is to maximize the quantity $V(G, D)$

$$\begin{aligned} V(G, D) &= \int_{\mathbf{x}} \mathbb{P}_r(\mathbf{x}) \log(D(\mathbf{x})) dx + \int_{\mathbf{z}} \mathbb{P}_z(\mathbf{z}) \log(1 - D(G(\mathbf{z}))) dz \\ &= \int_{\mathbf{x}} \mathbb{P}_r(\mathbf{x}) \log(D(\mathbf{x})) + \mathbb{P}_g(\mathbf{x}) \log(1 - D(\mathbf{x})) dx. \end{aligned}$$

For any $(a, b) \in R^2 \backslash \{0, 0\}$, the function $y \to a\log y + b\log(1 - y)$ reaches its maximum in $[0, 1]$ at $\frac{a}{a+b}$.

Global Optimality of $\mathbb{P}_g = \mathbb{P}_r$

$$
\begin{aligned}
C(G) &:= \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_r}[\log(D^*(\mathbf{x}))] + \mathbb{E}_{\mathbf{z}\sim\mathbb{P}_z}[\log(1 - D^*(G(\mathbf{z})))] \\
&= \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_r}[\log(D^*(\mathbf{x}))] + \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_g}[\log(1 - D^*(\mathbf{x}))] \\
&= \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_r}\left[\log\frac{\mathbb{P}_r(\mathbf{x})}{\mathbb{P}_r(\mathbf{x}) + \mathbb{P}_g(\mathbf{x})}\right] + \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_g}\left[\log\frac{\mathbb{P}_g(\mathbf{x})}{\mathbb{P}_r(\mathbf{x}) + \mathbb{P}_g(\mathbf{x})}\right] \\
&= -\log 4 + KL(\mathbb{P}_r||\frac{\mathbb{P}_r + \mathbb{P}_g}{2}) + KL(\mathbb{P}_g||\frac{\mathbb{P}_r + \mathbb{P}_g}{2})
\end{aligned}
$$

- If G and D have enough capacity, and at each step of training, D is allowed to reach its optimum given G, and $\mathbb{P}_g$ is updated so as to improve the criterion $C(G)$.
- Therefore, $\mathbb{P}_g$ will **converges to** $\mathbb{P}_r$ (Jensen-Shannon Divergence (JSD)).
- Finally, the global minimum of the virtual training criterion $C(G)$ is achieved if and only if $\mathbb{P}_g = \mathbb{P}_r$.

Outline

Training of GAN

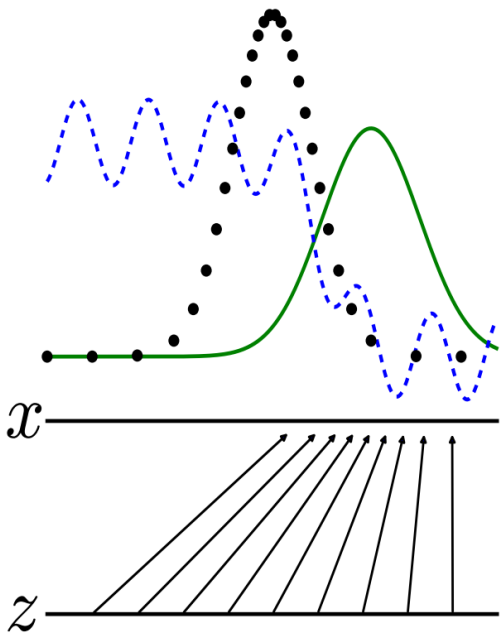
- Both G and D are neural networks.
- Use SGD-like algorithm (e.g., Adam) on two minibatches simultaneously.
  - A minibatch of training examples.
  - A minibatch of generated samples.
  - By iterative optimizing D and G by SGD-like algorithms, we can approximately solve the minimax objective.
    - The objective for training D is

$$\max_D \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_r}[\log D(\mathbf{x})] + \mathbb{E}_{\tilde{\mathbf{x}}\sim\mathbb{P}_g}[\log(1 - D(\tilde{\mathbf{x}}))].$$

    - The objective for training G is

$$\min_G \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_g}[\log(1 - D(\tilde{\mathbf{x}}))].$$

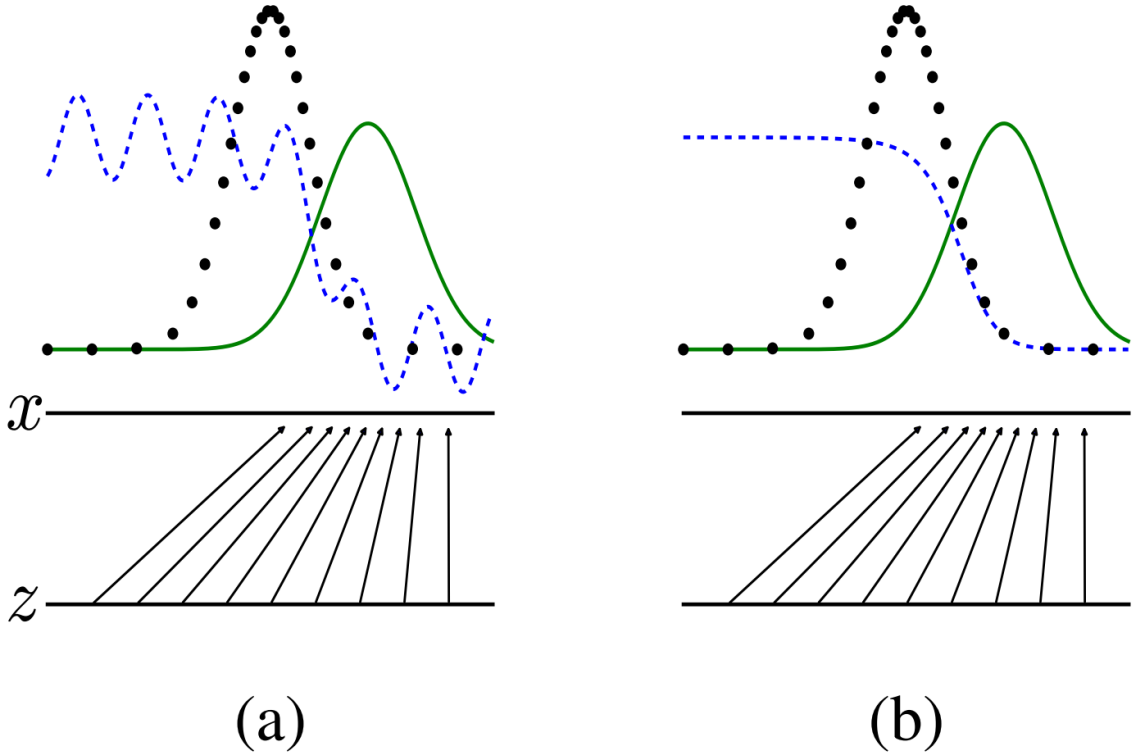
Training of GAN



(a)

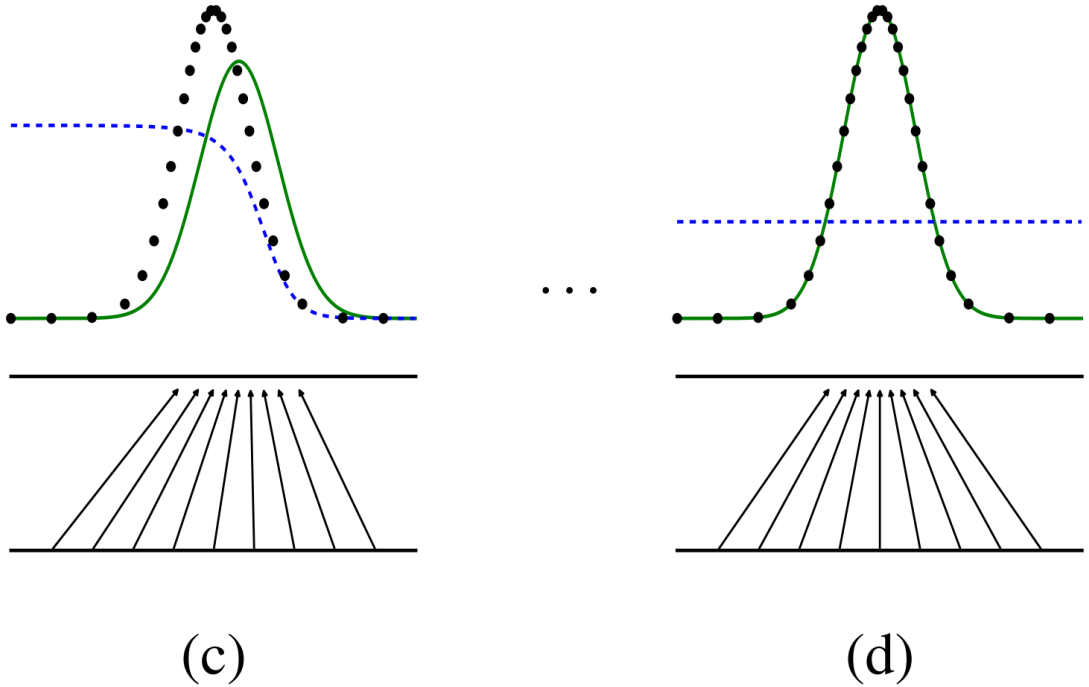
- GAN is trained by simultaneously updating the discriminative distribution (D, blue, dashed line) so that it discriminates between samples from the data generating distribution (black, dotted line) $\mathbb{P}_r$ from those of the generative distribution $\mathbb{P}_g$ (green, solid line).
- The lower horizontal line is the domain from which $\mathbf{z}$ is sampled, in this case uniformly. The horizontal line above is part of the domain of $\mathbf{x}$.
- The upward arrows show how the mapping $\mathbf{x} = G(\mathbf{z})$ imposes the non-uniform distribution $\mathbb{P}_g$ on transformed samples.

Training of GAN



- (a) Consider an adversarial pair near convergence: $\mathbb{P}_g$ is similar to $\mathbb{P}_r$ and D is a partially accurate classifier.
- (b) In the step of optimizing D, it is trained to discriminate samples from data, converging to $\frac{\mathbb{P}_r(\mathbf{x})}{\mathbb{P}_r(\mathbf{x})+\mathbb{P}_g(\mathbf{x})}$.

Training of GAN



- (c) After an update of G, gradients of D have guided $G(\mathbf{z})$ to flow to regions that are more likely to be classified as data.
- (d) After several steps of training, if G and D have enough capacity, they will reach a point at which both cannot improve because $\mathbb{P}_g = \mathbb{P}_r$. The discriminator is unable to differentiate between the two distributions, i.e. $D(\mathbf{x}) = \frac{1}{2}$.

Heuristic Training Objective

- The original minimax objective results in the following optimization objective for G

$$\min_G \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_g}[\log(1 - D(\tilde{\mathbf{x}}))].$$

- The modified objective for G advocated by Goodfellow et al. (2014) is

$$\min_G -\mathbb{E}_{\mathbf{x}\sim\mathbb{P}_g}[\log(D(\tilde{\mathbf{x}}))].$$

- The motivation is that the original minimax objective leads to vanishing gradients as the discriminator saturates.

