Understanding
Deep Generative Models

2018. 01. 04

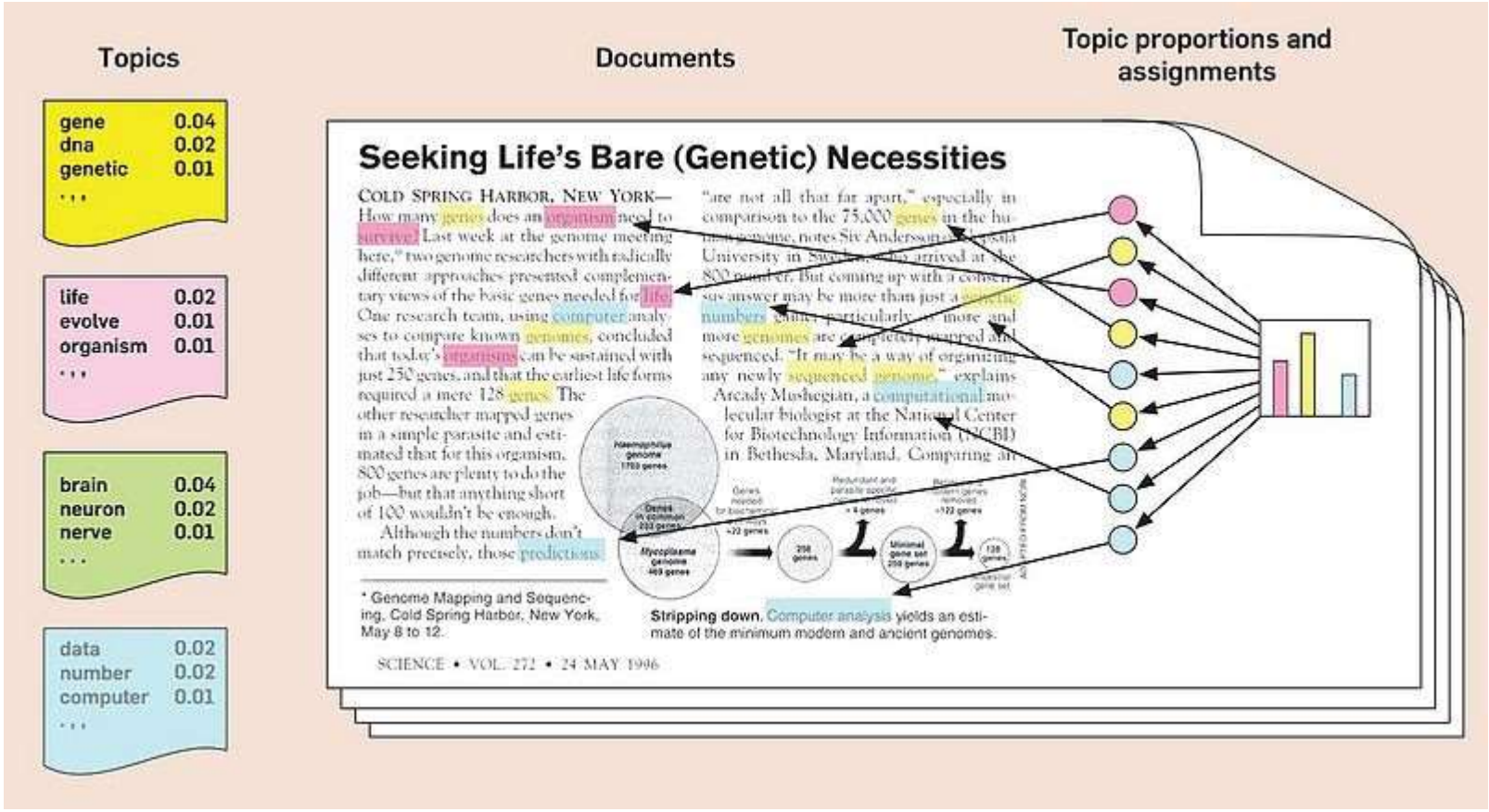
DMQA Lab Seminar

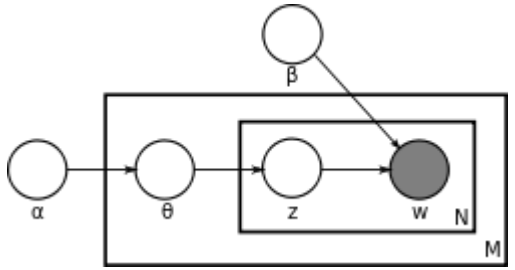
Deep Generative Models

Generative Models

Generative models → try to learn data generating process

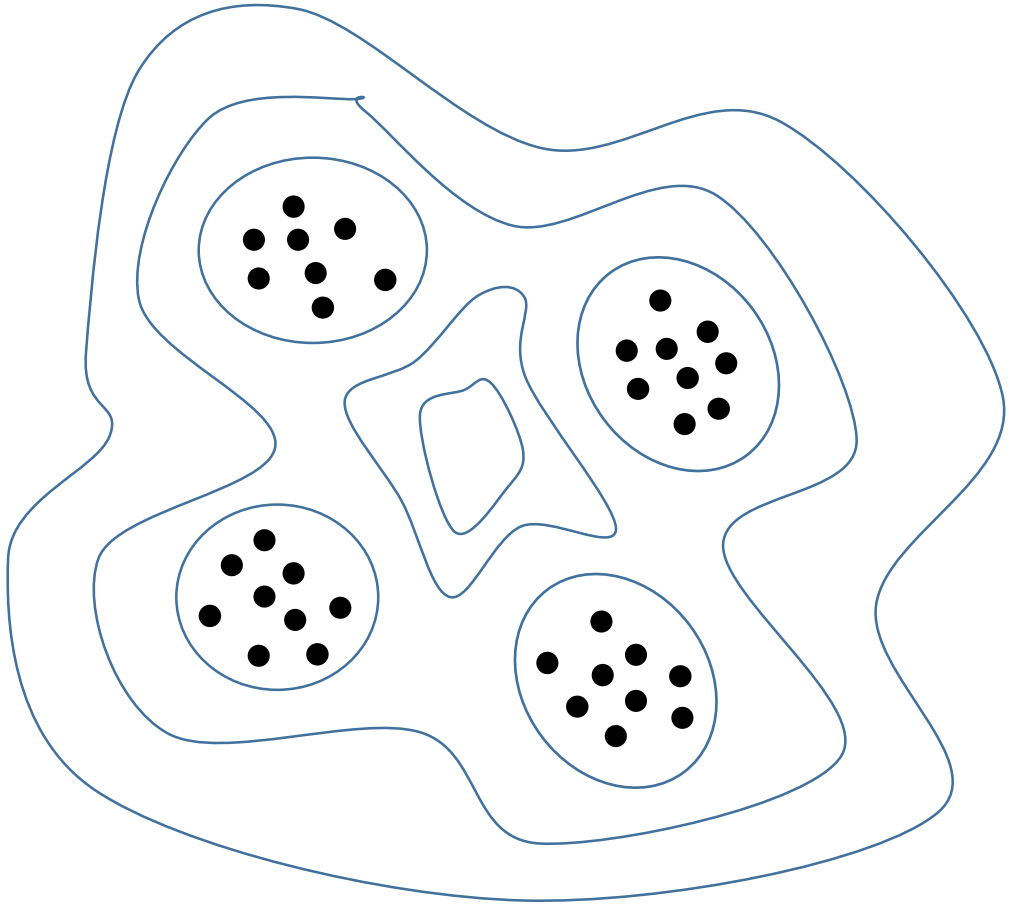
Motivating example: latent Dirichlet allocation

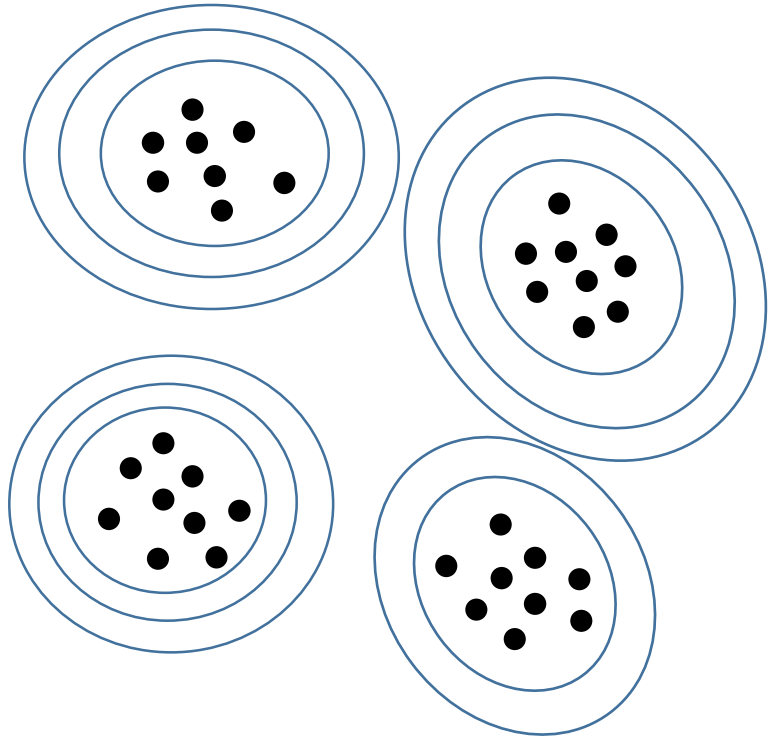




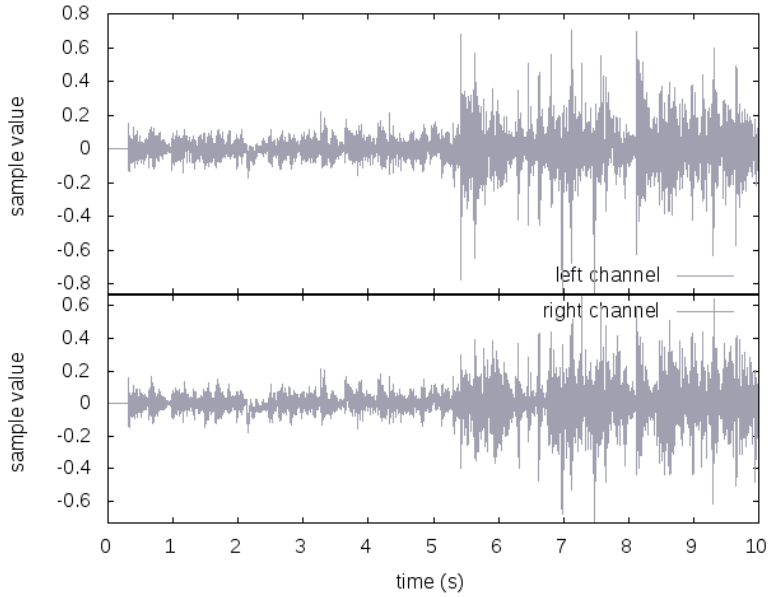
Generative Models with Latent Variables

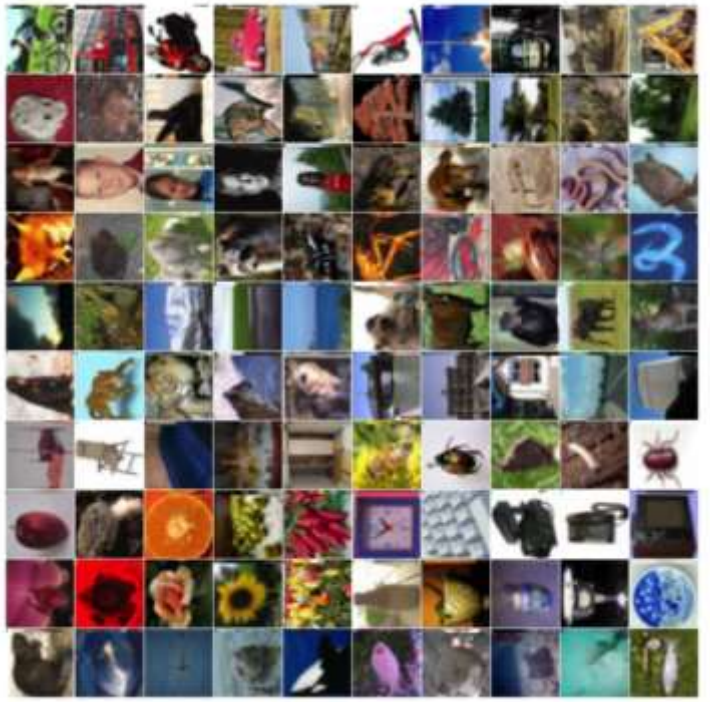
Introducing latent variable can make problems tractable (e.g. mixture distribution)





Deep Generative Models

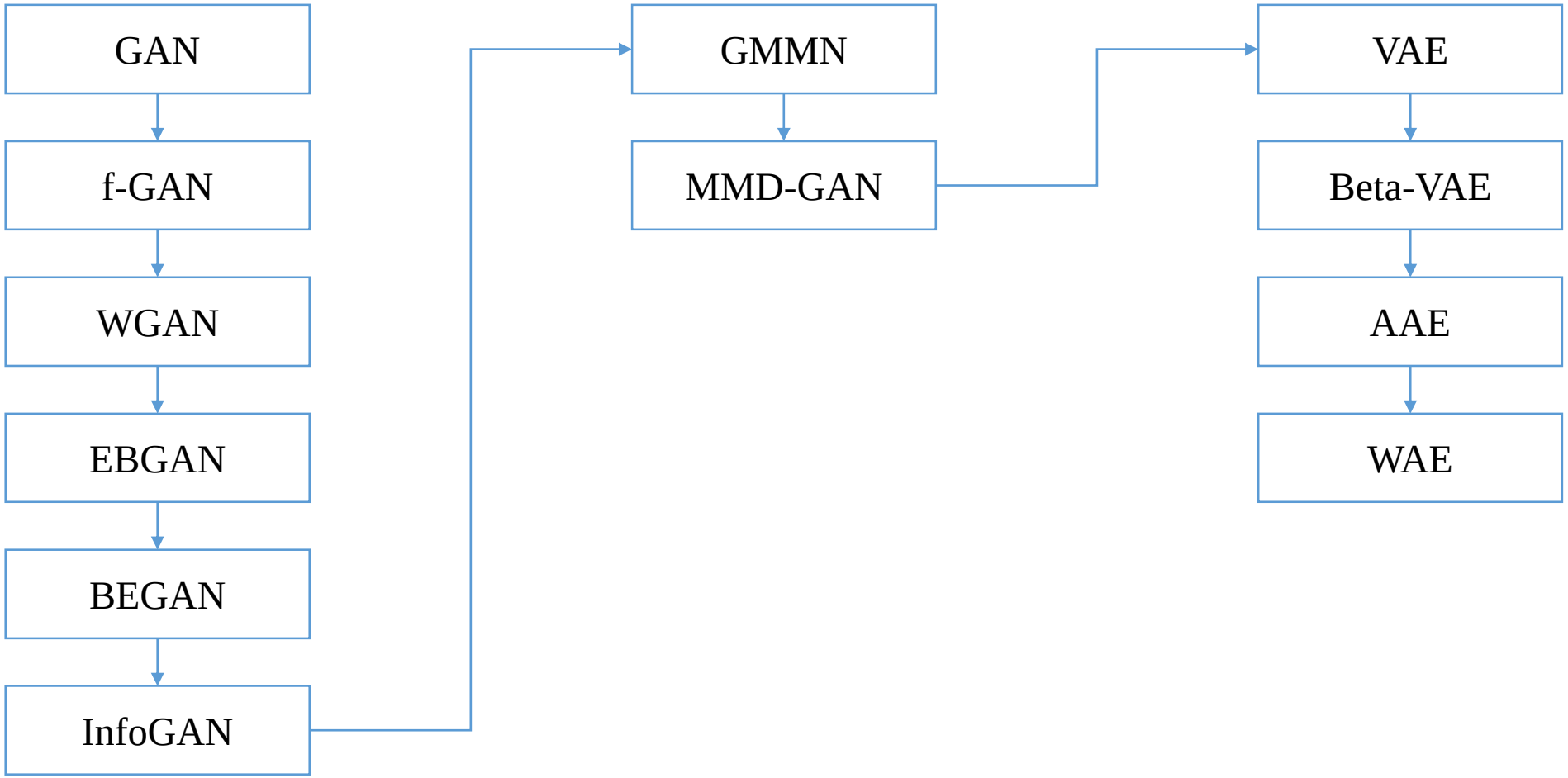




Conventional generative models are not tractable with

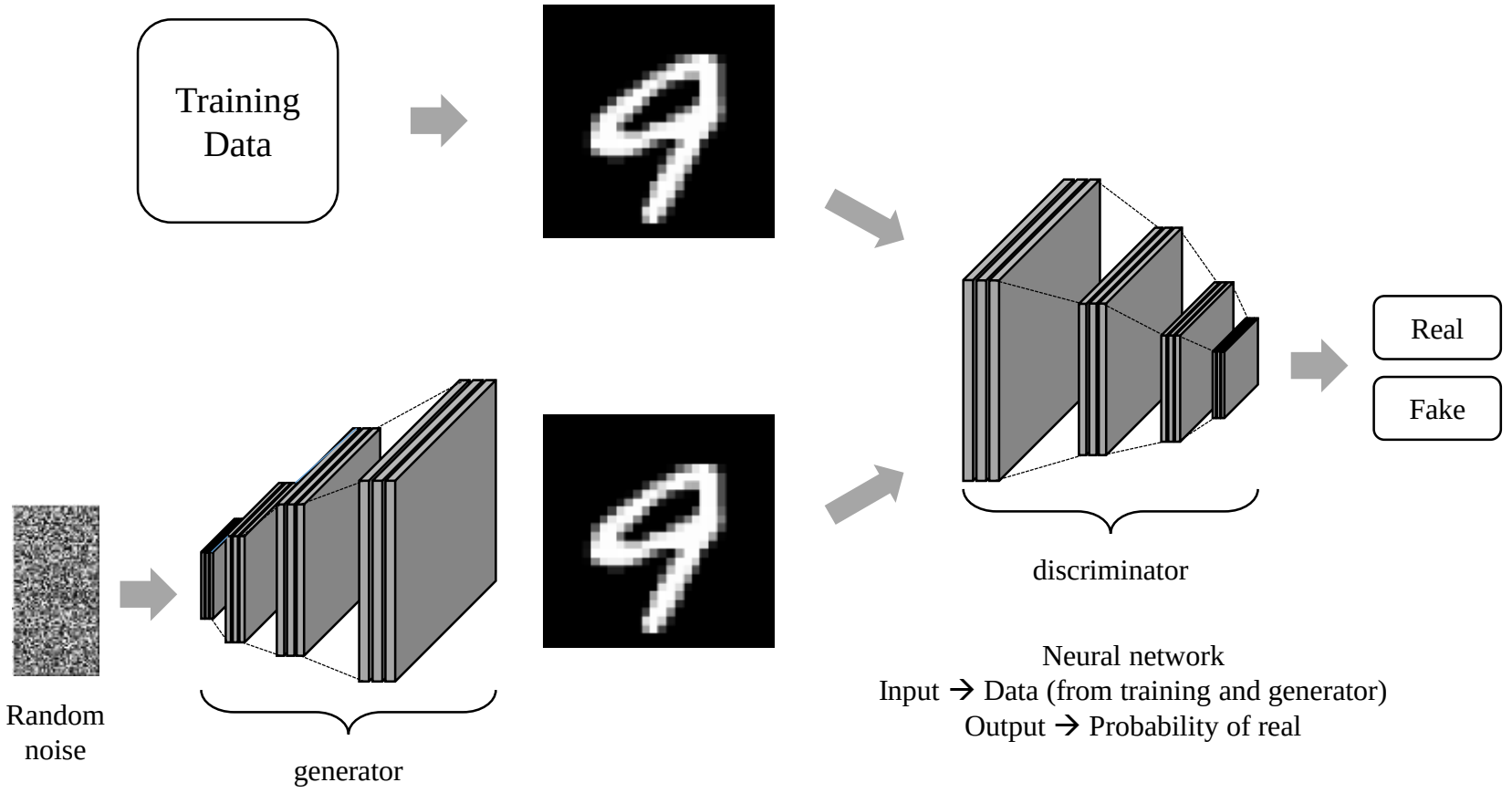
high-dimensional complex data

Deep Generative Models



GANs, GMMN, and VAE

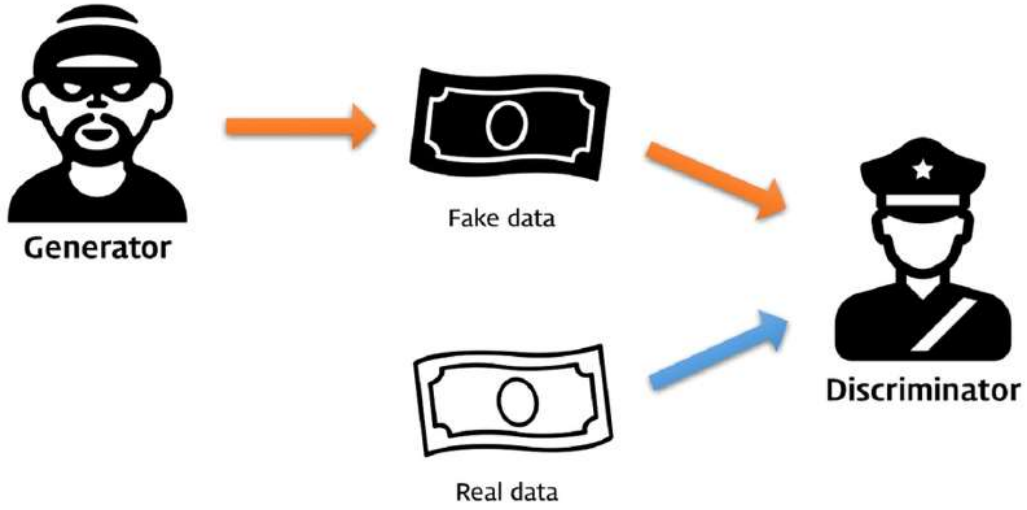
Generative Adversarial Networks



Generative Adversarial Networks

Real

Generated



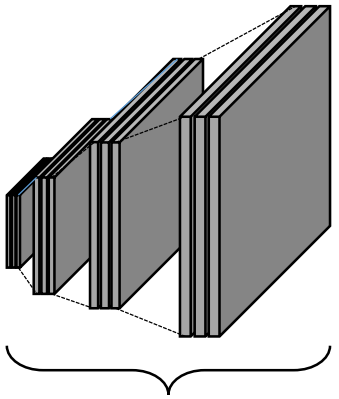
Adversarial learning process

Generative Adversarial Networks

Random Noise
e.g. Uniform (0, 1)
of 100 dimension
(Latent Distribution)

z1	z2	…	z99	z100
0.8642	-0.3331	…	-0.0322	0.1234

100-dimensional vector
(Latent Variable)

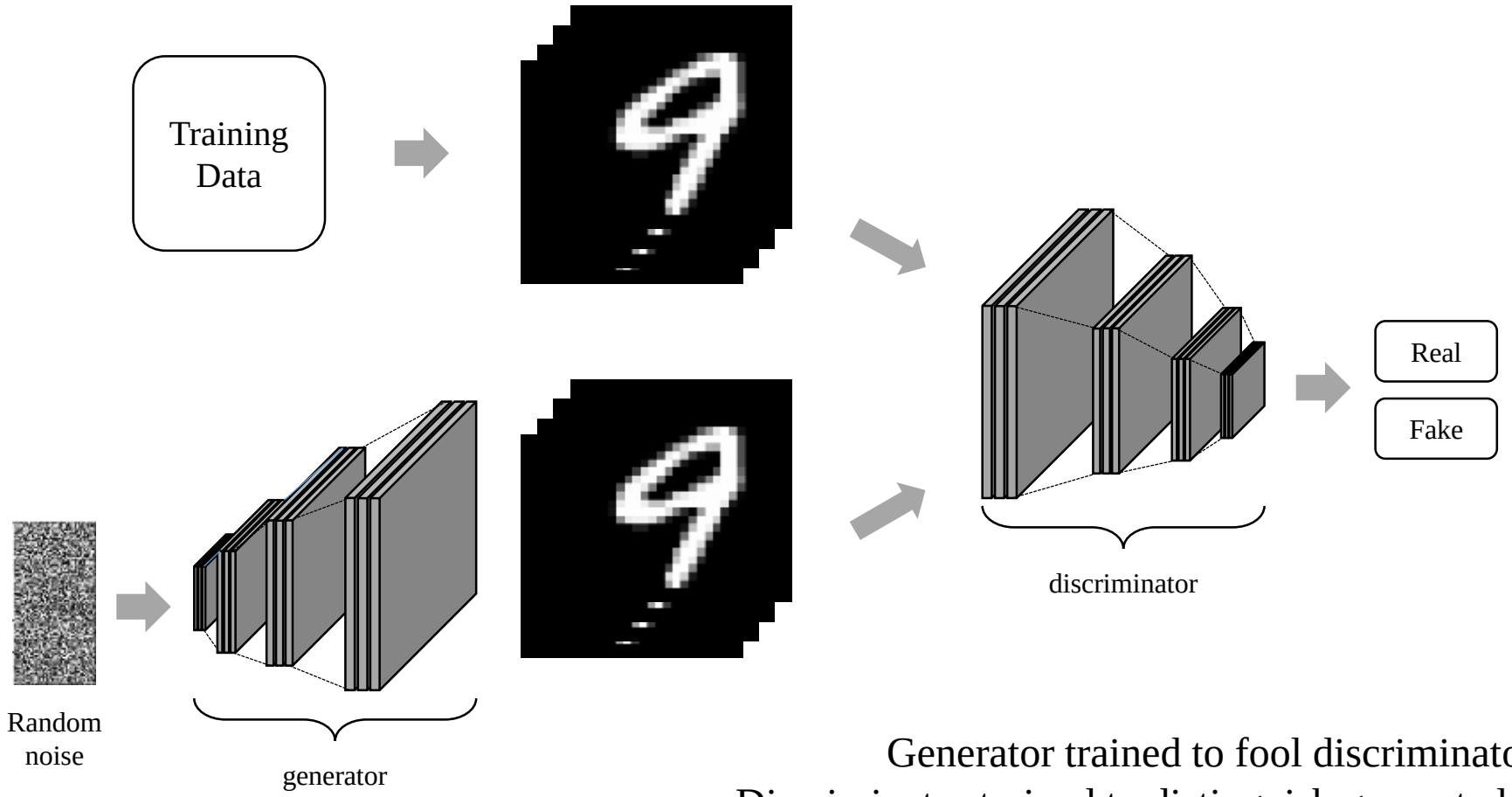


generator

Image

Generating Process of GANs

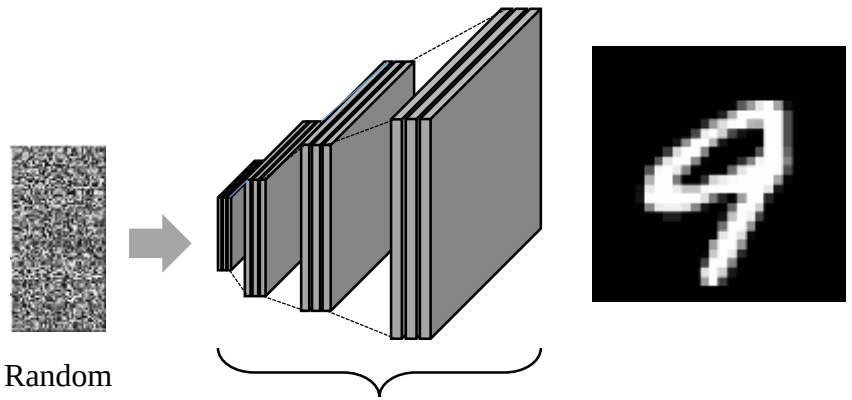
Generative Adversarial Networks



Generator trained to fool discriminator
Discriminator trained to distinguish generated from reals

Training of GANs (Alternating Minimization)

Generative Moment Matching Network



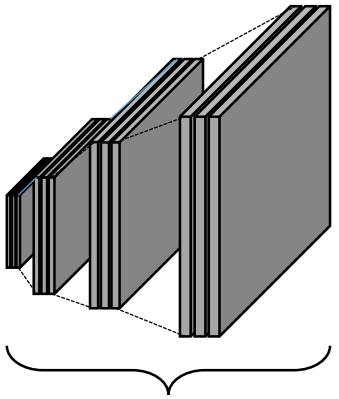
Neural network
Input → noise (from prior)
Output → random sample (from model distribution)

Generative Moment Matching Network

Random Noise
e.g. Uniform (0, 1)
of 100 dimension
(Latent Distribution)

z1	z2	…	z99	z100
0.8642	-0.3331	…	-0.0322	0.1234

100-dimensional vector
(Latent Variable)

