초짜 대학원생의 입장에서 이해하는

# VARIANTS OF GANs

Jaejun Yoo
Ph.D. Candidate @KAIST

13th May, 2017

# 안녕하세요 저는


유재준

KAIST Korea Advanced Institute of Science and Technology BiSPL Bio Imaging Signal Processing Lab.

- Ph.D. Candidate
- Medical Image Reconstruction, Topological Data Analysis, EEG
- http://jaejunyoo.blogspot.com/
- CVPR 2017 NTIRE Challenge: Ranked 3rd

# 이 강의의 목표

1. GAN에 대한 더 깊은 이해
2. 이후 GAN 연구의 흐름을 따라가기 위한 기반 다지기
   - 기존의 GAN이 가지고 있는 문제점과 그 이유에 대한 이해
   - Variants of GAN을 소개하되 주요 문제점을 해결하거나 큰 틀에서 새로운 방향을 제시한 논문들 위주 소개

# BACKGROUND

# PREREQUISITES

## Generative Models

"FACE IMAGES"

# PREREQUISITES

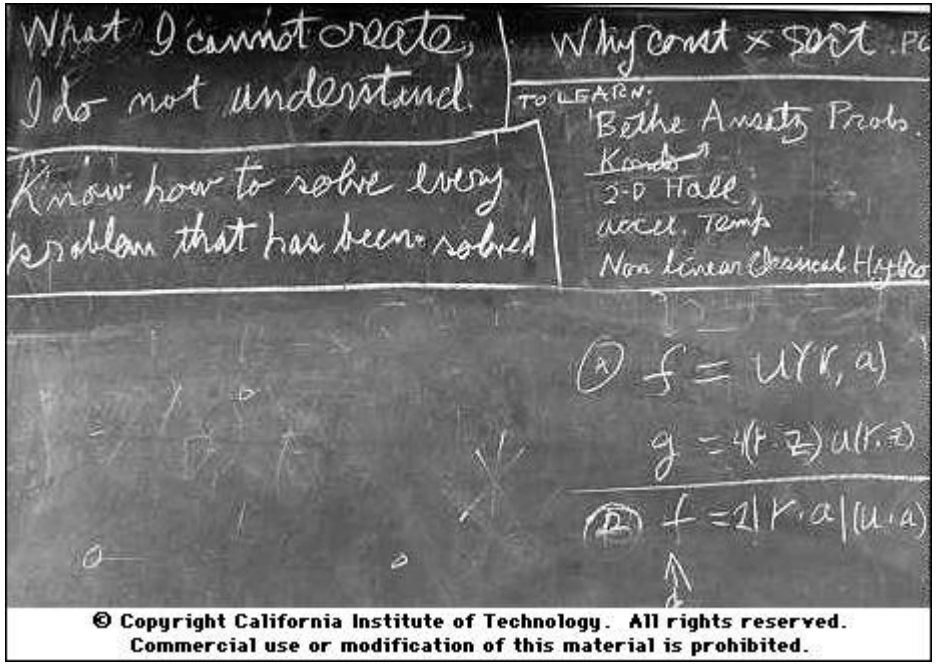
## Generative Models

**Generated Images by Neural Network**

* Figure adopted from *BEGAN* paper released at 31. Mar. 2017
David Berthelot et al. Google (link)

# PREREQUISITES

## Generative Models

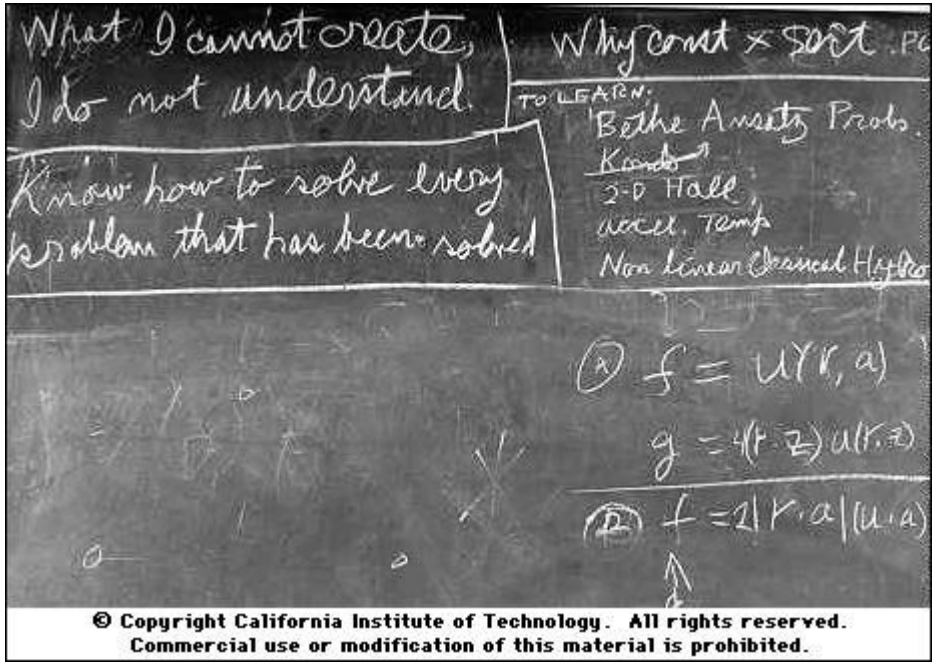




**“What I cannot create, I do not understand”**

# PREREQUISITES

## Generative Models



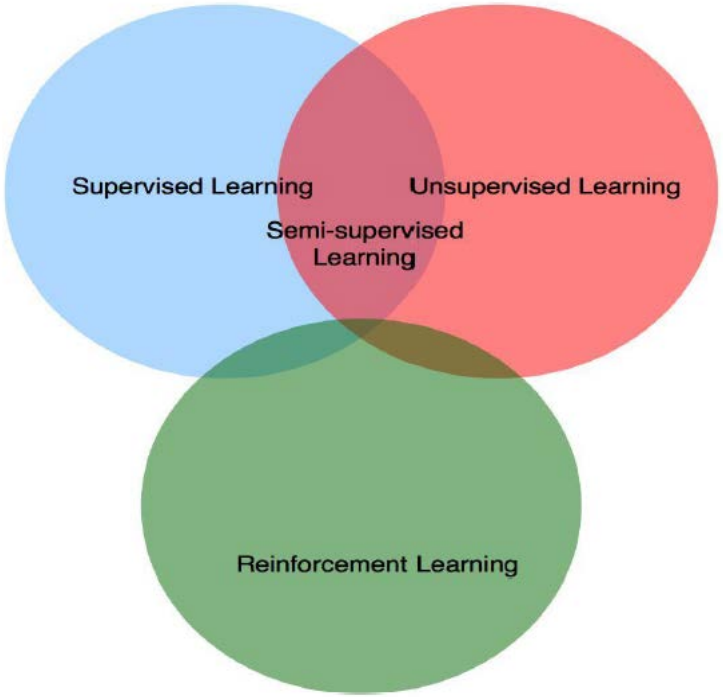


**"What I cannot create, I do not understand"**

**If the network can learn how to draw cat and dog separately,**
**it must be able to classify them, i.e. feature learning follows naturally.**