Vanishing Gradient

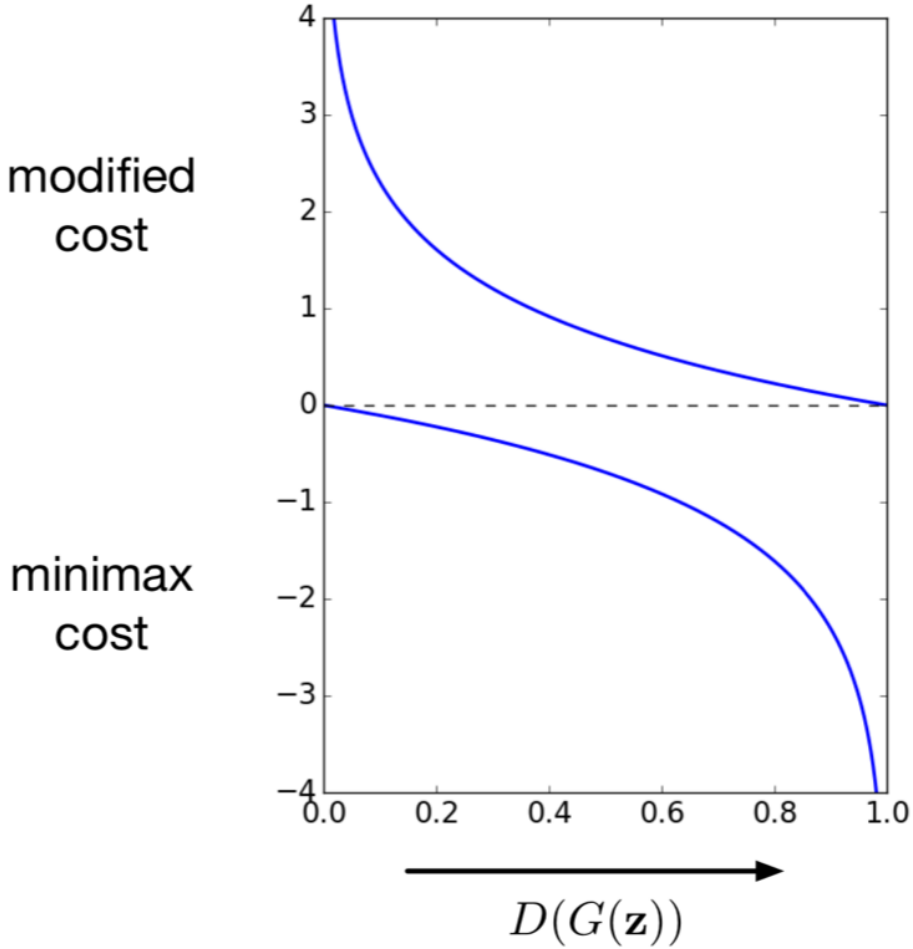
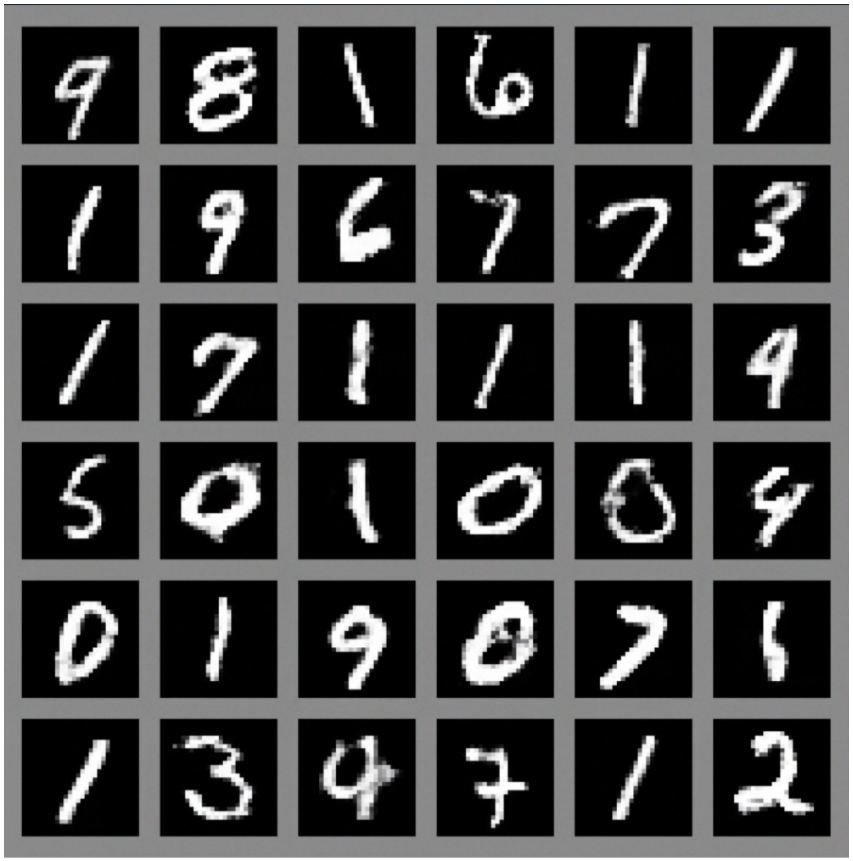
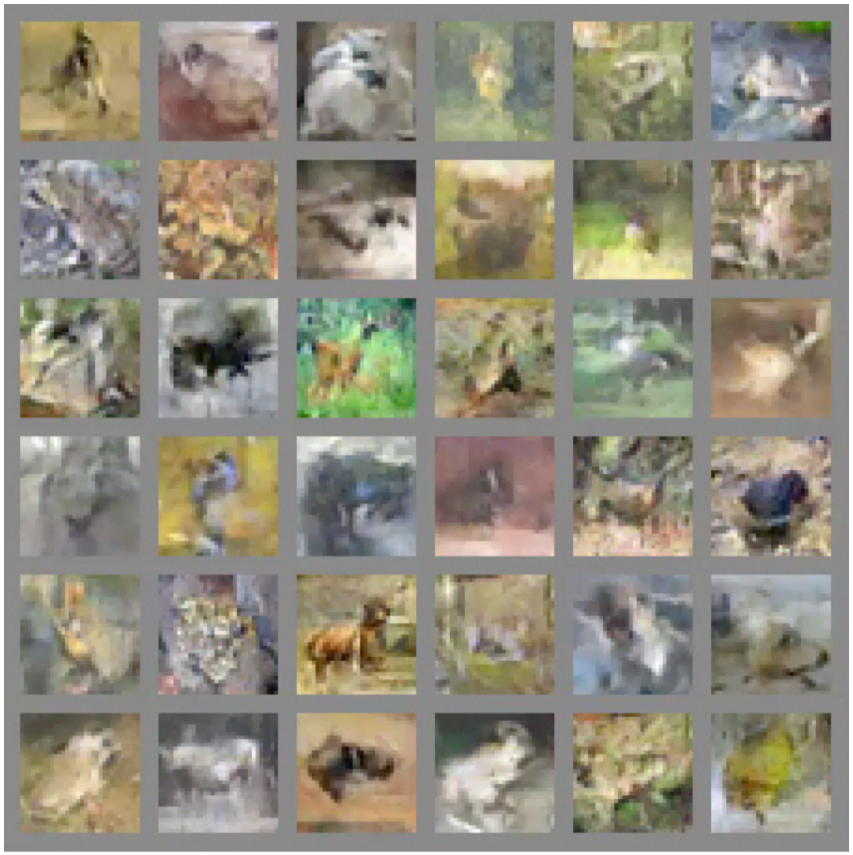


Figure 4: The curve of loss with respect to $D(G(\mathbf{z}))$. The minimax objective's gradients for G converge to 0 when the discriminator is nearly perfect, which makes it hard to learn the generator.

Samples from GAN



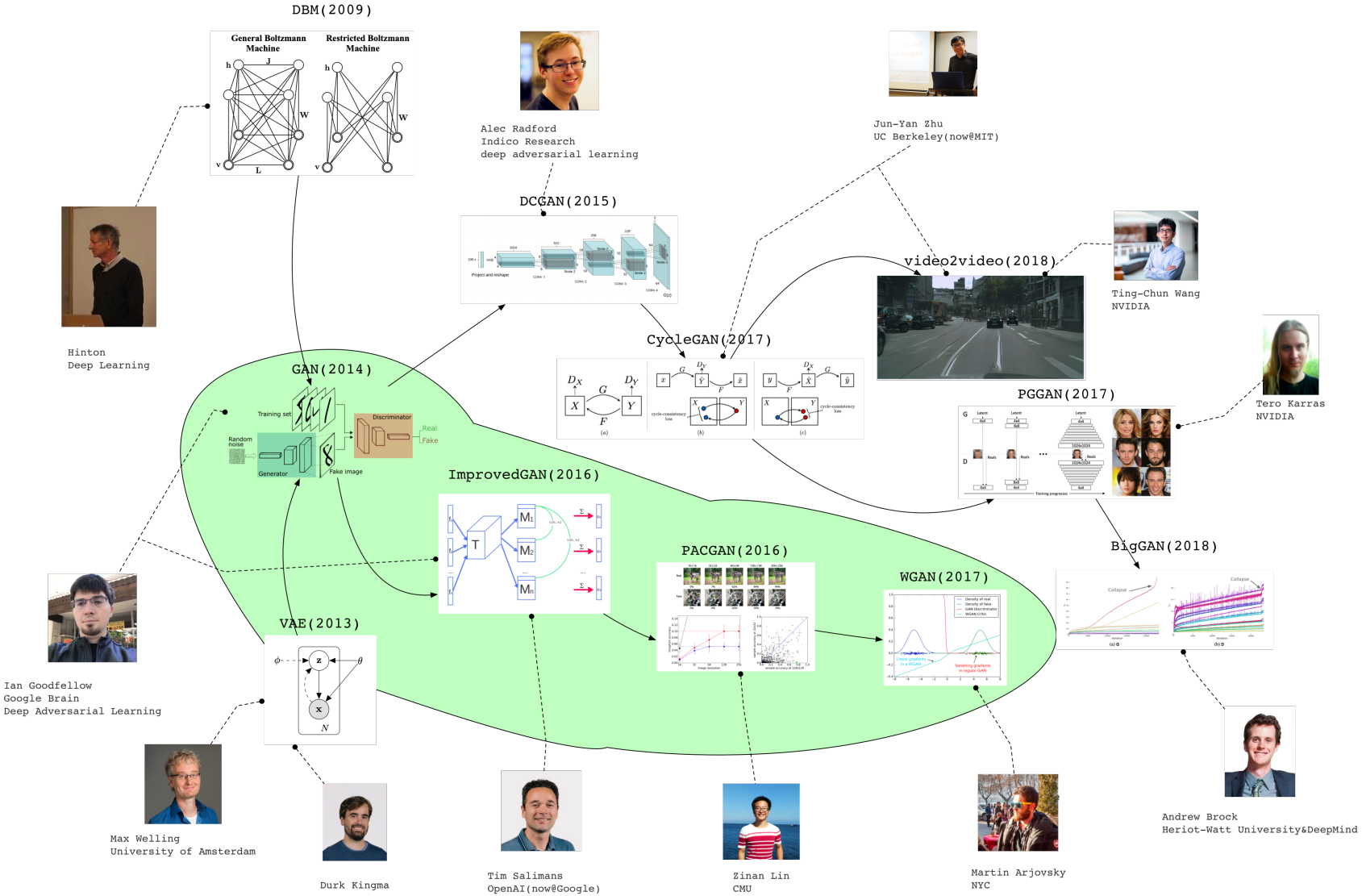
MNIST



CIFAR-10

Outline

Toward Improving the Training of GANs



Mode Collapse

- A common problem of iterative optimization is that $\min_G \max_D V(G, D) \neq \max_D \min_G V(G, D)$.
- When D is in the inner loop, the optimizer converges to correct distribution.
- When G is in the inner loop, the optimizer places all mass on most likely point.
- Mode collapse is an empirically observed phenomenon that distribution of $G(\mathbf{z})$ is likely to fail to capture all modes of $\mathbb{P}_r$.

Mode Collapse

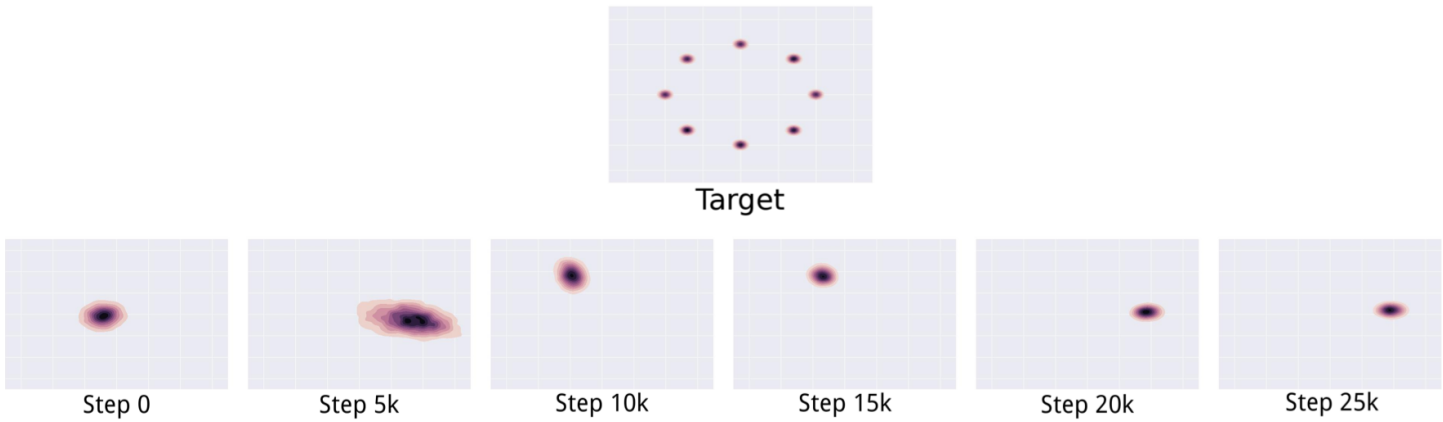


Figure 5: Illustration of mode collapse. The figures at the bottom line are results of GAN. During all the six training steps, GAN fails to capture all 9 modes. In fact, GAN only recover one mode. The mode captured drifts all the time.

Unrolled GAN

- Unrolled GAN[1] changes the way GAN is training to mitigating the pain of mode collapse.
- Main idea: The generator G fails to fool D. Giving G an edge in the minimax game may help.
- Unrolled GAN prevents mode collapse by backpropagating through a set of k updates of the discriminator to update generator parameters once, so G's chance of overfitting to a specific D reduces.
- Though k updates of D are used for the update of G, only the first update of D is kept for next round's training. So D's chance of overfitting to a specific G reduces.

[1] Metz, Luke, et al. "Unrolled generative adversarial networks." arXiv preprint arXiv:1611.02163 (2016).