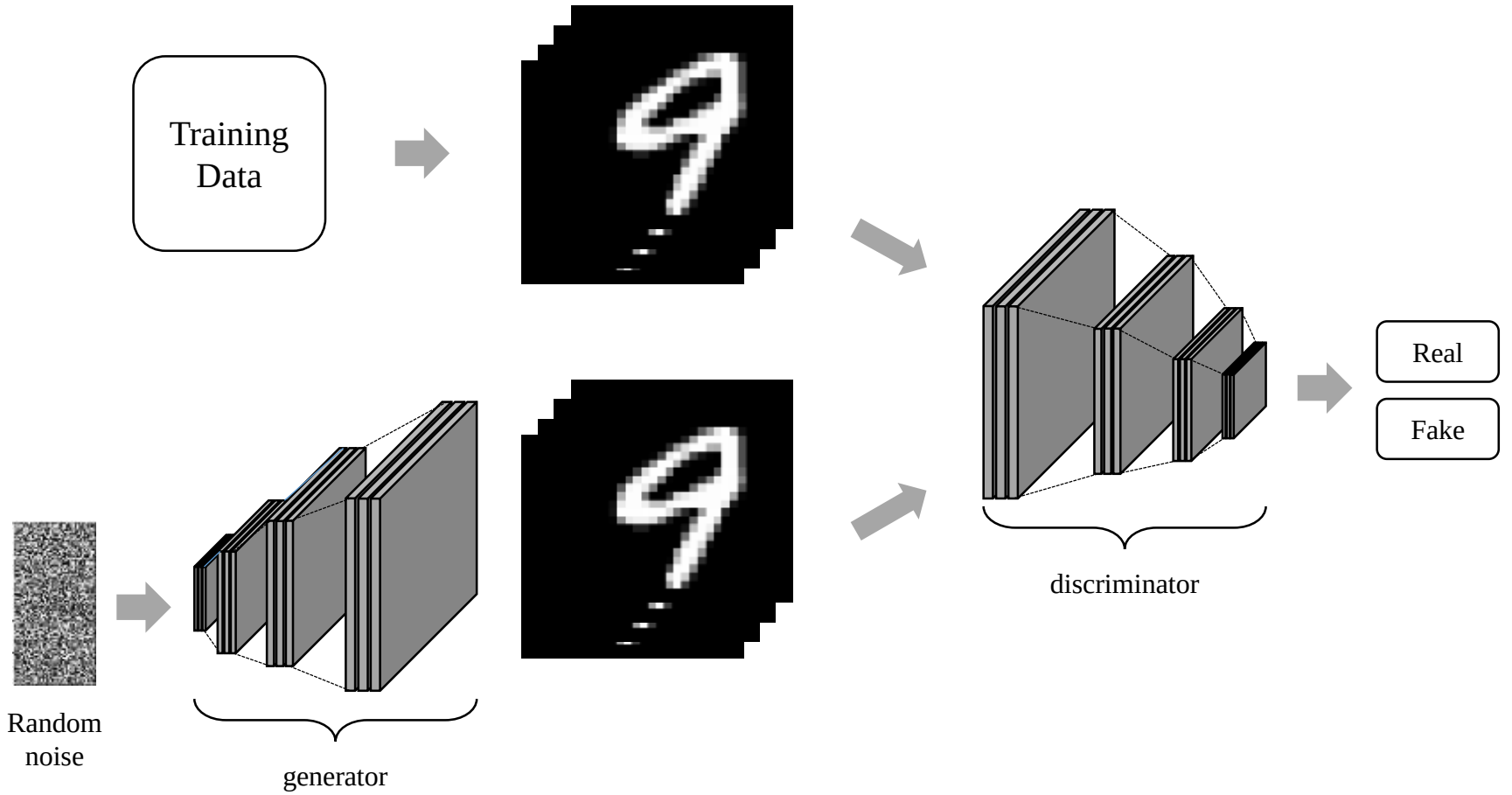


generator

Image

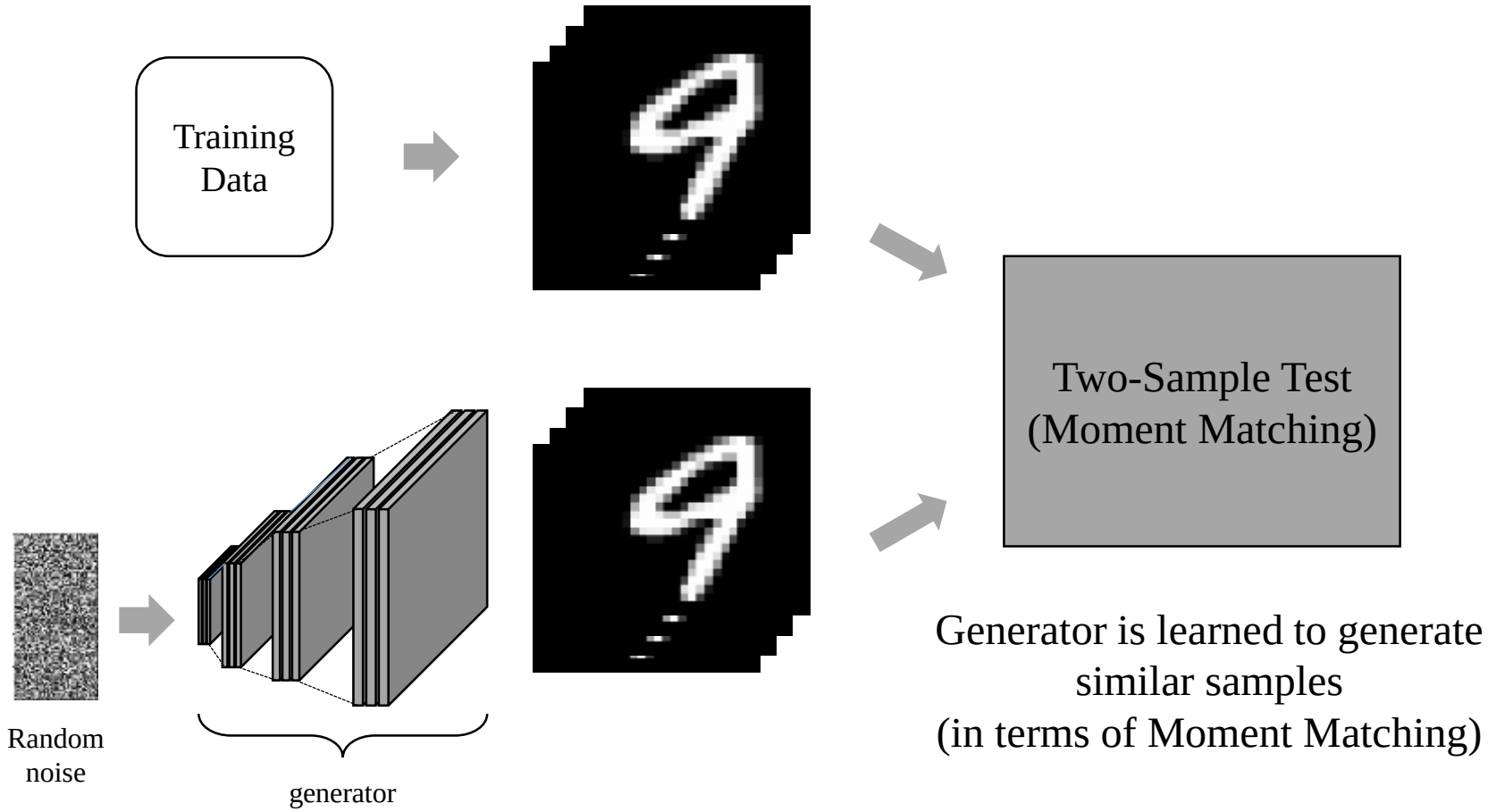
Generating Process of GMMN

Generative Moment Matching Network



Discriminator in GANs determines how true samples and generated samples are similar

Generative Moment Matching Network

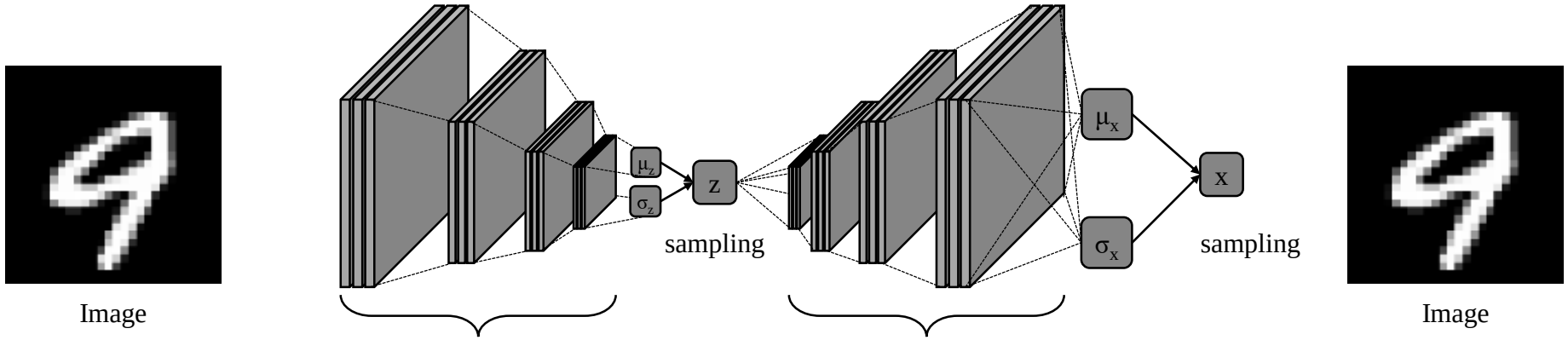


Generator is learned to generate similar samples (in terms of Moment Matching)

In GMMN, a method called **Moment Matching** is used to measure the similarity between true samples and generated samples

Variational Autoencoder

Gaussian Encoder – Gaussian Decoder



Image

Image

encoder
$q_\phi(z|x)$

decoder
$p_\theta(x|z)$

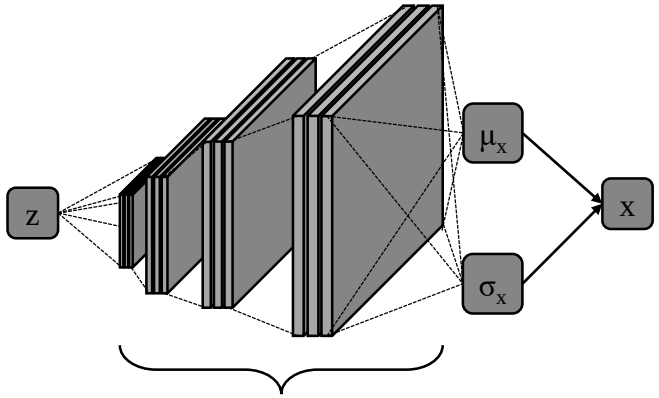
Neural network
Input → Data
Output → Parameters of
the latent distribution (μ, σ)

Neural network
Input → Latent vectors
Output → Parameters of
the model distribution (μ, σ)

Variational Autoencoder

z1	z2	…	z99	z100
0.8642	-0.3331	…	-0.0322	0.1234

100-dimensional vector
(Latent Variable)

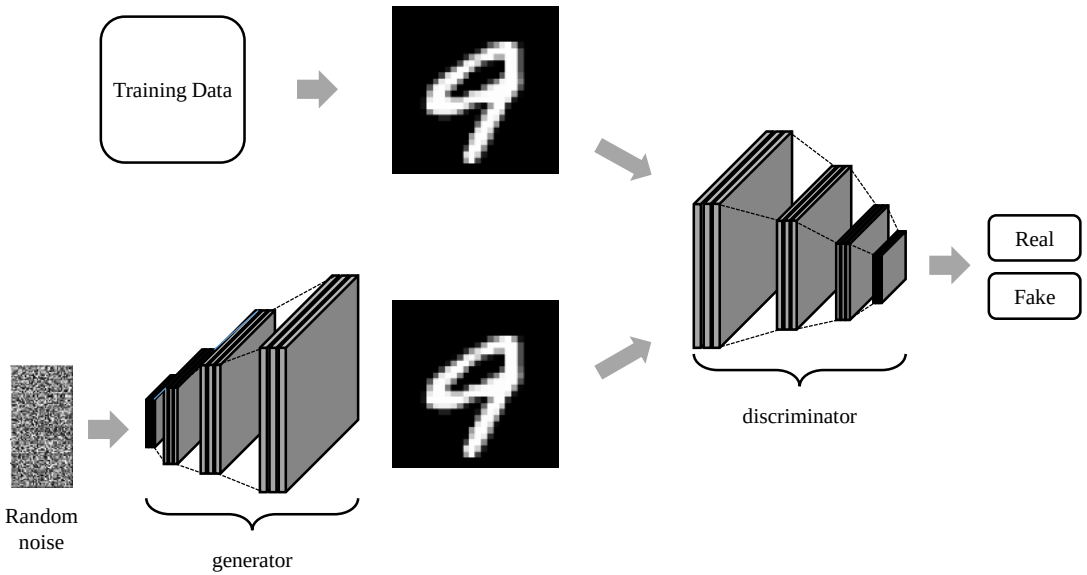


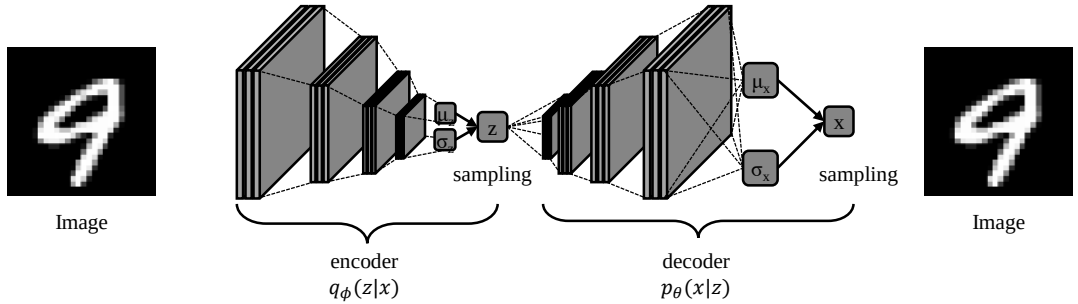
decoder
$p_\theta(x|z)$

Image

Generating Process of VAE

What's Different





GANs

- G generates samples – G is a transformation (noise to sample)
  → Implicit density / likelihood free
- Random noise distribution does not change (prior)
  → There is no recognition model
- Min-max problem (adversarial)
- Suffers from mode collapse
- Unstable training (hard to balance G and D)
- Try to minimize the discrepancy between data distribution and model distribution
- Possible to utilize discrete random noise

VAE

- Decoder generates samples – decoder is an inference model (latent to parameters)
  → Explicit density
- Latent variable distribution changes (posterior)
  → There is a recognition model (encoder)
- Minimization (or maximization) problem
- Suffers from blurry image (low quality images)
- Stable training
- Try to maximize model likelihood given data
- Hard to impose discrete latent distribution

Generative Adversarial Networks

Generative Adversarial Networks

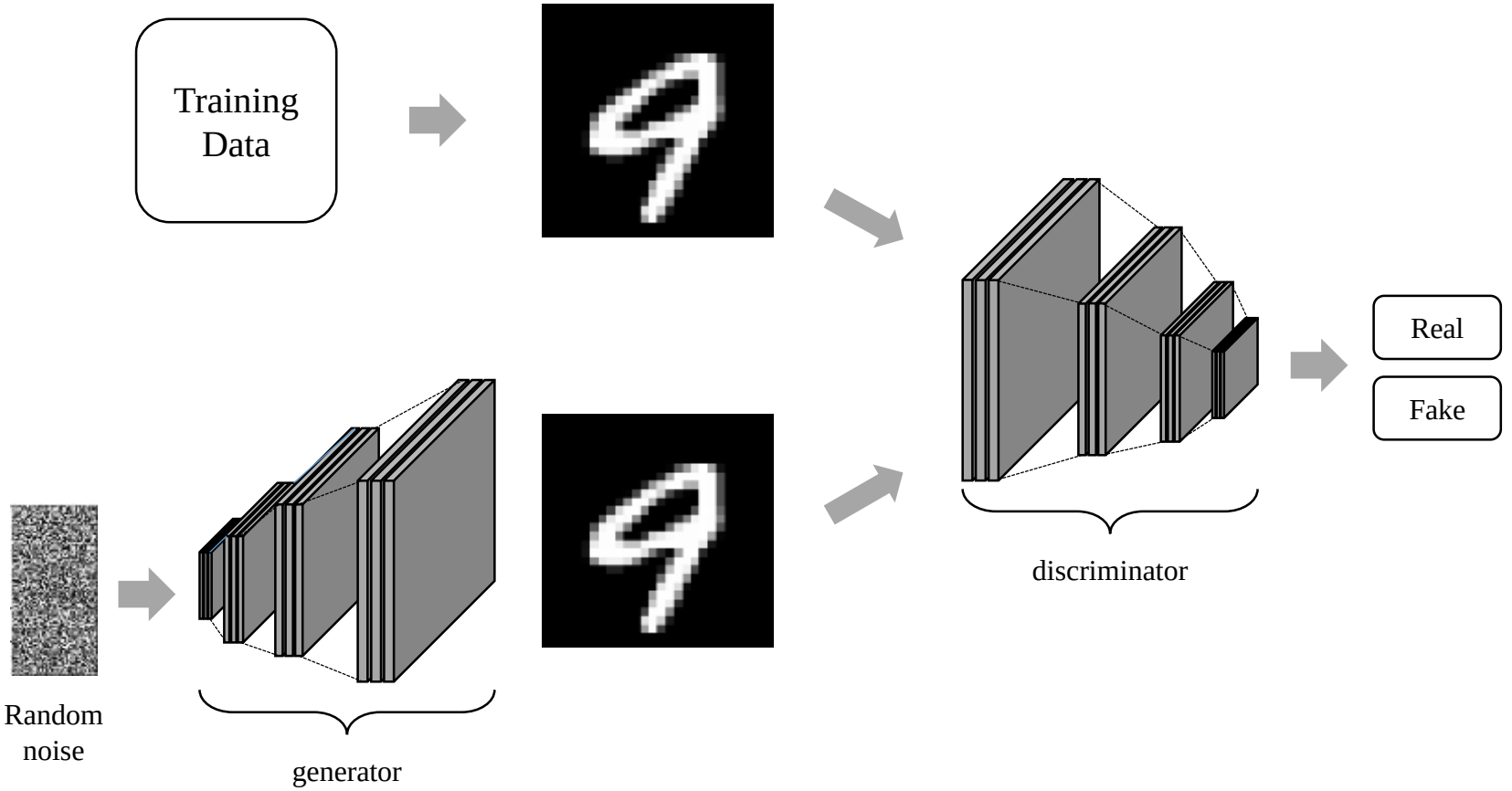
Generative Adversarial Nets

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



Generator tries to make generated samples to be classified as real
Discriminator tries to distinguish generated samples and real data

Generative Adversarial Networks

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

$$D(x) = \text{Output of } D = \text{probability of } x \text{ is determined to be real by } D$$
$$G(z) = \text{Output of } G = \text{generated sample by } G \text{ given a latent vector } z$$
$$\theta_G,\ \theta_D = \text{neural network parameters of } G \text{ and } D, \text{ respectively}$$

Generator tries to make generated samples to be classified as real
Discriminator tries to distinguish generated samples and real data

Generative Adversarial Networks

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

$$\min_{\theta_G} \mathbb{E}_{z \sim p(z)}[\log(1 - D(G(z)))]$$

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

Generator tries to make generated samples to be classified as real
Discriminator tries to distinguish generated samples and real data

Generative Adversarial Networks

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

Proposition 1. *For G fixed, the optimal discriminator D is*

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

Generative Adversarial Networks

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

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

$$= \min_{\theta_G} \mathbb{E}_{x\sim p_{\text{data}(x)}}[\log D^*_G(x)] + \mathbb{E}_{x\sim p_G(x)}[\log(1 - D^*_G(x))]$$

$$= \min_{\theta_G} \mathbb{E}_{x\sim p_{\text{data}}(x)}\left[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}\right] + \mathbb{E}_{x\sim p_G(x)}\left[\log \frac{p_G(x)}{p_{\text{data}}(x) + p_G(x)}\right]$$

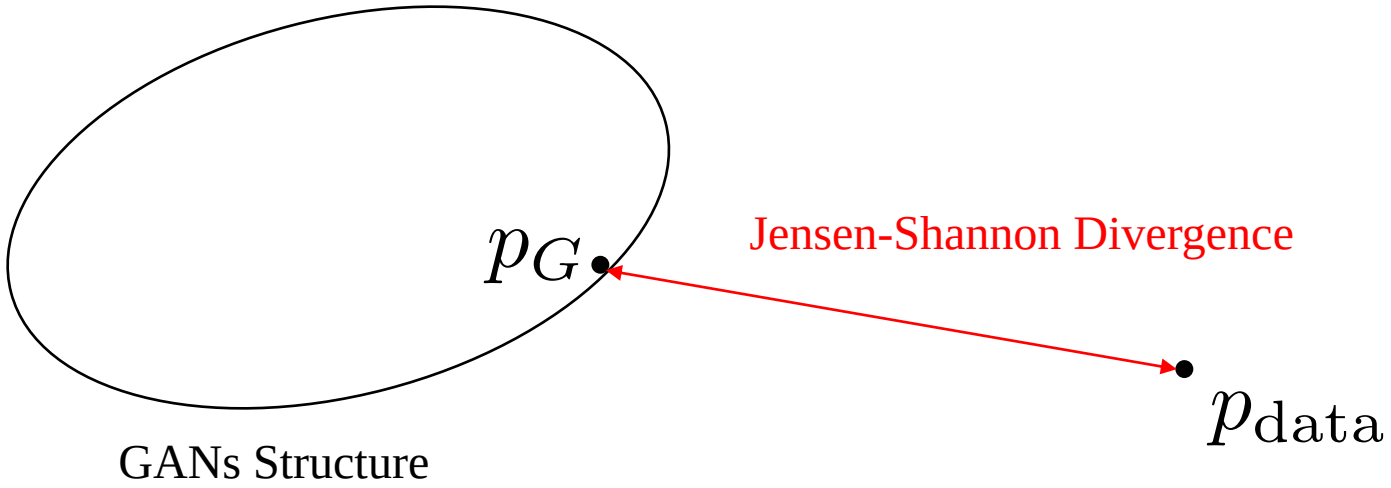
$$= \min_{\theta_G} -\log 2 + \mathbb{E}_{x\sim p_{\text{data}}(x)}\left[\log \frac{p_{\text{data}}(x)}{\frac{1}{2}(p_{\text{data}}(x) + p_G(x))}\right] - \log 2 + \mathbb{E}_{x\sim p_G(x)}\left[\log \frac{p_G(x)}{\frac{1}{2}(p_{\text{data}}(x) + p_G(x))}\right]$$

$$= \min_{\theta_G} -\log 4 + \mathcal{D}_{\text{KL}}\left(p_{\text{data}} \middle\| \frac{p_{\text{data}} + p_G}{2}\right) + \mathcal{D}_{\text{KL}}\left(p_G \middle\| \frac{p_{\text{data}} + p_G}{2}\right)$$

$$= \min_{\theta_G} -\log 4 + 2\mathcal{D}_{\text{JS}}\left(p_{\text{data}} \| p_G\right)$$

G is trained to minimize Jensen-Shannon divergence between data distribution and generator distribution!

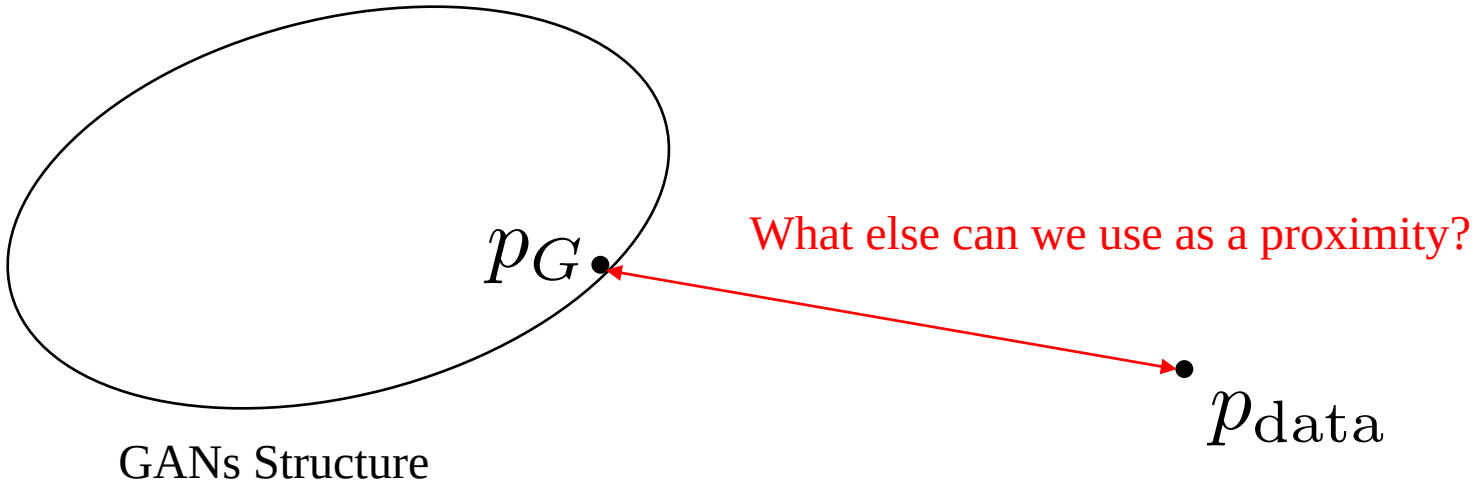
Generative Adversarial Networks



GANs Structure

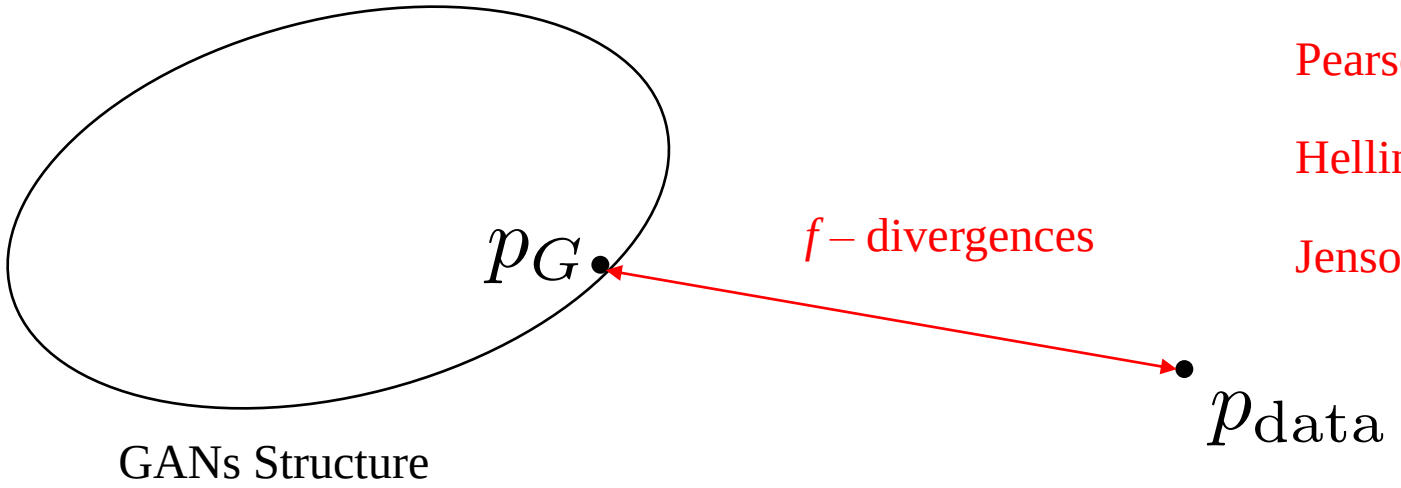
G is trained to minimize Jensen-Shannon divergence between data distribution and generator distribution!

Generative Adversarial Networks



GANs Structure

$f-$GAN



GANs Structure

Kullback-Leibler

Reverse Kullback-Leibler

Pearson chi-square

Hellinger

Jenson-Shannon

Generalized GANs from minimizing JS divergence to minimizing an arbitrary *f*-divergence

$f-$GAN

f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization

Sebastian Nowozin
Machine Intelligence and Perception Group
Microsoft Research
Cambridge, UK
Sebastian.Nowozin@microsoft.com

Botond Cseke
Machine Intelligence and Perception Group
Microsoft Research
Cambridge, UK
botcse@microsoft.com

Ryota Tomioka
Machine Intelligence and Perception Group
Microsoft Research
Cambridge, UK
ryoto@microsoft.com

Abstract

Generative neural samplers are probabilistic models that implement sampling using feedforward neural networks: they take a random input vector and produce a sample from a probability distribution defined by the network weights. These models are expressive and allow efficient computation of samples and derivatives, but cannot be used for computing likelihoods or for marginalization. The *generative-adversarial* training method allows to train such models through the use of an auxiliary discriminative neural network. We show that the generative-adversarial approach is a special case of an existing more general variational divergence estimation approach. We show that any f-divergence can be used for training generative neural samplers. We discuss the benefits of various choices of divergence functions on training complexity and the quality of the obtained generative models.

$f-$GAN

f-divergence

$$\mathcal{D}_f\left(p\|q\right)=\int q(x)f\left(\frac{p(x)}{q(x)}\right)dx,\ \text{s.t. } f(1)=0$$

$$\mathcal{D}_{\mathrm{KL}}\left(p\|q\right)=\int p(x)\log\left(\frac{p(x)}{q(x)}\right)dx\Longrightarrow f(u)=u\log u$$

$$\mathcal{D}_{\mathrm{RKL}}\left(p\|q\right)=\int q(x)\log\left(\frac{q(x)}{p(x)}\right)dx\Longrightarrow f(u)=-\log u$$

$$\mathcal{D}_{\mathrm{JS}}\left(p\|q\right)=\frac{1}{2}\int\left\{p(x)\log\left(\frac{2\cdot p(x)}{p(x)+q(x)}\right)+q(x)\log\left(\frac{2\cdot q(x)}{p(x)+q(x)}\right)\right\}dx$$

$$\Longrightarrow f(u)=-(u+1)\log\frac{u+1}{2}+u\log u$$

$f-$GAN

So, how can we train GANs with f-divergences?