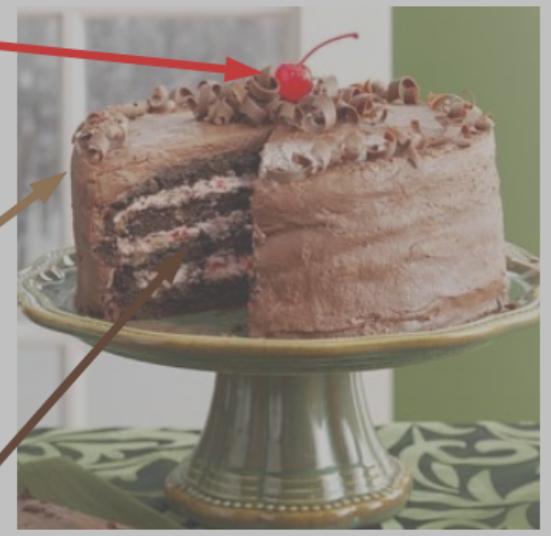
# PREREQUISITES

## Taxonomy of Machine Learning




From **David silver**, Reinforcement learning (UCL course on RL, 2015)

- "Pure" Reinforcement Learning (cherry)
  - The machine predicts a scalar reward given once in a while.
  - **A few bits for some samples**
- Supervised Learning (icing)
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - **10→10,000 bits per sample**
- Unsupervised/Predictive Learning (cake)
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**



- **(Yes, I know, this picture is slightly offensive to RL folks. But I'll make it up)**

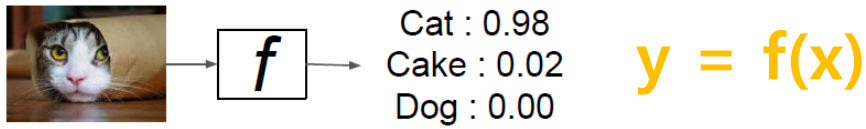
From **Yann Lecun**, (NIPS 2016)

# PREREQUISITES

Introduction

## Supervised Learning

- More flexible solution
  - Get probability of the label for given data instead of label itself




kakaobrain

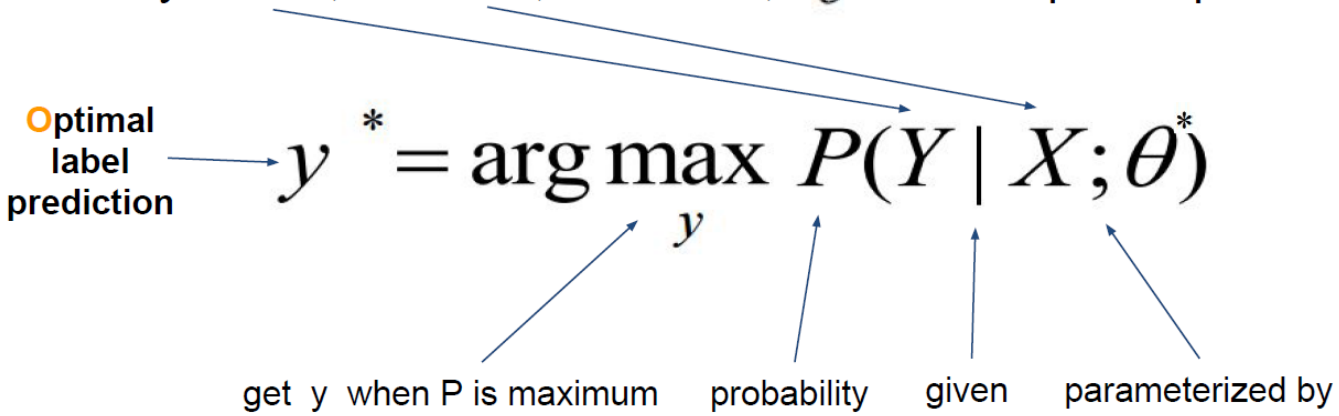
Slide adopted from **Namju Kim**, Kakao brain (SlideShare, AI Forum, 2017)

# PREREQUISITES

Introduction

## Supervised Learning

- Mathematical notation of **classifying** ( greedy policy )
  - y : label, x : data, z : latent, $\theta^*$: fixed optimal parameter

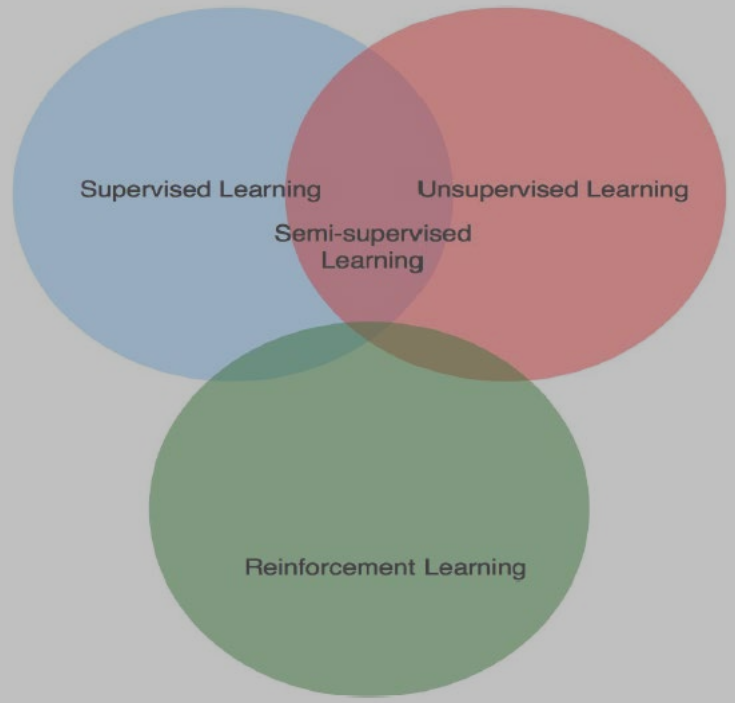



kakaobrain

Slide adopted from **Namju Kim**, Kakao brain (SlideShare, AI Forum, 2017)

# PREREQUISITES

## Taxonomy of Machine Learning




From **David silver**, Reinforcement learning (UCL course on RL, 2015)

- **"Pure" Reinforcement Learning (cherry)**
  - The machine predicts a scalar reward given once in a while.
  - **A few bits for some samples**
- **Supervised Learning (icing)**
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - **10→10,000 bits per sample**
- **Unsupervised/Predictive Learning (cake)**
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**

- **(Yes, I know, this picture is slightly offensive to RL folks. But I'll make it up)**

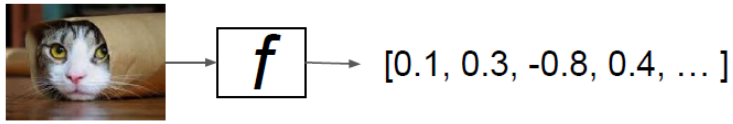
From **Yann Lecun**, (NIPS 2016)

# PREREQUISITES

Introduction

## Unsupervised Learning

- Find deterministic function ***f*** : $z = f(x)$, x : data, z : **latent**




kakaobrain

Slide adopted from **Namju Kim**, Kakao brain (SlideShare, AI Forum, 2017)

# PREREQUISITES



## Unsupervised Learning

- More challenging than supervised learning :
  - No label or curriculum → self learning

- Some NN solutions :
  - Boltzmann machine
  - Auto-encoder or Variational Inference
  - Generative Adversarial Network



Slide adopted from **Namju Kim**, Kakao brain (SlideShare, AI Forum, 2017)