Unrolled GAN

- The parameter update formula of SGD algorithm on the original GAN is (A single sample is used for simplicity of illustration.)

$$\begin{aligned} \theta_D &= \theta_D + \eta \frac{\partial L(\theta_D, \theta_G)}{\theta_D}, \\ \theta_G &= \theta_G - \eta \frac{\partial L(\theta_D, \theta_G)}{\theta_G}. \end{aligned} \tag{4}$$

- The parameter update formula of SGD algorithm on unrolled GAN 's discriminator is the same as the original GAN is

$$\theta_D = \theta_D + \eta \frac{\partial L(\theta_D, \theta_G)}{\theta_D}. \tag{5}$$

Unrolled GAN

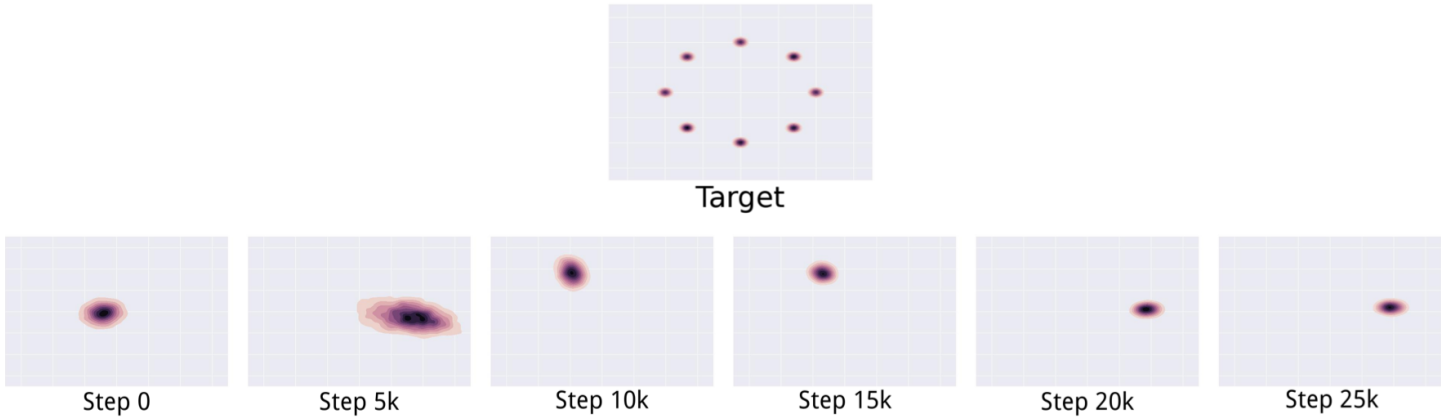
- Unrolled GAN's generator's training signal comes from k updates of the its discriminator. Thus the update formula is

$$\begin{aligned} \theta_D^0 &= \theta_D, \\ \theta_D^{k+1} &= \theta_D^k + \eta^k \frac{\partial L(\theta_D^k, \theta_G)}{\theta_D^k}, \\ \theta_G &= \theta_G - \eta \frac{\partial L_K(\theta_D, \theta_G)}{\theta_G}, \end{aligned} \tag{6}$$

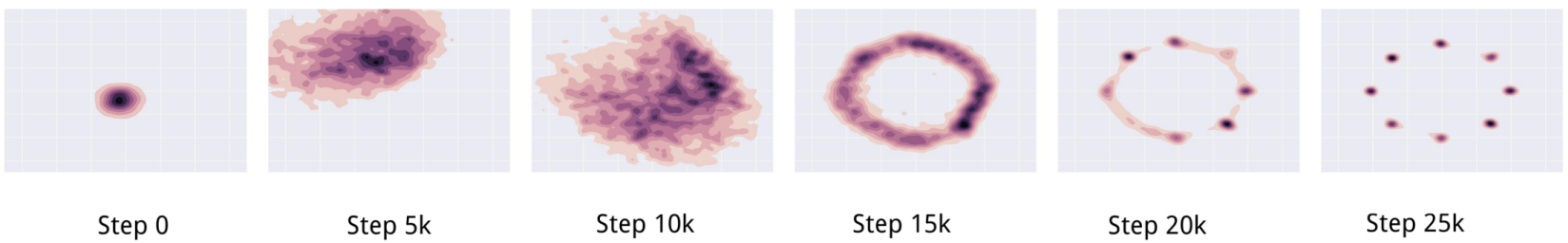
  where $L_K(\theta_D, \theta_G) := L(\theta_D^K, \theta_G)$ is the surrogate loss for the generator.

Unrolled GAN's Result

Recall the mode collapse of GAN.



Unrolled GAN captures all the modes finally.

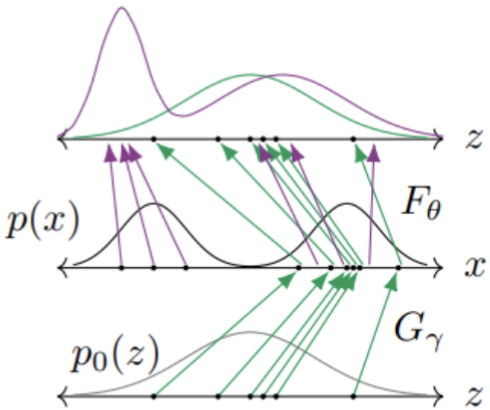


VEEGAN

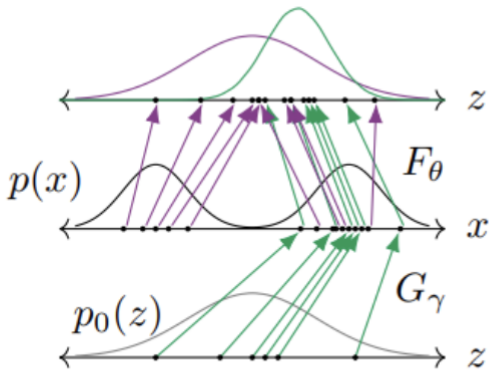
- VEEGAN[2] tries to mitigate problem of mode collapse by introducing a reconstruction process.
- VEEGAN introduces a reconstructor network which is learned both to map the true data distribution $\mathbb{P}_r$ to a Gaussian and to approximately invert the generator network G.
- We assume that F is parameterized by θ and G is parameterized by γ, i.e., $F = F_\theta$ and $G = G_\gamma$.
- Formally, the role of reconstructor network F_θ is to
  - map the data distribution to a Gaussian, i.e., $F_\theta(\mathbf{x}) \approx N(0, I)$, where $\mathbf{x} \sim \mathbb{P}_r$;
  - approximately reverse the action of the generator, i.e., $F_\theta(G_\gamma(\mathbf{z})) \approx \mathbf{z}$.

[2] Srivastava, Akash, et al. "Veegan: Reducing mode collapse in gans using implicit variational learning." Advances in Neural Information Processing Systems. 2017.

VEEGAN



(a) Suppose F_θ is trained to approximately invert G_γ. Then applying F_θ to true data is likely to produce a non-Gaussian distribution, allowing us to detect mode collapse.



(b) When F_θ is trained to map the data to a Gaussian distribution, then treating $F_\theta \circ G_\gamma$ as an autoencoder provides learning signal to correct $G\gamma$.

Figure 6: How a reconstructor network F_θ helps detect mode collapse and correct generator. If the same reconstructor network maps both the true data and the generated data to a Gaussian distribution, then the generated data is likely to coincide with true data.

VEEGAN

- Mathematically, to measure whether F_θ approximately inverts G_γ, a L_2 loss between $\mathbf{z}$ and $F_\theta(G_\gamma(\mathbf{z}))$ is used, the same as the loss of autoencoder.
- Mathematically, to quantify whether F_θ maps the true data distribution to a Gaussian, the cross entropy loss $H(\mathbf{z}, F_\theta(\mathbf{x}))$ is used.
- The two losses combined leads to

$$L_{F,G} = \mathbb{E}[\|\mathbf{z} - F_\theta(G_\gamma(\mathbf{z}))\|_2^2] + H(\mathbf{z}, F_\theta(\mathbf{x})). \tag{7}$$

VEEGAN's Objective

- The second part of the loss $L_{F,G}$ cannot be easily computed and minimized. We can introduce variational bound to derive an easier surrogate loss.
- Denote standard Gaussian by $p_0 = N(0, I)$ and the distribution of $F_\theta(\mathbf{x})$ by $p_\theta(\mathbf{z})$. We have

$$\begin{aligned} H(\mathbf{z}, F_\theta(\mathbf{x})) &= -\int p_0(\mathbf{z}) \log p_\theta(\mathbf{z}) d\mathbf{z} \\ &= -\int p_0(\mathbf{z}) \log \int p(\mathbf{x}) p_\theta(\mathbf{z}|\mathbf{x}) d\mathbf{x} d\mathbf{z} \\ &\leq \mathbf{KL}[q_\gamma(\mathbf{x}|z) p_0(\mathbf{z}) || p_\theta(\mathbf{z}|x) p(\mathbf{x})] - \mathbb{E}[\log p_0(\mathbf{z})]. \end{aligned} \tag{8}$$

  where expectation is taken w.r.t. the joint distribution $p_0(\mathbf{z}) q_\gamma(\mathbf{x}|\mathbf{z})$ and $q_\gamma(\mathbf{x}|\mathbf{z})$ is the introduced variational distribution parameterized by the network G_γ.

VEEGAN's Objective

- Even the surrogate objective given by variational upper bound is hard to optimize, since q_γ is implicitly defined by a stochastic network and $p(\mathbf{x})$ is unknown.
- A discriminator network $D_w(\mathbf{x}, \mathbf{z})$ is introduced to be forced the value of the density ratio $D_w(\mathbf{x}, \mathbf{z}) = \log \frac{q_\gamma(\mathbf{x}|\mathbf{z})p_0(\mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x})p(\mathbf{x})}$.
- This requirement can be satisfied by the following objective for the discriminator $-\mathbb{E}_\gamma[\log(\sigma(D_w(\mathbf{x}, \mathbf{z})))] - \mathbb{E}_\theta[\log(1 - \sigma(D_w(\mathbf{x}, \mathbf{z})))]$, where σ is the logistic function. (It can be easily derived through the optimal condition of D in GAN's theory.)
- Thus, by sampling we can estimate the surrogate objective by

$$\frac{1}{N}\sum_{i=1}^{N} D_w(\mathbf{z}^i, \mathbf{x}_g^i) + \frac{1}{N}\sum_{i=1}^{N} d(\mathbf{z}^i, \mathbf{x}_g^i).$$

where $(\mathbf{z}^i, \mathbf{x}_g^i) \sim p_0(\mathbf{z})q_\gamma(\mathbf{x}|\mathbf{z})$.

VEEGAN's Result

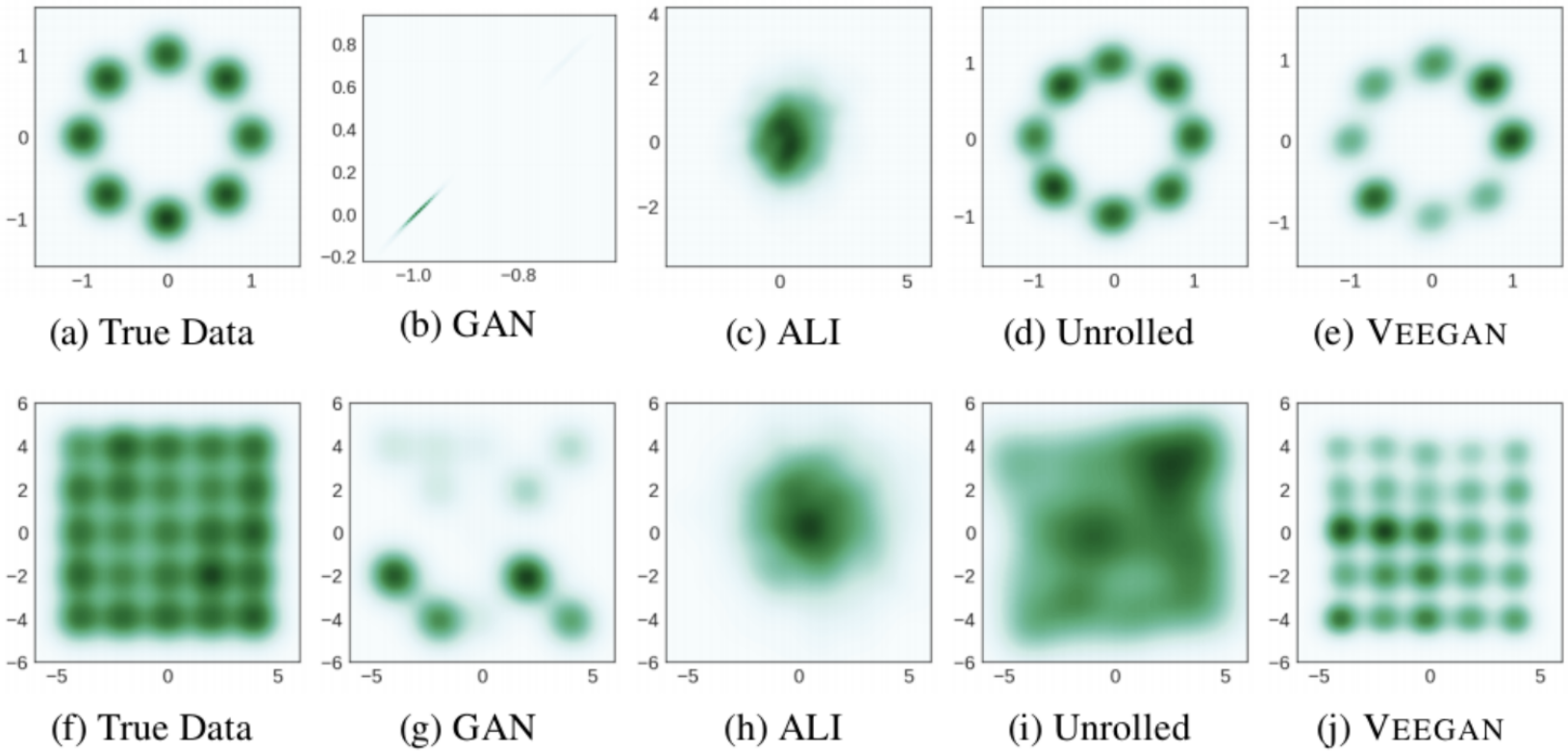


Figure 7: Comparison of some GANs' results. VEEGAN performs best at the bottom line.

PACGAN

- PACGAN[3] eases the pain of mode collapse by changing input to the discriminator.
- Assumption: one of the main reasons GANs suffer from mode collapse is because most of the popular GANs unnecessarily restrict the discriminator to be a function of a single input.
- In other words, diversity (or lack of diversity) in the generated samples is easier to detect if the discriminator is allowed to make the decision based on multiple samples jointly.
- The idea of packing: show multiple samples simultaneously instead of one as the discriminator's input.
- A theoretical result: packed GAN naturally penalizes the generator that exhibits strong mode collapse.
- Note that we need to increase the input layer of the discriminator by a factor of m if we are packing m samples.