We want to solve the following problem:

$$\min_{\theta_G} \mathcal{D}_f \left(p_{\text{data}} \| p_G\right)$$

$$\Longleftrightarrow \min_{\theta_G} \int p_G(x) f\left(\frac{p_{\text{data}}(x)}{p_G(x)}\right) dx$$

However, we do not know p_{data} and p_G,

but only have samples drawn from them, respectively

$f-$GAN

Variational Estimation of f-divergences

Estimating divergence functionals and the likelihood ratio by convex risk minimization

XuanLong Nguyen
Dept. of Statistical Science
Duke University
xuanlong.nguyen@stat.duke.edu

Martin J. Wainwright
Dept. of Statistics, and Dept. of EECS
University of California, Berkeley
wainwrig@stat.berkeley.edu

Michael I. Jordan
Dept. of Statistics, and Dept. of EECS
University of California, Berkeley
jordan@stat.berkeley.edu

Revised April 15, 2009

Technical Report 764
Department of Statistics
University of California, Berkeley

Abstract

We develop and analyze M-estimation methods for divergence functionals and the likelihood ratios of two probability distributions. Our method is based on a non-asymptotic variational characterization of f-divergences, which allows the problem of estimating divergences to be tackled via convex empirical risk optimization. The resulting estimators are simple to implement, requiring only the solution of standard convex programs. We present an analysis of consistency and convergence for these estimators. Given conditions only on the ratios of densities, we show that our estimators can achieve optimal minimax rates for the likelihood ratio and the divergence functionals in certain regimes. We derive an efficient optimization algorithm for computing our estimates, and illustrate their convergence behavior and practical viability by simulations.[1]

$f-$GAN

Fenchel (Convex) Conjugate of f at t: $f^*(t) = \sup_{u \in \mathrm{dom} f} \{ut - f(u)\}$

$$
\begin{aligned}
\mathcal{D}_f(p\|q) &= \int q(x) f\left(\frac{p(x)}{q(x)}\right) dx \\
&= \int q(x) \sup_{t \in \mathrm{dom} f} \left\{ t\frac{p(x)}{q(x)} - f^*(t) \right\} dx \\
&= \int \sup_{t \in \mathrm{dom} f} \{tp(x) - q(x) f^*(t)\} dx \\
&= \int \sup_{T \in \mathcal{T}} \{p(x)T(x) - q(x) f^*(T(x))\} dx \\
&\geq \sup_{T \in \mathcal{T}} \left\{ \int p(x)T(x)dx - \int q(x) f^*(T(x))dx \right\} \\
&= \sup_{T \in \mathcal{T}} \left\{ \mathbb{E}_{x \sim p(x)}[T(x)] - \mathbb{E}_{x \sim q(x)}[f^*(T(x))] \right\} \quad \text{the lower bound is tight if } T^*(x) = f'\left(\frac{p(x)}{q(x)}\right)
\end{aligned}
$$

$f-$GAN

$$\min \ \mathcal{D}_f\left(p \| q\right) \Longleftrightarrow \min \ \{\mathbb{E}_{x\sim p(x)}[T(x)] - \mathbb{E}_{x\sim q(x)}[f^*(T(x))]\}$$

$$\min_{\theta_G} \mathcal{D}_f\left(p_{\text{data}} \| p_G\right) \Longleftrightarrow \min_{\theta_G} \{\mathbb{E}_{x\sim p_{\text{data}}(x)}[T(x)] - \mathbb{E}_{x\sim p_G(x)}[f^*(T(x))]\}$$ *f*-GAN objective

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

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

$$= \min_{\theta_G} \mathbb{E}_{x\sim p_{\text{data}(x)}}[\log D^*_G(x)] + \mathbb{E}_{x\sim p_G(x)}[\log(1 - D^*_G(x))]$$ Vanilla GAN objective

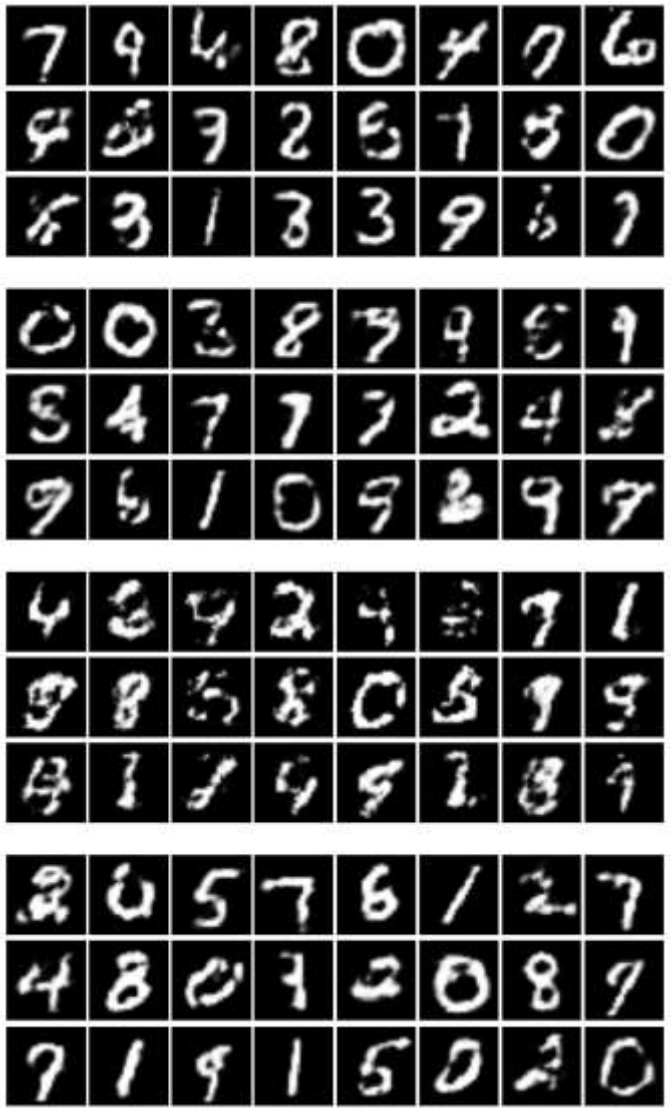
$f-$ GAN

$$\min_{\theta_G} \{\mathbb{E}_{x\sim p_{\text{data}}(x)}[T(x)] - \mathbb{E}_{x\sim p_G(x)}[f^*(T(x))]\}$$

Name	$D_f(P\|Q)$	Generator $f(u)$	$T^*(x)$
Kullback-Leibler	$\int p(x) \log \frac{p(x)}{q(x)} \mathrm{d}x$	$u \log u$	$1 + \log \frac{p(x)}{q(x)}$
Reverse KL	$\int q(x) \log \frac{q(x)}{p(x)} \mathrm{d}x$	$-\log u$	$-\frac{q(x)}{p(x)}$
Pearson χ^2	$\int \frac{(q(x)-p(x))^2}{p(x)} \mathrm{d}x$	$(u-1)^2$	$2(\frac{p(x)}{q(x)} - 1)$
Squared Hellinger	$\int \left(\sqrt{p(x)} - \sqrt{q(x)}\right)^2 \mathrm{d}x$	$(\sqrt{u}-1)^2$	$(\sqrt{\frac{p(x)}{q(x)}} - 1) \cdot \sqrt{\frac{q(x)}{p(x)}}$
Jensen-Shannon	$\frac{1}{2}\int p(x) \log \frac{2p(x)}{p(x)+q(x)} + q(x) \log \frac{2q(x)}{p(x)+q(x)} \mathrm{d}x$	$-(u+1)\log \frac{1+u}{2} + u \log u$	$\log \frac{2p(x)}{p(x)+q(x)}$
GAN	$\int p(x) \log \frac{2p(x)}{p(x)+q(x)} + q(x) \log \frac{2q(x)}{p(x)+q(x)} \mathrm{d}x - \log(4)$	$u \log u - (u+1)\log(u+1)$	$\log \frac{p(x)}{p(x)+q(x)}$

Name	Output activation g_f	dom_{f^*}	Conjugate $f^*(t)$	$f'(1)$
Kullback-Leibler (KL)	v	$\mathbb{R}$	$\exp(t-1)$	1
Reverse KL	$-\exp(-v)$	$\mathbb{R}_-$	$-1 - \log(-t)$	-1
Pearson χ^2	v	$\mathbb{R}$	$\frac{1}{4}t^2 + t$	0
Squared Hellinger	$1 - \exp(-v)$	$t < 1$	$\frac{t}{1-t}$	0
Jensen-Shannon	$\log(2) - \log(1+\exp(-v))$	$t < \log(2)$	$-\log(2-\exp(t))$	0
GAN	$-\log(1+\exp(-v))$	$\mathbb{R}_-$	$-\log(1-\exp(t))$	$-\log(2)$

f – GAN



Kullback-Leibler

Reverse Kullback-Leibler

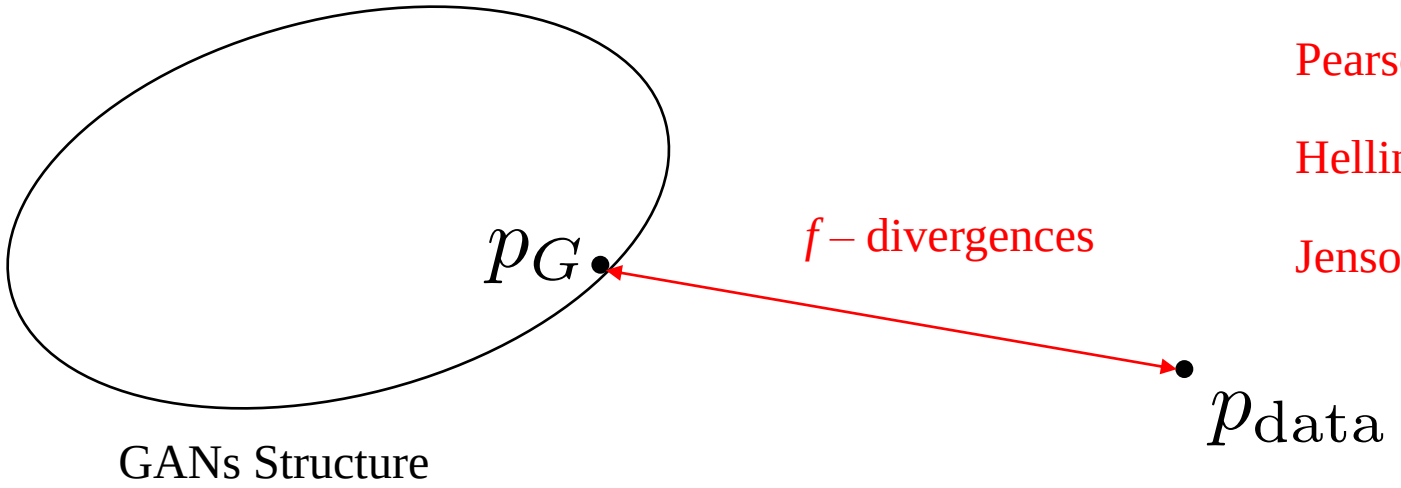
Squared Hellinger

Jensen-Shannon (Vanilla GAN)

f – GAN

(a) GAN (b) KL (c) Squared Hellinger

Wasserstein GAN



GANs Structure

Kullback-Leibler

Reverse Kullback-Leibler

Pearson chi-square

Hellinger

Jenson-Shannon

What else?

Wasserstein GAN

Wasserstein GAN

Martin Arjovsky[1], Soumith Chintala[2], and Léon Bottou[1,2]

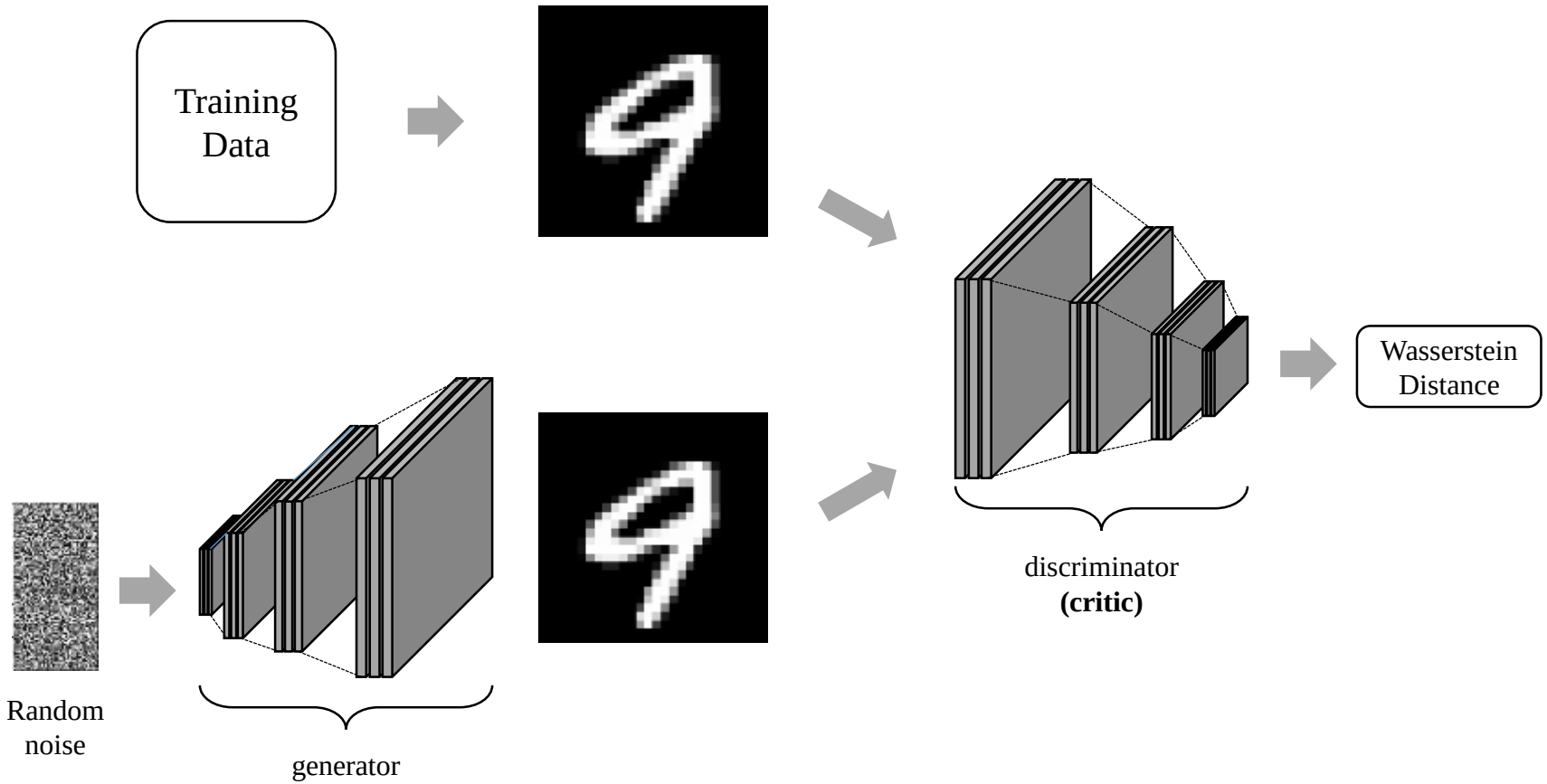
[1]Courant Institute of Mathematical Sciences
[2]Facebook AI Research

1 Introduction

The problem this paper is concerned with is that of unsupervised learning. Mainly, what does it mean to learn a probability distribution? The classical answer to this is to learn a probability density. This is often done by defining a parametric family of densities $(P_\theta)_{\theta \in \mathbb{R}^d}$ and finding the one that maximized the likelihood on our data: if we have real data examples $\{x^{(i)}\}_{i=1}^m$, we would solve the problem

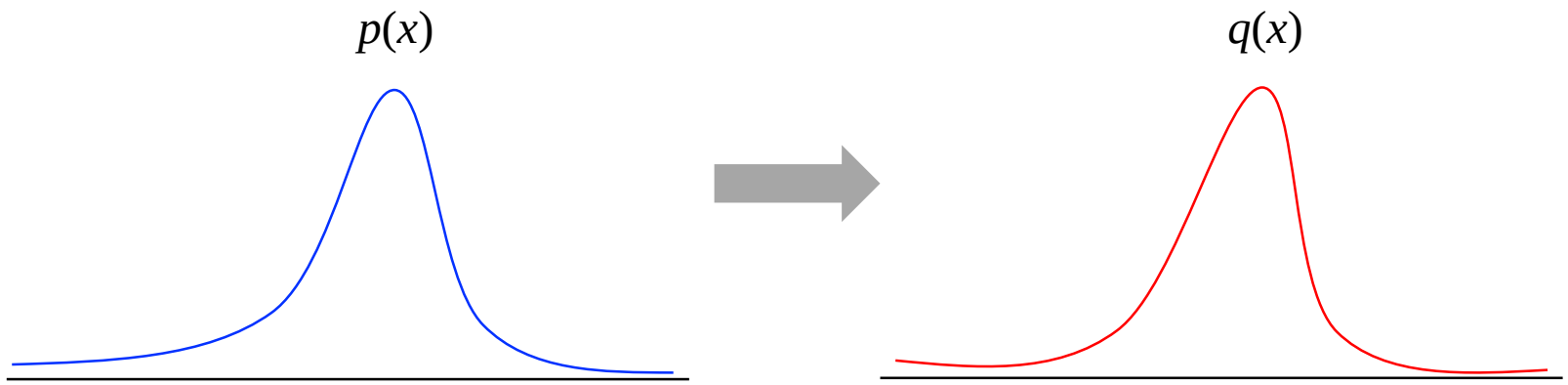
$$\max_{\theta \in \mathbb{R}^d} \frac{1}{m} \sum_{i=1}^{m} \log P_\theta(x^{(i)})$$

Wasserstein GAN

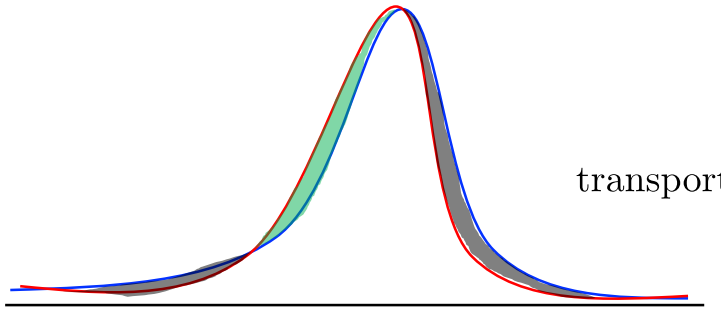


Wasserstein GAN

Optimal Transport Cost / Earth Mover's Distance



Regard $p(x)$ as a pile of earth, and we wish to transport the mass in such a way that it is transformed into the distribution $q(x)$



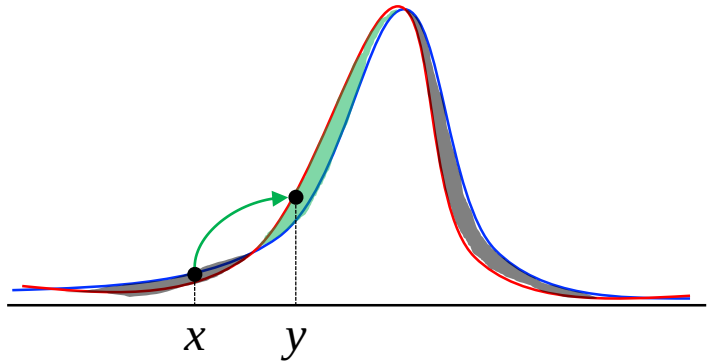
transportation plan: $\gamma(x, y) =$ amount of mass to move from x to y

We have to move the mass in gray region to green region

Wasserstein GAN

Optimal Transport Cost / Earth Mover's Distance

transportation plan: $\gamma(x, y) =$ amount of mass to move from x to y



Also, there is a cost to move the mass from *x* to *y* is given by a function

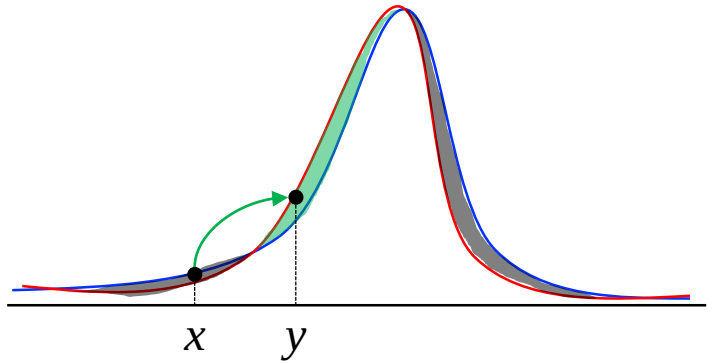
$$c(x, y) \mapsto [0, \infty)$$

$$c(x, y) = \|x - y\|_1$$

$$c(x, y) = \|x - y\|_2^2$$

Wasserstein GAN

Optimal Transport Cost / Earth Mover's Distance



Also, there is a cost to move the mass from *x* to *y* is given by a function

$$\text{cost}(\gamma; c) = \int \int c(x, y)\gamma(x, y)dxdy$$

$$\text{optimal cost}(\gamma; c) = \inf_{\gamma \in \Gamma(p,q)} \int \int c(x, y)\gamma(x, y)dxdy = \inf_{\gamma \in \Gamma(p,q)} \mathbb{E}_{(x,y)\in\gamma}[c(x, y)]$$

Wasserstein GAN

$$W_1(p_{\text{data}}, p_G) = \inf_{\gamma \in \Gamma(p_{\text{data}}, p_G)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|_1]$$

How can we minimize the loss function ???

By Kantorovich-Rubinstein duality, we have

$$W_1(p_{\text{data}}, p_G) = \sup_{\|f\|_L \leq K} \{\mathbb{E}_{x\sim p_{\text{data}}}[f(x)] - \mathbb{E}_{x\sim p_G}[f(x)]\}$$

Wasserstein GAN

$$W_1(p_{\text{data}}, p_G) = \sup_{\|f\|_L \leq K} \left\{ \mathbb{E}_{x \sim p_{\text{data}}(x)}[f(x)] - \mathbb{E}_{x \sim p_G(x)}[f(x)] \right\}$$

Applying generative adversarial network settings (G, D)

$$W_1(p_{\text{data}}, p_G) = \max_{\theta_D} \left\{ \mathbb{E}_{x \sim p_{\text{data}}(x)}[D(x)] - \mathbb{E}_{x \sim p_G(x)}[D(x)] \right\}$$

$$W_1(p_{\text{data}}, p_G) = \max_{\theta_D} \left\{ \mathbb{E}_{x \sim p_{\text{data}}(x)}[D(x)] - \mathbb{E}_{z \sim p(z)}[D(G(z))] \right\}$$

Wasserstein GAN

Now we have Wasserstein GAN problem that minimizes Wassersten-1 distance between data distribution and the generator distribution

$$\min_{\theta_G} W_1(p_{\text{data}}, p_G) = \min_{\theta_G} \max_{\theta_D} \left\{ \mathbb{E}_{x \sim p_{\text{data}}(x)}[D(x)] - \mathbb{E}_{z \sim p(z)}[D(G(z))] \right\}$$

Wasserstein GAN Problem

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

Vanilla GAN Problem

Wasserstein GAN

So, why Wassersten-1 distance?

$$\left.\begin{array}{r}\text{(Kullback-Leibler divergence) } \mathcal{D}_{\mathrm{KL}}(p_\theta, p) \to 0 \\ \text{(Reverse KL divergence) } \mathcal{D}_{\mathrm{KL}}(p, p_\theta) \to 0\end{array}\right\} \quad (1)$$

$$\left.\begin{array}{r}\text{(Jensen-Shannon divergence) } \mathcal{D}_{\mathrm{JS}}(p_\theta, p) \to 0 \\ \text{(Total variation distance) } \mathrm{TV}(p_\theta, p) \to 0\end{array}\right\} \quad (2)$$

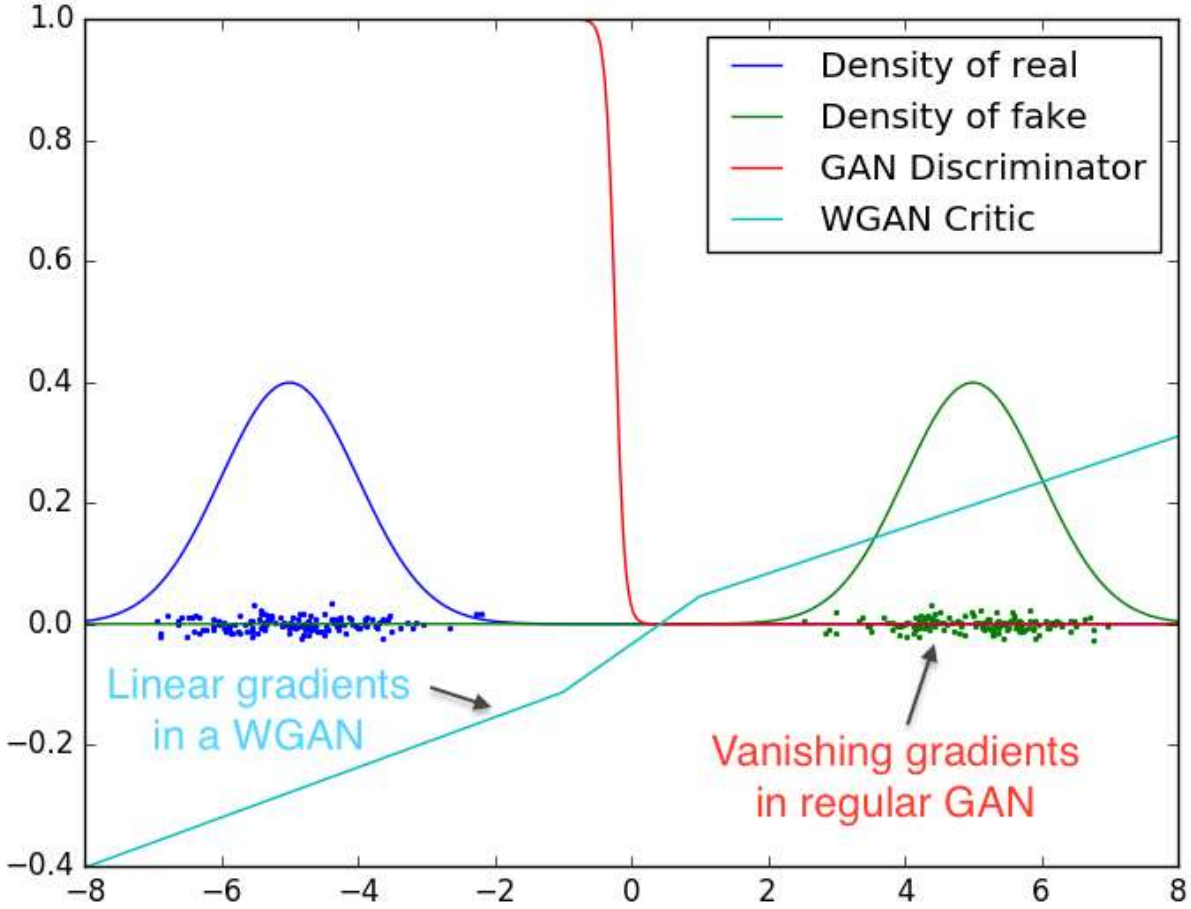
$$\text{(Wasserstein-1 distance) } W_1(p_\theta, p) \to 0 \quad (3)$$

(3) is the most strict condition
That is, we can find a case satisfying (1) or (2) but not satisfying (3)

Wasserstein GAN

So, why Wassersten-1 distance?

Optimal D's



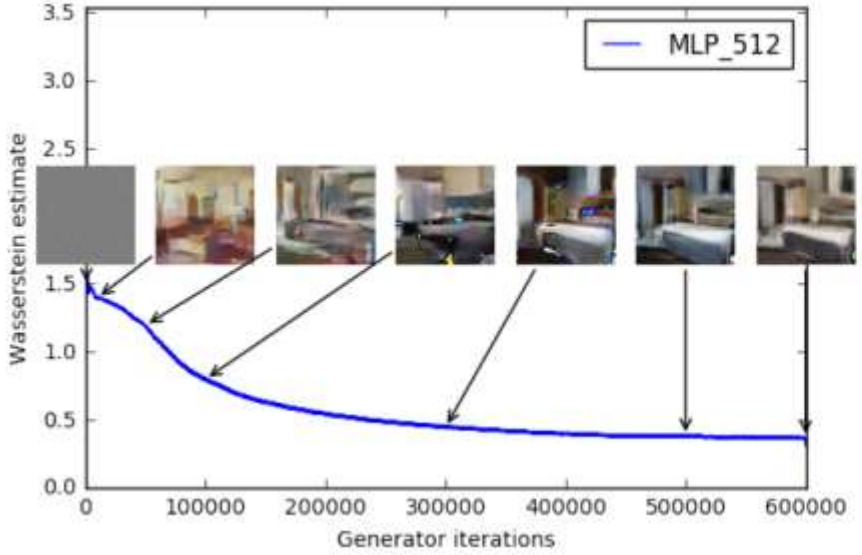
GAN D learns very quickly to distinguish real and fake
→ Optimal D is locally saturated and its gradient vanishes

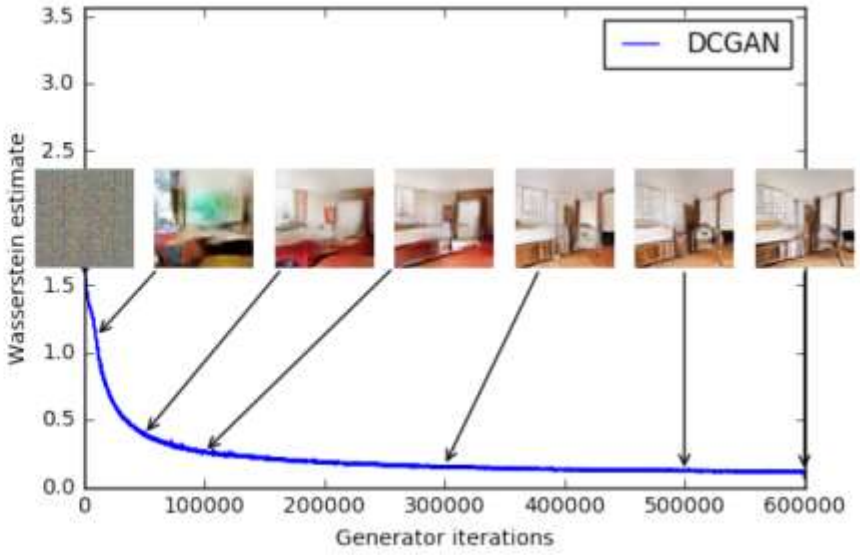
WGAN Critic does not saturate, and converges to a linear function
→ Optimal critic have no problem at all

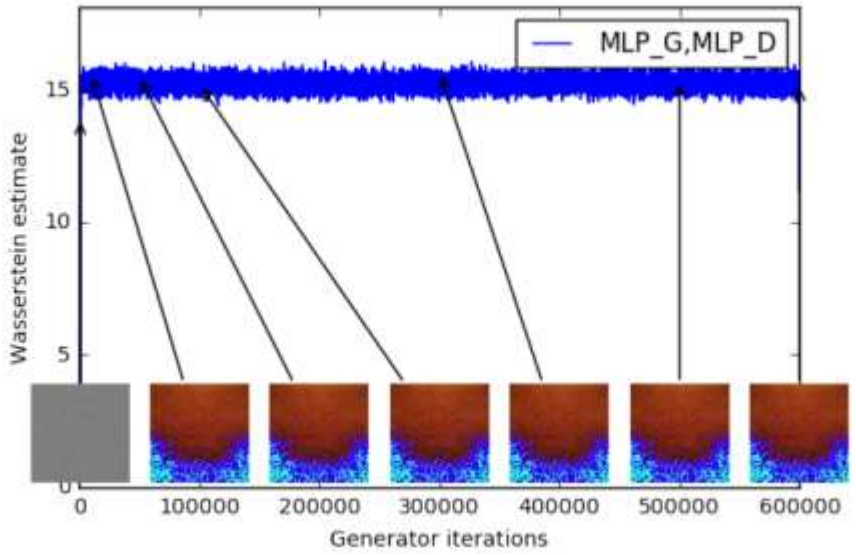
Additionally, using critic does not accompany mode collapse unlike GANs

Wasserstein GAN

Loss function value – Visual quality (Wasserstein-1)

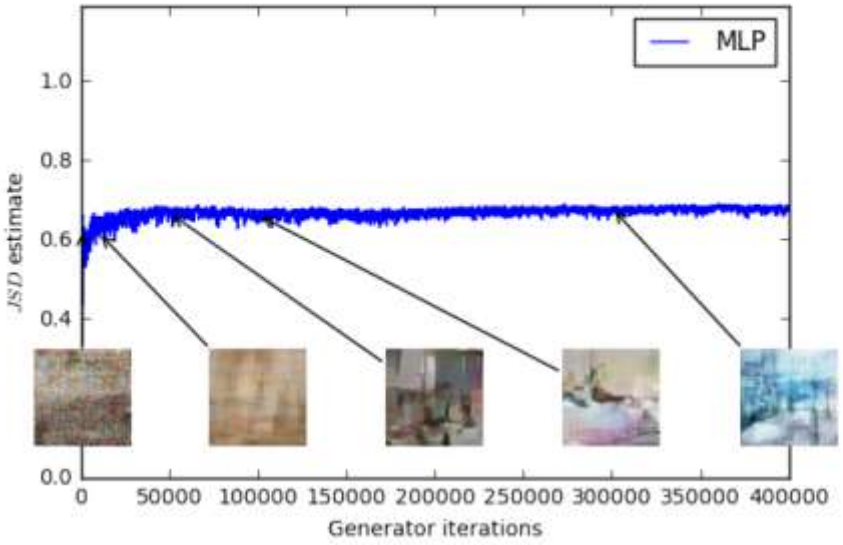


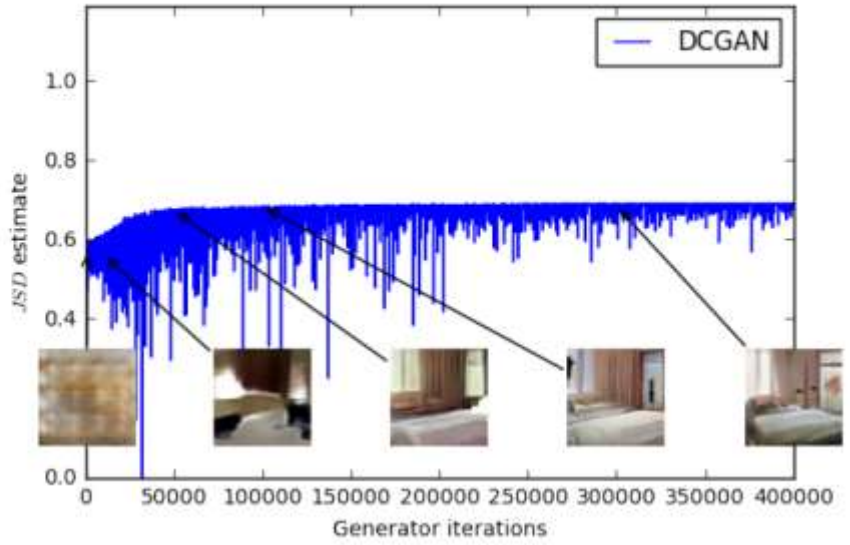


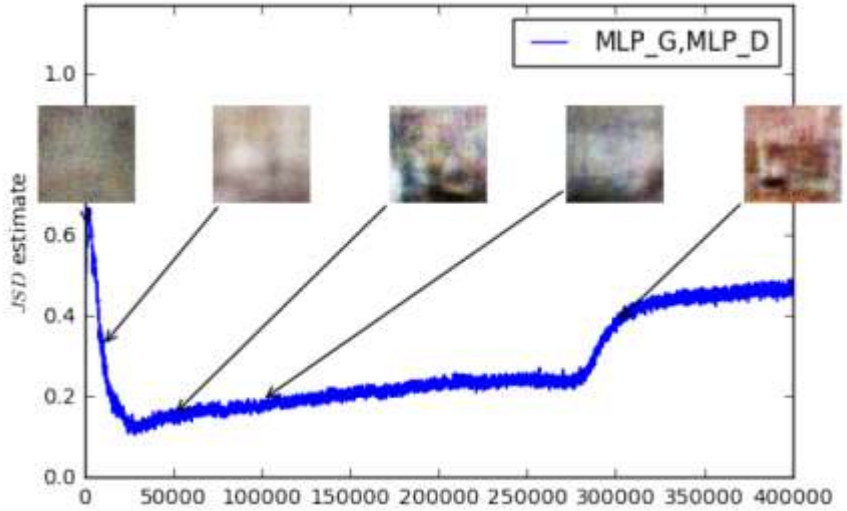


Wasserstein GAN

Loss function value – Visual quality (JSD)