# PREREQUISITES

Introduction

## Unsupervised Learning

- More challenging than supervised learning :
  - No label or curriculum → self learning

- Some NN solutions :
  - Boltzmann machine
  - Auto-encoder or Variational Inference
  - Generative Adversarial Network

kakaobrain

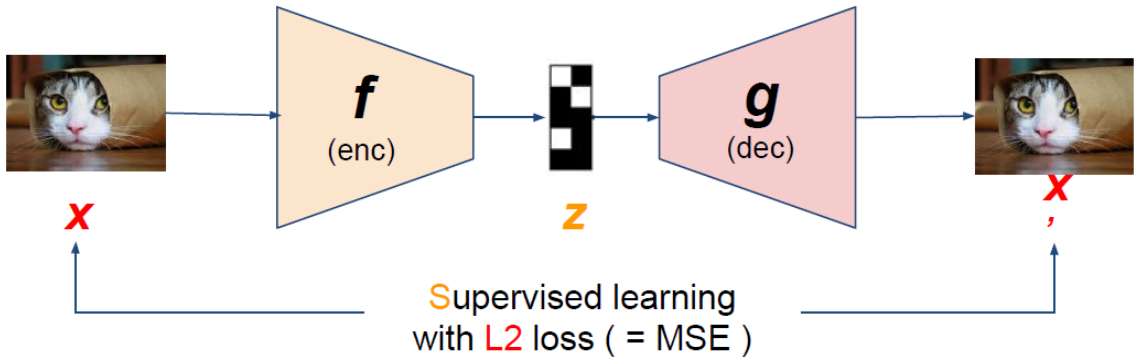
Slide adopted from **Namju Kim**, Kakao brain (SlideShare, AI Forum, 2017)

# PREREQUISITES

Autoencoders

## Stacked autoencoder - SAE

- Use data itself as label → Convert UL into reconstruction SL
- $z = f(x)$, $x = g(z)$ → $x = g(f(x))$
- https://github.com/buriburisuri/sugartensor/blob/master/sugartensor/example/mnist_sae.py




kakaobrain

Slide adopted from **Namju Kim**, Kakao brain (SlideShare, AI Forum, 2017)

# PREREQUISITES

Autoencoders

## Variational autoencoder - VAE

- Kingma et al, “Auto-Encoding Variational Bayes”, 2013.

- Generative Model + Stacked Autoencoder
  - Based on **Variational approximation**

**Variational approximations** Variational methods define a lower bound

$$\mathcal{L}(\boldsymbol{x};\boldsymbol{\theta}) \leq \log p_{\text{model}}(\boldsymbol{x};\boldsymbol{\theta}). \tag{7}$$

kakaobrain

Slide adopted from **Namju Kim**, Kakao brain (SlideShare, AI Forum, 2017)

# PREREQUISITES

Autoencoders

## Variational autoencoder - VAE

- Kingma et al, “Auto-Encoding Variational Bayes”, 2013.

- Generative Model + Stacked Autoencoder
  - Based on **Variational approximation**

**Variational approximations** Variational methods define a lower bound

$$\widetilde{\mathcal{L}}^B(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = -D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z})) + \frac{1}{L}\sum_{l=1}^{L}(\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}^{(i,l)})) \quad (7)$$

where $\mathbf{z}^{(i,l)} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^{(i,l)}, \mathbf{x}^{(i)})$ and $\boldsymbol{\epsilon}^{(l)} \sim p(\boldsymbol{\epsilon})$

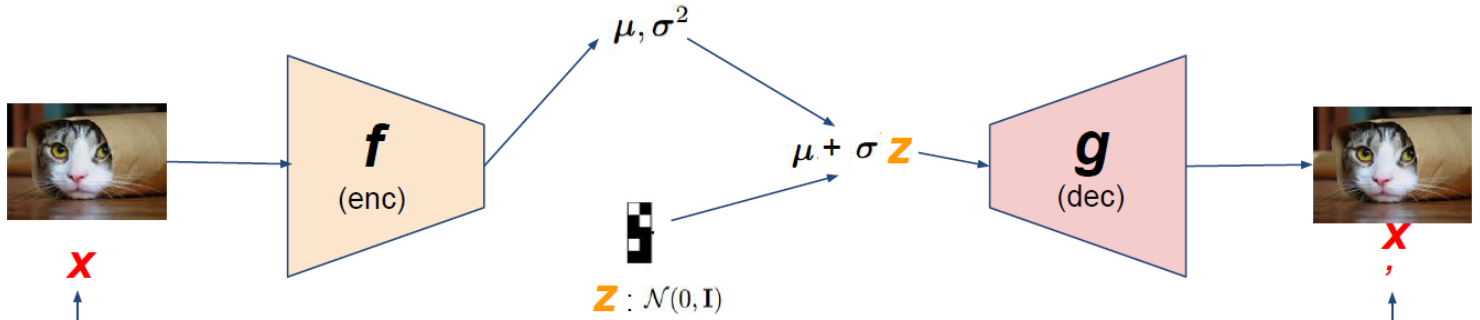
Slide adopted from **Namju Kim**, Kakao brain (SlideShare, AI Forum, 2017)

# PREREQUISITES

Autoencoders

## Variational autoencoder - VAE

- Training
- https://github.com/buriburisuri/sugartensor/blob/master/sugartensor/example/mnist_vae.py




$$\tilde{\mathcal{L}}^B(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = -D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z})) + \frac{1}{L}\sum_{l=1}^{L}(\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}^{(i,l)}))$$

where $\mathbf{z}^{(i,l)} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^{(i,l)}, \mathbf{x}^{(i)})$ and $\boldsymbol{\epsilon}^{(l)} \sim p(\boldsymbol{\epsilon})$

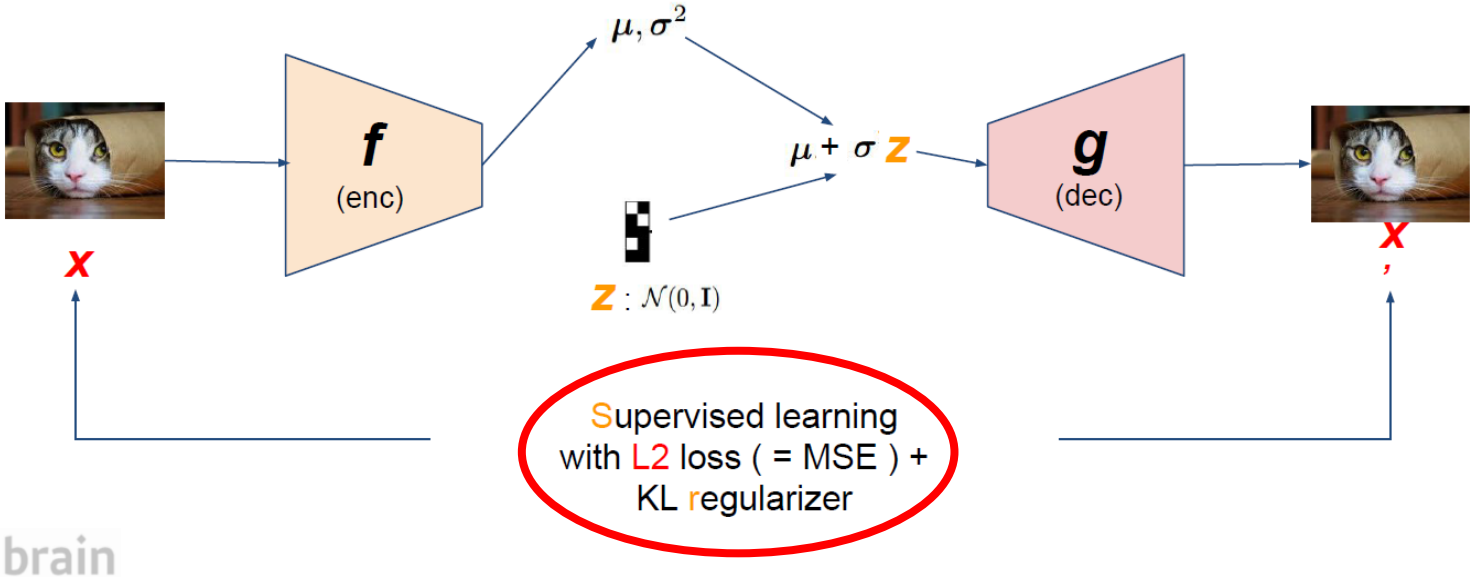
Slide adopted from **Namju Kim**, Kakao brain (SlideShare, AI Forum, 2017)