[3] Lin, Zinan, et al. "Pacgan: The power of two samples in generative adversarial networks." Advances in Neural Information Processing Systems. 2018.

Comparison of GAN and PACGAN

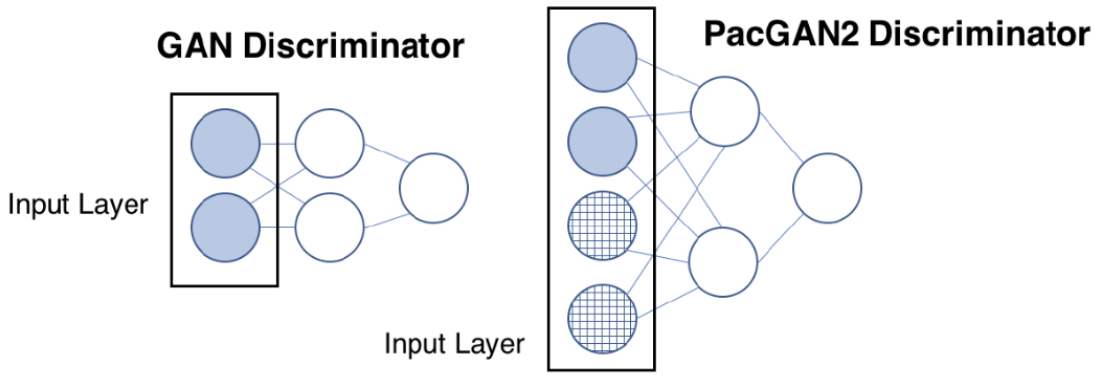


Figure 8: GAN sees one picture each time while PACGAN sees two. It's also possible to use even more pictures as the input.

PACGAN's Result

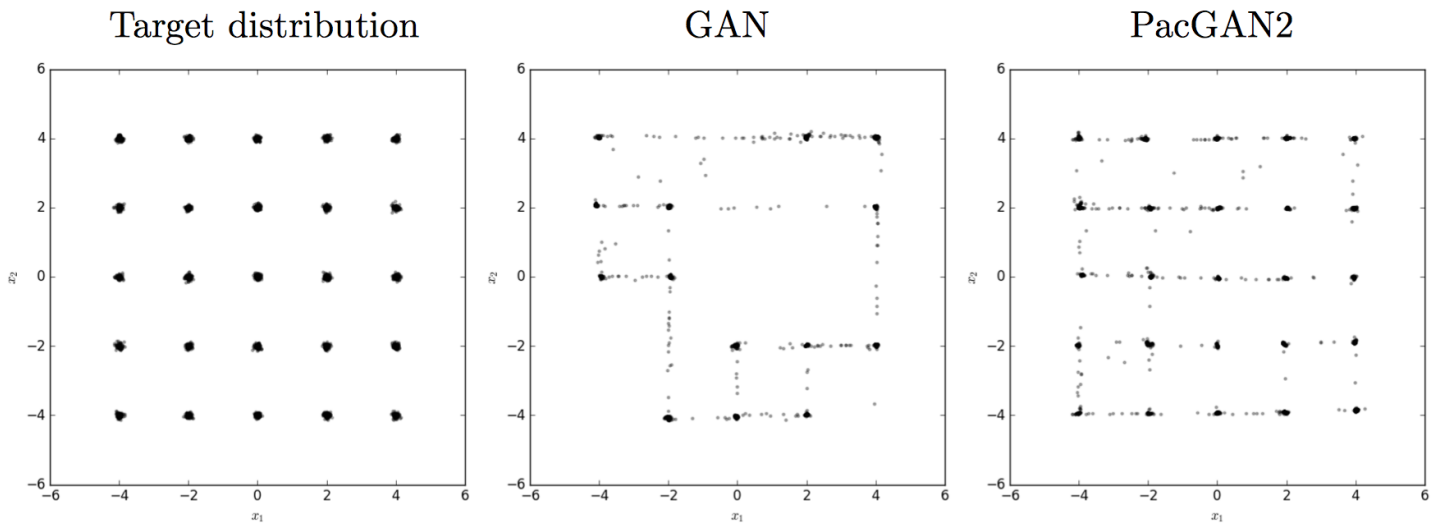


Figure 9: PACGAN captures all modes while GAN fails most of them.

Semi-supervised GAN

- When partial labels exist, it is possible and beneficial to use this additional information in the training of GANs.
- Semi-supervised GAN[4] uses additional partial labels to help the training of GAN.
- The idea: one can always add an additional class "Fake" to the original set of classes, turning the classifier into a discriminator.
- Given unlabeled data(real or generated), semi-supervised GAN trains GAN as before. Given labeled data, it could be trained to recognize the individual real class.
- Very few labels turn out to be quite helpful.

[4] Salimans, Tim, et al. "Improved techniques for training gans." Advances in Neural Information Processing Systems. 2016.

Semi-supervised GAN's Objective

- Recall that GAN's loss for the discriminator is

$$-\mathbb{E}_{\mathbf{x}\sim\mathbb{P}_r}[\log D(\mathbf{x})] - \mathbb{E}_{\tilde{\mathbf{x}}\sim\mathbb{P}_g}[\log(1 - D(\tilde{\mathbf{x}}))].$$

- Semi-supervised GAN's loss for the discriminator is defined as

$$\mathbb{E}_{\mathbf{x},y\sim\mathbb{P}_r}[\log \mathbb{P}_D(y|\mathbf{x})] - \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_g}[\log \mathbb{P}_D(y = K + 1|\mathbf{x})],$$

  where $\mathbb{P}_D(y = K + 1|\mathbf{x}) = 1 - D(\mathbf{x})$ is the probability of the sample being fake.

Semi-supervised GAN's Objective

- We can decompose this loss into a supervised loss and an unsupervised loss by

$$L_{\text{supervised}} = -\mathbb{E}_{\mathbf{x},y\sim\mathbb{P}_r} \log P_D(y|\mathbf{x}, y < K+1), \tag{9}$$

$$L_{\text{unsupervised}} = -\mathbb{E}_{\mathbf{x}\sim\mathbb{P}_r}[\log \mathbb{P}_D(y = K+1|\mathbf{x})] \tag{10}$$

$$-\mathbb{E}_{\tilde{\mathbf{x}}\sim\mathbb{P}_g}[\log(1 - \mathbb{P}_D(y = K+1|\tilde{\mathbf{x}}))]. \tag{11}$$

- In this decomposition, $L_{\text{unsupervised}}$ is exactly the same as the original GAN's loss for discriminator, and what is the additional supervision is clear.

Semi-supervised GAN

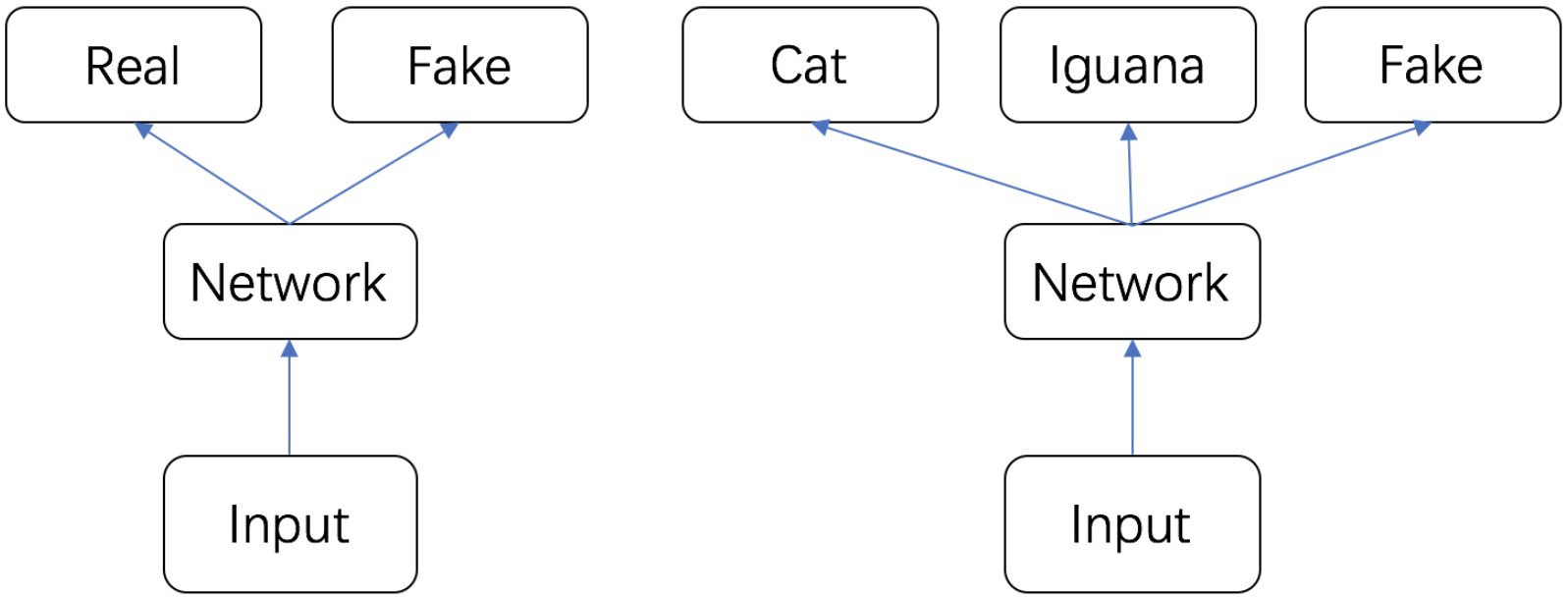


Figure 10: Left figure illustrates GAN's discriminator. Right figure illustrates Semi-supervised GAN's discriminator.

Semi-supervised GAN

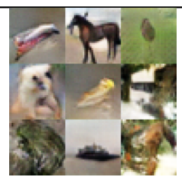
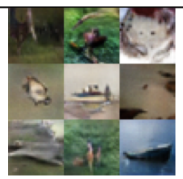
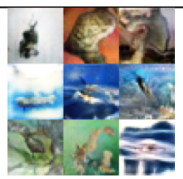
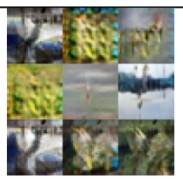
Samples							
Model	Real data	Our methods	-VBN+BN	-L+HA	-LS	-L	-MBF
Score $\pm$ std.	$11.24 \pm .12$	$8.09 \pm .07$	$7.54 \pm .07$	$6.86 \pm .06$	$6.83 \pm .06$	$4.36 \pm .04$	$3.87 \pm .03$

Figure 11: The result reported by Salimans et al 2016. Pay attention to the "-L" result versus "our model".

Feature Matching

- Feature matching[5] is another technique to ease the pain of model collapse. It defines a loss term which is a L_2 norm between the feature(the output of some middle layer of the discriminator) of generated images and that of real ones.
- Feature matching training objective of GAN
  - Let $f(\mathbf{x})$ denote activations on an intermediate layer of the discriminator.
  - Recall in the original GAN, the training objective for the generator is
$$\min_G \mathbb{E}_{\mathbf{z}\sim\mathbb{P}_z}[\log(1 - D(G(\mathbf{z})))].$$
  - By feature matching, the generator's objective is redefined as
$$\|\mathbb{E}_{\mathbf{x}\sim\mathbb{P}_r} f(\mathbf{x}) - \mathbb{E}_{\mathbf{z}\sim\mathbb{P}_z} f(G(\mathbf{z}))\|_2^2.$$
  - The loss has transformed from log-loss to L_2 loss and it's the statistics of the real data to be matched by the output of the generator.
  - The discriminator's objective remains unchanged.

[5] Gulrajani, Ishaan, et al. "Improved training of wasserstein gans." Advances in Neural Information Processing Systems. 2017.

WGAN

- WGAN[6] investigates the nature of the original GAN's objective and proposes a more informative alternative objective based on its findings.
- Recall that during the training of GAN, under an ideal discriminator, the generator is expected to minimizes the JS divergence between $\mathbb{P}_r$ and $\mathbb{P}_g$, i.e.,

$$JS(\mathbb{P}_r||\mathbb{P}_g) = KL(\mathbb{P}_r||\frac{\mathbb{P}_r + \mathbb{P}_g}{2}) + KL(\mathbb{P}_g||\frac{\mathbb{P}_r + \mathbb{P}_g}{2}), \quad (12)$$

  where KL denote the K-L divergence.
- There is one problem related to this formulation of loss. JS loss fails to provide non-zero gradient for generator sometimes.

[6] Arjovsky, Martin, Soumith Chintala, and Lon Bottou. "Wasserstein generative adversarial networks." International Conference on Machine Learning. 2017.

Problem of JS divergence

- See the figure below. In R^2 space, when $p(\mathbf{x})$ is the uniform distribution on the segment AB and $q(\mathbf{x})$ is the uniform distribution on the segment CD, what is the JS divergence between p and q?
- By some calculation, one can show:

$$JS(p||q) = \begin{cases} \log 2, & \theta \neq 0 \\ 0, & \theta = 0 \end{cases}$$

- A constant JS divergence leads to 0 gradient which makes it impossible for q to find a reasonable direction to converge to p.