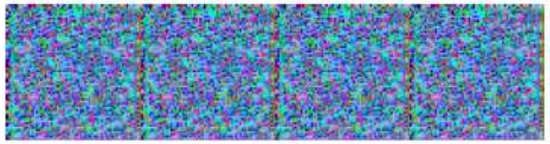




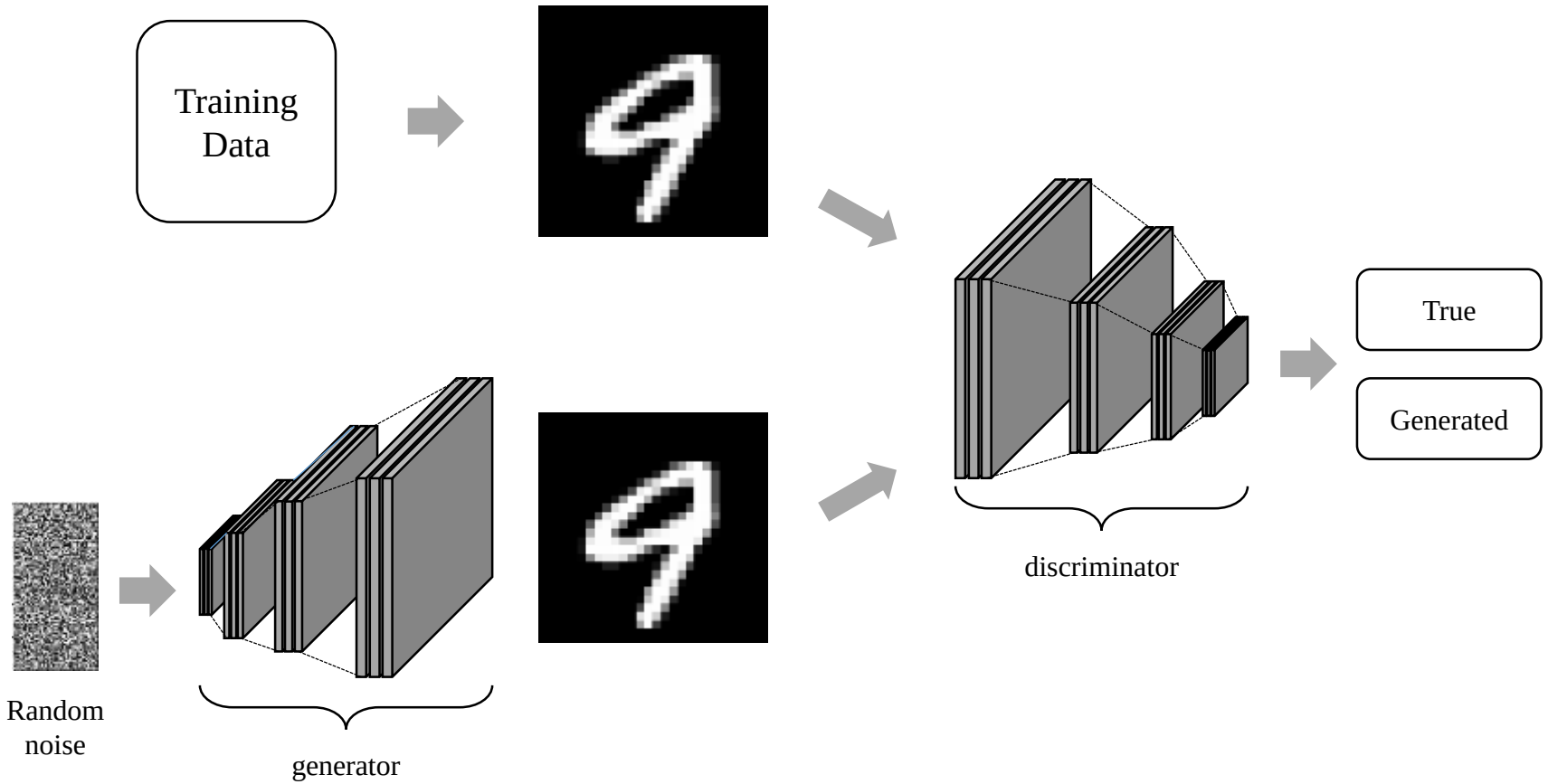
Wasserstein GAN

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			
			

Wasserstein GAN

DCGAN	LSGAN	WGAN (clipping)	WGAN-GP (ours)
Gated multiplicative nonlinearities everywhere in G and D			
			
tanh nonlinearities everywhere in G and D			
			
101-layer ResNet G and D			
			

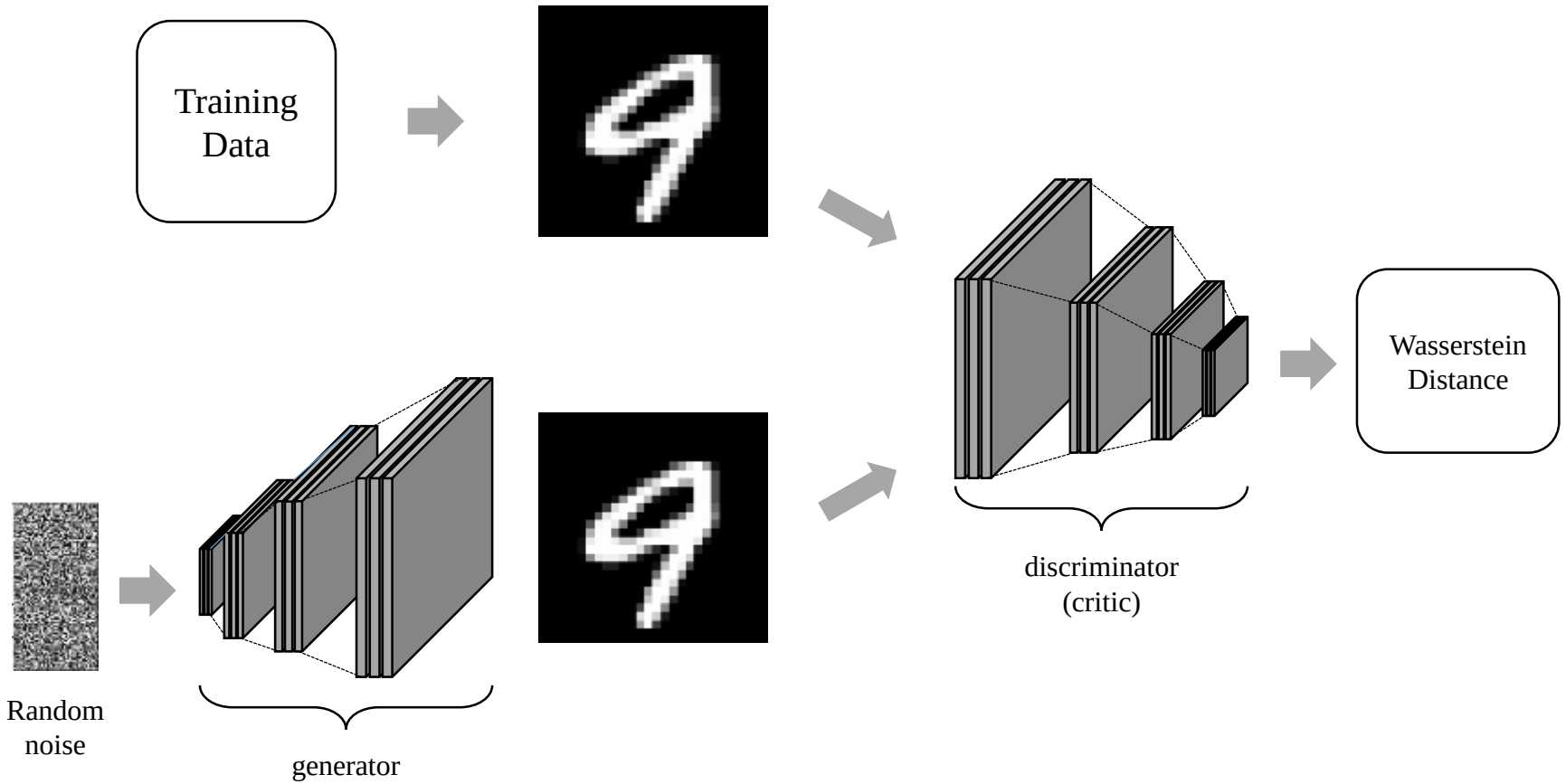
Energy-Based Generative Adversarial Networks



Vanilla GANs (JS Divergence)

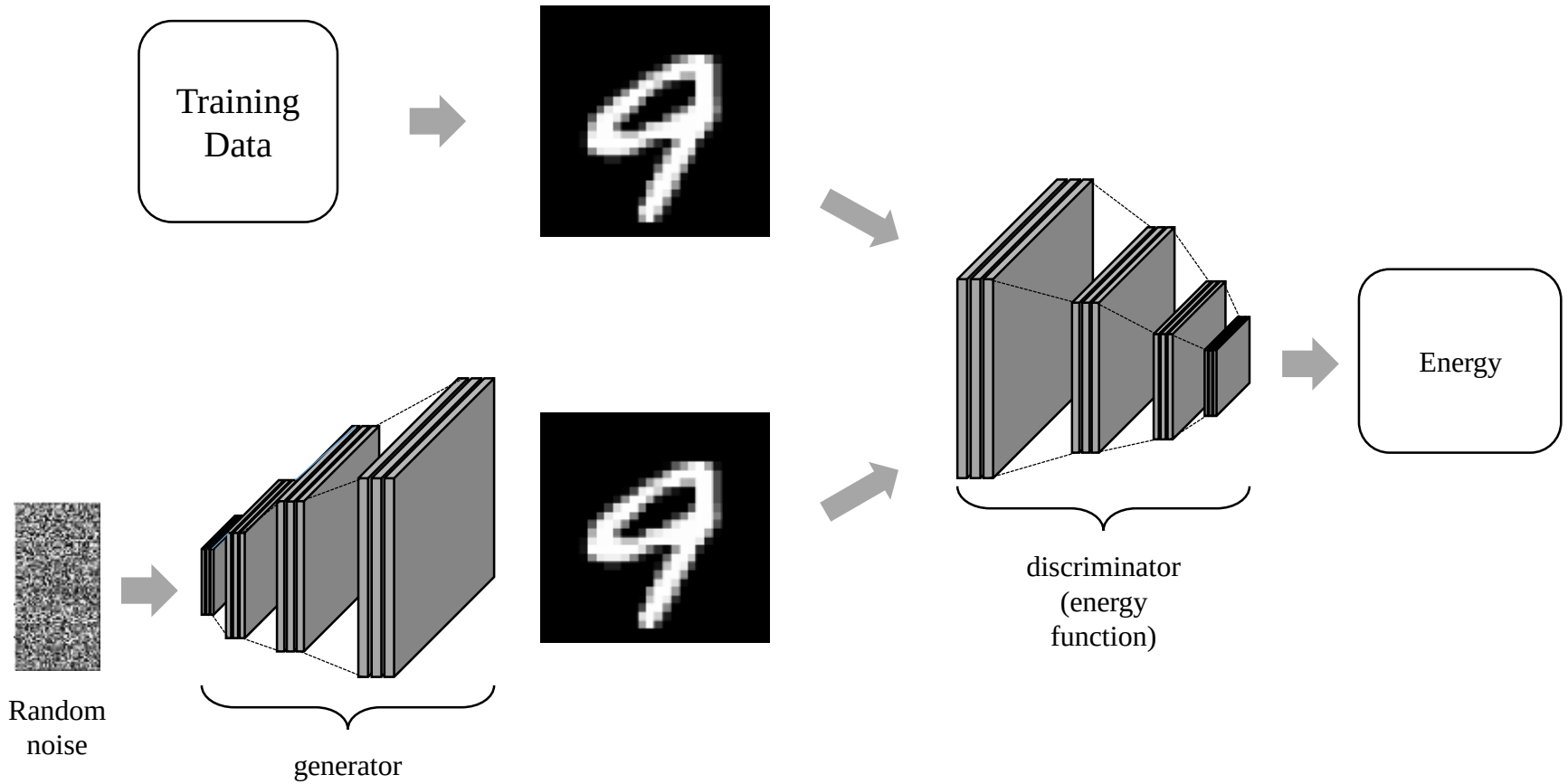
f–GANs (*f*–Divergence)

Energy-Based Generative Adversarial Networks



Wasserstein GANs (Wasserstein-1 Distance)

Energy-Based Generative Adversarial Networks



Energy-Based GANs (Energy)

Energy-Based Generative Adversarial Networks

Published as a conference paper at ICLR 2017

ENERGY-BASED GENERATIVE ADVERSARIAL NETWORKS

Junbo Zhao, Michael Mathieu and Yann LeCun
Department of Computer Science, New York University
Facebook Artificial Intelligence Research
{jakezhao, mathieu, yann}@cs.nyu.edu

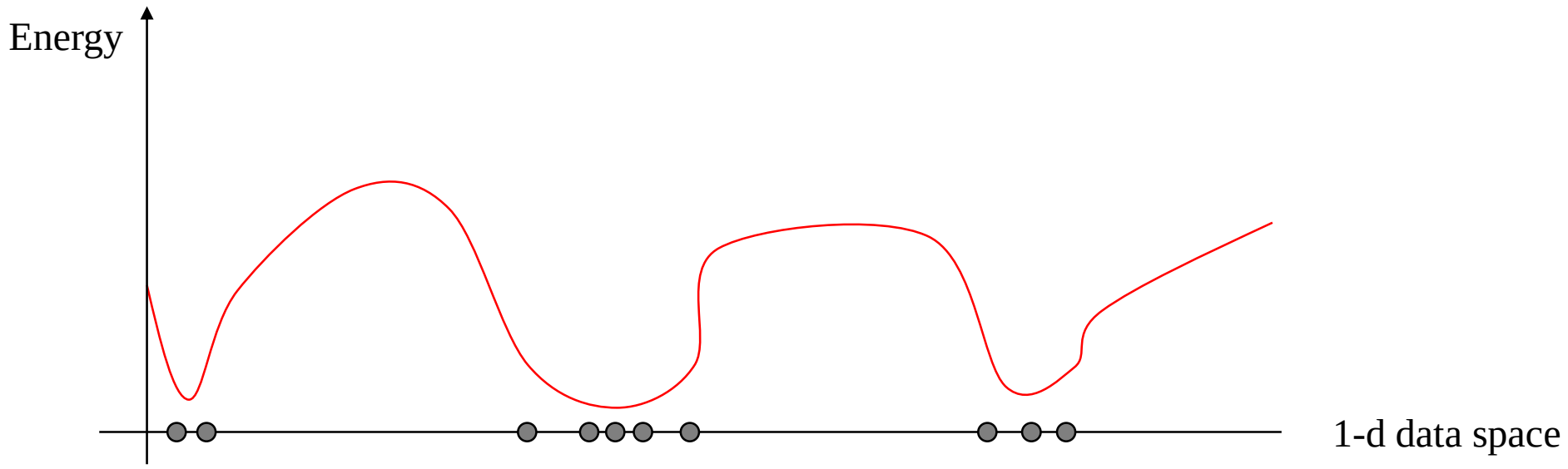
ABSTRACT

We introduce the "Energy-based Generative Adversarial Network" model (EBGAN) which views the discriminator as an energy function that attributes low energies to the regions near the data manifold and higher energies to other regions. Similar to the probabilistic GANs, a generator is seen as being trained to produce contrastive samples with minimal energies, while the discriminator is trained to assign high energies to these generated samples. Viewing the discriminator as an energy function allows to use a wide variety of architectures and loss functionals in addition to the usual binary classifier with logistic output. Among them, we show one instantiation of EBGAN framework as using an auto-encoder architecture, with the energy being the reconstruction error, in place of the discriminator. We show that this form of EBGAN exhibits more stable behavior than regular GANs during training. We also show that a single-scale architecture can be trained to generate high-resolution images.

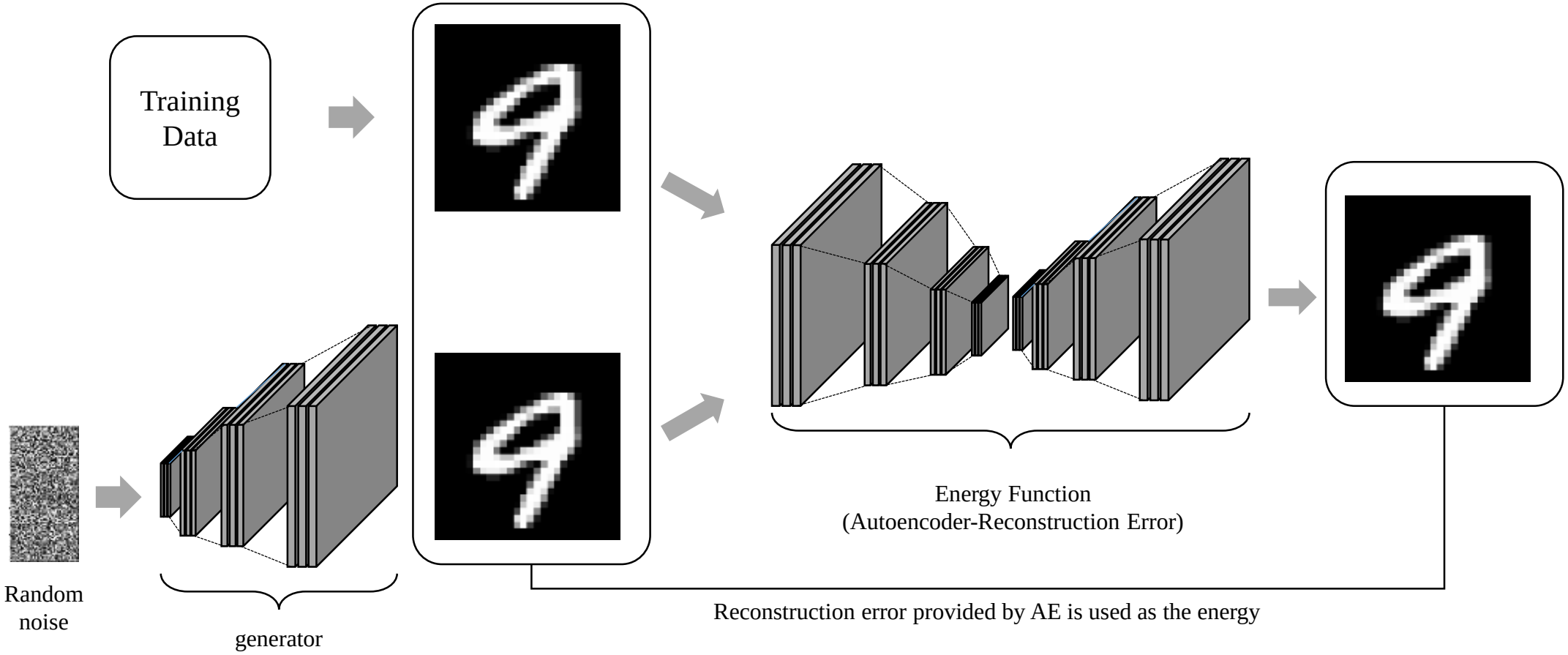
Energy-Based Generative Adversarial Networks

Energy Function: A function assigns energy (a scalar value) to every point of space (lower the better)

In unsupervised learning, low energy is assigned where data are observed



Energy-Based Generative Adversarial Networks



AE is trained to assign low energy (good reconstruction) to real data

AE is trained to assign high energy (bad reconstruction) to generated data

G is trained to assign low energy (good reconstruction) to generated data

Energy-Based Generative Adversarial Networks

AE is trained to assign low energy (good reconstruction) to real data

AE is trained to assign high energy (bad reconstruction) to generated data

G is trained to assign low energy (good reconstruction) to generated data

$$\min_{\theta_G} \max_{\theta_D} \mathbb{E}_{x\sim p_{\text{data}}(x)}[D(x)] + \mathbb{E}_{x\sim p_G(x)}[(m - D(x))_+] + \mathbb{E}_{x\sim p_G(x)}[D(x)]$$

$$\min_{\theta_G} \max_{\theta_D} \mathbb{E}_{x\sim p_{\text{data}}(x)}[D(x)] + \mathbb{E}_{z\sim p(z)}[(m - D(G(z)))_+] + \mathbb{E}_{z\sim p(z)}[D(G(z))]$$

For optimal D^* and G^*, $\int \mathbb{I}_{\{x:p_{\text{data}}(x)<p_{G^*}(x)\}}(p_{\text{data}}(x) - p_{G^*}(x))dx = 0$

that is, $p_{\text{data}} = p_{G^*}$ $a.e.$

Energy-Based Generative Adversarial Networks

DCGAN

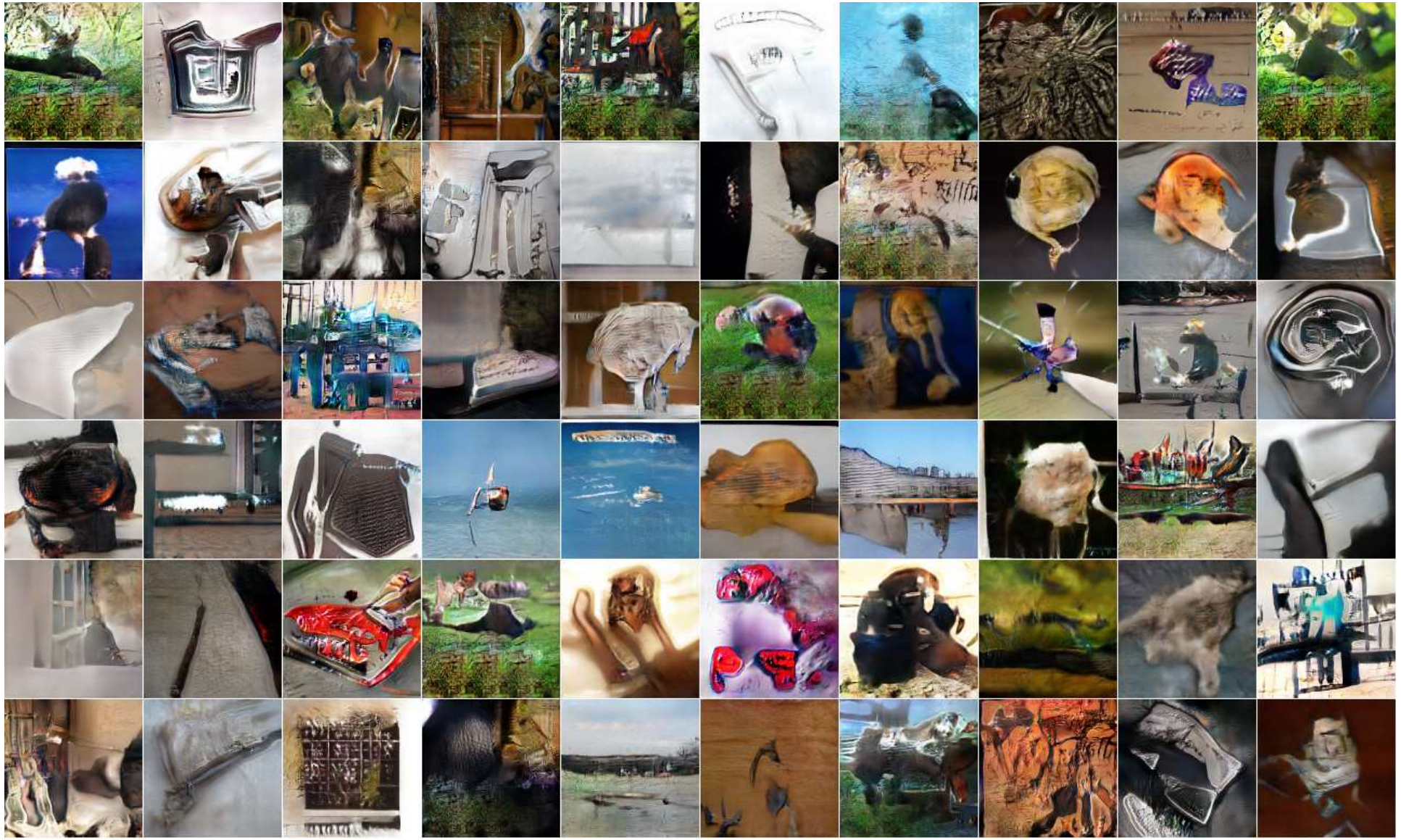
EBGAN

Energy-Based Generative Adversarial Networks

DCGAN

EBGAN

Energy-Based Generative Adversarial Networks



ImageNet 128 by 128

Energy-Based Generative Adversarial Networks

ImageNet 256 by 256

Boundary Equilibrium Generative Adversarial Networks

BEGAN: Boundary Equilibrium Generative Adversarial Networks

David Berthelot, Thomas Schumm, Luke Metz
Google
{dberth,fwiffo,lmetz}@google.com

Abstract

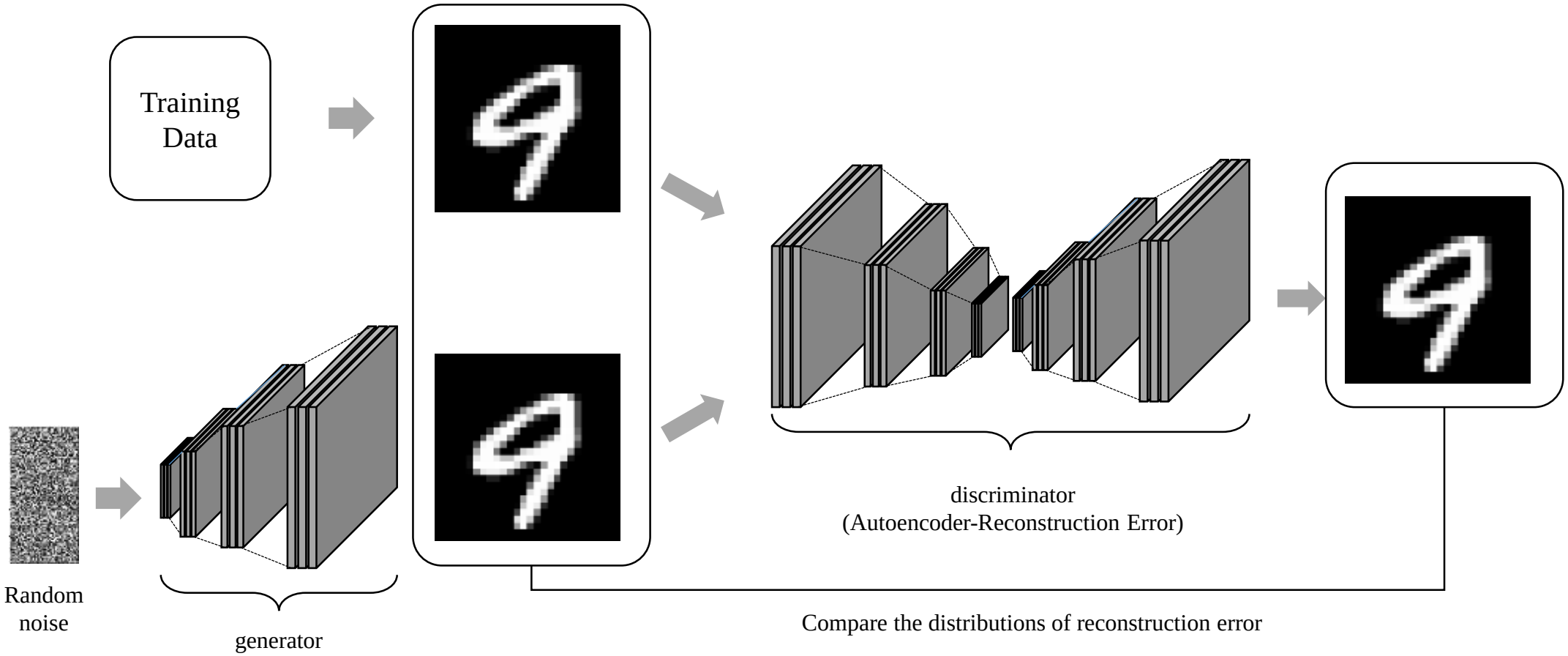
We propose a new equilibrium enforcing method paired with a loss derived from the Wasserstein distance for training auto-encoder based Generative Adversarial Networks. This method balances the generator and discriminator during training. Additionally, it provides a new approximate convergence measure, fast and stable training and high visual quality. We also derive a way of controlling the trade-off between image diversity and visual quality. We focus on the image generation task, setting a new milestone in visual quality, even at higher resolutions. This is achieved while using a relatively simple model architecture and a standard training procedure.