# PREREQUISITES

Autoencoders

## Variational autoencoder - VAE

- Training
- https://github.com/buriburisuri/sugartensor/blob/master/sugartensor/example/mnist_vae.py




kakaobrain

Slide adopted from **Namju Kim**, Kakao brain (SlideShare, AI Forum, 2017)

# PREREQUISITES

Autoencoders

## Variational autoencoder - VAE

- Results

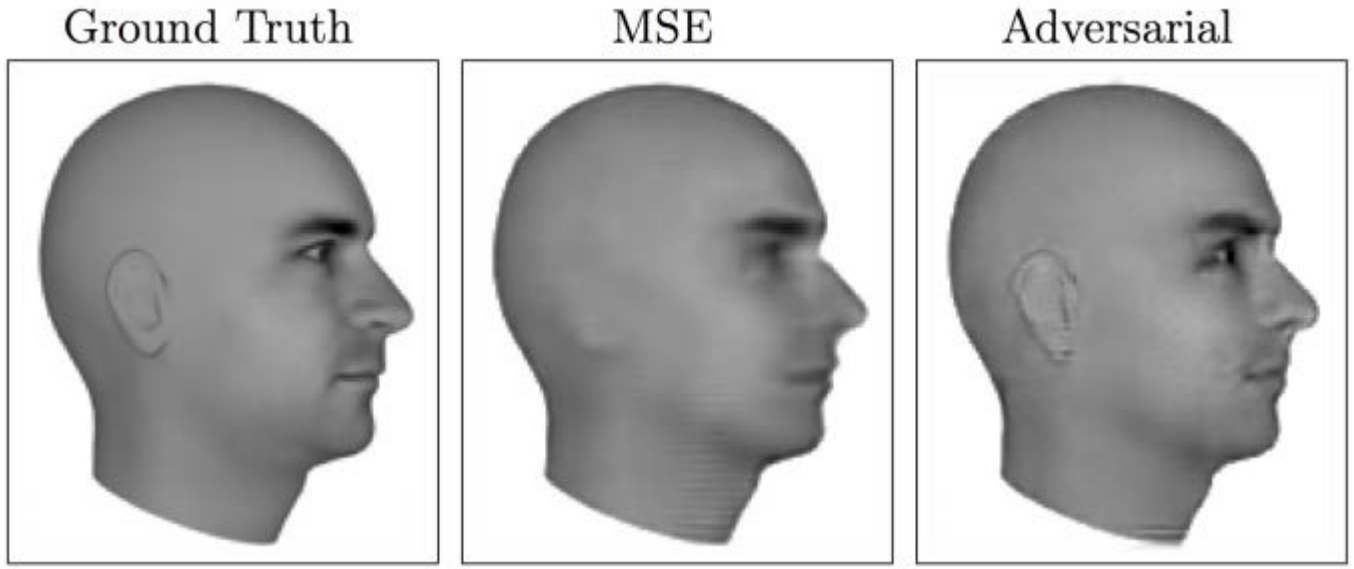
kakaobrain

Slide adopted from **Namju Kim**, Kakao brain (SlideShare, AI Forum, 2017)

# PREREQUISITES

Autoencoders

## Variational autoencoder - VAE




kakaobrain

Slide adopted from **Namju Kim**, Kakao brain (SlideShare, AI Forum, 2017)

* Figure adopted from NIPS 2016 Tutorial: GAN paper, Ian Goodfellow 2016

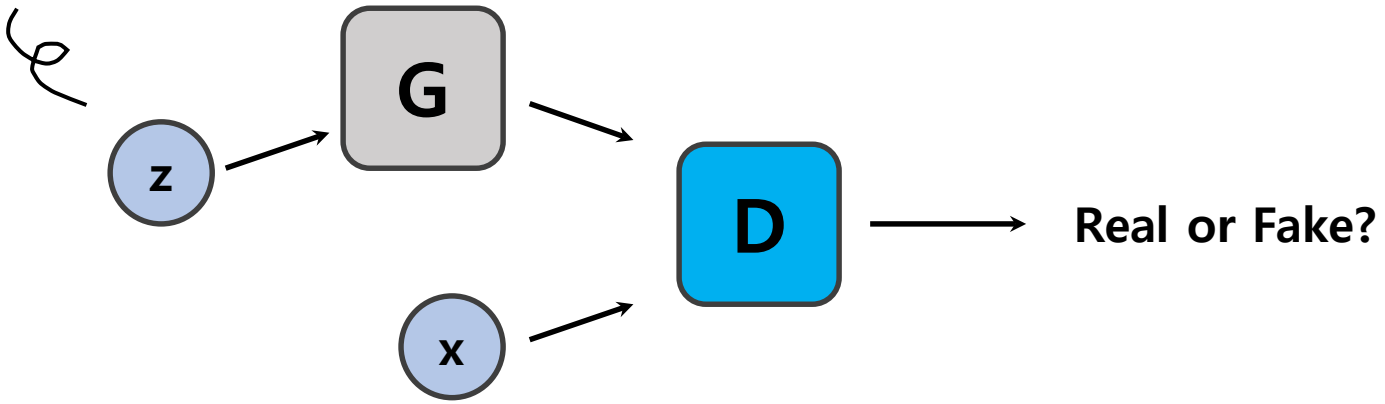
# GAN

# SCHEMATIC OVERVIEW

## Diagram of Standard GAN

$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z \sim p_z(z)}[log(1 - D(G(z)))]$$