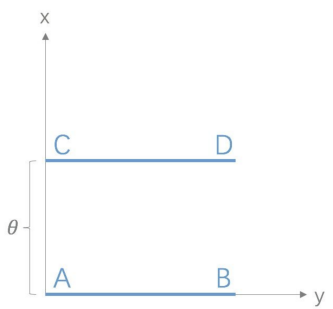


Figure 12: An example showing two uniform distribution on R^2

Discontinuity of JS Divergence

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

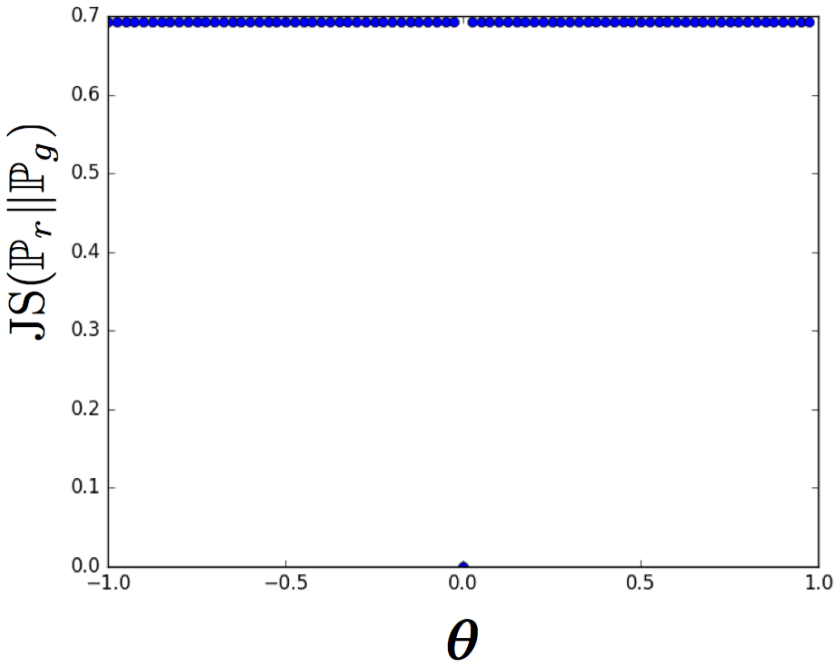


Figure 13: A constant JS divergence makes it hard to train generator.

Problem of JS Divergence

- The intuition behind this phenomenon of zero gradient is that the two distributions have no overlap with each other thus a constant JS divergence.
- A generative model's basic assumption is that there exist a low dimensional representation for the real data in the high dimensional space, i.e., a manifold in the high dimensional space. And we are interested in learning the low dimensional representation.
- In the real data's high dimensional space, the probability of two low dimensional curve having "significant" overlap is 0. Therefore, JS divergence is nearly always an insufficient signal to train GANs.
- Can we find a better measurement?

Wasserstein-1 Distance

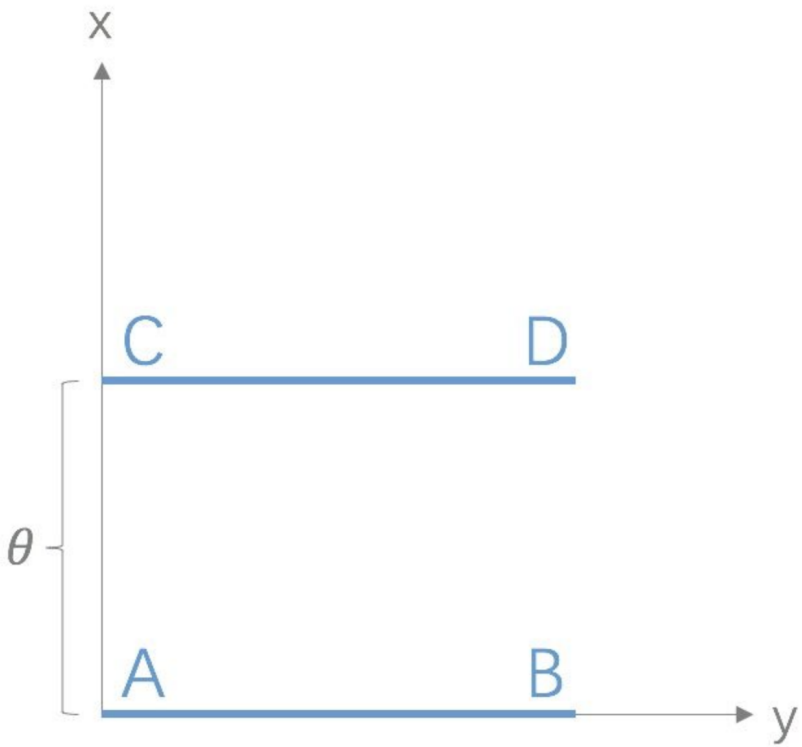
- An alternative for JS divergence is the Earth-Mover (also called Wasserstein-1) distance

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|]. \tag{13}$$

- One can view EM distance from $\mathbb{P}_r$ to $\mathbb{P}_g$ as the minimum cost of transporting mass to transform the distribution $\mathbb{P}_r$ into the distribution $\mathbb{P}_g$.
- Under mild condition, EM distance is continuous everywhere and differentiable almost everywhere.

Wasserstein-1 Distance

- $\mathbb{P}_r$ and $\mathbb{P}_g$ are uniformly distributed on segment AB and CD respectively.



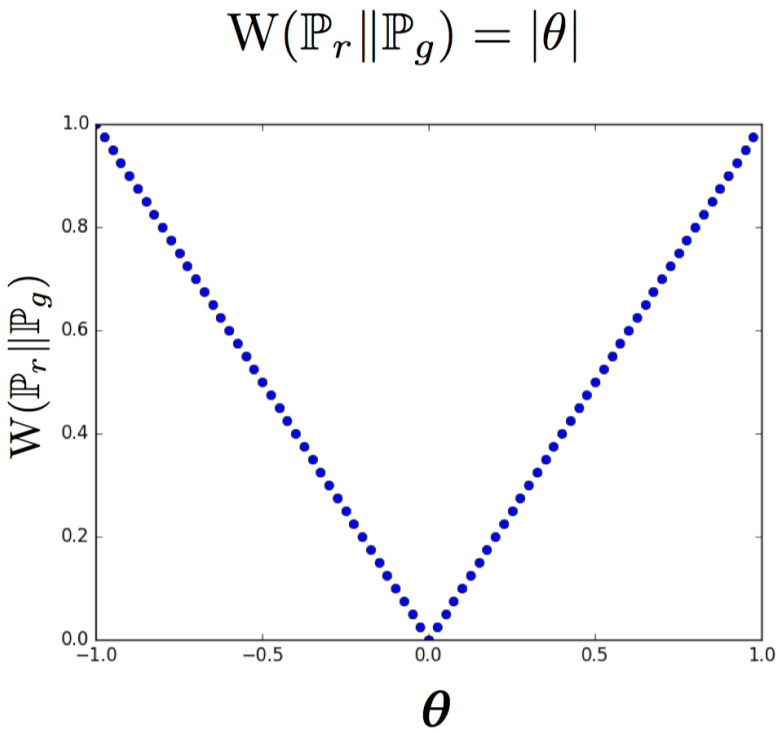


Figure 14: The Wasserstein-1 distance between uniform distributions on AB and CD.

WGAN

- Recall Wasserstein-1 distance's definition is

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|]. \tag{14}$$

- $W(\mathbb{P}_r, \mathbb{P}_g)$ might have nicer properties than $JS(\mathbb{P}_r, \mathbb{P}_g)$.
- However, the infimum operator is intractable.
- By Kantorovich-Rubinstein duality, it can be transformed into

$$W(\mathbb{P}_r, \mathbb{P}_g) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_g}[f(x)], \tag{15}$$

  where the supremum operator is over all the 1-Lipschitz functions $f : \mathcal{X} \mapsto \mathbb{R}$.
- Then, we can write down the WGAN's minimax objective

$$\min_G \max_{D\in\mathcal{D}} \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_r}[D(\mathbf{x})] - \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_g}[D(\tilde{\mathbf{x}})], \tag{16}$$

  where $\mathcal{D}$ is the set of all 1-Lipshitz functions.

WGAN's Objective

- WGAN's objective is

$$\min_{G} \max_{D \in \mathcal{D}} \mathbb{E}_{\mathbf{x} \sim \mathbb{P}_r}[D(\mathbf{x})] - \mathbb{E}_{\mathbf{x} \sim \mathbb{P}_g}[D(\tilde{\mathbf{x}})]. \tag{17}$$

  where $\mathcal{D}$ is the set of all 1-Lipshitz functions.
- We can use neural networks to approximately represent the function set $\mathcal{D}$, thus make it possible to use BP to learn the discriminator.
- Under this objective, discriminator has changed its role from classifying real/generated data into maximize the distance between real/generated distribution.
- One open question is how to enforce the Lipschitz continuity constraint on neural networks.
  - Arjovsky et al. (2017) proposed to clip the weights of the critic to lie within a compact space $[-c, c]$.
  - This strategy results in a subset of the set of all the k-Lipschitz functions where k is a function of c.

Gradient Penalty

- There are two problems related to the clipping weights strategy.
  - The model's full capacity is restricted.
  - Gradients seems to either explode or vanishing during experiment.
- A property of the optimal WGAN critic $f^*(\mathbf{x})$ can be proved. With probability 1, for $\tilde{\mathbf{x}} \sim \mathbb{P}_g$ and $\mathbf{x} \sim \mathbb{P}_r$, for all points $\mathbf{x}_t$ on the straight line between $\mathbf{x}$ and $\tilde{\mathbf{x}}$, we have

$$\bigtriangledown f^*(\mathbf{x}_t) = \frac{\mathbf{x} - \mathbf{x}_t}{\|\mathbf{x} - \mathbf{x}_t\|}. \tag{18}$$

  Therefore, the optimal WGAN critic has gradient norm 1 at $\mathbf{x}_t$.

WGAN-GP

- WGAN-GP[7] adds a regularizer to the WGAN's objective according to the fixed norm property at its critic, which leads to

$$L = \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_r}[D(\mathbf{x})] - \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_g}[D(\tilde{\mathbf{x}})] + \lambda\mathbb{E}_{\hat{\mathbf{x}}}[(\bigtriangledown_{\|\hat{x}\|}D(\hat{x})^2 - 1)]^2, \tag{19}$$

where $\hat{\mathbf{x}} = \xi\mathbf{x} + (1-\xi)\tilde{\mathbf{x}}$, $\xi \sim U(0,1)$, $\mathbf{x} \sim \mathbb{P}_r$, $\tilde{\mathbf{x}} \sim \mathbb{P}_g$.

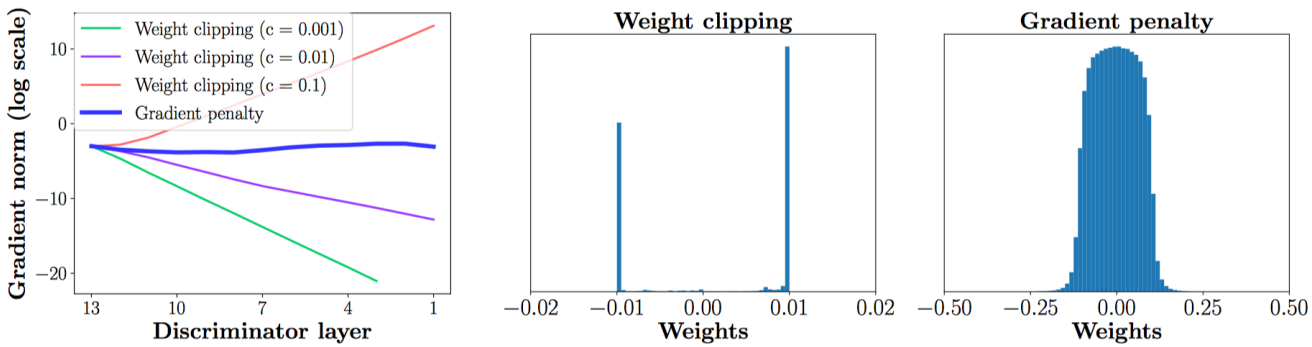
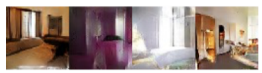
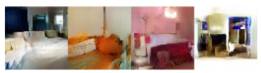
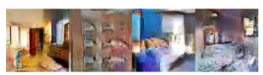
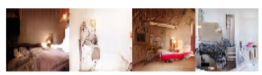
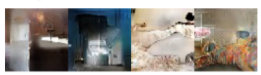
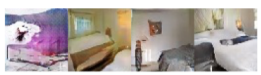
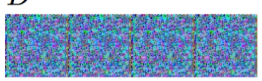
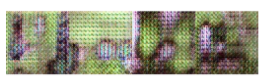


Figure 15: Different weight's norm of WGAN with weight clipping and WGAN-GP. WGAN-GP is more robust.