Boundary Equilibrium Generative Adversarial Networks

EBGAN (64 by 64)

BEGAN (128 by 128)

Boundary Equilibrium Generative Adversarial Networks



Boundary Equilibrium Generative Adversarial Networks

$$\mathcal{L}(x) = \|x - D(x)\|_1 \Longrightarrow \text{Reconstruction error of } x$$

$$\mathcal{L}(x) \sim q_{\text{data}},\ x \sim p_{\text{data}} \Longrightarrow \text{Error distribution of real samples}$$

$$\mathcal{L}(x) \sim q_{\text{G}},\ x \sim p_{\text{G}} \Longrightarrow \text{Error distribution of generated samples}$$

We want to make two error distributions similar

$$W_1(q_{\text{data}}, q_G) = \inf_{\gamma \in \Gamma(q_{\text{data}}, q_G)} \mathbb{E}_{(x,y)\sim\gamma}[|x - y|]$$

Boundary Equilibrium Generative Adversarial Networks

$$W_1(q_{\text{data}}, q_G) = \inf_{\gamma \in \Gamma(q_{\text{data}}, q_G)} \mathbb{E}_{(x,y)\sim\gamma}[|x - y|]$$

Instead of using the Kantorovich-Rubinstein duality, in BEGAN simply utilize lower bound

$$\inf \ \mathbb{E}[|x - y|] \geq \inf |\mathbb{E}[x - y]| = |\mathbb{E}[x] - \mathbb{E}[y]|$$

$$W_1(q_{\text{data}}, q_G) \geq |\mathbb{E}_{x\sim q_{\text{data}}}[x] - \mathbb{E}_{y\sim q_G}[y]|$$

Boundary Equilibrium Generative Adversarial Networks

$$W_1(q_{\text{data}}, q_G) \geq |\mathbb{E}_{x \sim q_{\text{data}}}[x] - \mathbb{E}_{y \sim q_G}[y]|$$

We want the discriminator to maximize the lower bound to maximize the Wassertein-1 distance

Two possible cases to maximize the lower bound

$$\begin{cases} W_1(q_{\text{data}}, q_G) \geq \mathbb{E}_{x \sim q_{\text{data}}}[x] - \mathbb{E}_{y \sim q_G}[y] \\ \mathbb{E}_{x \sim q_{\text{data}}}[x] \to \infty \\ \mathbb{E}_{y \sim q_G}[y] \to 0 \end{cases}$$

$$\begin{cases} W_1(q_{\text{data}}, q_G) \geq \mathbb{E}_{y \sim q_G}[y] - \mathbb{E}_{x \sim q_{\text{data}}}[x] \\ \mathbb{E}_{x \sim q_{\text{data}}}[x] \to 0 \\ \mathbb{E}_{y \sim q_G}[y] \to \infty \end{cases}$$

In BEGAN, the second case is selected since minimizing the reconstruction error of the real data naturally leads to autoencoder task

Boundary Equilibrium Generative Adversarial Networks

Finally we have,

D-loss $$\mathcal{L}_D = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|x - D(x)\|_1] - \mathbb{E}_{z \sim p(z)}[\|G(z) - D(G(z))\|_1]$$

G-loss $$\mathcal{L}_G = -\mathcal{L}_D$$

Similar to Wasserstein GANs, but two main differences

(1) BEGAN matches distributions between reconstruction errors not between samples

(2) No K-Lipschitz condition is necessary because of not using the Kantorovich-Rubinstein duality

→ Weight/gradient clipping not required

Boundary Equilibrium Generative Adversarial Networks

In practice it is crucial to maintain a balance between the generator and discriminator losses

In the paper, G and D are at equilibrium when

$$\mathbb{E}_{x \sim p_{\text{data}}(x)}[\mathcal{L}(x)] = \mathbb{E}_{z \sim p(z)}[\mathcal{L}(G(z))]$$

Moreover, the point of equilibrium can be tuned with a parameter gamma

$$\gamma = \frac{\mathbb{E}_{z \sim p(z)}[\mathcal{L}(G(z))]}{\mathbb{E}_{x \sim p_{\text{data}}(x)}[\mathcal{L}(x)]}$$

Boundary Equilibrium Generative Adversarial Networks

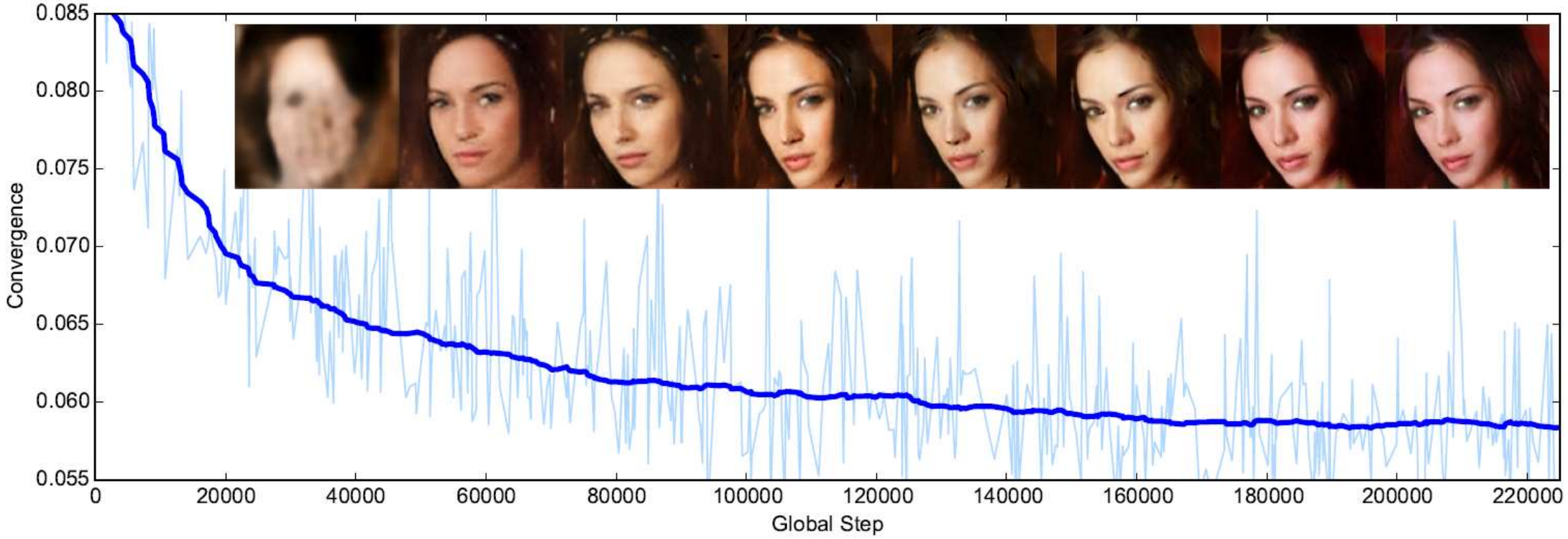
$$\gamma = \frac{\mathbb{E}_{z \sim p(z)}[\mathcal{L}(G(z))]}{\mathbb{E}_{x \sim p_{\text{data}}(x)}[\mathcal{L}(x)]}$$

(quality) 0 → 1 (diversity)

Boundary Equilibrium Generative Adversarial Networks

$$\mathcal{M}_{global} = \mathcal{L}(x) + |\gamma\mathcal{L}(x) - \mathcal{L}(G(z_G))|$$

Convergence measure – Visual quality



InfoGAN

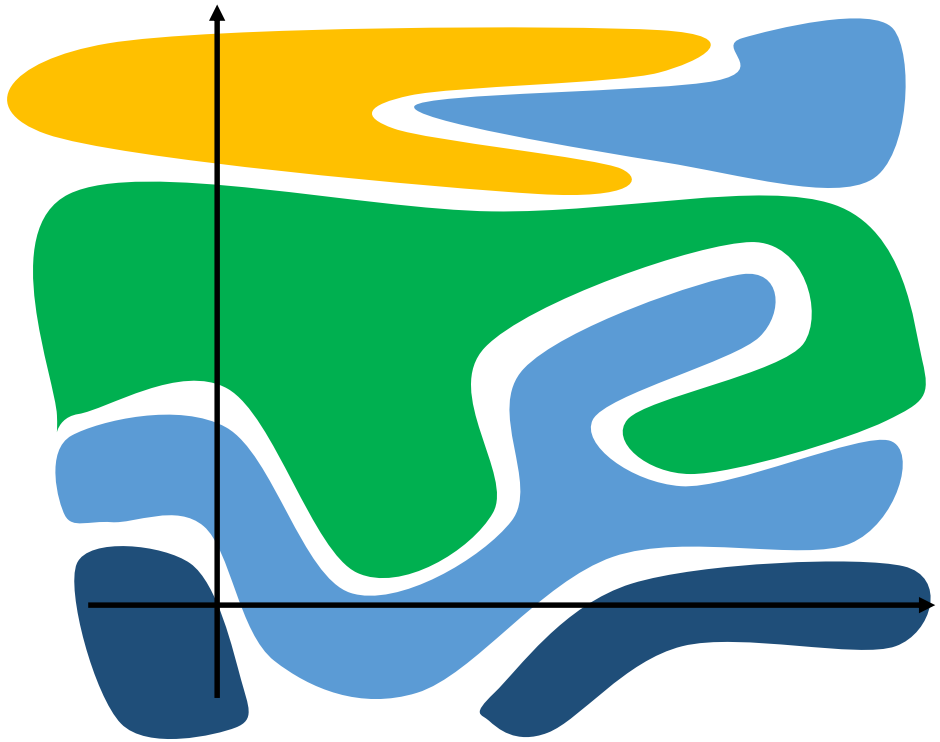
InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets

Xi Chen[†‡], **Yan Duan**[†‡], **Rein Houthooft**[†‡], **John Schulman**[†‡], **Ilya Sutskever**[‡], **Pieter Abbeel**[†‡]
† UC Berkeley, Department of Electrical Engineering and Computer Sciences
‡ OpenAI

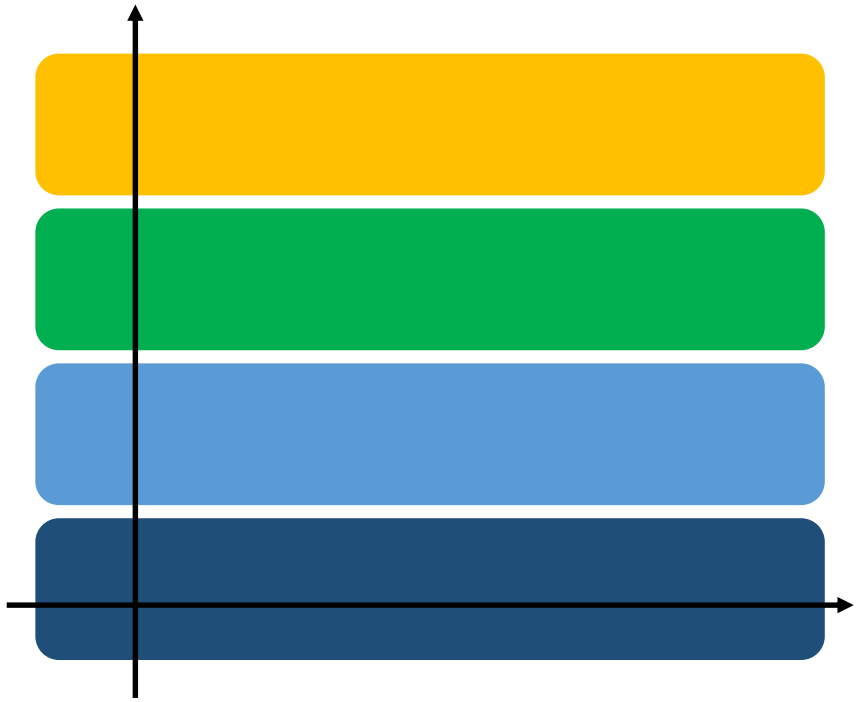
Abstract

This paper describes InfoGAN, an information-theoretic extension to the Generative Adversarial Network that is able to learn disentangled representations in a completely unsupervised manner. InfoGAN is a generative adversarial network that also maximizes the mutual information between a small subset of the latent variables and the observation. We derive a lower bound of the mutual information objective that can be optimized efficiently. Specifically, InfoGAN successfully disentangles writing styles from digit shapes on the MNIST dataset, pose from lighting of 3D rendered images, and background digits from the central digit on the SVHN dataset. It also discovers visual concepts that include hair styles, presence/absence of eyeglasses, and emotions on the CelebA face dataset. Experiments show that InfoGAN learns interpretable representations that are competitive with representations learned by existing supervised methods.

InfoGAN

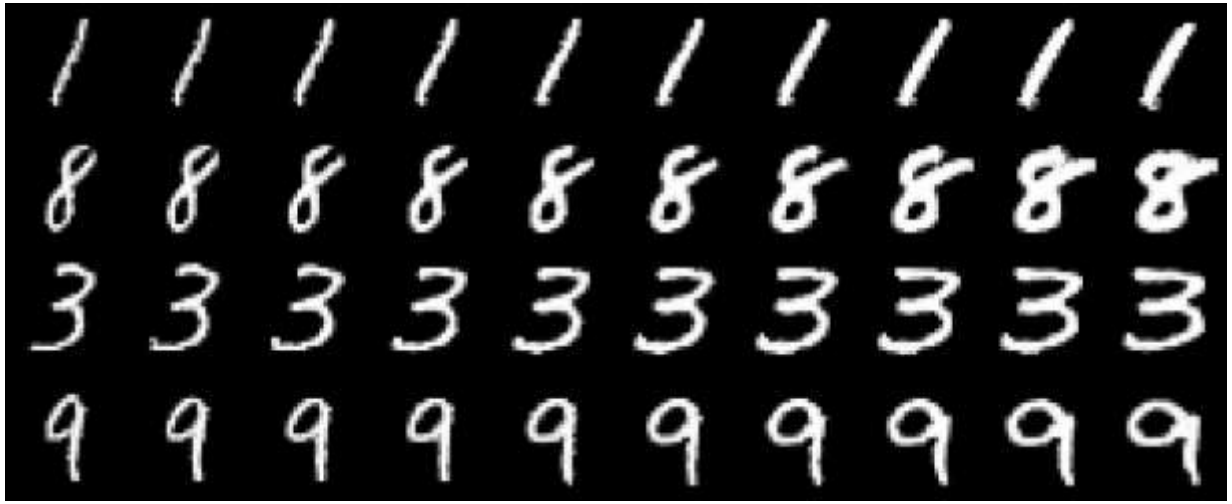


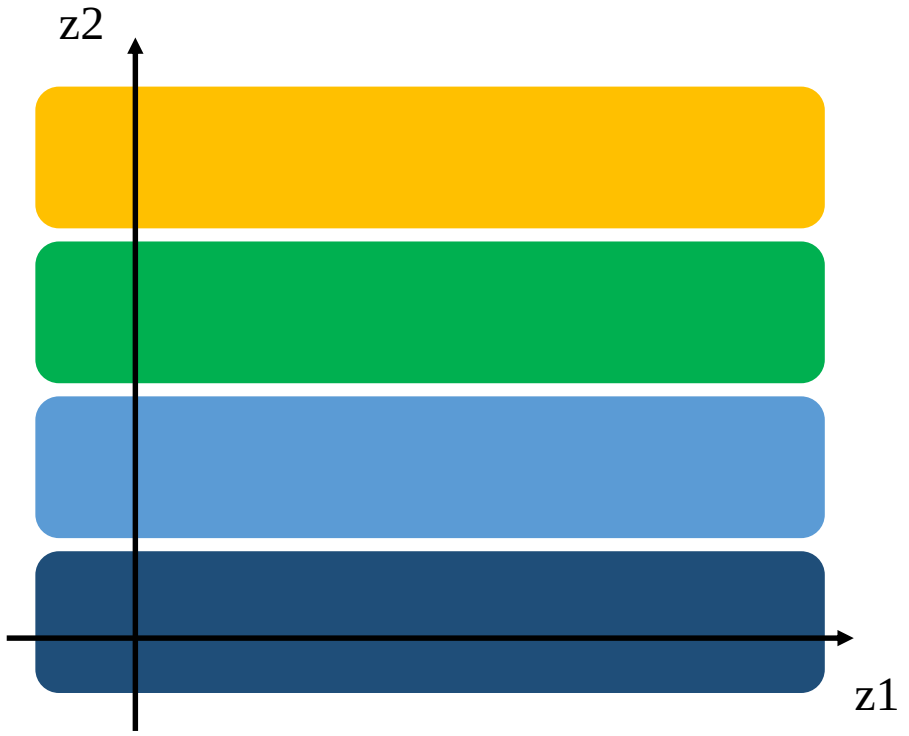
Entangled Representation (hard to interpret)



Disentangled Representation (easier to interpret)

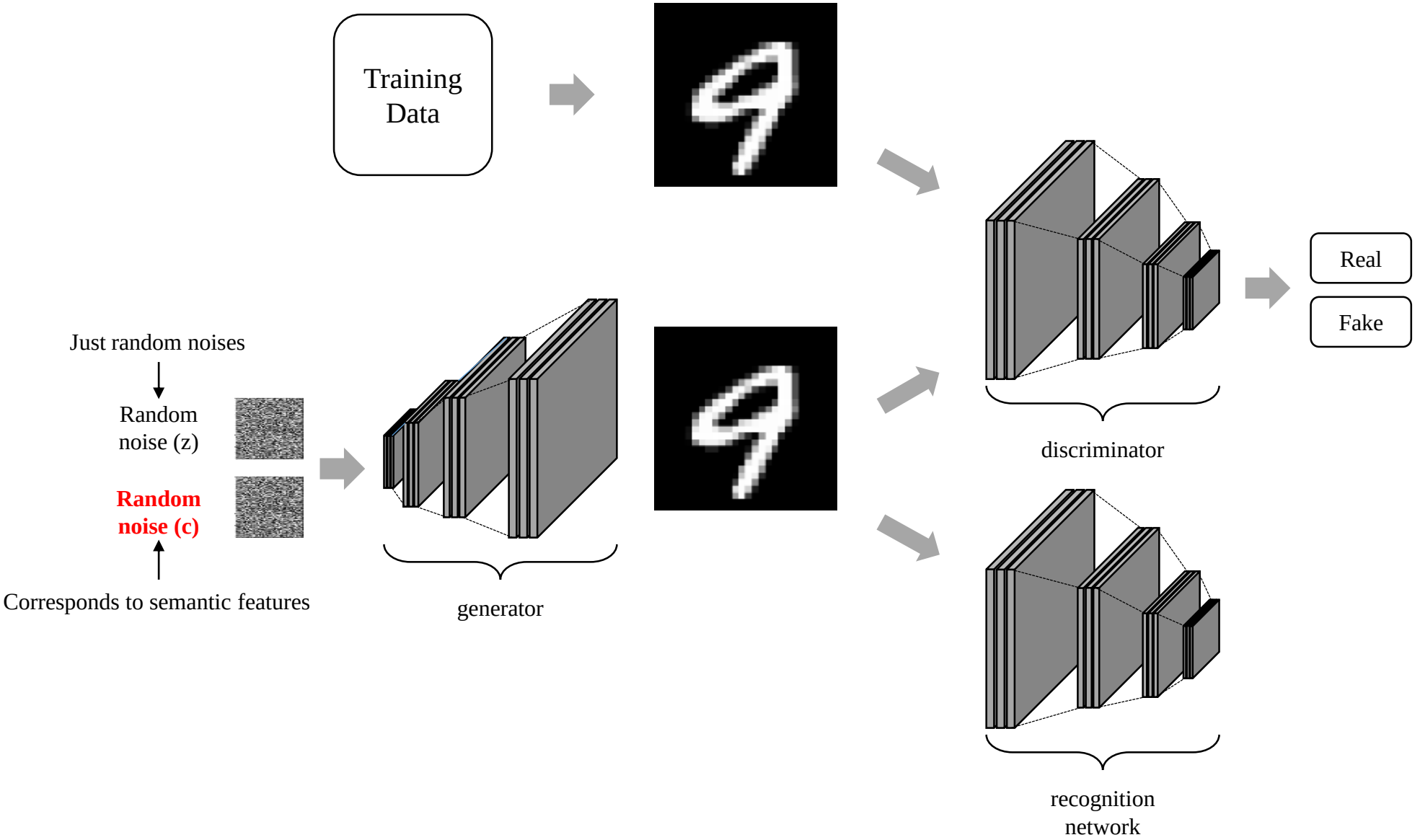
InfoGAN





Example:
z1 axis corresponds to the “width” of digits
z2 axis corresponds to the “type” of digits

InfoGAN


Training Data
Just random noises
Random noise (z)
Random noise (c)
Corresponds to semantic features
generator
discriminator
Real
Fake
recognition network

InfoGAN

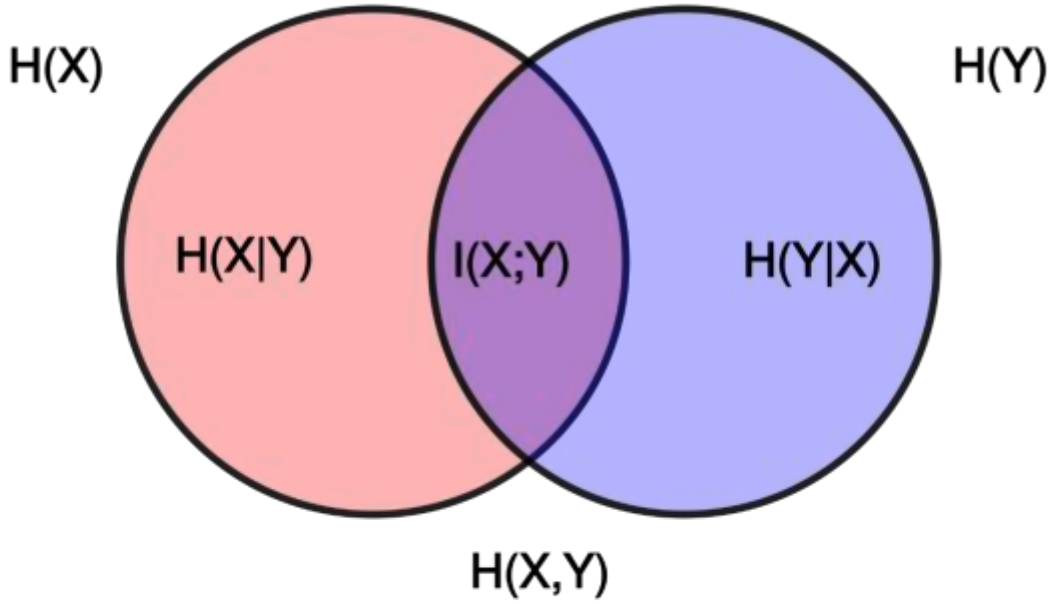
$$\min_{\theta_G} \max_{\theta_D} \mathbb{E}_{x \sim p_{\text{data}(x)}}[\log D(x)] + \mathbb{E}_{z \sim p(z)}[\log(1 - D(G(z)))]$$

$$\min_{\theta_G} \max_{\theta_D} \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p(z),\ c \sim p(c)}[\log(1 - D(G(z, c)))]$$

InfoGAN

Mutual Information: Measure of dependence between two random variables
→ amount of information obtained about *X* through observing *Y*

$$I(X;Y) = H(Y) - H(Y|X) = H(X) - H(X|Y)$$



InfoGAN

InfoGAN tries to maximize the mutual information of the latent code *c* and the output of *G*

$$\min_{\theta_G} \max_{\theta_D} \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p(z),\ c \sim p(c)}[\log(1 - D(G(z,c)))] - \lambda \cdot I(c; G(z,c))$$

$$I(c; G(z,c)) = H(G(z,c)) - H(G(z,c); c) = H(c) - H(c; G(z,c))$$

Have no idea of probability density of *G*(*z*,*c*), but we know *p*(*c*)

InfoGAN

Variational Mutual Information Maximization

→ We can obtain a lower bound of the posterior $p(c|\underline{G}(z,c))$

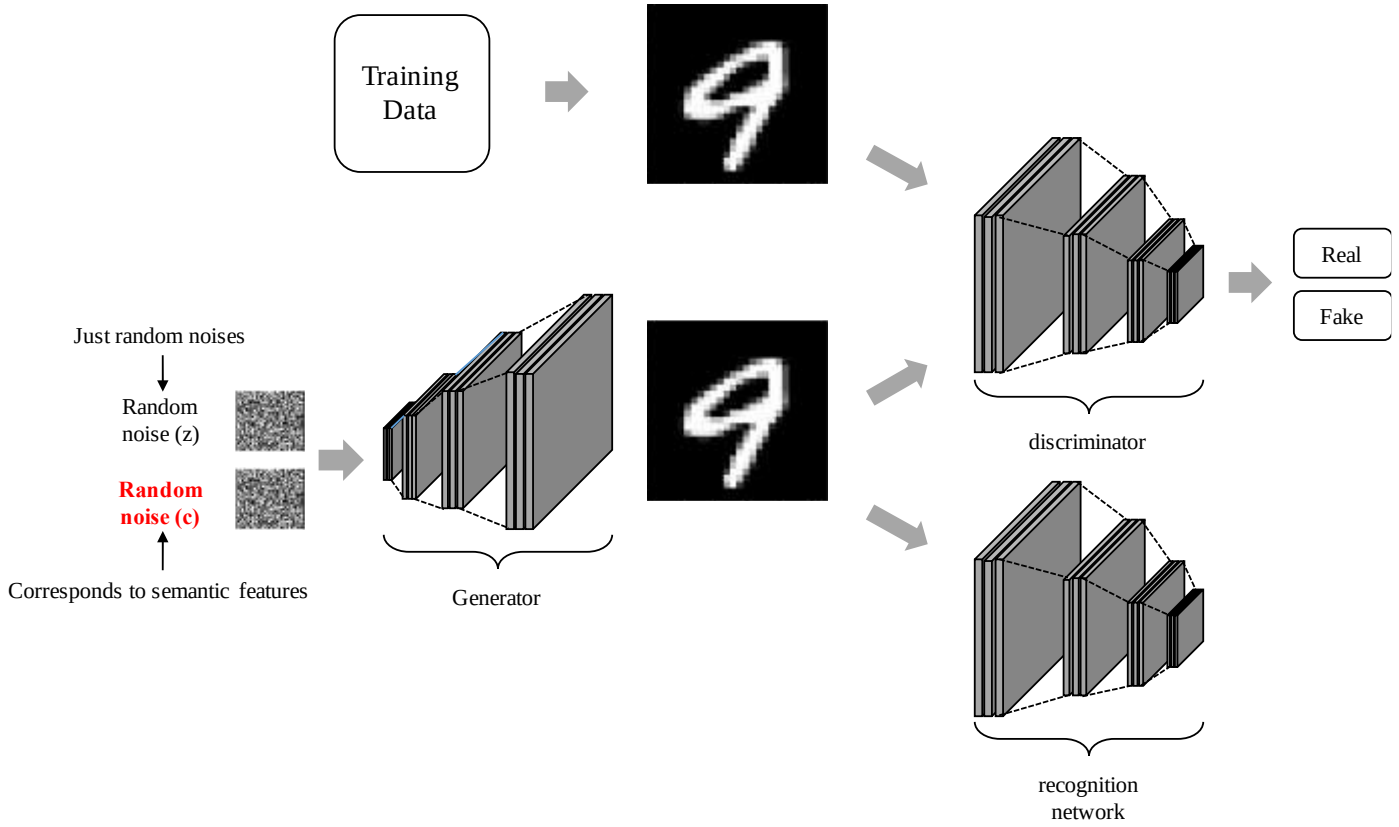
Let $q(c|x)$ be an approximation of $p(c|x)$

$$
\begin{aligned}
I(c; G(z,c)) &= -H(c|G(z,c)) + H(c) \\
&= \mathbb{E}_{x\sim G(z,c)}\mathbb{E}_{c'\sim p(c|x)}[\log p(c'|x)] + H(c) \\
&= \mathbb{E}_{x\sim G(z,c)}\left[\mathcal{D}_{\mathrm{KL}}(p(c|x)||q(c|x)) + \mathbb{E}_{c'\sim p(c|x)}[\log q(c'|x)]\right] + H(c) \\
&\geq \mathbb{E}_{x\sim G(z,c)}\mathbb{E}_{c'\sim p(c|x)}[\log q(c'|x)] + H(c) \\
&= \mathbb{E}_{c\sim p(c),\ x\sim G(z,c)} \log q(c|x) + H(c) \\
&= L_I(G,q)
\end{aligned}
$$

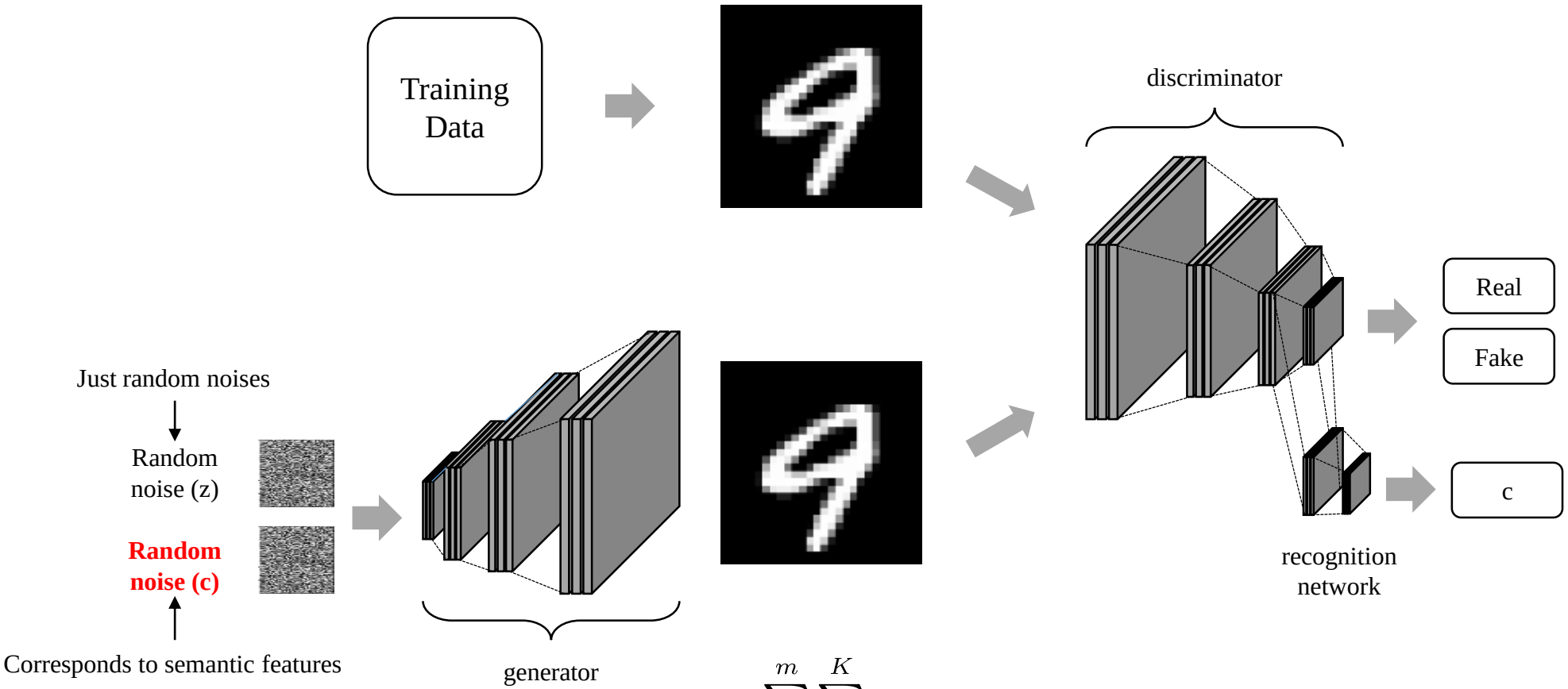
q is parameterized by an additional neural network

InfoGAN

$$\min_{\theta_G, \theta_q} \max_{\theta_D} \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p(z),\ c \sim p(c)}[\log(1 - D(G(z, c)))] - \lambda \cdot L_I(G, q)$$