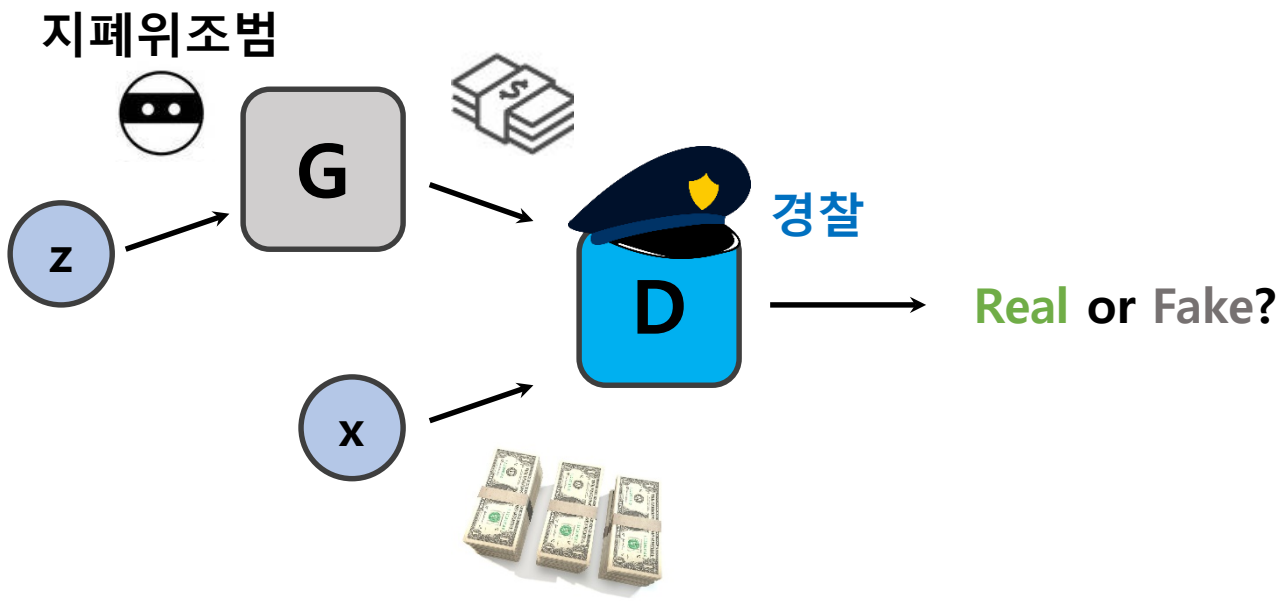
Gaussian noise as an input for G



# SCHEMATIC OVERVIEW

## Diagram of Standard GAN

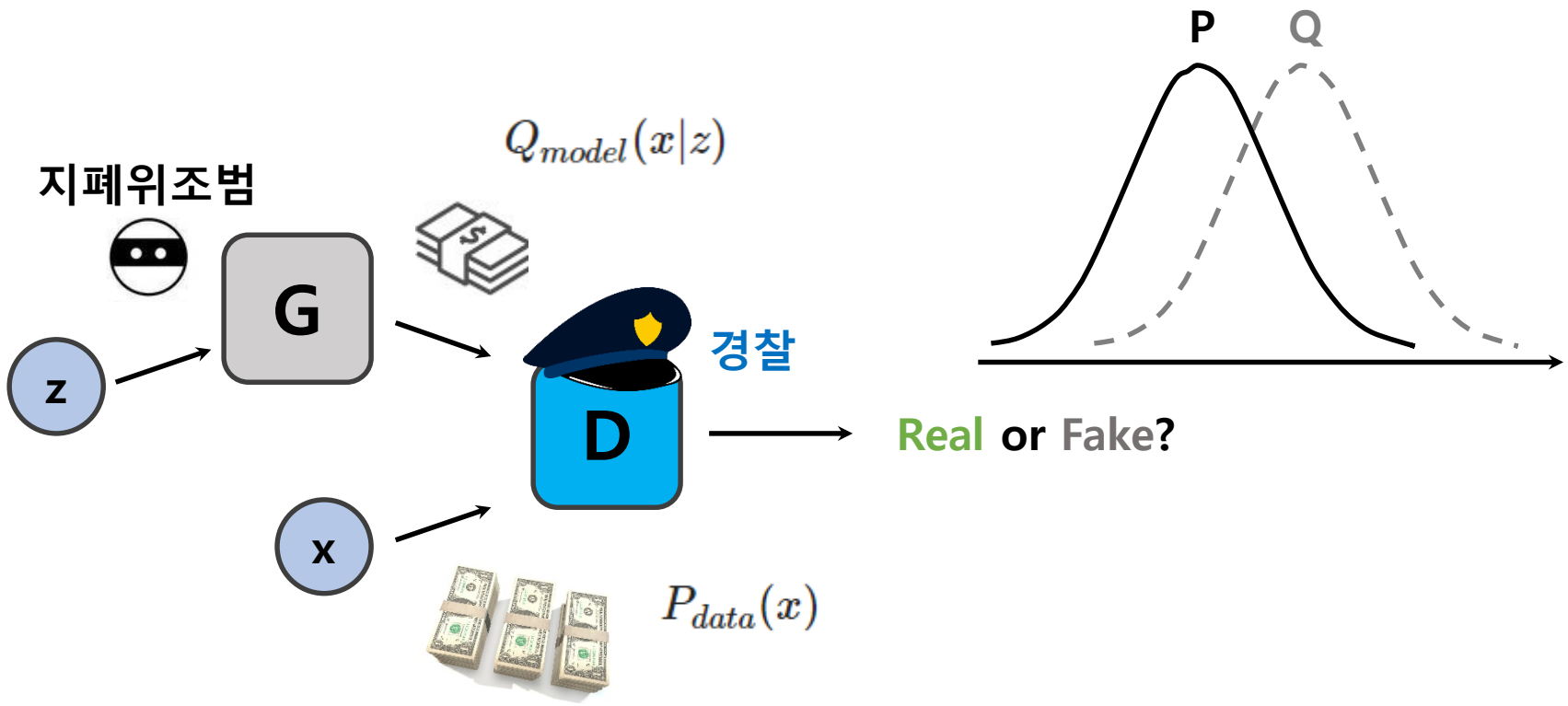
$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z \sim p_z(z)}[log(1 - D(G(z)))]$$



# SCHEMATIC OVERVIEW

## Diagram of Standard GAN

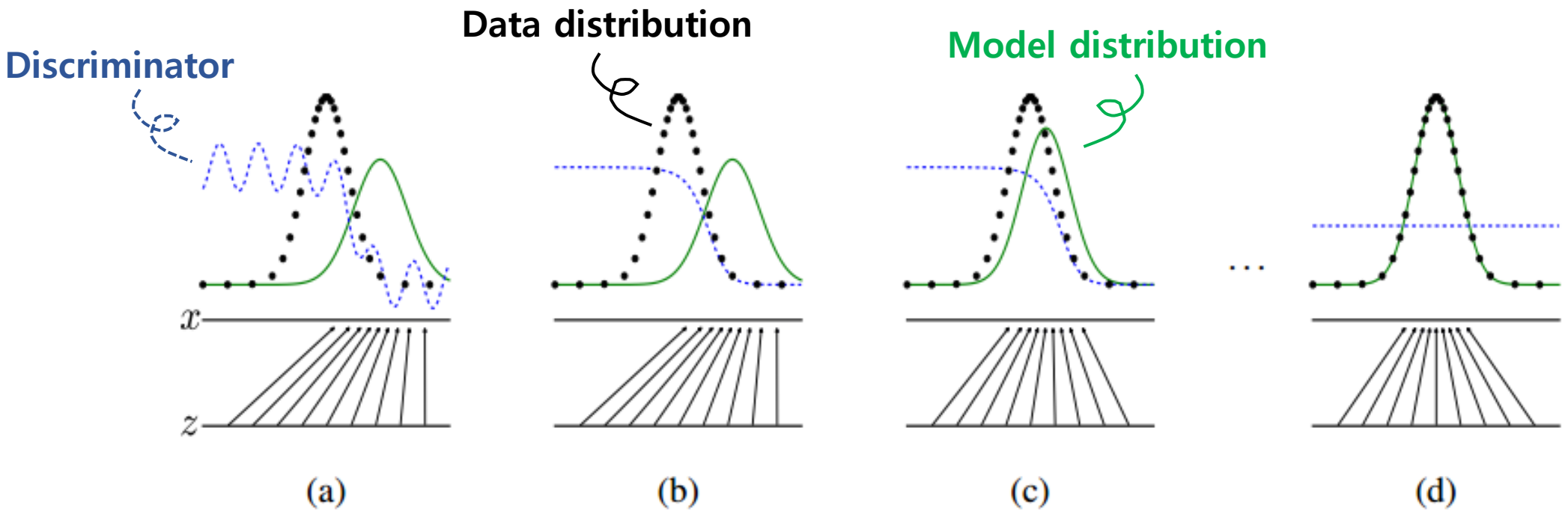
$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z \sim p_z(z)}[log(1 - D(G(z)))]$$