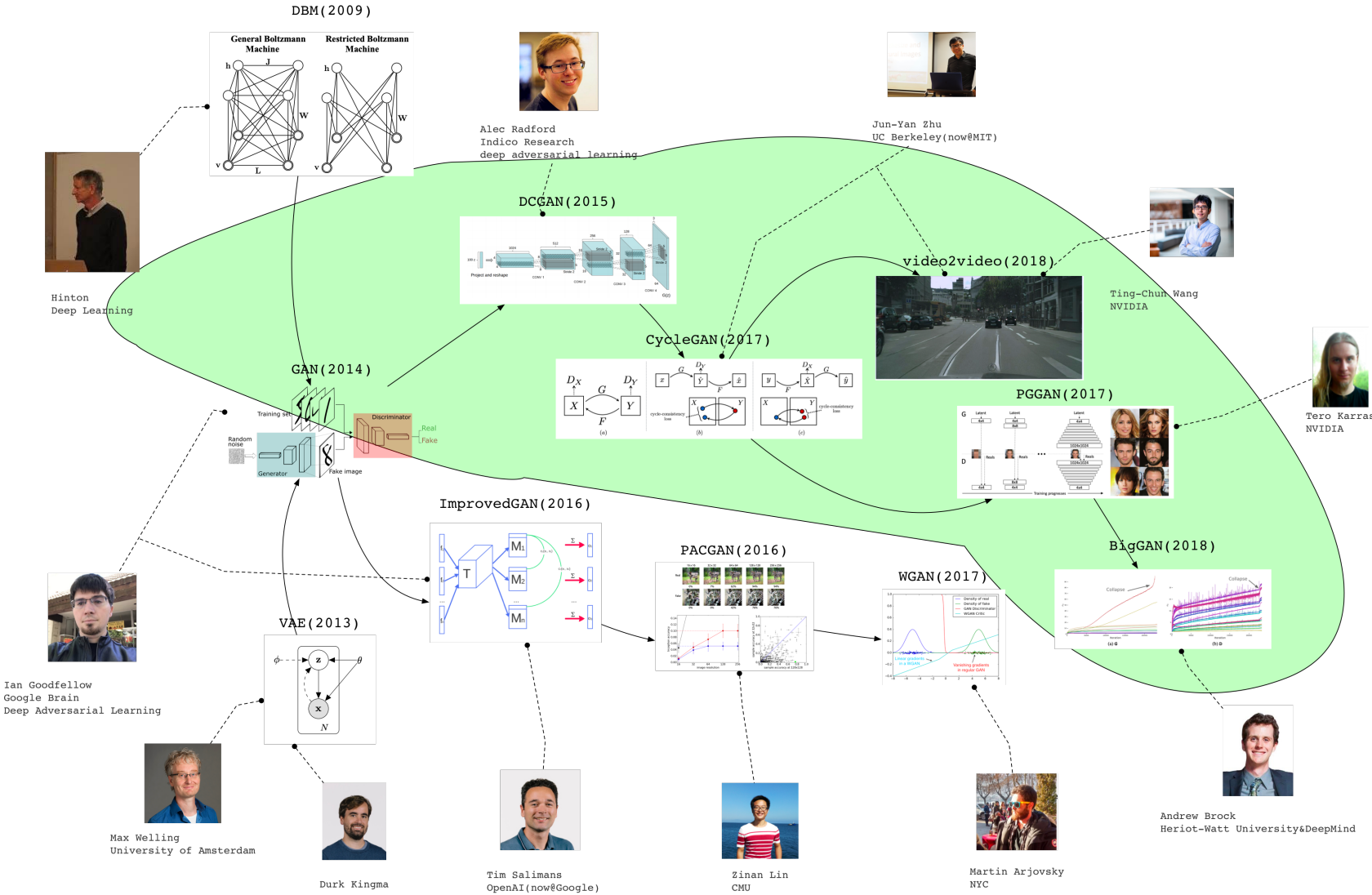
[7] Gulrajani, Ishaan, et al. "Improved training of wasserstein gans." Advances in Neural Information Processing Systems. 2017.

Empirical Results

- WGAN-GP is more robust to variations in training setups.

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				

Now we are going to introduce several applications



CycleGAN

- Image-to-Image translation aims to learn the mapping between an input image and an output image using a training set of aligned image pairs.
- To put it formally, the goal of image-to-image translation is to learn a mapping $G : X \mapsto Y$ such that the distribution of images $G(X)$ is indistinguishable from the distribution Y.
- In practice, it is rare that many aligned image pairs are available.
- CycleGAN[8] proposed a framework to learn transformation across domains with unpaired data.

[8] Zhu, Jun-Yan, et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks." Proceedings of the IEEE International Conference on Computer Vision. 2017.

CycleGAN

- Though we lack supervision in the form of paired training data, we can still exploit supervision at the level of sets, i.e., to learn the mapping G that maps domain X to target domain Y.
- In this formulation, we can learn the mapping G in GAN's framework through

$$L_{GAN}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{data}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{data}}[\log(1 - D_Y(G(x)))].$$

- There are two problems of this formulation.
  - There's no guarantee of a meaningful individual x's transformation.
  - It has been found hard to optimize the adversarial objective in isolation in practice.

CycleGAN

- Adding more structure to the objective relieves the issues above.
- Cycle consistency: If we have $G : X \mapsto Y$ and $F : Y \mapsto X$ as two translators, G, F should be bijections and inverses of each other. Formally speaking, we have $F(G(x)) = x$ and $G(F(x)) = x$.
- CycleGAN's loss has two parts.
  - The Adversarial loss: $L_{GAN}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{data}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{data}}[\log(1 - D_Y(G(x)))]$, where G is the translator from X to Y and D_Y is the discriminator.
  - A similar loss is defined for $F : Y \mapsto X$.
  - Cycle Consistency Loss:

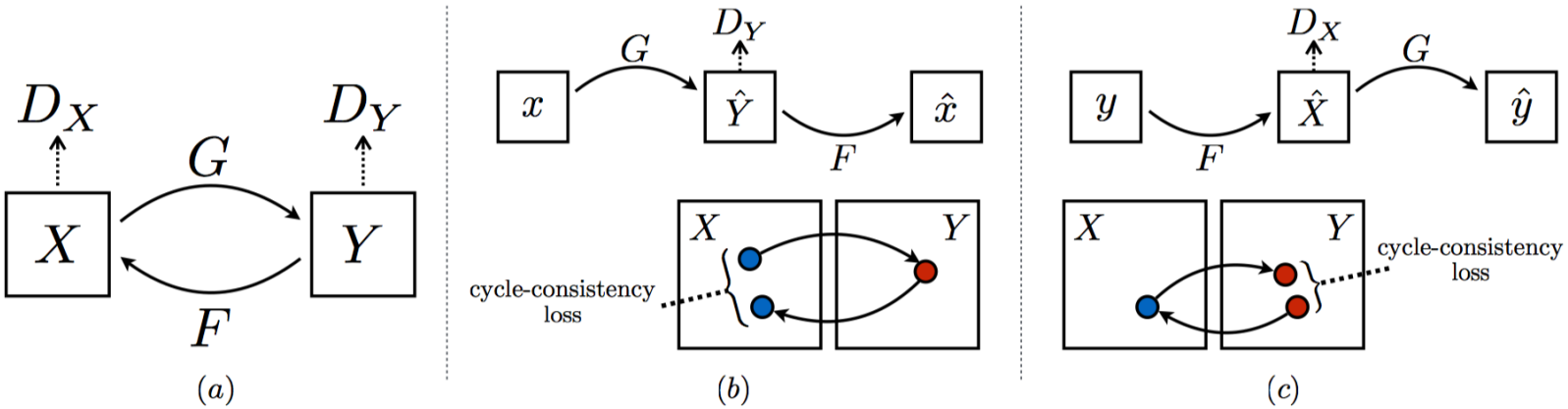
$$L_{cyc}(G, F) = \mathbb{E}_{y \sim p_{data}(y)}[\|G(F(y)) - y\|_1] + \mathbb{E}_{x \sim p_{data}}[\|F(G(x)) - x\|_1].$$

CycleGAN

- Full loss is three terms combined, i.e.,

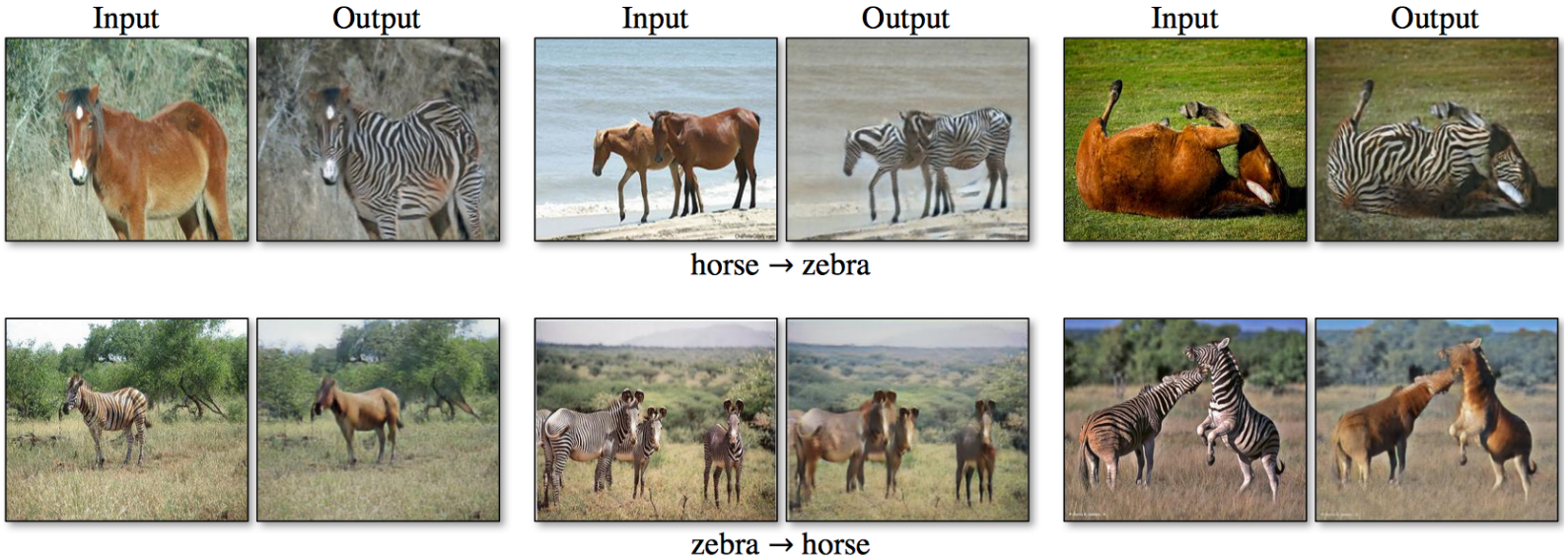
$$\begin{aligned} L(G, F, D_X, D_Y) = & L_{GAN}(G, D_Y, X, Y) \\ & + L_{GAN}(F, D_X, Y, X) \\ & + L_{cyc}(G, F). \end{aligned} \tag{20}$$

- the minimax objective is $\min_{G,F} \max_{D_X, D_Y} L(G, F, D_X, D_Y)$.



CycleGAN

- Some result:



CycleGAN

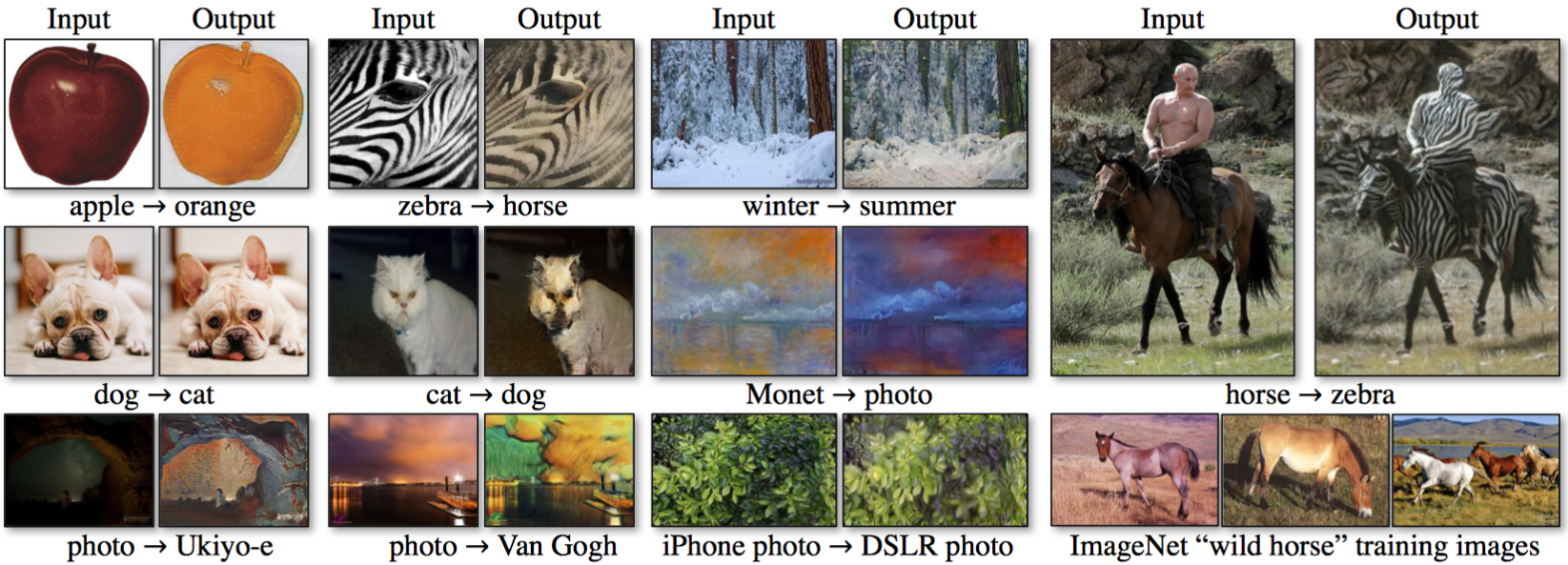


Figure 16: Some failure case.

Progressive GAN

- Training GAN is no easy job. Training GAN for the generation of high resolution image generation is even harder.
  - Higher resolution image means higher dimension, thus harder optimization.
  - Higher resolution makes it easier to tell the generated images apart from training images, thus makes it harder to train the generator.
- Even using improved minimax objective like WGAN-GP's may be insufficient to get nice result.
- Progressive GAN[9] offers a training technique orthogonal to the adjustment of the minimax objective to generate high-resolution images.

[9] Karras, Tero, et al. "Progressive Growing of GANs for Improved Quality, Stability, and Variation." International Conference on Learning Representations (2018).

Progressive GAN

- Progressive GAN improves image quality by growing the model size throughout training.
- The key idea is progressive training, namely to grow both the generator and discriminator progressively, starting from a low resolution and adding new layers that model increasingly fine details as training progresses.
  - At early stage, the generation of smaller images is substantially more stable because there is less class information and fewer modes.
  - Increasing the resolution little by little means that the model is only faced with harder challenge when it has been equipped with some knowledge.
- Progressive GAN also reduces the typical training time for getting high resolution images, since training on smaller images is cheaper.