InfoGAN

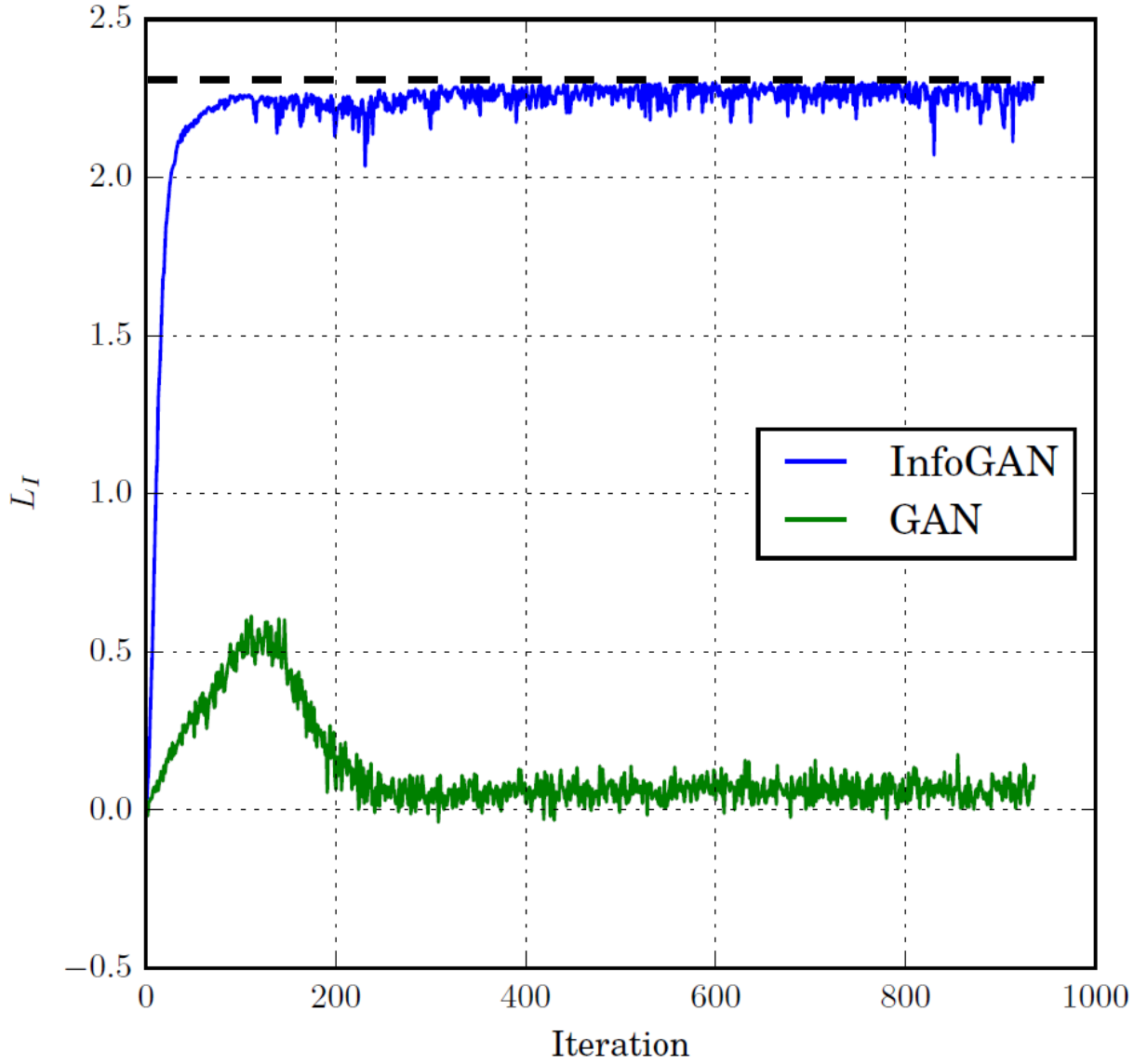


$$\mathbb{E}_{c\sim p(c),\ x\sim G(z,c)} \log q(c|x)$$

$$\sum_{i=1}^{m}\sum_{k=1}^{K} c_{\text{gen},i,k} \log c_{q,i,k} \ \text{(Cross-Entropy)}$$

$$\sum_{i=1}^{m} (c_{\text{gen},i} - c_{q,i})^2 \ \text{(MSE)}$$

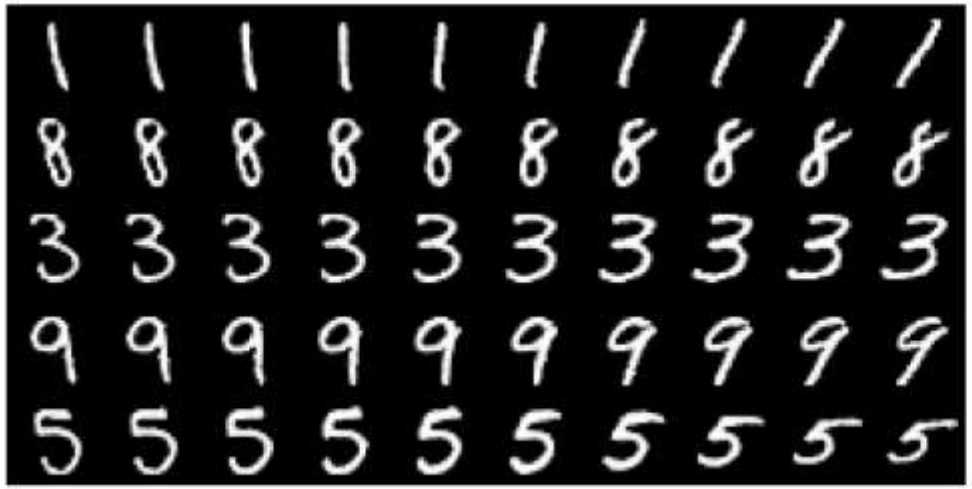
InfoGAN


2.5
2.0
1.5
1.0
0.5
0.0
−0.5
L_I
0
200
400
600
800
1000
Iteration
InfoGAN
GAN

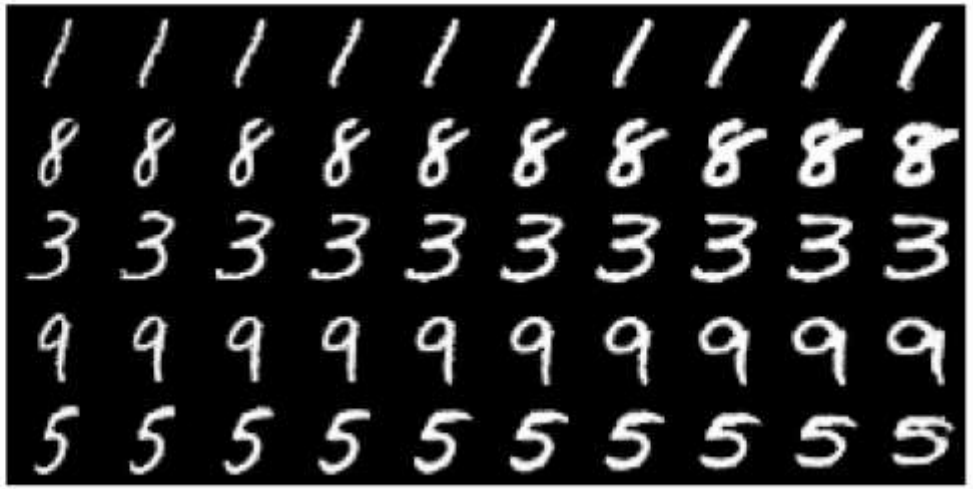
InfoGAN

(a) Varying c_1 on InfoGAN (Digit type)

(b) Varying c_1 on regular GAN (No clear meaning)



(c) Varying c_2 from -2 to 2 on InfoGAN (Rotation)



(d) Varying c_3 from -2 to 2 on InfoGAN (Width)

InfoGAN

(a) Azimuth (pose)

(b) Elevation

(c) Lighting

(d) Wide or Narrow

Generative Moment Matching Networks

Generative Moment Matching Network

Generative Moment Matching Networks

Yujia Li[1] YUJIALI@CS.TORONTO.EDU
Kevin Swersky[1] KSWERSKY@CS.TORONTO.EDU
Richard Zemel[1,2] ZEMEL@CS.TORONTO.EDU
[1]Department of Computer Science, University of Toronto, Toronto, ON, CANADA
[2]Canadian Institute for Advanced Research, Toronto, ON, CANADA

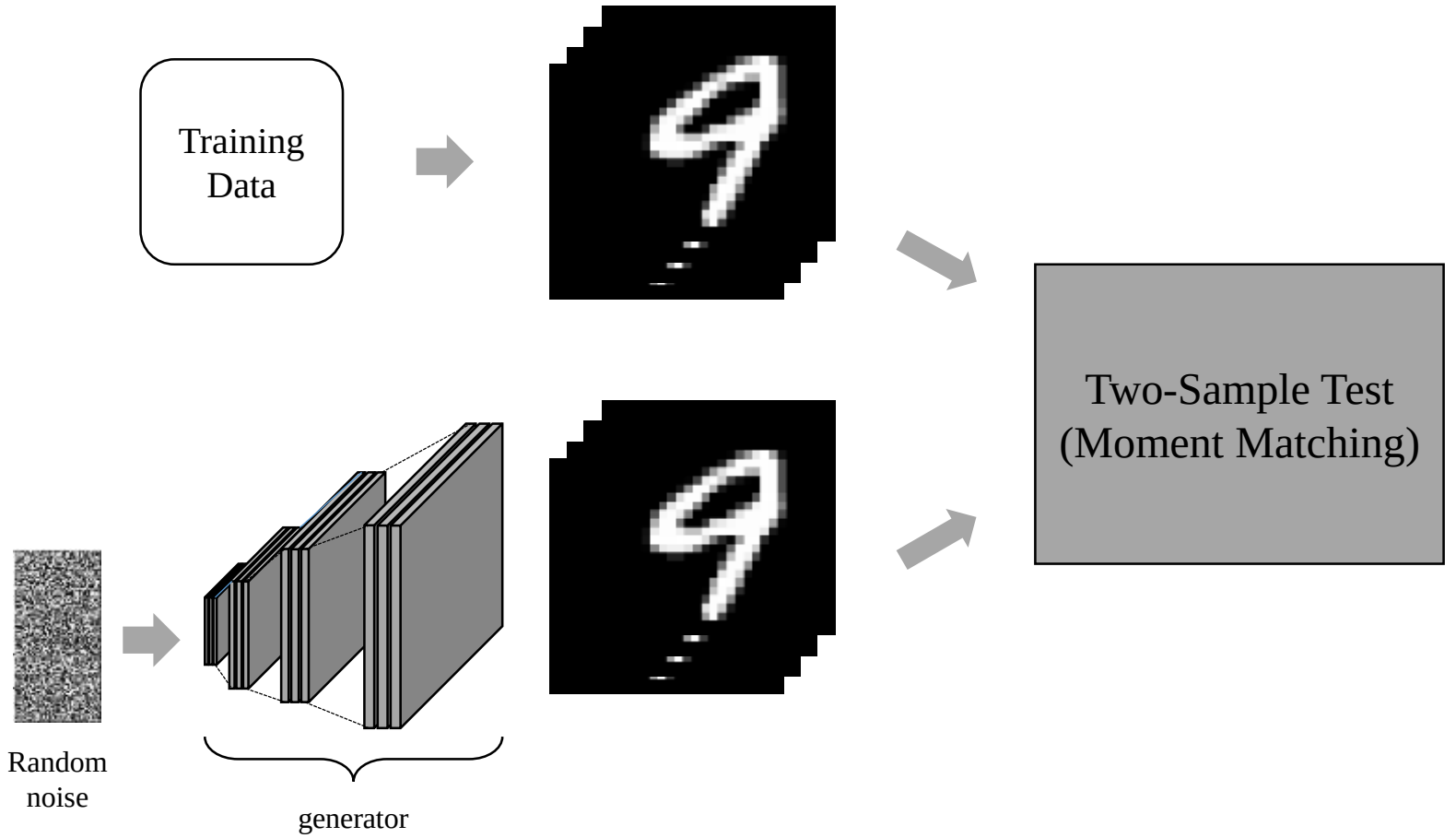
Abstract

We consider the problem of learning deep generative models from data. We formulate a method that generates an independent sample via a single feedforward pass through a multilayer preceptron, as in the recently proposed generative adversarial networks (Goodfellow et al., 2014). Training a generative adversarial network, however, requires careful optimization of a difficult minimax program. Instead, we utilize a technique from statistical hypothesis testing known as maximum mean discrepancy (MMD), which leads to a simple objective that can be interpreted as matching all orders of statistics between a dataset and samples from the model, and can be trained by backpropagation. We further boost the performance of this approach by combining our generative network with an auto-encoder network, using MMD to learn to generate codes that can then be decoded to produce samples. We show that the combination of these techniques yields excellent generative models compared to baseline approaches as measured on MNIST and the Toronto Face Database.

Kiros et al., 2014), machine translation (Cho et al., 2014; Sutskever et al., 2014), and more. Despite their successes, one of the main bottlenecks of the supervised approach is the difficulty in obtaining enough data to learn abstract features that capture the rich structure of the data. It is well recognized that a promising avenue is to use unsupervised learning on unlabelled data, which is far more plentiful and cheaper to obtain.

A long-standing and inherent problem in unsupervised learning is defining a good method for evaluation. Generative models offer the ability to evaluate generalization in the data space, which can also be qualitatively assessed. In this work we propose a generative model for unsupervised learning that we call generative moment matching networks (GMMNs). GMMNs are generative neural networks that begin with a simple prior from which it is easy to draw samples. These are propagated deterministically through the hidden layers of the network and the output is a sample from the model. Thus, with GMMNs it is easy to quickly draw independent random samples, as opposed to expensive MCMC procedures that are necessary in other models such as Boltzmann machines (Ackley et al., 1985; Hinton, 2002; Salakhutdinov & Hinton, 2009). The structure of a GMMN is most analogous to the recently pro-

Generative Moment Matching Network



Generative Moment Matching Network

Maximum Mean Discrepancy

$$\begin{matrix} x_1, \cdots, x_n \sim p_x \\ y_1, \cdots, y_m \sim p_y \end{matrix} \implies \text{want to test } p_x = p_y$$

$$\mathcal{L}_{\text{MMD}}(p_x, p_y; f) = \left\| \frac{1}{n} \sum_{i=1}^{n} f(x_i) - \frac{1}{m} \sum_{i=1}^{m} f(y_i) \right\|_2^2$$

$$\text{if } f(x) = x \implies \left\| \frac{1}{n} \sum_{i=1}^{n} x_i - \frac{1}{n} \sum_{i=1}^{m} y_i \right\| \text{ (difference of sample mean)}$$

$$\text{if } f(x) = x^2 \implies \left\| \frac{1}{n} \sum_{i=1}^{n} x_i^2 - \frac{1}{n} \sum_{i=1}^{m} y_i^2 \right\| \text{ (difference of the second moment)}$$

Generative Moment Matching Network

Maximum Mean Discrepancy

$$\mathcal{L}_{\text{MMD}}(p_x, p_y; f) = \left\| \frac{1}{n}\sum_{i=1}^{n} f(x_i) - \frac{1}{m}\sum_{i=1}^{m} f(y_i) \right\|_2^2$$

$$= \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} f(x_i)^T f(x_j) - \frac{2}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m} f(x_i)^T f(y_j) + \frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m} f(y_i)^T f(y_j)$$

$$= \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} k(x_i, x_j) - \frac{2}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m} k(x_i, y_j) + \frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m} k(y_i, y_j)$$

using $k(x, y) = \exp\left(-\frac{1}{2\sigma}(x-y)^2\right) \Longrightarrow$ difference of moments of all the orders

with Gaussian kernel, k,

Moment Matching

$$p_x = p_y \Longleftrightarrow \mathcal{L}_{\text{MMD}}(p_x, p_y; f) = 0$$

Generative Moment Matching Network

Training

Algorithm 1: GMMN minibatch training

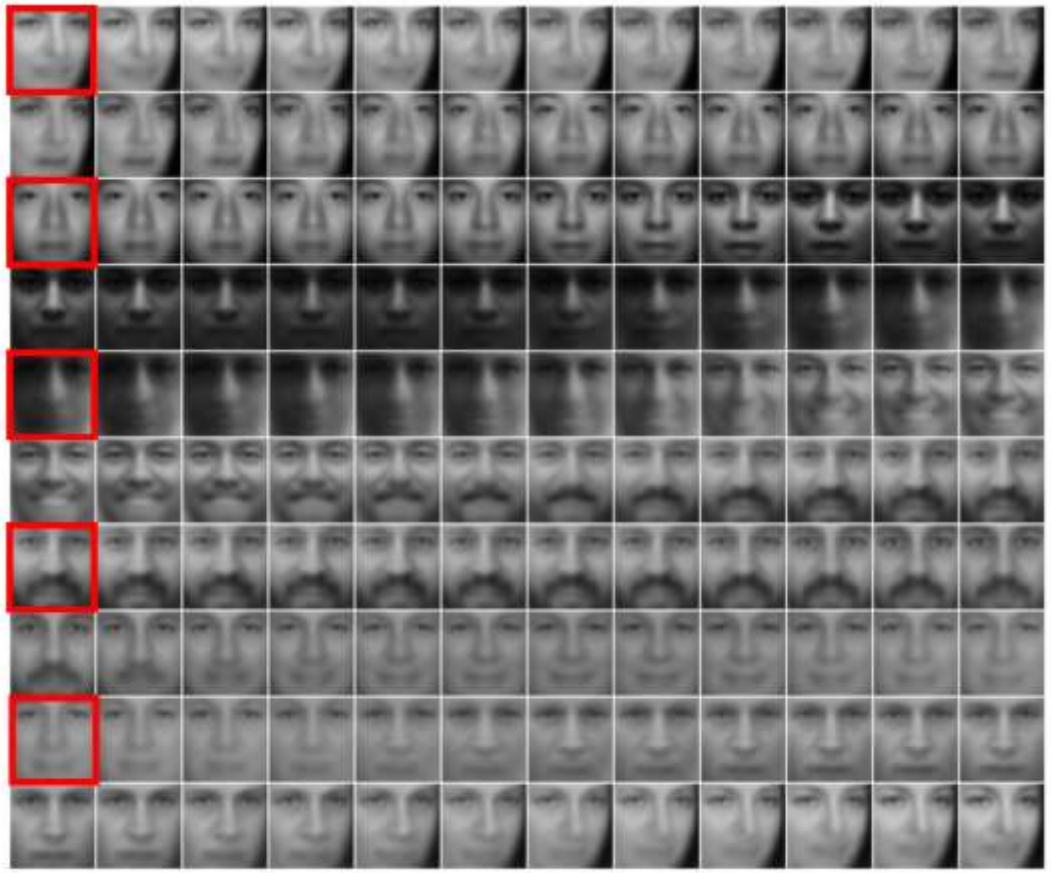
Input : Dataset $\{\mathbf{x}_1^d, ..., \mathbf{x}_N^d\}$, prior $p(\mathbf{h})$, network $f(\mathbf{h}; \mathbf{w})$ with initial parameter $\mathbf{w}^{(0)}$

Output: Learned parameter $\mathbf{w}^*$

while *Stopping criterion not met* **do**
  Get a minibatch of data $\mathbf{X}^d \leftarrow \{\mathbf{x}_{i_1}^d, ..., \mathbf{x}_{i_b}^d\}$
  Get a new set of samples $\mathbf{X}^s \leftarrow \{\mathbf{x}_1^s, ..., \mathbf{x}_b^s\}$
  Compute gradient $\frac{\partial \mathcal{L}_{\text{MMD}}}{\partial \mathbf{w}}$ on $\mathbf{X}^d$ and $\mathbf{X}^s$
  Take a gradient step to update $\mathbf{w}$
end

Generative Moment Matching Network

(a) MNIST interpolation



(b) TFD interpolation

MMD – GAN

MMD GAN: Towards Deeper Understanding of Moment Matching Network

Chun-Liang Li[1,*] **Wei-Cheng Chang**[1,*] **Yu Cheng**[2] **Yiming Yang**[1] **Barnabás Póczos**[1]
[1] Carnegie Mellon University, [2]AI Foundations, IBM Research
{chunlial,wchang2,yiming,bapoczos}@cs.cmu.edu chengyu@us.ibm.com
(* denotes equal contribution)

Abstract

Generative moment matching network (GMMN) is a deep generative model that differs from Generative Adversarial Network (GAN) by replacing the discriminator in GAN with a two-sample test based on kernel maximum mean discrepancy (MMD). Although some theoretical guarantees of MMD have been studied, the empirical performance of GMMN is still not as competitive as that of GAN on challenging and large benchmark datasets. The computational efficiency of GMMN is also less desirable in comparison with GAN, partially due to its requirement for a rather large batch size during the training. In this paper, we propose to improve both the model expressiveness of GMMN and its computational efficiency by introducing *adversarial kernel learning* techniques, as the replacement of a fixed Gaussian kernel in the original GMMN. The new approach combines the key ideas in both GMMN and GAN, hence we name it *MMD GAN*. The new distance measure in MMD GAN is a meaningful loss that enjoys the advantage of weak* topology and can be optimized via gradient descent with relatively small batch sizes. In our evaluation on multiple benchmark datasets, including MNIST, CIFAR-10, CelebA and LSUN, the performance of MMD GAN significantly outperforms GMMN, and is competitive with other representative GAN works.

MMD – GAN

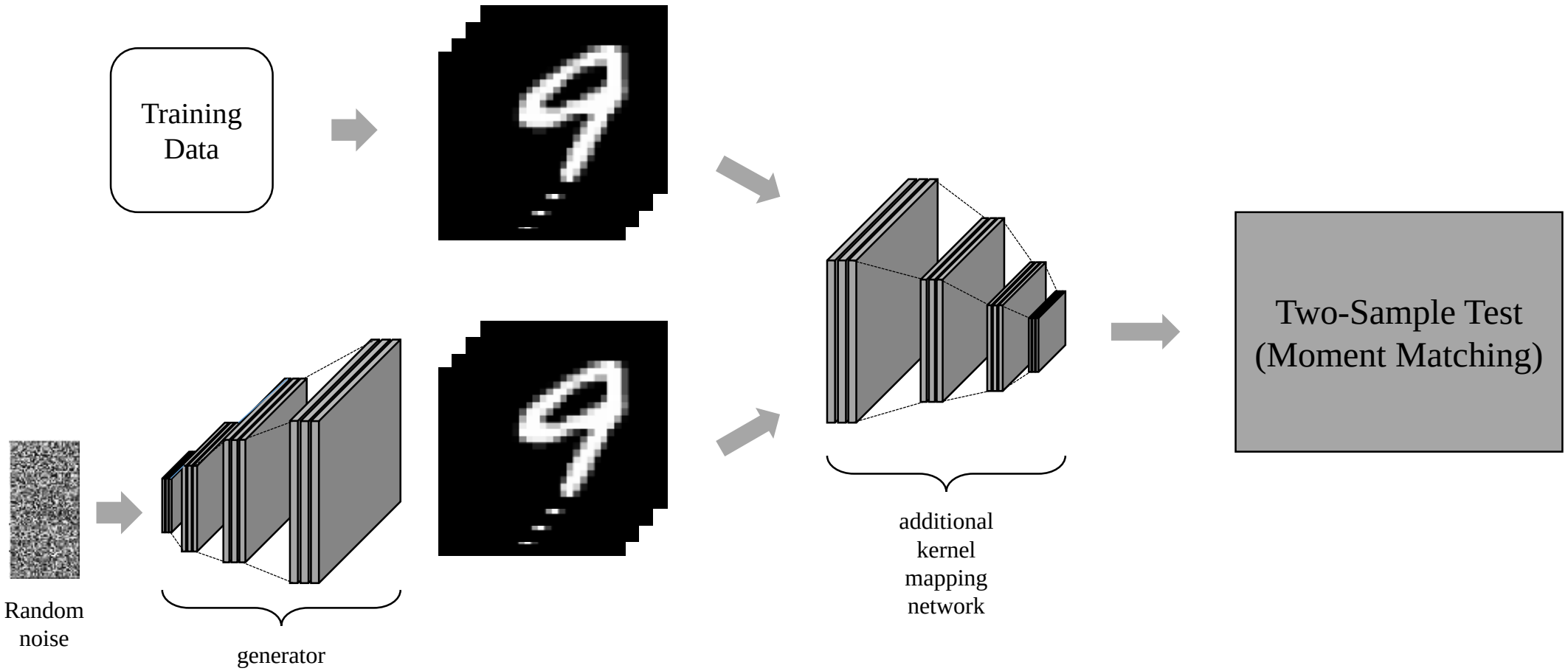
Generative Moment Matching Network: $\min_{\theta_G} \mathcal{L}_{\text{MMD}}(p_{\text{data}}, p_G; k)$

Better results with: $\min_{\theta_G} \max_{k \in \mathcal{K}} \mathcal{L}_{\text{MMD}}(p_{\text{data}}, p_G; k)$

Apply adversarial learning & NN: $\min_{\theta_G} \max_{\theta_f} \mathcal{L}_{\text{MMD}}(p_{\text{data}}, p_G; k \circ f)$

Feature extraction before apply kernel trick

MMD – GAN


Training Data
Random noise
generator
additional kernel mapping network
Two-Sample Test (Moment Matching)

MMD – GAN

(a) GMMN-D MNIST (b) GMMN-C MNIST (c) MMD GAN MNIST

(d) GMMN-D CIFAR-10 (e) GMMN-C CIFAR-10 (f) MMD GAN CIFAR-10

MMD – GAN

(a) WGAN MNIST

(b) WGAN CelebA

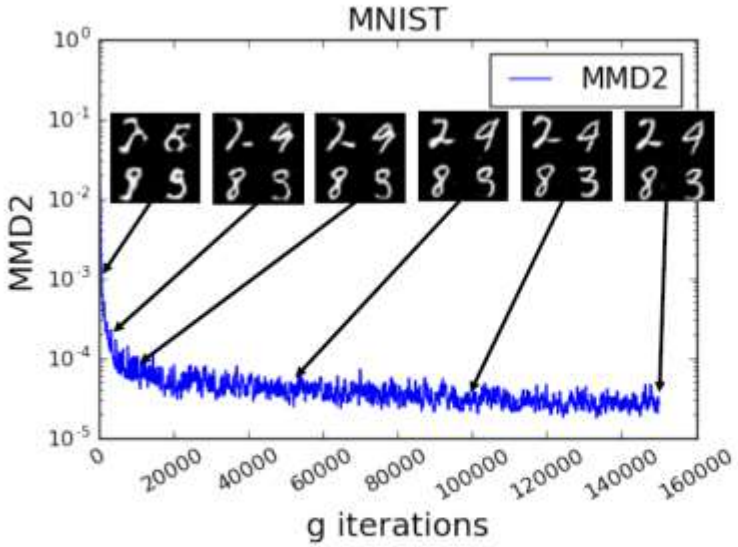
(c) WGAN LSUN

(d) MMD GAN MNIST

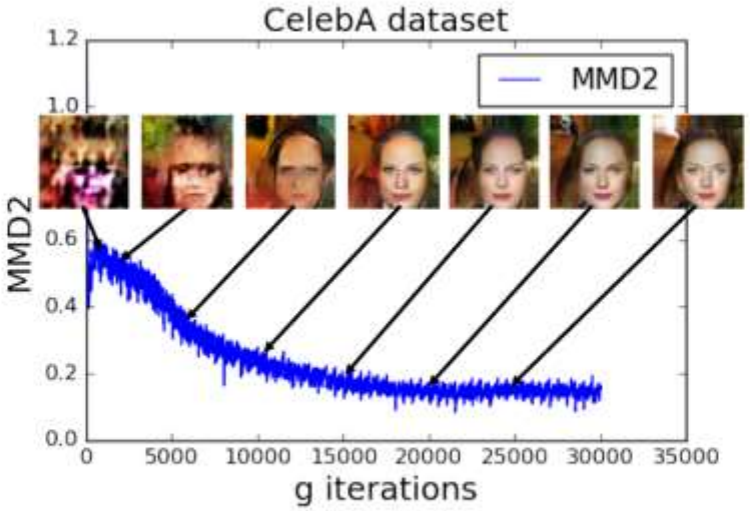
(e) MMD GAN CelebA

(f) MMD GAN LSUN

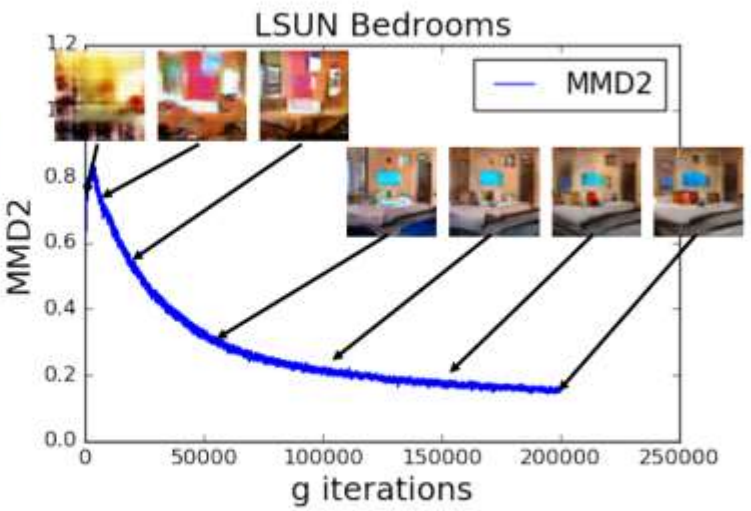
MMD – GAN



(a) MNIST



(b) CelebA

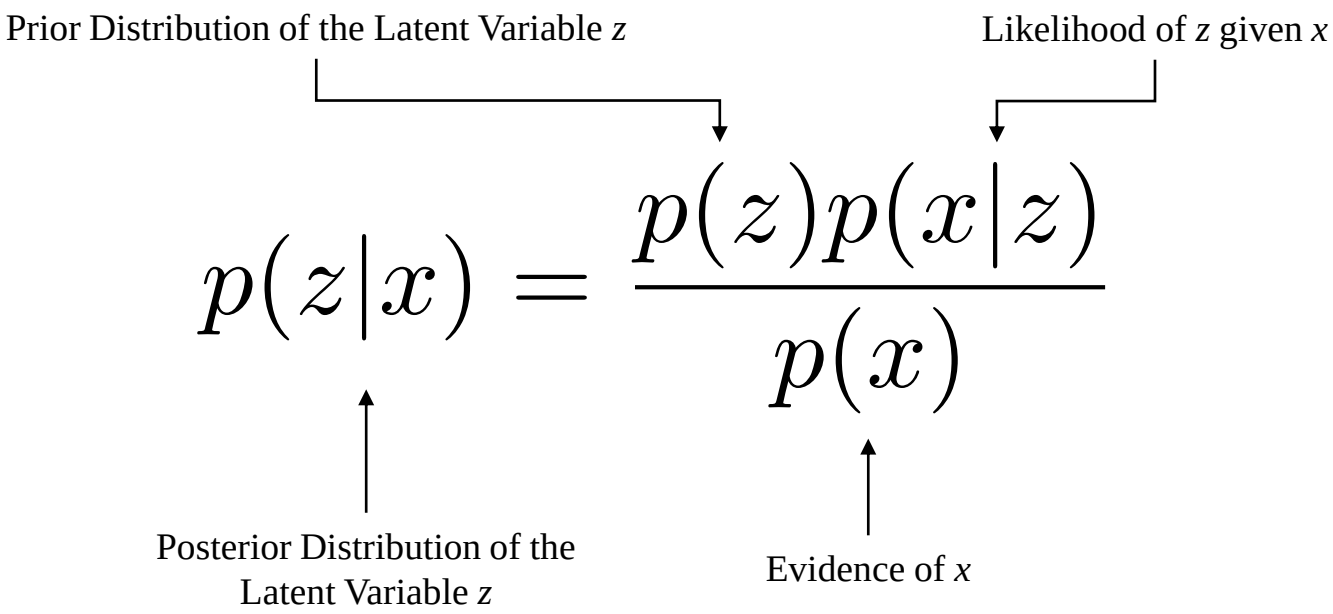


(c) LSUN Bedrooms

Variational Autoencoder

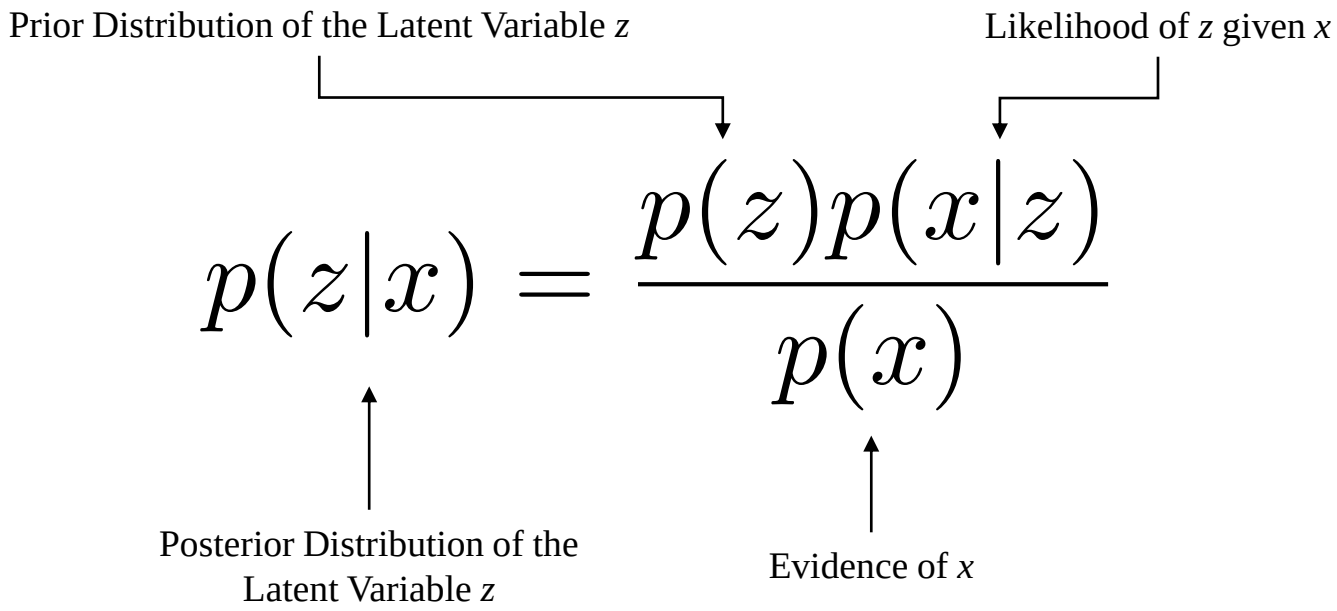
Variational Inference

(1) Bayesian Inference Problem with Latent Variables



Variational Inference

(1) Bayesian Inference Problem with Latent Variables



$$p(x) = \int p(x, z)dz = \int p(z)p(x|z)dz$$

Variational Inference

(2) Bayesian Approximate Inference

Monte Carlo Methods

$$\underbrace{z_1, z_2, \cdots, z_n}_{\text{samples}} \sim p(z|x)$$

Approximate Inference

$$q(z) \approx p(z|x)$$

Variational Inference

(2) Bayesian Approximate Inference

Monte Carlo Methods

$$\underbrace{z_1, z_2, \cdots, z_n}_{\text{samples}} \sim p(z|x)$$

Approximate Inference

$$q(z) \approx p(z|x)$$

Family of q
Measure of proximity

Variational Inference

(2) Bayesian Approximate Inference

- Monte Carlo Methods
  - Naïve Monte Carlo
  - Metropolis-Hastings
  - Rejection Sampling
  - Gibbs Sampling
  - Importance Sampling
  - Reversible-Jump
- Approximate Inference
  - Laplace Approximation
  - Expectation Propagation
  - Variational Inference

Variational Inference

(3) Variational Inference

$$q^*(z) = \underset{q \in \mathcal{Q}}{\operatorname{argmin}}\ \mathcal{D}_{\mathrm{KL}}(q(z)||p(z|x))$$

Optimal Approximation

Class of functions

Target (Posterior)

What we do not know

→ ????