# SCHEMATIC OVERVIEW

## Diagram of Standard GAN

$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z \sim p_z(z)}[log(1 - D(G(z)))]$$




* Figure adopted from Generative Adversarial Nets, Ian Goodfellow et al. 2014

# THEORETICAL RESULTS

## Minimax problem of GAN

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z\sim p_x(z)}[log(1-D(G(z)))]$$

## TWO STEP APPROACH

**Show that…**

1. **The minimax problem of GAN has a global optimum at $p_g = p_{data}$**
2. **The proposed algorithm can find that global optimum**

# THEORETICAL RESULTS

## Proposition 1.

For $G$ fixed, the optimal discriminator $D$ is

$$D_G^*(x) = \frac{p_{data}\ (x)}{p_{data}\ (x) + p_g(x)}.$$

$$\begin{aligned} C(G) &= \max_D V(G,D) \\ &= \mathbb{E}_{x\sim p_{data}}\left[logD_G^*(x)\right] + \mathbb{E}_{z\sim p_z}\left[log(1-D_G^*(G(z)))\right] \\ &= \mathbb{E}_{x\sim p_{data}}\left[logD_G^*(x)\right] + \mathbb{E}_{x\sim p_g}\left[log(1-D_G^*(x))\right] \\ &= \mathbb{E}_{x\sim p_{data}}\left[log\frac{p_{data}\ (x)}{p_{data}\ (x)+p_g(x)}\right] + \mathbb{E}_{x\sim p_g}\left[log\frac{p_g(x)}{p_{data}\ (x)+p_g(x)}\right] \end{aligned}$$

# THEORETICAL RESULTS

## Main Theorem

The global minimum of the virtual training criterion $C(G)$ is achieved if and only if $p_g = p_{data}$. At that point, $C(G)$ achieves the value $-log(4)$.

For $p_g = p_{data}$, $D^*_G(x) = \frac{1}{2}$ and

$$C(G) = \mathbb{E}_{x \sim p_{data}} [-log(2)] + \mathbb{E}_{x \sim p_g} [-log(2)] = -log(4).$$

To show that this is the best possible value of $C(G)$:

$$\begin{aligned} C(G) &= -log(4) + KL\left(p_{data} || \frac{p_{data} + p_g}{2}\right) + KL\left(p_g || \frac{p_{data} + p_g}{2}\right) \\ &= -log(4) + 2 \cdot JSD(p_{data} || p_g). \end{aligned}$$

Here, JSD is always positive value and equal to $0$ only if two distributions match. Therefore, $C^* = -log(4)$ is the global minimum of $C(G)$ where the only solution is $p_g = p_{data}$.

# THEORETICAL RESULTS

## Convergence of the proposed algorithm

If $G$ and $D$ have enough capacity, and at each step of Algorithm 1, the discriminator is allowed to reach its optimum given G, and $p_g$ is updated so as to improve the criterion

$$\mathbb{E}_{x\sim p_{data}}\left[logD^*_G(x)\right] + \mathbb{E}_{x\sim p_g}\left[log(1-D^*_G(x))\right]$$

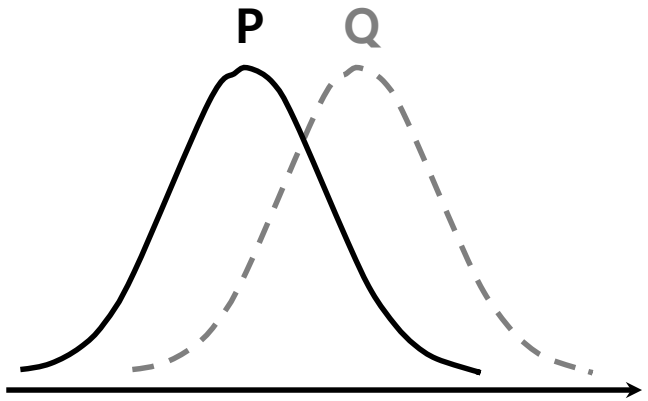
then $p_g$ converges to $p_{data}$.

*Proof*. Consider $V(G,D) = U(p_g, D)$ as a function of $p_g$ as done in the above criterion. Note that $U(p_g, D)$ is convex in $p_g$. The subderivatives of a supremum of convex functions include the derivative of the function at the point where the maximum is attained. This is equivalent to computing a gradient descent update for $p_g$ at the optimal $D$ given the corresponding $G$, $\sup_D U(p_g, D)$ is convex in $p_g$ with a unique global optima as proven in **Thm 1**, therefore with sufficiently small updates of $p_g$, $p_g$ converges to $p_x$, concluding the proof. ■

# SUMMARY

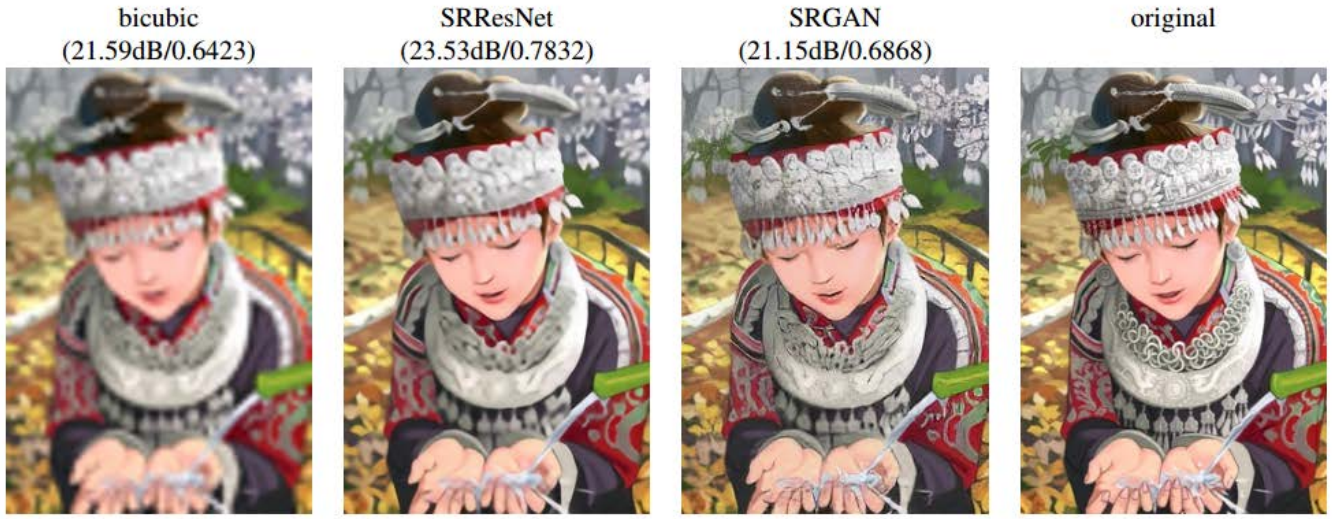
## TAKE HOME KEYPOINTS

- Supervised / Unsupervised / Reinforcement Learning
- Generative Models
- Variational Inference Technique
- Adversarial Training
- Reduce the gap between $Q_{model}$ and $P_{data}$