Progressive GAN

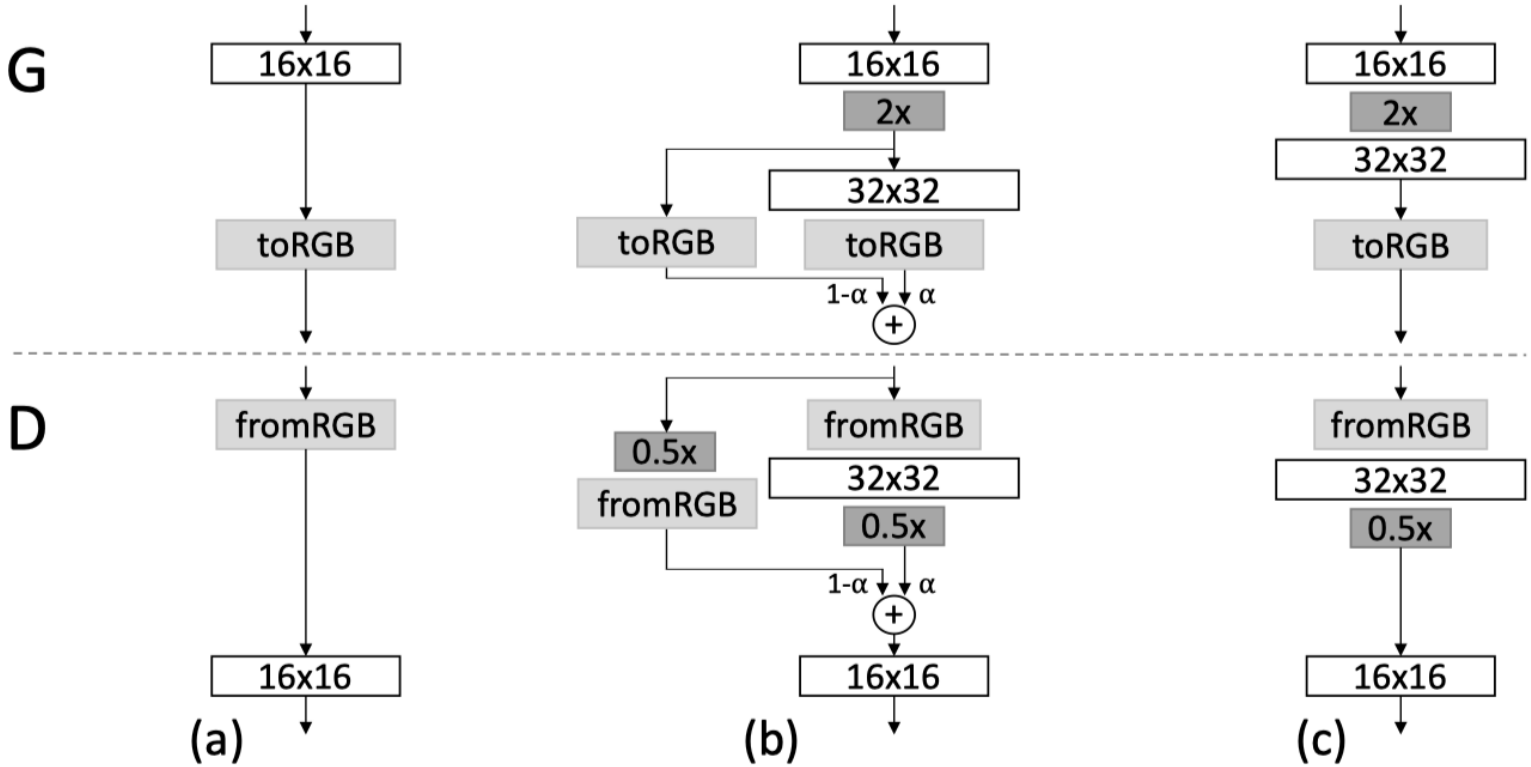


Figure 17: When doubling the resolution of the generator and discriminator, the new layers fade in smoothly, which is completed by a linearly increasing α from 0 to 1. $2\times$ and $0.5\times$ refer to doubling and halving the image resolution using nearest neighbor ltering and average pooling.

Progressive GAN

Figure 18: Face generated by Progressive GAN, 1024*1024 resolution.

BigGAN

- We have seen many exciting results from GANs.
- Yet, there is still a long way to go to generate high-resolution, diverse samples from complex datasets such as ImageNet.
- BigGAN[10] explores the idea of how scale may improve GANs.

[10] Brock, Andrew, Jeff Donahue, and Karen Simonyan. "Large Scale GAN Training for High Fidelity Natural Image Synthesis." International Conference on Learning Representations (2019).

BigGAN

- BigGAN is the largest scale network is attempted in the training of GANs yet.
  - 2-4 times as many parameters and 8 times the batch size compared to prior work.
- It is found that Gaussian truncation to sample $\mathbf{z}$ help stabilize training at this large scale.
- Many other tricks as well as multiple regularizations, such as a Gradient Penalty regularization and an orthogonal regularization(Gram matrix of the weight matrix enforced to be close to Identity matrix) are exploited.
  - The orthogonal regularization is

$$\lambda \| W^T W - I \|_F.$$

  - The purpose is to make different channels as orthogonal thus irrelevant as possible.

BigGAN

Figure 19: Figures are of 512*512 resolution. Classes such as dogs which are largely textural, and common in the dataset, are far easier to model than classes involving unaligned human faces or crowds.

Video-to-Video Synthesis

- The problem of video-to-video synthesis is to learn a mapping function from an input source video (e.g., a sequence of semantic segmentation masks) to an output photo-realistic video that precisely depicts the content of the source video.
- Directly applying image-to-image model to each frame of the source video often results in temporally incoherent video of low visual quality.
- Ting-Chun Wang et al.[11] proposed an approach for video-to-video synthesis under the GAN framework which has
  - careful design of generator and discriminator,
  - a spatio-temporal adversarial objective.

[11] Wang, Tingchun, et al. "Video-to-Video Synthesis." Neural Information Processing Systems (2018): 1152-1164.

Video-to-Video Synthesis

- Click here to see the demo online.

Video-to-Video Synthesis

- Problem formulation
  - Let $\mathbf{s}_1^T := (\mathbf{s}_1, \mathbf{s}_2, ...\mathbf{s}_T)$ be a a sequence of source video frames. Let $\mathbf{x}_1^T := (\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_T)$ be the sequence of corresponding real video frames.
  - The goal is to learn a mapping function that converts $\mathbf{s}_1^T$ to a sequence of output video frames $\tilde{\mathbf{x}}_1^T := (\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, ...\tilde{\mathbf{x}}_T)$ such that

$$p(\tilde{\mathbf{x}}_1^T | \mathbf{s}_1^T) = p(\mathbf{x}_1^T | \mathbf{s}_1^T).$$

  - Through matching the conditional video distribution, the model learns to generate photo-realistic, temporally coherent output sequences as if they are captured by a video camera.

Video-to-Video Synthesis

- Probability matching under GAN framework
  - Review that solving GAN's objective

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_r}[\log D(\mathbf{x})] + \mathbb{E}_{\tilde{\mathbf{x}}\sim\mathbb{P}_g}[\log(1 - D(\tilde{\mathbf{x}}))]$$

    is equivalent to minimize the Jensen-Shannon divergence between $\mathbb{P}_g$ and $\mathbb{P}_r$.
  - Similarly, the minimax objective

$$\max_D \min_G \mathbb{E}_{\mathbf{x}_1^T,\mathbf{s}_1^T}[\log D(\mathbf{x}_1^T, \mathbf{s}_1^T)] + \mathbb{E}_{\tilde{\mathbf{s}}_1^T}[\log(1 - D(G(\tilde{\mathbf{s}}_1^T), \tilde{\mathbf{s}}_1^T))]$$

    has the same solution as minimize the Jensen-Shannon divergence between $p(\tilde{\mathbf{x}}_1^T|\mathbf{s}_1^T)$ and $p(\mathbf{x}_1^T|\mathbf{s}_1^T)$, which is exactly what is needed here.

Factorization of Conditional Probability

- $p(\tilde{\mathbf{x}}_1^T|\mathbf{s}_1^T)$ could be quite complicated due to entanglement between frames at different time.
- To simplify the problem, a Markov assumption is made, i.e., $p(\tilde{\mathbf{x}}_1^T|\mathbf{s}_1^T)$ is assumed to have a product form given by

$$p(\tilde{\mathbf{x}}_1^T|\mathbf{s}_1^T) = \Pi_{t=1}^T p(\tilde{\mathbf{x}}_t|\tilde{\mathbf{x}}_{t-L}^{t-1}, \mathbf{s}_{t-L}^t).$$

- In other words, it is assumed that video frames can be generated sequentially, and the generation of the t-th frame $\tilde{\mathbf{x}}_t$ only depends on the history information from $t-L$ up to now.
- Under this assumption, a feed-forward network F is used to model the conditional distribution $p(\tilde{\mathbf{x}}_t|\tilde{\mathbf{x}}_{t-L}^{t-1}, \mathbf{s}_{t-L}^t)$ by $\tilde{\mathbf{x}}_t = F(\tilde{\mathbf{x}}_{t-L}^{t-1}, \mathbf{s}_{t-L}^t)$.
- Recursively applying the F leads to the final result $\tilde{\mathbf{x}}_1^T$.
- L is a parameter to be tuned. Empirically, small $L(e.g., L = 1)$ causes training instability, while large L increases training time and GPU memory usage but with minor quality improvement. $L = 2$ is used in the experiment.

Architecture of F

- Prior knowledge: Video signals contain a large amount of redundant information in consecutive frames. We can estimate the next frame by warping the current frame largely correctly except for the occluded areas.
- Based on this prior, F is assumed to have the form

$$F(\tilde{\mathbf{x}}_{t-L}^{t-1}, \mathbf{s}_{t-L}^{t}) = (\mathbf{1} - \tilde{m}_t) \odot \tilde{\mathbf{w}}_{t-1}(\tilde{\mathbf{x}}_{t-1}) + \tilde{m}_t \odot \tilde{\mathbf{h}}_t,$$

where $\odot$ is the element-wise product operator and $\mathbf{1}$ is an image of all ones.

  - $\tilde{\mathbf{w}}_{t-1} = W(\tilde{\mathbf{x}}_{t-L}^{t-1}, \mathbf{s}_{t-L}^{t})$ is the optical flow from $\tilde{\mathbf{x}}_{t-1}$ to $\tilde{\mathbf{x}}_t$, which capture the transformation between consecutive frames. W is called optical flow prediction network. $\tilde{\mathbf{x}}_{t-1}$ is warped based on $\tilde{\mathbf{w}}_{t-1}$.
  - $\tilde{\mathbf{h}}_t = H(\tilde{\mathbf{x}}_{t-L}^{t-1}, \mathbf{s}_{t-L}^{t})$ is the hallucinated image mainly targeting at the occluded areas, synthesized directly by the generator H.
  - $\tilde{m}_t = M(\tilde{\mathbf{x}}_{t-L}^{t-1}, \mathbf{s}_{t-L}^{t})$ is the soft occlusion mask with continuous values between 0 and 1. M denotes the mask prediction network. Soft mask is used to enable the generation of high resolution of the object gradually.

Multiple Discriminators

- Using multiple discriminators can mitigate the mode collapse problem during GAN's training.
- Two types of discriminators are designed.
  - Conditional image discriminator D_I: 1 for a true pair $(\mathbf{x}_t, \mathbf{s}_t)$ and 0 for a fake one $(\tilde{\mathbf{x}}_t, \mathbf{s}_t)$, so as to ensure that each output frame resembles a real image given the same source image;
  - Conditional video discriminator D_V: 1 for a true pair $(\mathbf{x}_{t-K}^{t-1}, \mathbf{w}_{t-K}^{t-1})$ 0 for a fake one $(\tilde{\mathbf{x}}_{t-K}^{t-1}, \mathbf{w}_{t-K}^{t-1})$, where $\mathbf{w}_{t-K}^{t-1}$ is the $K-1$ optical flow for the K consecutive real images $\mathbf{x}_{t-K}^{t-1}$ so as to ensure that consecutive output frames resemble the temporal dynamics of a real video given the same optical flow.

Multiple Discriminators

- How to realize these two discriminators? We need to define two sampling operators.
  - ϕ_I: a random image sampling operator such that $\phi_I(\mathbf{x}_1^T, \mathbf{s}_1^T) = (\mathbf{x}_i, \mathbf{s}_i)$, where i is uniformly distribution on $\{1, 2, ...T\}$. ϕ_I randomly samples a pair of images from $(\mathbf{x}_1^T, \mathbf{s}_1^T)$.
  - ϕ_V: a random frames sampling operator such that $\phi_V(\mathbf{w}_1^{T-1}, \mathbf{x}_1^T, \mathbf{s}_1^T) = (\mathbf{w}_{i-K}^{i-1}, \mathbf{x}_{i-K}^{i-1}, \mathbf{s}_{i-K}^{i-1})$ where i is an integer uniformly sampled from $K+1$ to $T+1$. ϕ_V retrieves K consecutive frames and the corresponding $K1$ optical flow images.

Learning Objective

- With the help of operators ϕ_I and ϕ_V, the objective can be written as

$$\min_F(\max_{D_I} L_I(F, D_I) + \max_{D_V} L_V(F, D_V)) + \lambda_W L_W(F), \quad (21)$$

where $L_I(F, D_I)$ is the GAN loss on images dened by the conditional image discriminator D_I, $\max_{D_V} L_V(F, D_V)$ is the GAN loss on K consecutive frames dened by D_V, and $L_W(F)$ is the flow estimation loss (the loss of the optical flow prediction network F.) λ_W is to be tuned, and 10 is used in the experiment based on grid search.