Variational Inference

(3) Variational Inference

$$\mathcal{D}_{\mathrm{KL}}(q(z)||p(z|x)) = \int q(z) \log \frac{q(z)}{p(z|x)} dz$$

$$= \int q(z) \log q(z) dz - \int q(z) \log p(z|x) dz$$

$$= \mathbb{E}_{z\sim q(z)}[\log q(z)] - \mathbb{E}_{z\sim q(z)}[\log p(z|x)]$$

$$= \mathbb{E}_{z\sim q(z)}[\log q(z)] - \mathbb{E}_{z\sim q(z)}\left[\log \frac{p(z,x)}{p(x)}\right]$$

$$= \mathbb{E}_{z\sim q(z)}[\log q(z)] - \mathbb{E}_{z\sim q(z)}[\log p(z,x)] + \log p(x)$$

$$= \mathbb{E}_{z\sim q(z)}[\log q(z)] - \mathbb{E}_{z\sim q(z)}[\log p(z)p(x|z)] + \log p(x)$$

Variational Inference

(3) Variational Inference

$$\min_{q \in \mathcal{Q}} \mathcal{D}_{\mathrm{KL}}(q(z)||p(z|x)) \iff \min_{q \in \mathcal{Q}} \mathbb{E}_{z \sim q(z)}[\log q(z)] - \mathbb{E}_{z \sim q(z)}[\log p(z)p(x|z)]$$

Now we can solve the problem!

If q is parametrized by \phi,

$$\min_{\phi} \mathcal{D}_{\mathrm{KL}}(q_\phi(z)||p(z|x)) \iff \min_{\phi} \mathbb{E}_{z \sim q_\phi(z)}[\log q_\phi(z)] - \mathbb{E}_{z \sim q_\phi(z)}[\log p(z)p(x|z)]$$

Variational Inference

(3) Variational Inference

$$\mathcal{D}_{\mathrm{KL}}(q(z)||p(z|x)) = \int q(z) \log \frac{q(z)}{p(z|x)} dz$$

$$= \mathbb{E}_{z\sim q(z)}[\log q(z)] - \mathbb{E}_{z\sim q(z)}[\log p(z)p(x|z)] + \log p(x)$$

$$\underbrace{\log p(x)}_{\text{Evidence}} = \underbrace{\mathcal{D}_{\mathrm{KL}}(q(z)||p(z|x))}_{\text{KLD: Nonnegative}} \underbrace{-\mathbb{E}_{z\sim q(z)}[\log q(z)] + \mathbb{E}_{z\sim q(z)}[\log p(z)p(x|z)]}_{\text{Evidence Lower Bound (ELBO)}}$$

Variational Autoencoder

Auto-Encoding Variational Bayes

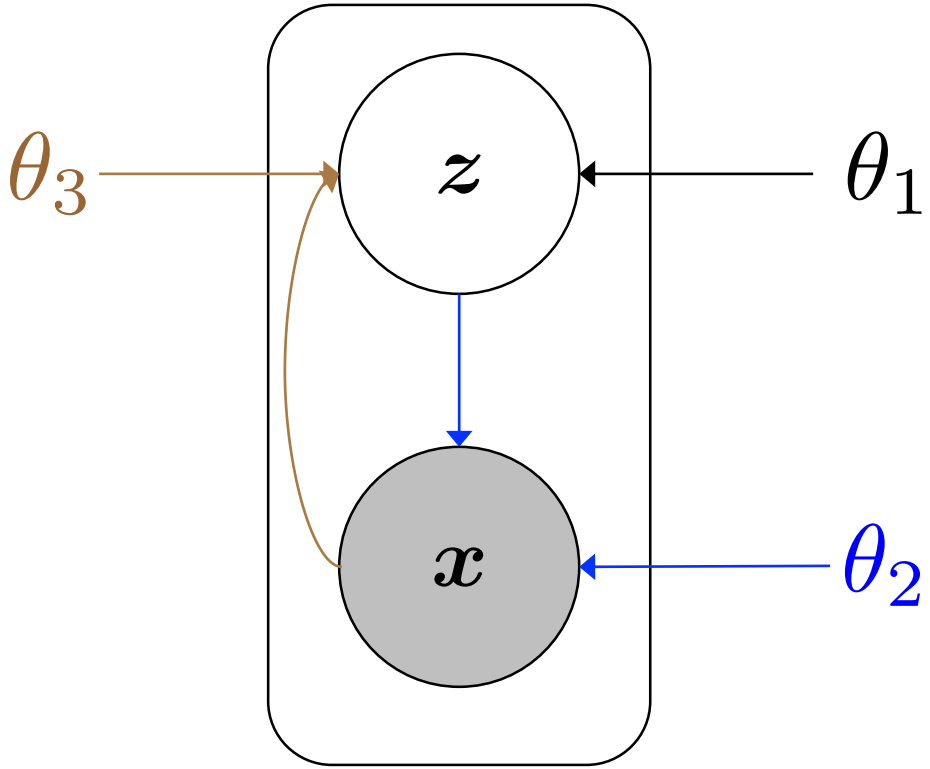
Diederik P. Kingma
Machine Learning Group
Universiteit van Amsterdam
dpkingma@gmail.com

Max Welling
Machine Learning Group
Universiteit van Amsterdam
welling.max@gmail.com

Abstract

How can we perform efficient inference and learning in directed probabilistic models, in the presence of continuous latent variables with intractable posterior distributions, and large datasets? We introduce a stochastic variational inference and learning algorithm that scales to large datasets and, under some mild differentiability conditions, even works in the intractable case. Our contributions is two-fold. First, we show that a reparameterization of the variational lower bound yields a lower bound estimator that can be straightforwardly optimized using standard stochastic gradient methods. Second, we show that for i.i.d. datasets with continuous latent variables per datapoint, posterior inference can be made especially efficient by fitting an approximate inference model (also called a recognition model) to the intractable posterior using the proposed lower bound estimator. Theoretical advantages are reflected in experimental results.

Variational Autoencoder



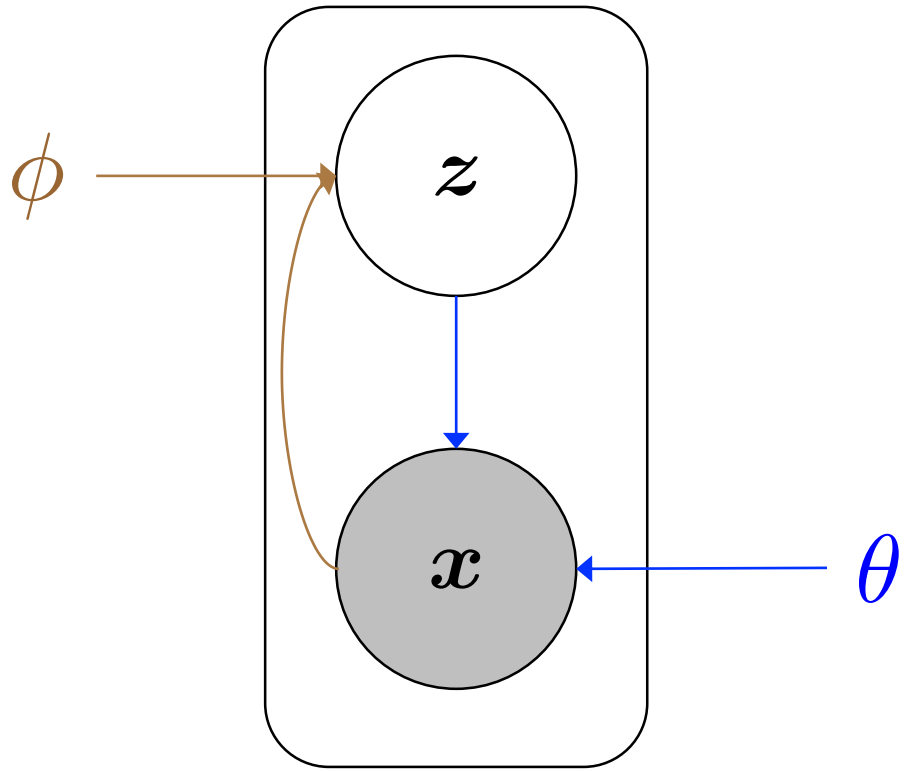
$$p_{\theta_1}(z)$$

$$p_{\theta_2}(x|z)$$

$$p_{\theta_3}(z|x)$$

$$p_{\theta_1,\theta_2}(x) = \int p_{\theta_1}(z)p_{\theta_2}(x|z)dz$$

Variational Autoencoder



$$p(z)$$

$$p_\theta(x|z)$$

$$q_\phi(z|x)$$

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

Variational Autoencoder

(1) VAE problem includes not only posterior (approximate) inference
(2) But also involves maximum likelihood inference of the generating process

Based on the maximum likelihood principle, we want to maximize the marginal likelihood of \theta given data x

$$\log p_\theta(x) = \mathcal{D}_{\mathrm{KL}}(q_\phi(z|x)||p_{\mathrm{true}}(z|x)) + \mathbb{E}_{z\sim q_\phi(z|x)}[-\log q_\phi(z|x) + \log p(z) + \log p_\theta(x|z)]$$

$$\log p_\theta(x) \geq \mathbb{E}_{z\sim q_\phi(z|x)}[-\log q_\phi(z|x) + \log p(z) + \log p_\theta(x|z)]$$

$$\max_{\theta,\phi} \mathbb{E}_{z\sim q_\phi(z|x)}[-\log q_\phi(z|x) + \log p(z) + \log p_\theta(x|z)]$$

Maximizing the ELBO implies the following two:
a) Maximizing the marginal likelihood (for generating process, or the decoder)
b) Minimizing the KL divergence of the approximate from the true posterior

Variational Autoencoder

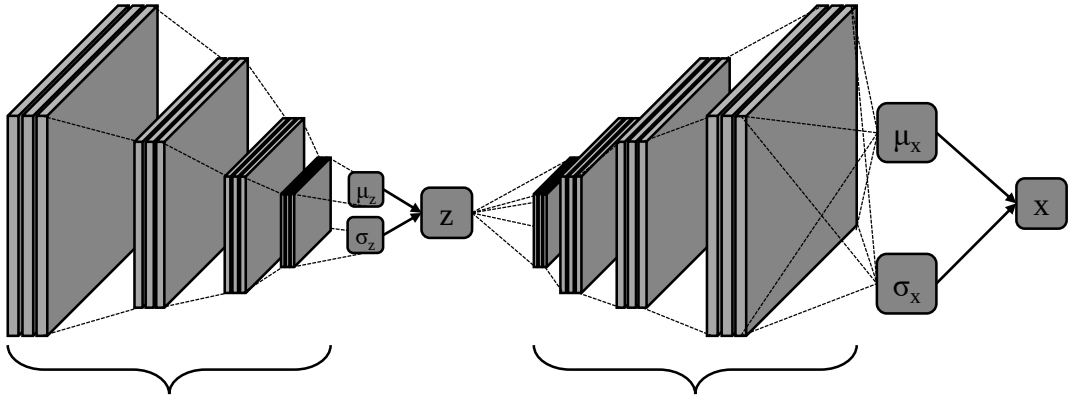
$$\begin{aligned}\mathbb{E}_{z\sim q_\phi(z|x)}[-\log q_\phi(z|x) + \log p(z) + \log p_\theta(x|z)] &= -\mathbb{E}_{z\sim q_\phi(z|x)}[\log q_\phi(z|x) - \log p(z)] + \mathbb{E}_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] \\ &= -\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log \frac{q_\phi(z|x)}{p(z)}\right] + \mathbb{E}_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] \\ &= -\mathcal{D}_{\mathrm{KL}}(q_\phi(z|x)||p(z)) + \mathbb{E}_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)]\end{aligned}$$

$$\max_{\theta,\phi} \; -\mathcal{D}_{\mathrm{KL}}(q_\phi(z|x)||p(z)) + \mathbb{E}_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)]$$

$$\min_{\theta,\phi} \; \mathcal{D}_{\mathrm{KL}}(q_\phi(z|x)||p(z)) - \mathbb{E}_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)]$$

(1) Guide variational approximate posterior to match the prior $p(z)$
(2) Maximize the expected likelihood of the generative model given data x
(or guide encoder-decoder to reconstruct the data)

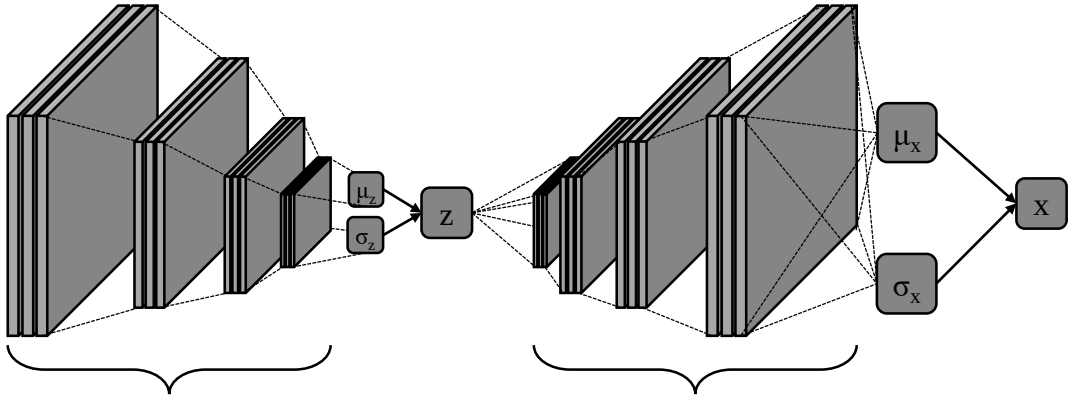
Variational Autoencoder



$$\min_{\theta,\phi} \mathcal{D}_{\mathrm{KL}}(q_\phi(z|x)||p(z)) - \mathbb{E}_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)]$$

Variational Autoencoder

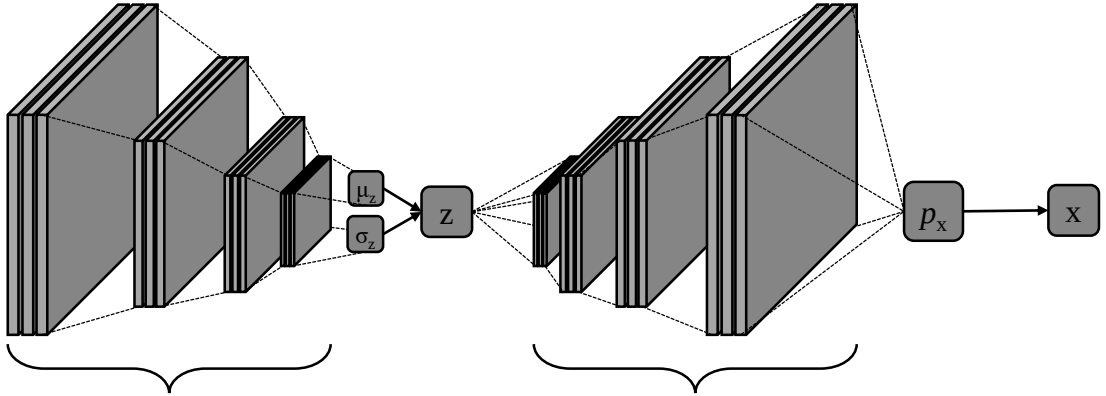
Example 1: Gaussian Encoder – Gaussian Decoder VAE



$$q_\phi(z_i|x_i) = \mathcal{N}(z_i; \mu(x_i), \sigma(x_i)^2\mathbf{I}) \qquad p_\theta(x_i|z_i) = \mathcal{N}(x_i; \mu(z_i), \sigma(z_i)^2\mathbf{I})$$

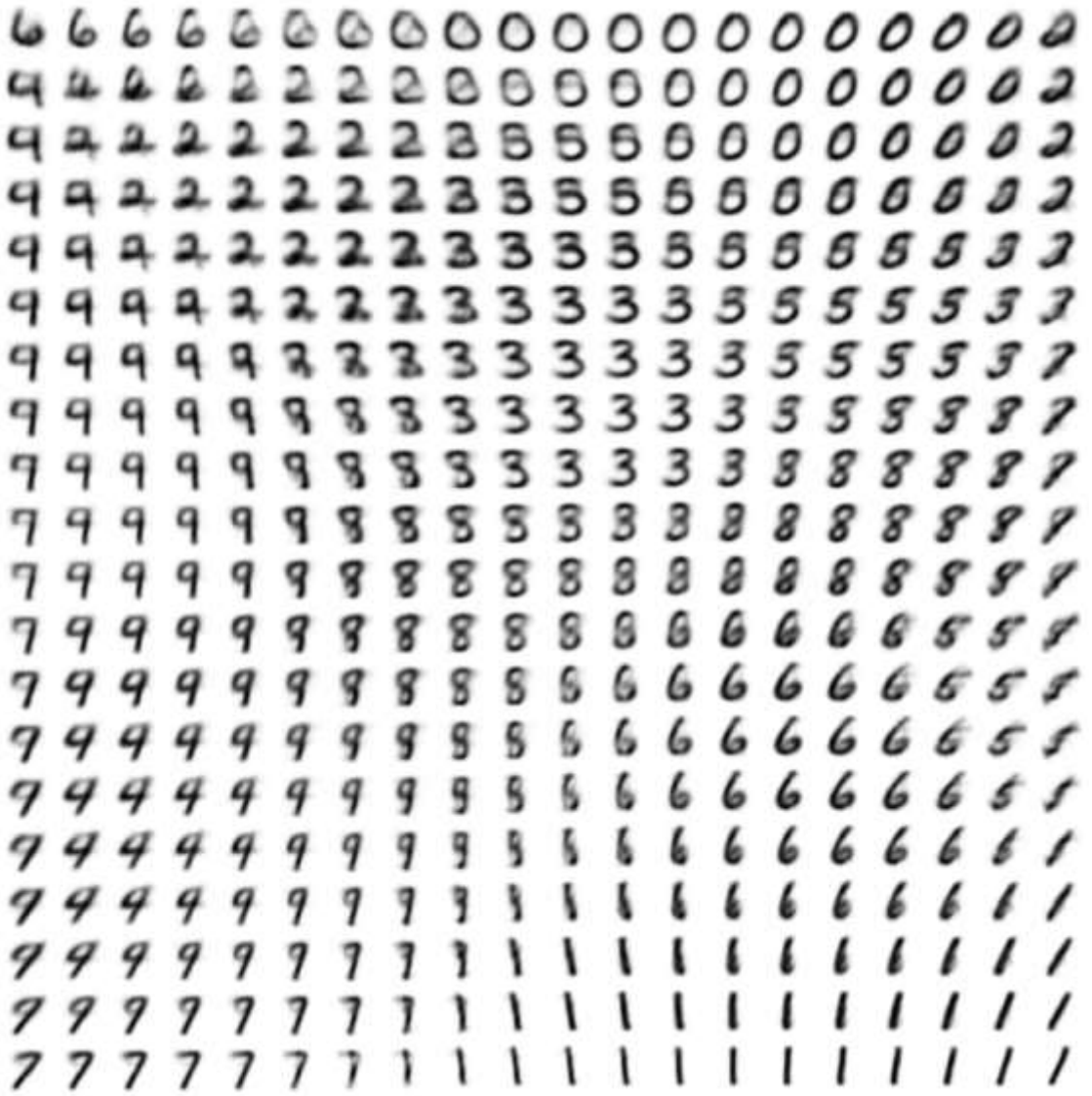
Variational Autoencoder

Example 2: Gaussian Encoder – Bernoulli Decoder VAE



$$q_\phi(z_i|x_i) = \mathcal{N}(z_i; \mu(x_i), \sigma(x_i)^2\mathbf{I}) \qquad p_\theta(x_i|z_i) = \text{Bernoulli}(x_i; p(z_i))$$

Variational Autoencoder



Beta Variational Autoencoder

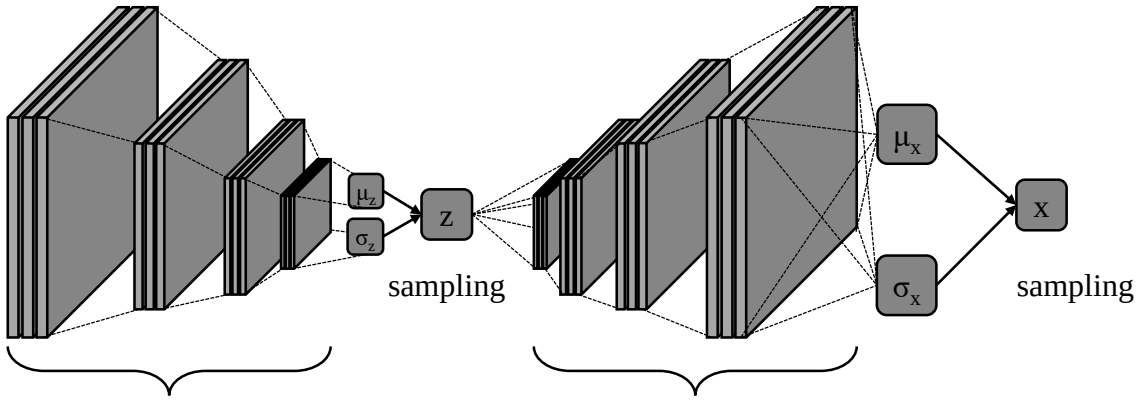
β-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, Alexander Lerchner
Google DeepMind
{irinah,lmatthey,arkap,cpburgess,glorotx,
botvinick,shakir,lerchner}@google.com

Abstract

Learning an interpretable factorised representation of the independent data generative factors of the world without supervision is an important precursor for the development of artificial intelligence that is able to learn and reason in the same way that humans do. We introduce β-VAE, a new state-of-the-art framework for automated discovery of interpretable factorised latent representations from raw image data in a completely unsupervised manner. Our approach is a modification of the variational autoencoder (VAE) framework. We introduce an adjustable hyperparameter β that balances latent channel capacity and independence constraints with reconstruction accuracy. We demonstrate that β-VAE with appropriately tuned $\beta > 1$ qualitatively outperforms VAE ($\beta = 1$), as well as state of the art unsupervised (InfoGAN) and semi-supervised (DC-IGN) approaches to disentangled factor learning on a variety of datasets (*celebA*, *faces* and *chairs*). Furthermore, we devise a protocol to quantitatively compare the degree of disentanglement learnt by different models, and show that our approach also significantly outperforms all baselines quantitatively. Unlike InfoGAN, β-VAE is stable to train, makes few assumptions about the data and relies on tuning a single hyperparameter β, which can be directly optimised through a hyperparameter search using weakly labelled data or through heuristic visual inspection for purely unsupervised data.

Beta Variational Autoencoder

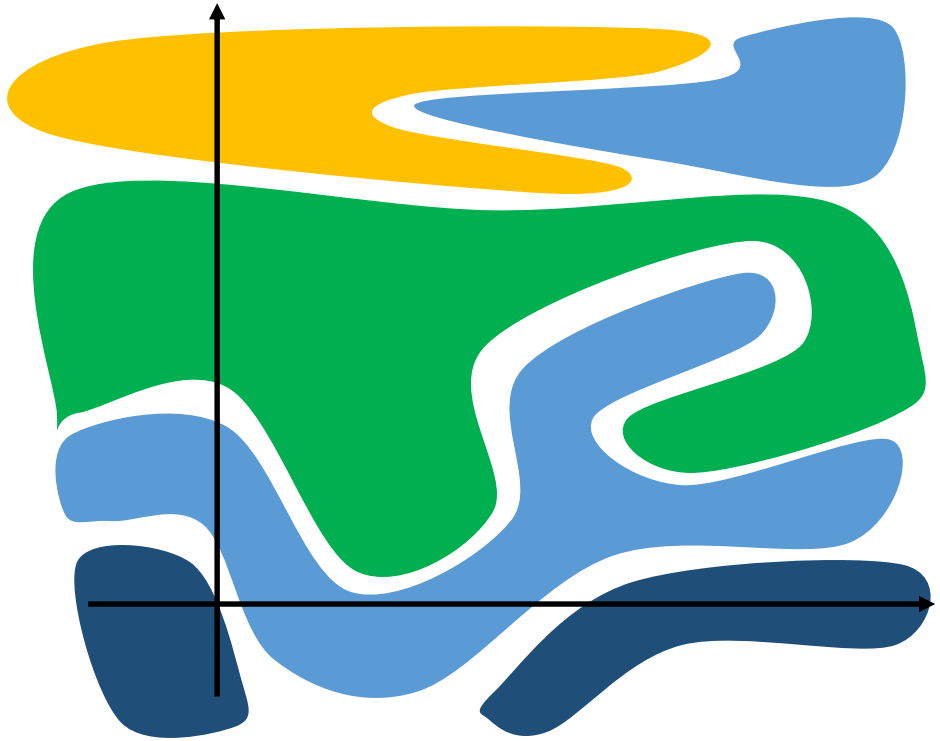


Vanilla VAE $$\min_{\theta,\phi} \mathcal{D}_{\mathrm{KL}}(q_\phi(z|x)||p(z)) - \mathbb{E}_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)]$$

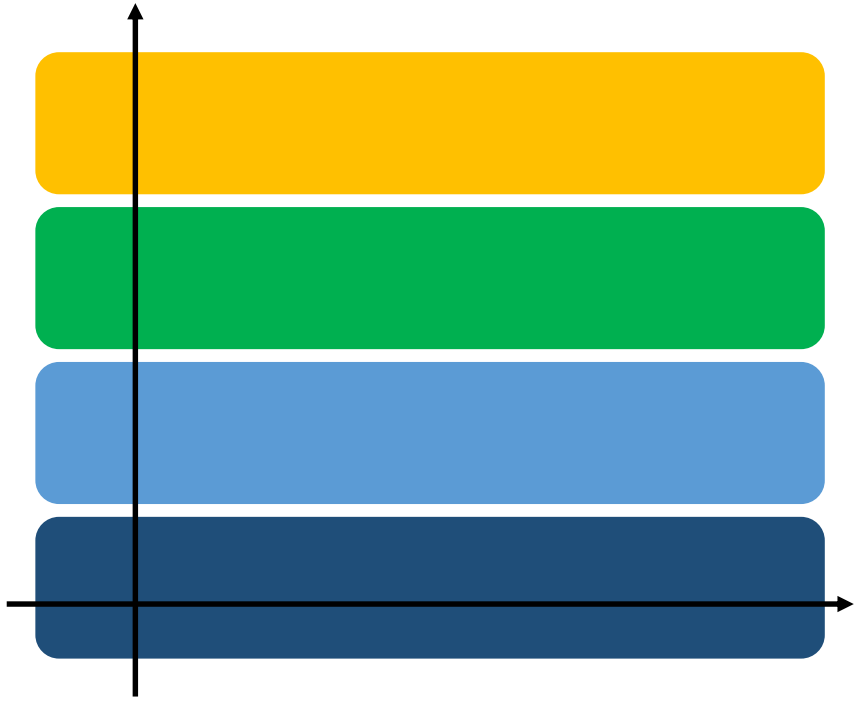
Beta VAE $$\min_{\theta,\phi} \beta \cdot \mathcal{D}_{\mathrm{KL}}(q_\phi(z|x)||p(z)) - \mathbb{E}_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)]$$

More weights on KL divergence term

Beta Variational Autoencoder

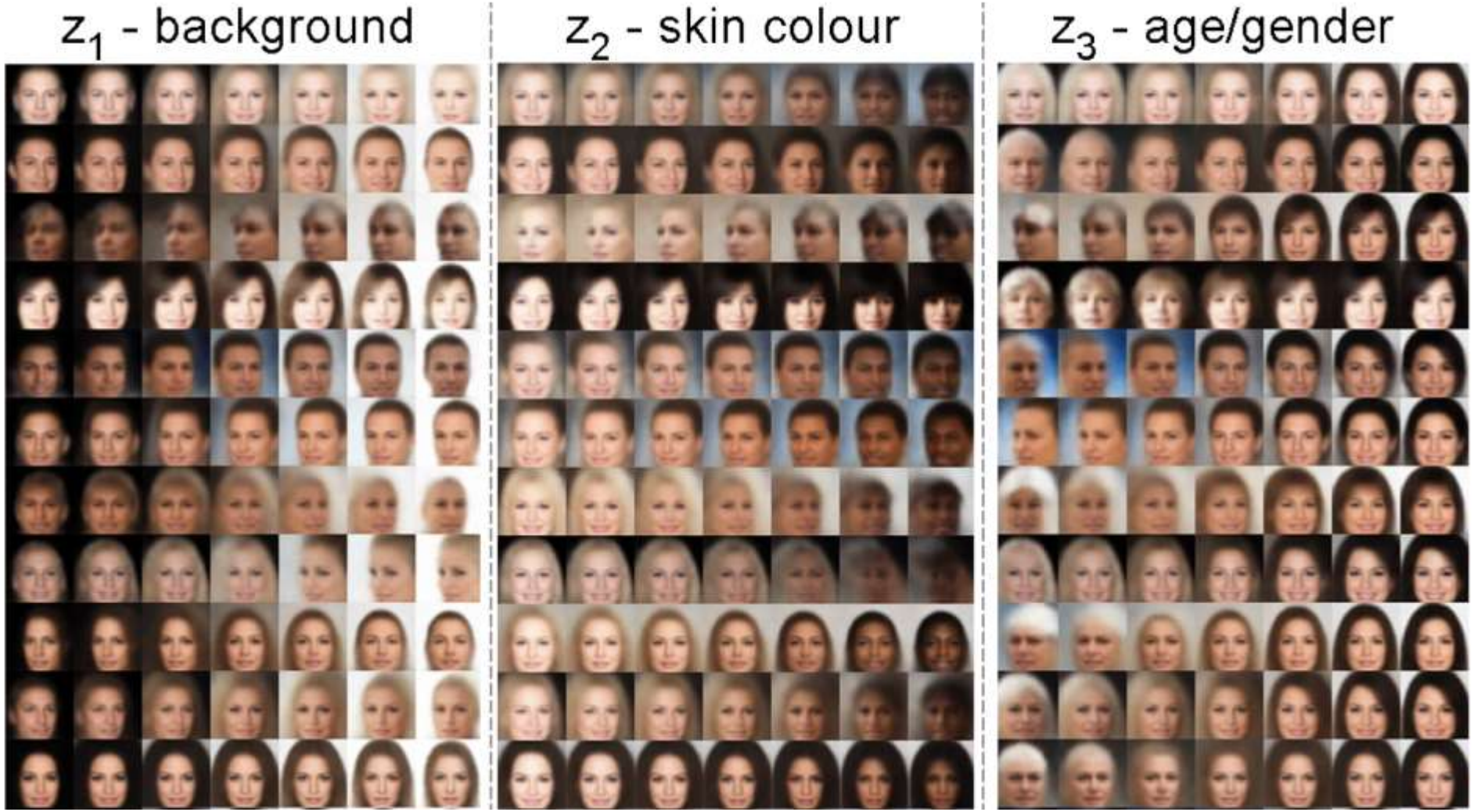


Entangled Representation (hard to interpret)