# RELATED WORKS (APPLICATIONS)

**Super-resolution**



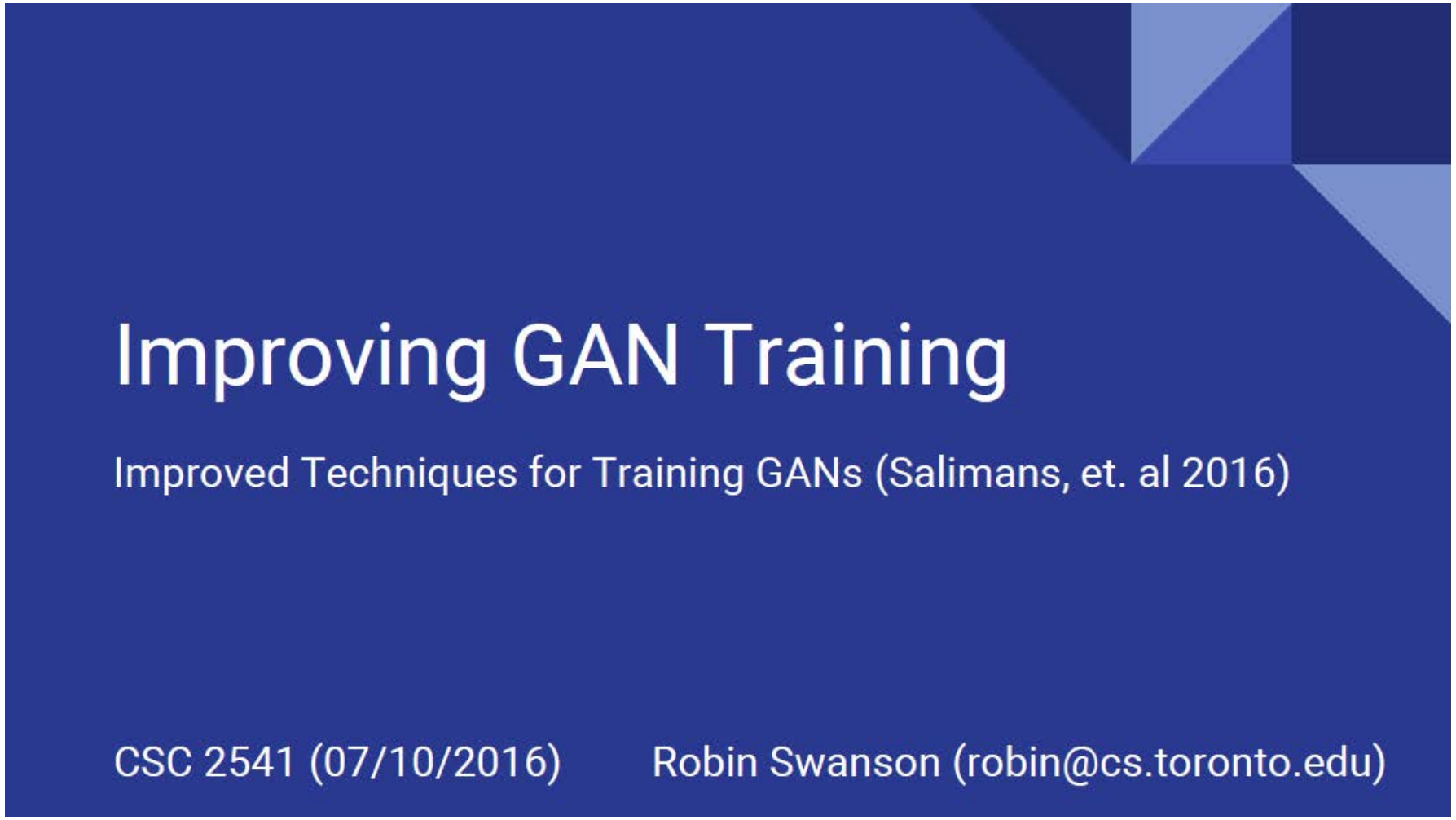

* SRGAN Christian Ledwig et al. 2017

**Domain Adaptation**

**Img2Img Translation**

* CycleGAN Jun-Yan Zhu et al. 2017

# DIFFICULTIES



# DIFFICULTIES

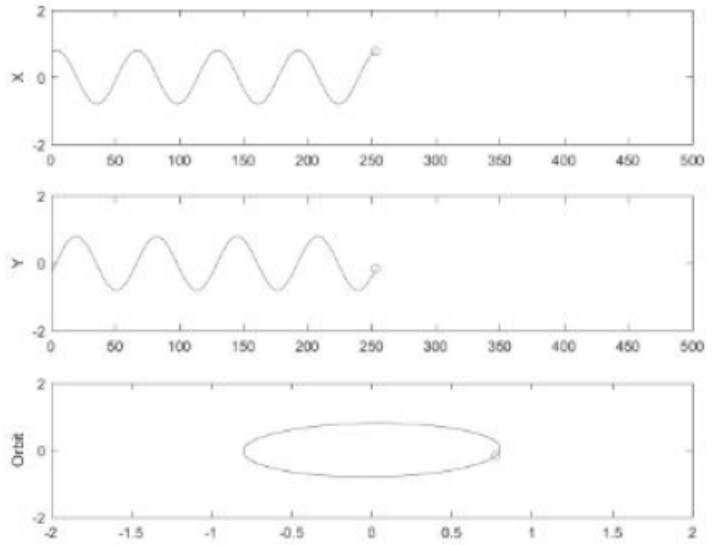
## Training GANs is Difficult

- General Case is hard to solve
  - Cost functions are non-convex
  - Parameters are continuous
  - Extreme Dimensionality
- Gradient descent can’t solve everything
  - Reducing cost of generator could increase cost of discriminator
  - And vice-versa

# DIFFICULTIES CONVERGENCE OF THE MODEL

## Simple Example

- Player 1 minimizes f(x) = xy
- Player 2 minimizes f(y) = -xy
- Gradient descent enters a stable orbit
- Never reaches x = y = 0




(Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. 2016. MIT Press)

# DIFFICULTIES MODE COLLAPSE (SAMPLE DIVERSITY)

this small bird has a pink breast and crown, and black primaries and secondaries.

this magnificent fellow is almost all black with a red crest, and white cheek patch

the flower has petals that are bright pinkish purple with white stigma

this white and yellow flower have thin white petals and a round yellow stamen

(Reed et al 2016)