Learning Objective

- Recall that the learning objective is $\min_F(\max_{D_I} L_I(F, D_I) + \max_{D_V} L_V(F, D_V)) + \lambda_W L_W(F)$.
- GAN loss on images is

$$L_I(F, D_I) := \mathbb{E}_{\phi_I(\mathbf{x}_1^T, \mathbf{s}_1^T)}[\log D_I(\mathbf{x}_i, \mathbf{s}_i)] + \mathbb{E}_{\phi_I(\mathbf{x}_1^T, \mathbf{s}_1^T)}[\log(1 - D_I(\tilde{x}_i, \mathbf{s}_i))].$$

- GAN loss on K consecutive frames is

$$\begin{aligned} L_V(F, D_V) :=& \mathbb{E}_{\phi_V(\mathbf{w}_1^{T-1}, \mathbf{x}_1^T, \mathbf{s}_1^T)}[\log D_V(\mathbf{x}_{i-K}^{i-1}, \mathbf{w}_{i-K}^{i-1})] \\ &+ \mathbb{E}_{\phi_V(\mathbf{w}_1^{T-1}, \mathbf{x}_1^T, \mathbf{s}_1^T)}[\log(1 - D_V(\tilde{\mathbf{x}}_{i-K}^{i-1}, \mathbf{w}_{i-K}^{i-1}))]. \end{aligned}$$

Learning Objective

- Flow estimation loss is

$$L_W(F) := \frac{1}{T-1} \sum_{t=1}^{T-1} (\|\tilde{\mathbf{w}}_t - \mathbf{w}_t\|_1 + \|\tilde{\mathbf{w}}_t - \mathbf{x}_{t+1}\|_1),$$

  where the summation over $T-1$ is because there is one flow loss for every frame but the last one. Each flow loss contains two parts, the first being the endpoint error between the ground truth and the estimated flow, the second being the warping loss when the flow warps the previous frame to the next frame.

GraphSGAN

- We have seen a lot applications on images using GAN.
- GraphSGAN[12] proposes a framework based on semi-supervised GAN to learn on graph.
- It's a work combining network embedding and GAN.

[12] Ding, Ming, et al. " Semi-supervised Learning on Graphs with Generative Adversarial Nets." Proceedings of the 27th ACM International Conference on Information and Knowledge Management (2018).

GraphSGAN

- In a graph G, we have labeled node set $L = \{(v_i, l_i)\}$ and unlabeled node set $U = \{u_i\}$. The graph is denoted by $G = (L + U, E)$ where E is the edge set.
- Feature vectors of each node i is denoted by p_i.
- The problem studied is to predict labels of U.

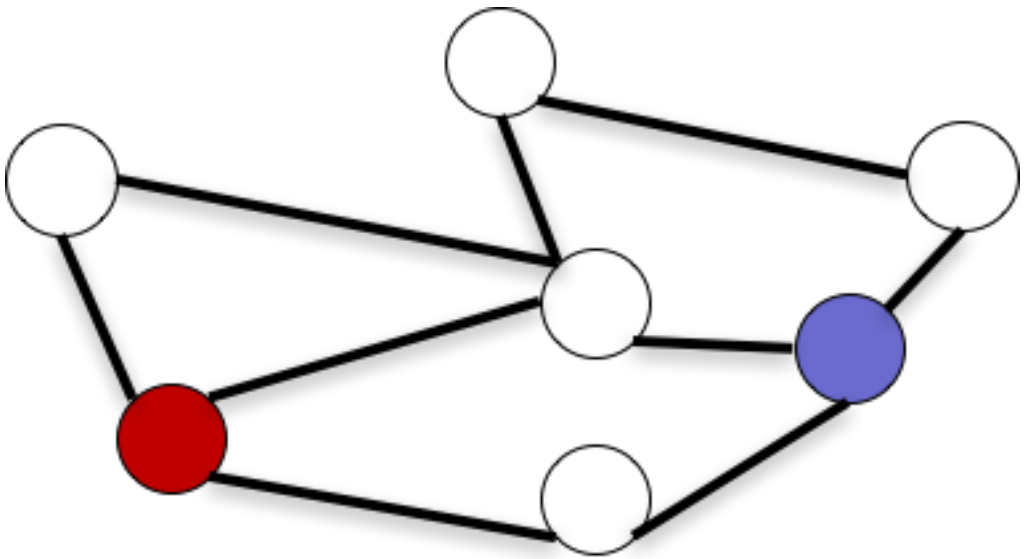


Figure 20: A partially labeled graph. GraphSGAN aims at learning a model to predict the labels of white nodes.

GraphSGAN

- Why may GAN help this problem setting?

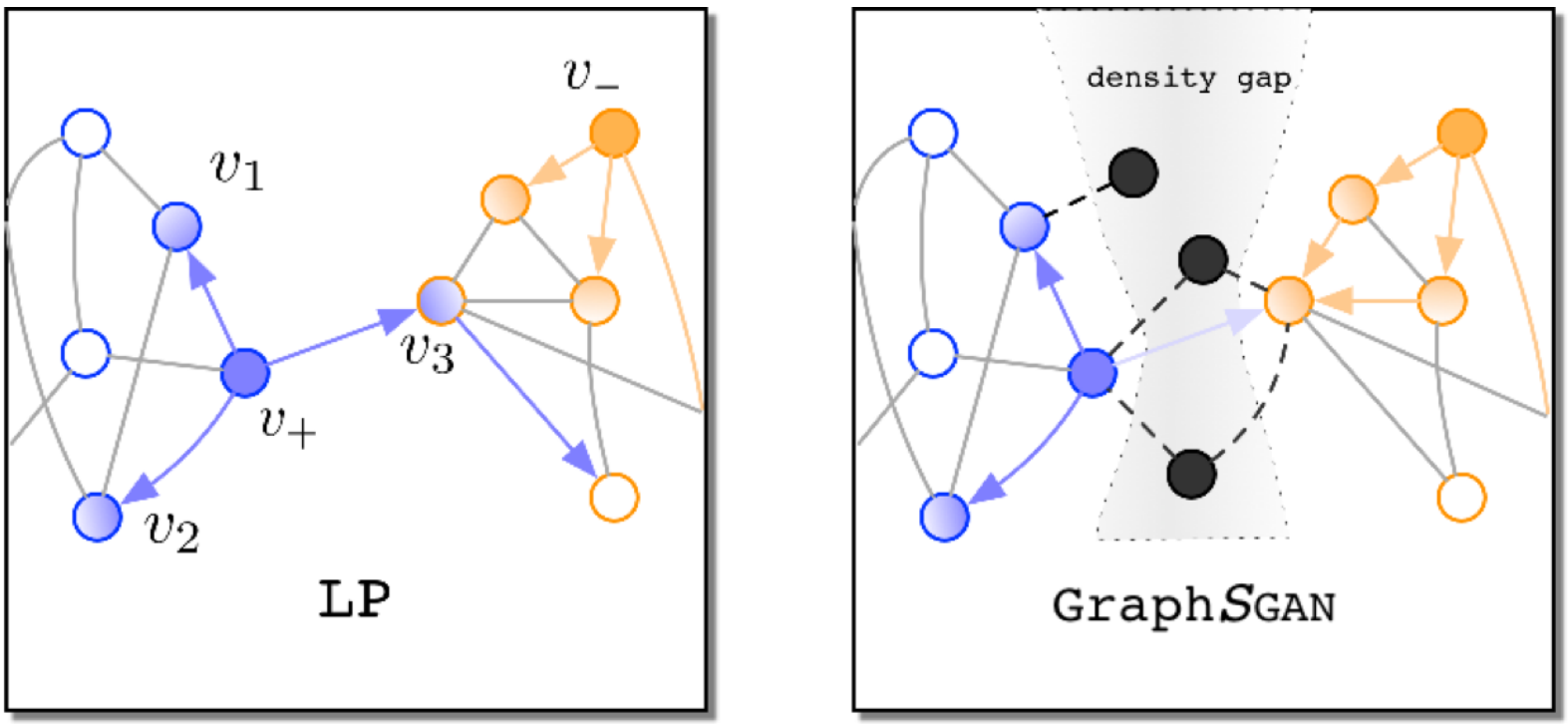


Figure 21: Using GAN's generated fake data, the influence between clusters are reduced.

Discriminative Loss

- There are four parts in discriminative loss.
  - Supervised loss:

$$\text{loss}_{\text{sup}} = -\mathbb{E}_{\mathbf{x}_i \in X^L}[\log \mathbb{P}(y_i|\mathbf{x}_i, y_i < M)]$$

    .
  - Unsupervised loss:

$$\text{loss}_{\text{un}} = -\mathbb{E}_{\mathbf{x}_i \in X^U} \log(1 - \mathbb{P}(M|\mathbf{x}_i)) - \mathbb{E}_{\mathbf{x}_i \in G(z)} \log \mathbb{P}(M|\mathbf{x}_i)$$

- Entropy loss:

$$\text{loss}_{\text{ent}} = --\mathbb{E}_{\mathbf{x}_i \in X^U} \sum_{y=0}^{M-1} \mathbb{P}(y|\mathbf{x}_i, y_i < M) \log \mathbb{P}(y|\mathbf{x}_i, y_i < M).$$

- Pull-away loss:

$$\text{loss}_{\text{pt}} = \frac{1}{m(m-1)} \sum_{i=1}^{m} \sum_{j \neq i} \frac{h^{(n)}(\mathbf{x}_i)^T h^{(n)}(\mathbf{x}_j)}{\|h^{(n)}(\mathbf{x}_i)^T h^{(n)}(\mathbf{x}_j)\|},$$

  where h is some middle layer.

Generative Loss

- Feature matching loss:

$$\text{loss}_{\text{fm}} = \|\mathbb{E}_{\mathbf{x}_i \in X^U} h^{(n)}(\mathbf{x}_i) - \mathbb{E}_{\mathbf{x}_j \in G(z)} h^{(n)}(\mathbf{x}_j)\|_2^2.$$

- Pull-away loss:

$$\text{loss}_{\text{pt}} = \frac{1}{m(m-1)} \sum_{i=1}^{m} \sum_{j \neq i} \frac{h^{(n)}(\mathbf{x}_i)^T h^{(n)}(\mathbf{x}_j)}{\|h^{(n)}(\mathbf{x}_i)^T h^{(n)}(\mathbf{x}_j)\|}.$$

Result

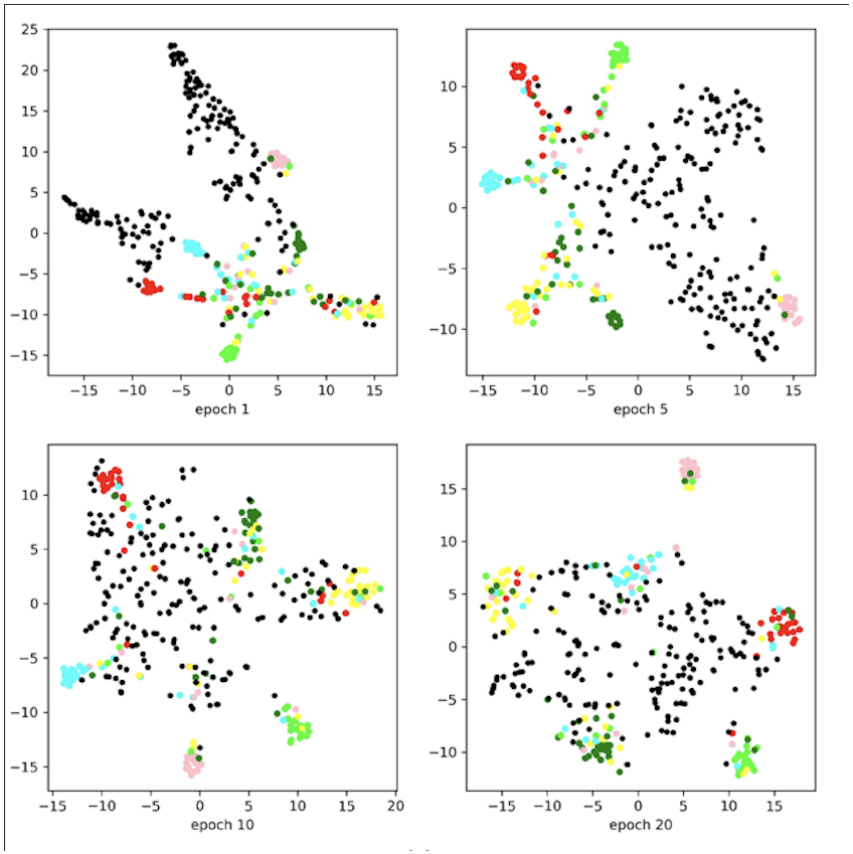


Figure 22: The figure shows that generated data(black dot) is indeed distributed in the density gaps where the real data have few existence.

Result

Category	Method	Recall@K
Regularization	LP	16.2
	ManiReg	47.7
Embedding	DeepWalk	25.8
	SemiEmb	48.6
	Planetoid	50.1
Original	*DIEL*	40.5
Our Method	GraphSGAN	**51.8**
	Upper bound	61.7

Figure 23: Results on DIEL dataset.

Conclusion

- GANs are generative models that use a minimax game to approximate an intractable cost function or a density ratio.
- GANs can simulate many cost functions and complex distributions, yet GANs are hard to train. Thus many works are devoted to improving the training of GANs.
- GANs show great potential in real world applications. Applications include learning from very few labeled examples, real image/artwork generation, video synthesis, and GCN etc.

Overview

Thanks.

HP: http://keg.cs.tsinghua.edu.cn/jietang/
Email: jietang@tsinghua.edu.cn