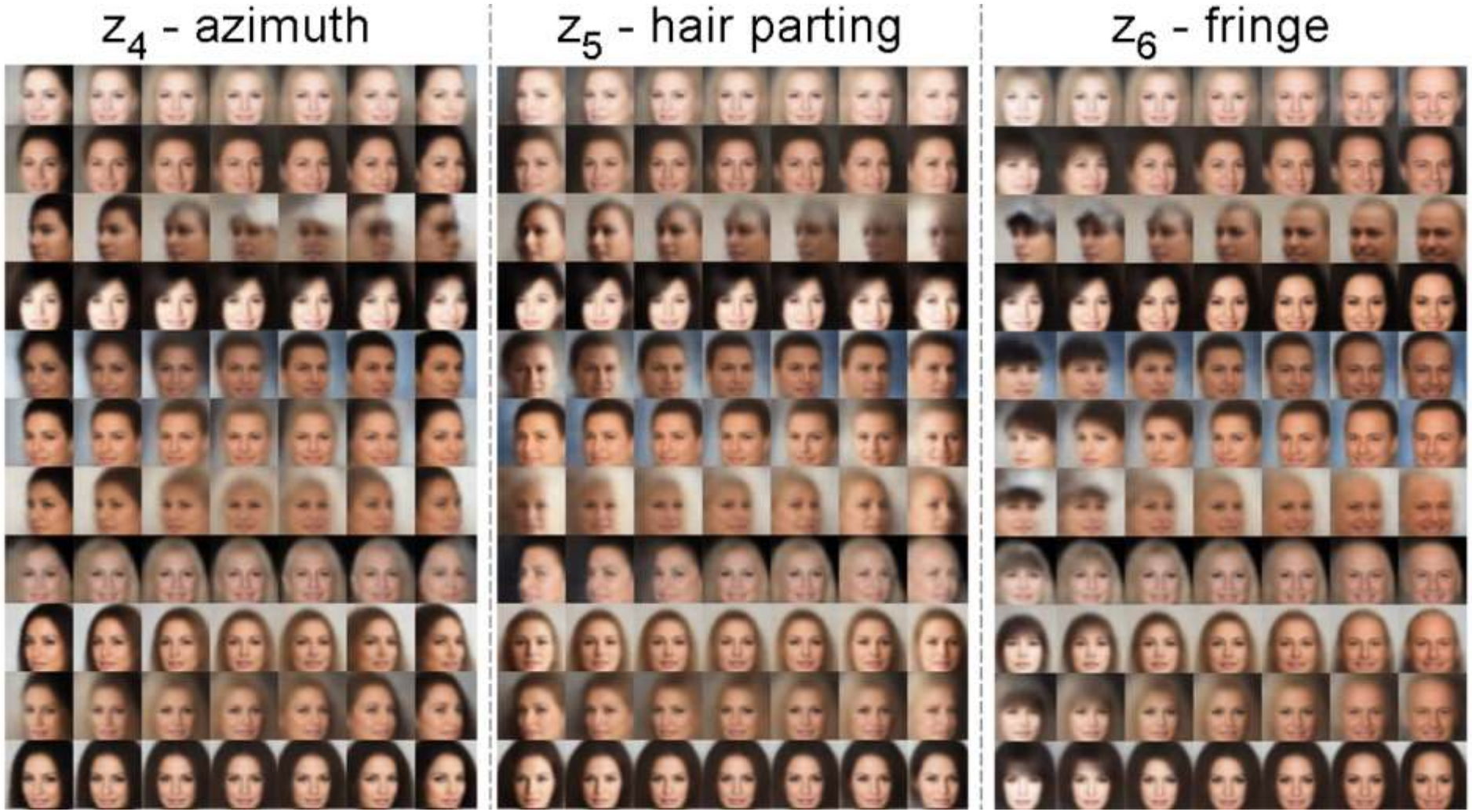


Disentangled Representation (easier to interpret)

Beta Variational Autoencoder



Beta Variational Autoencoder



Adversarial Autoencoder

Adversarial Autoencoders

Alireza Makhzani
University of Toronto
makhzani@psi.toronto.edu

Jonathon Shlens & Navdeep Jaitly
Google Brain
{shlens,ndjaitly}@google.com

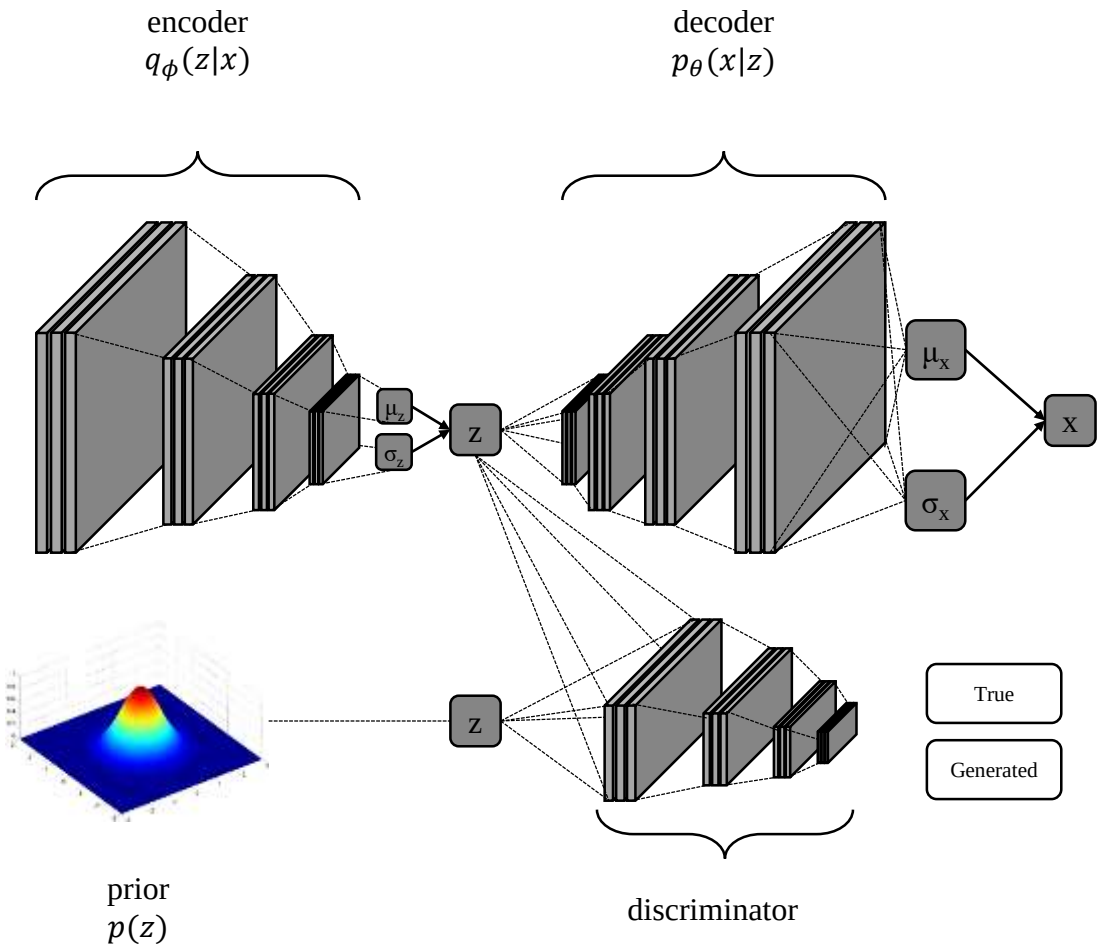
Ian Goodfellow
OpenAI
goodfellow.ian@gmail.com

Brendan Frey
University of Toronto
frey@psi.toronto.edu

Abstract

In this paper, we propose the "adversarial autoencoder" (AAE), which is a probabilistic autoencoder that uses the recently proposed generative adversarial networks (GAN) to perform variational inference by matching the aggregated posterior of the hidden code vector of the autoencoder with an arbitrary prior distribution. Matching the aggregated posterior to the prior ensures that generating from any part of prior space results in meaningful samples. As a result, the decoder of the adversarial autoencoder learns a deep generative model that maps the imposed prior to the data distribution. We show how the adversarial autoencoder can be used in applications such as semi-supervised classification, disentangling style and content of images, unsupervised clustering, dimensionality reduction and data visualization. We performed experiments on MNIST, Street View House Numbers and Toronto Face datasets and show that adversarial autoencoders achieve competitive results in generative modeling and semi-supervised classification tasks.

Adversarial Autoencoder



Adversarial Autoencoder

VAE $$\min_{\theta,\phi} \mathcal{D}_{\mathrm{KL}}(q_\phi(z|x)||p(z)) - \mathbb{E}_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)]$$

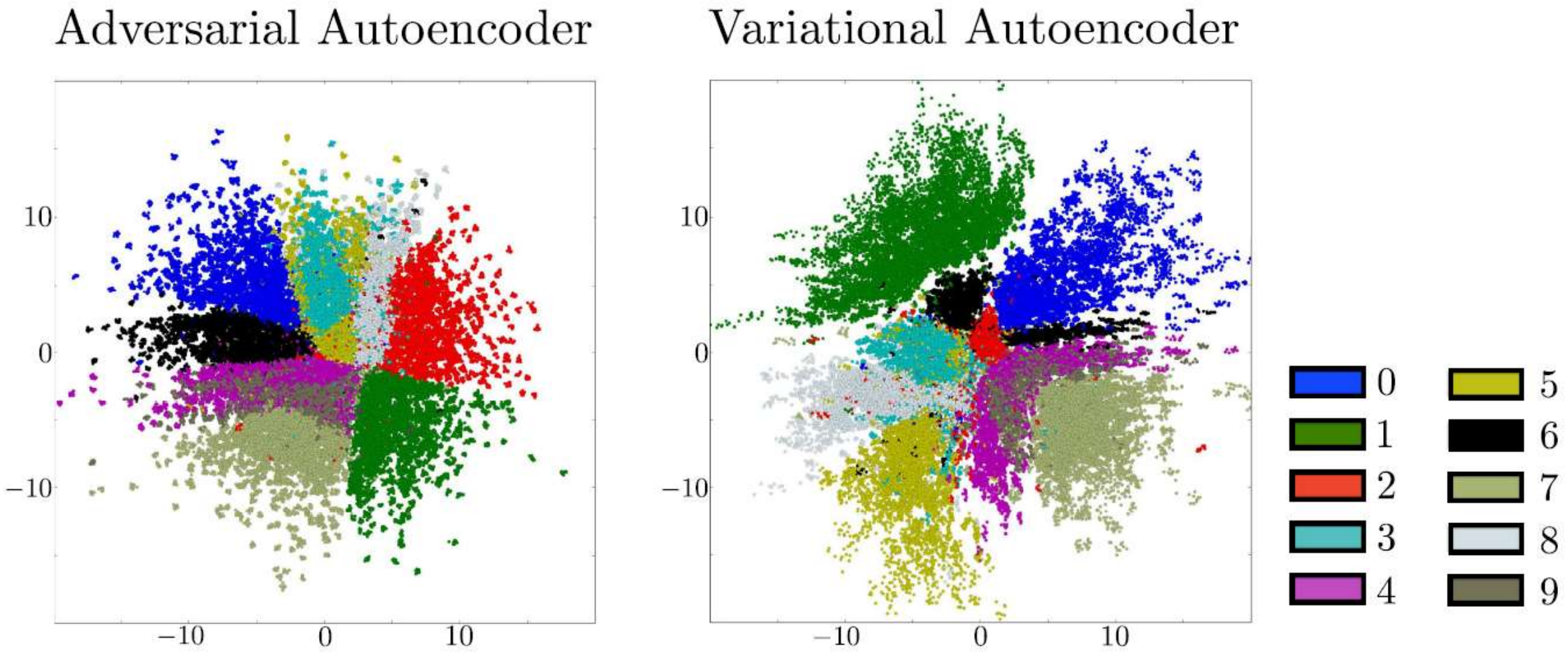
Minimize the discrepancy between prior p(z) and posterior q(z|x)

AAE $$\min_{\theta,\phi} \max_{\psi} \mathbb{E}_{z\sim p(z)}[\log D(z)] + \mathbb{E}_{z\sim q(z)}[\log(1 - D(z))] - \mathbb{E}_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)]$$

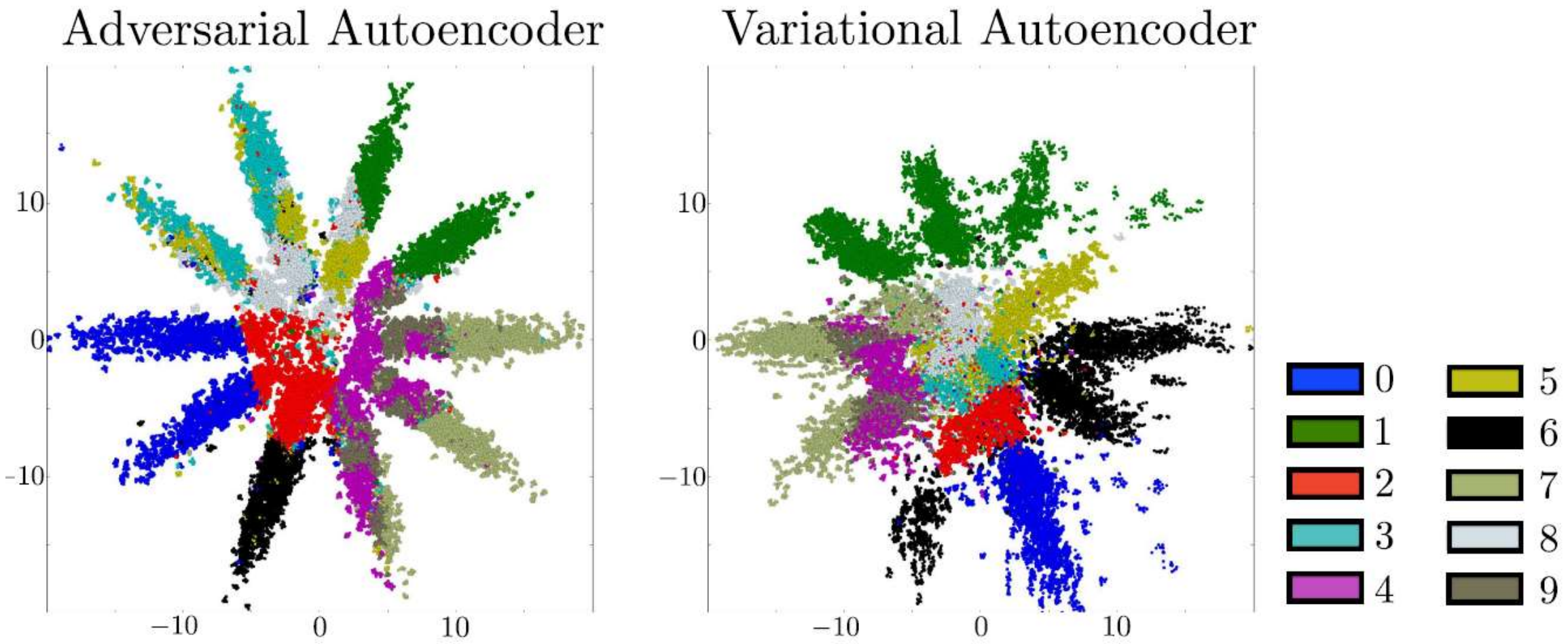
Minimize the discrepancy between prior p(z) and **aggregated** posterior q(z)

$$q(z) = \int q_\phi(z|x)p_{\mathrm{data}}(x)dx \rightarrow \text{aggregated posterior distribution}$$

Adversarial Autoencoder

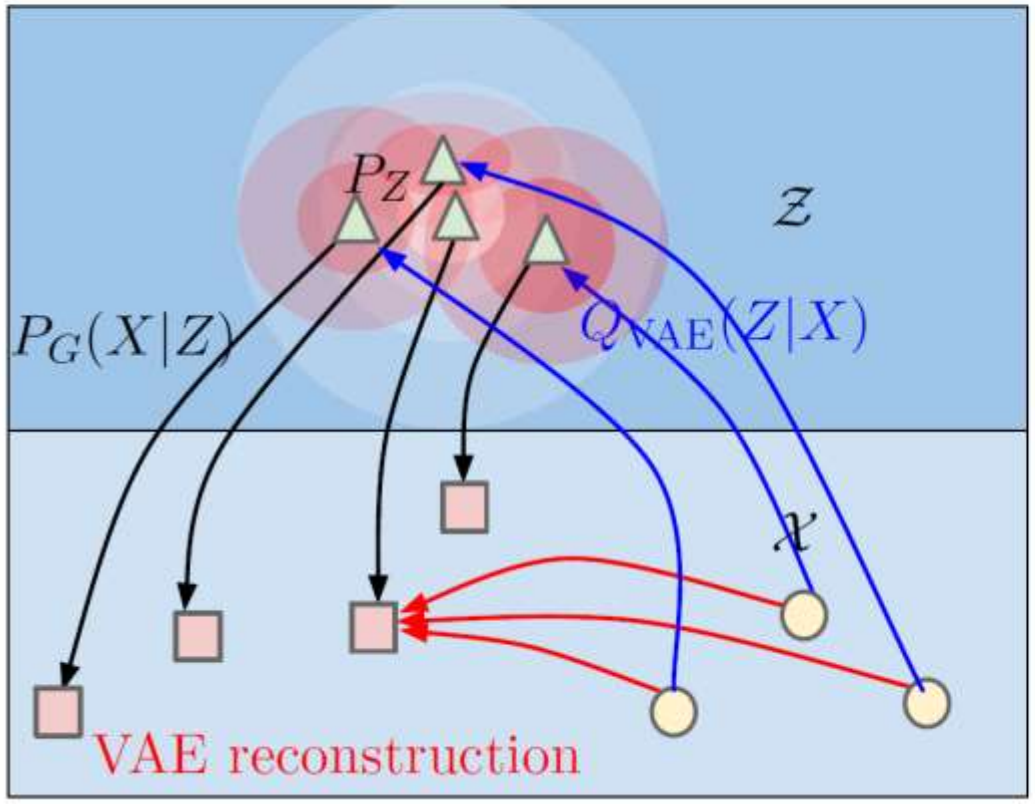


Adversarial Autoencoder



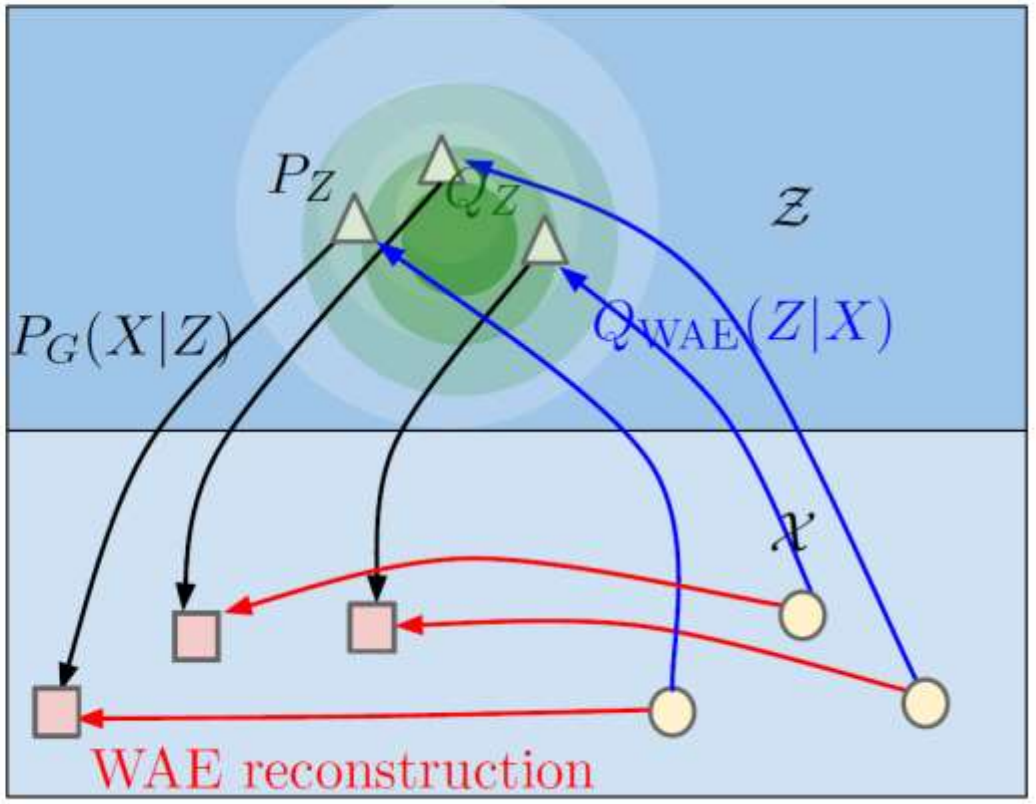
Adversarial Autoencoder

(a) VAE



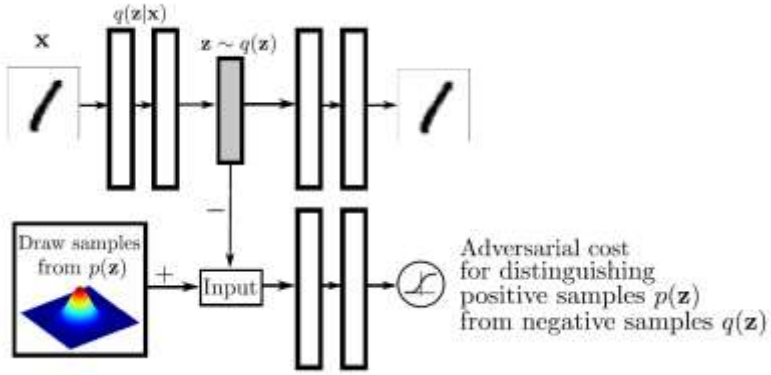
$$\min_{\theta,\phi} \mathcal{D}_{\text{KL}}(q_\phi(z|x)||p(z)) - \mathbb{E}_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)]$$

(b) WAE and AAE

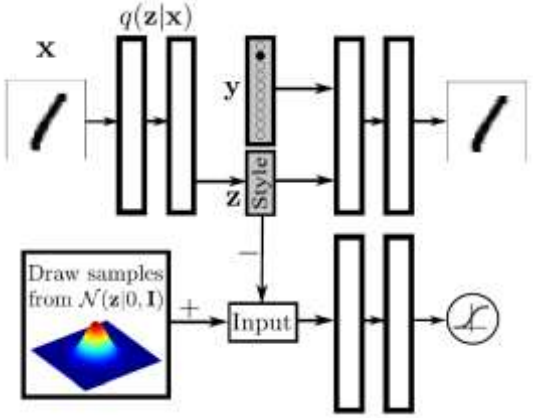


$$\min_{\theta,\phi,\psi} \mathbb{E}_{x\sim p_{\text{data}}(x)}\mathbb{E}_{z\sim q_\phi(z|x)}[c(x, G_\theta(z))] + \lambda \cdot \mathcal{D}_\psi(q_z||p_z)$$

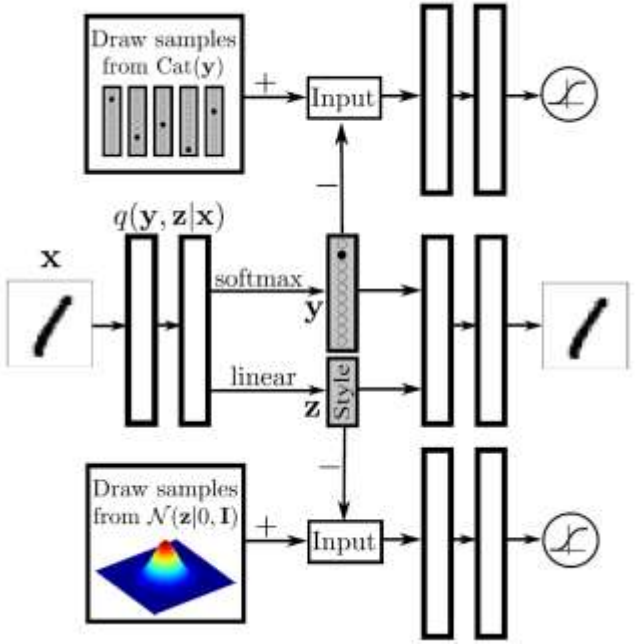
Adversarial Autoencoder



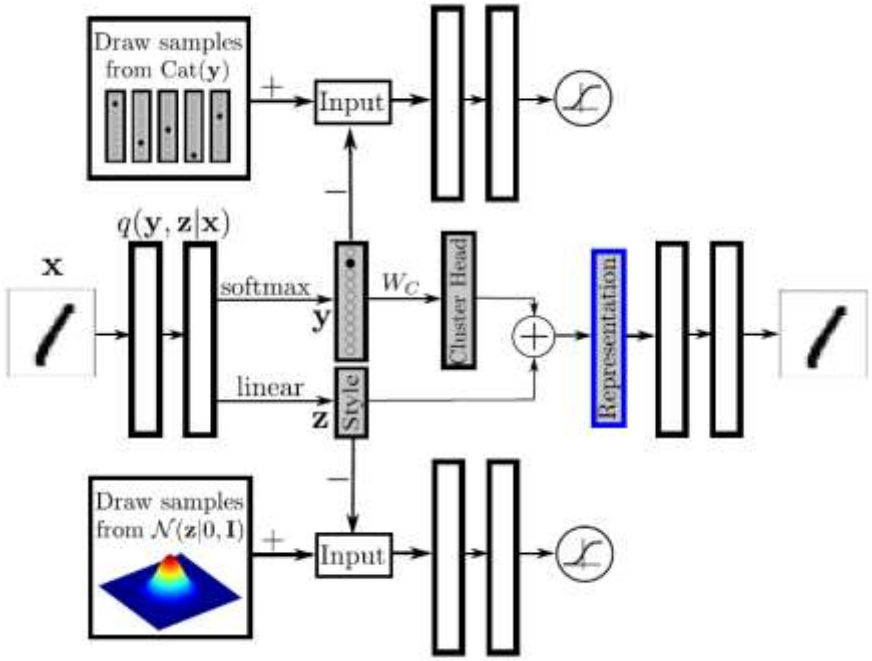
AAE Basic Form



Supervised AAE



Semi-Supervised AAE



Dimensionality Reduction AAE

Wasserstein Autoencoder

Wasserstein Auto-Encoders

Ilya Tolstikhin[1], Olivier Bousquet[2], Sylvain Gelly[2], and Bernhard Schölkopf[1]

[1]Max Planck Institute for Intelligent Systems
[2]Google Brain

Abstract

We propose the Wasserstein Auto-Encoder (WAE)—a new algorithm for building a generative model of the data distribution. WAE minimizes a penalized form of the Wasserstein distance between the model distribution and the target distribution, which leads to a different regularizer than the one used by the Variational Auto-Encoder (VAE) [1]. This regularizer encourages the encoded training distribution to match the prior. We compare our algorithm with several other techniques and show that it is a generalization of adversarial auto-encoders (AAE) [2]. Our experiments show that WAE shares many of the properties of VAEs (stable training, encoder-decoder architecture, nice latent manifold structure) while generating samples of better quality, as measured by the FID score.

Wasserstein Autoencoder

$$W_1(p_{\text{data}}, p_G) = \inf_{\gamma \in \Gamma(p_{\text{data}}, p_G)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|_1]$$

$$W_1(p_{\text{data}}, p_G) = \sup_{\|f\|_L \leq K} \left\{ \mathbb{E}_{x\sim p_{\text{data}}(x)}[f(x)] - \mathbb{E}_{x\sim p_G(x)}[f(x)] \right\}$$

$$W_1(p_{\text{data}}, p_G) = \max_{\theta_D} \left\{ \mathbb{E}_{x\sim p_{\text{data}}(x)}[D(x)] - \mathbb{E}_{z\sim p(z)}[D(G(z))] \right\}$$

Wasserstein Autoencoder

$$W_c(p_{\text{data}}, p_G) = \inf_{\gamma \in \Gamma(p_{\text{data}}, p_G)} \mathbb{E}_{(x,y)\sim\gamma}[c(x,y)]$$

$$= \inf_{\{q: q_z = p_z\}} \mathbb{E}_{x\sim p_{\text{data}}(x)} \mathbb{E}_{z\sim q(z|x)}[c(x, G(z))]$$

q_z is the aggregated posterior when $x \sim p_{data}(x)$ and $z \sim q(z|x)$

$$\inf_{q(z|x)} \mathbb{E}_{x\sim p_{\text{data}}(x)} \mathbb{E}_{z\sim q(z|x)}[c(x, G(z))] + \lambda \cdot \mathcal{D}(q_z || p_z)$$

WAE problem $\min_{\theta,\phi,\psi} \mathbb{E}_{x\sim p_{\text{data}}(x)} \mathbb{E}_{z\sim q_\phi(z|x)}[c(x, G_\theta(z))] + \lambda \cdot \mathcal{D}_\psi(q_z || p_z)$

Wasserstein Autoencoder

WAE problem $\min_{\theta,\phi,\psi} \mathbb{E}_{x\sim p_{\text{data}}(x)}\mathbb{E}_{z\sim q_\phi(z|x)}[c(x, G_\theta(z))] + \lambda \cdot \mathcal{D}_\psi(q_z||p_z)$

WAE – GAN

$$\min_{\theta,\phi} \max_{\psi} \mathbb{E}_{x\sim p_{\text{data}}(x)}\mathbb{E}_{z\sim q_\phi(z|x)}[c(x, G_\theta(z))] + \lambda \cdot (\mathbb{E}_{z\sim p_z(z)}[\log D_\psi(z)] + \mathbb{E}_{z\sim q_z(z)}[\log(1 - D_\psi(z))])$$

WAE – MMD

$$\min_{\theta,\phi} \mathbb{E}_{x\sim p_{\text{data}}(x)}\mathbb{E}_{z\sim q_\phi(z|x)}[c(x, G_\theta(z))] + \lambda \cdot \mathcal{L}_{\text{MMD}}(q_z, p_z)$$

Wasserstein Autoencoder

VAE

WAE-MMD

WAE-GAN