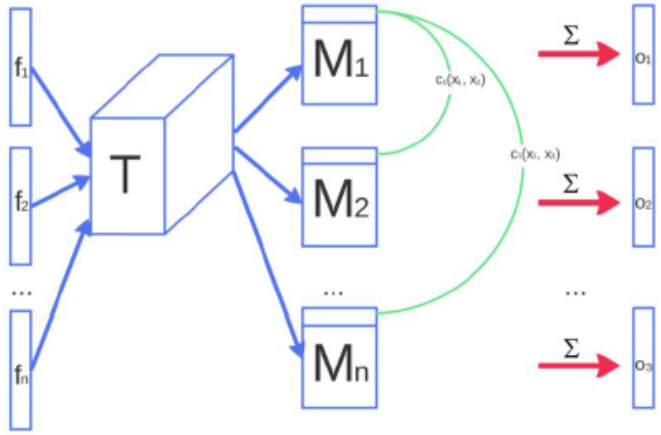

(Reed et al, submitted to ICLR 2017)

* Slide adopted from NIPS 2016 Tutorial, Ian Goodfellow

# DIFFICULTIES

## Minibatch Discrimination

- Discriminator looks at generated examples independently
- Can't discern generator collapse
- Solution: Use other examples as side information
- KL divergence does not change
- JS favours high entropy

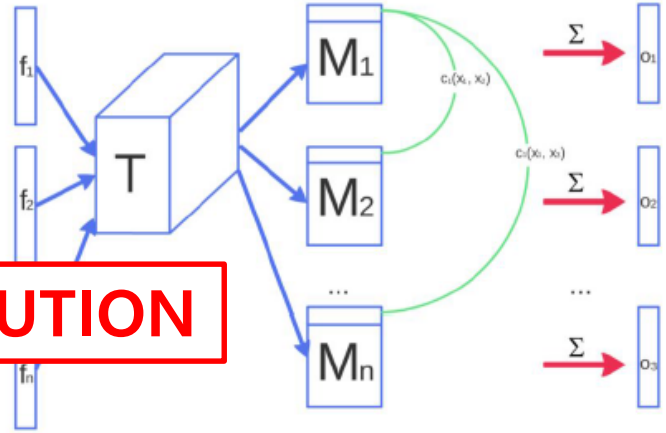



(Ferenc Huszár - http://www.inference.vc/understanding-minibatch-discrimination-in-gans/)

# DIFFICULTIES

## Minibatch Discrimination

- Discriminator looks at generated examples independently
- Can't discern generator collapse
- Solution: Us **TEMPORARY SOLUTION** side information
- KL divergence does not change
- JS favours high entropy




(Ferenc Huszár - http://www.inference.vc/understanding-minibatch-discrimination-in-gans/)

# DIFFICULTIES HOW TO EVALUATE THE QUALITY?

## Ask Somebody

- Solution: Amazon Mechanical Turk
- Problem:
  - "TASK IS HARD."
  - Humans are slow, and unreliable, and ...
- Annotators learn from mistakes


Your score on this question is **6/9**

(http://infinite-chamber-35121.herokuapp.com/cifar-minibatch/)

# DIFFICULTIES

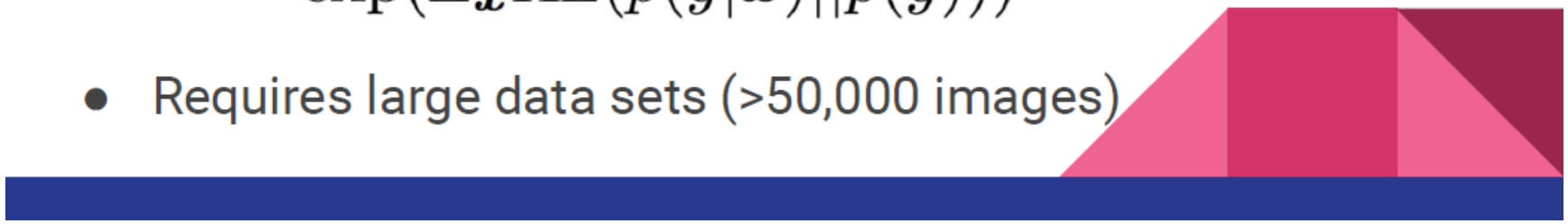
## HOW TO EVALUATE THE QUALITY?

### Inception Score

- Run output through Inception Model
- Images with meaningful objects should have a label distribution (p(y|x)) with low entropy
- Set of output images should be varied
- Proposed score:

$$\exp(\mathbb{E}_{\boldsymbol{x}}\mathrm{KL}(p(y|\boldsymbol{x})||p(y)))$$

- Requires large data sets (>50,000 images)



# DIFFICULTIES HOW TO EVALUATE THE QUALITY?

## Inception Score

- Run output through Inception Model
- Images with meaningful objects should have a label distribution (p(y|x)) with low entropy
- Set of outpu **TEMPORARY SOLUTION**
- Proposed score:

$$\exp(\mathbb{E}_{\boldsymbol{x}}\mathrm{KL}(p(y|\boldsymbol{x})||p(y)))$$

- Requires large data sets (>50,000 images)

# SUMMARY

## “TRAINING GAN IS HARD”

- Power balance (NO learning)
- Convergence (oscillation)
- Mode collapse
- Evaluation (GAN training loss is intractable)

# SUMMARY

## “TRAINING GAN IS HARD”

- Power balance (NO learning)
- Co
- Mode collapse
- Evaluation (GAN training loss is intractable)

**HOW TO SOLVE THESE PROBLEMS?**

# DCGAN

LET'S CAREFULLY SELECT THE ARCHITECTURE!

# MOTIVATION

## TRAINING IS TOOOO HARD…

(Ahhh…it just does not work…orz…)

# SCHEMATIC OVERVIEW

## Guideline for stable learning

Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

# SCHEMATIC OVERVIEW

## Guideline for stable learning

Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

"However, *after extensive model exploration* we identified a family of architectures that resulted in stable training across a range of datasets and allowed for higher resolution and deeper generative models."

# SCHEMATIC OVERVIEW

## Guideline for stable learning

Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.

**"Most GANs today are at least loosely based on the DCGAN architecture."**

**- NIPS 2016 Tutorial by Ian Goodfellow**

"However, *after extensive model exploration* we identified a family of architectures that resulted in stable training across a range of datasets and allowed for higher resolution and deeper generative models."

# KEYPOINTS

**Okay, learning is finished and the model converged. Then…**

**"How to show that our network or generator learned A MEANINGFUL FUNCTION?"**

# KEYPOINTS

## Okay, learning is finished and the model converged. Then…

### Show that

- The generator **DOES NOT MEMORIZED** the images.
- There are **NO SHARP TRANSITION** while walking in the latent space.
- The generator **UNDERSTANDS** the feature of the data.

# RESULTS

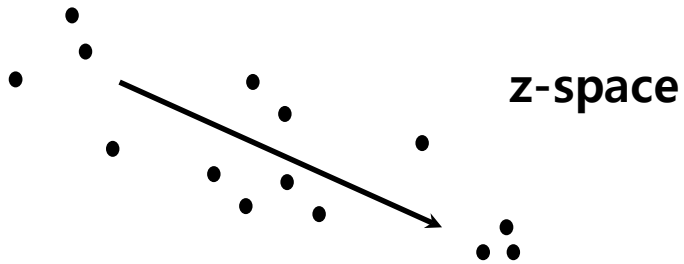
## What can GAN do?

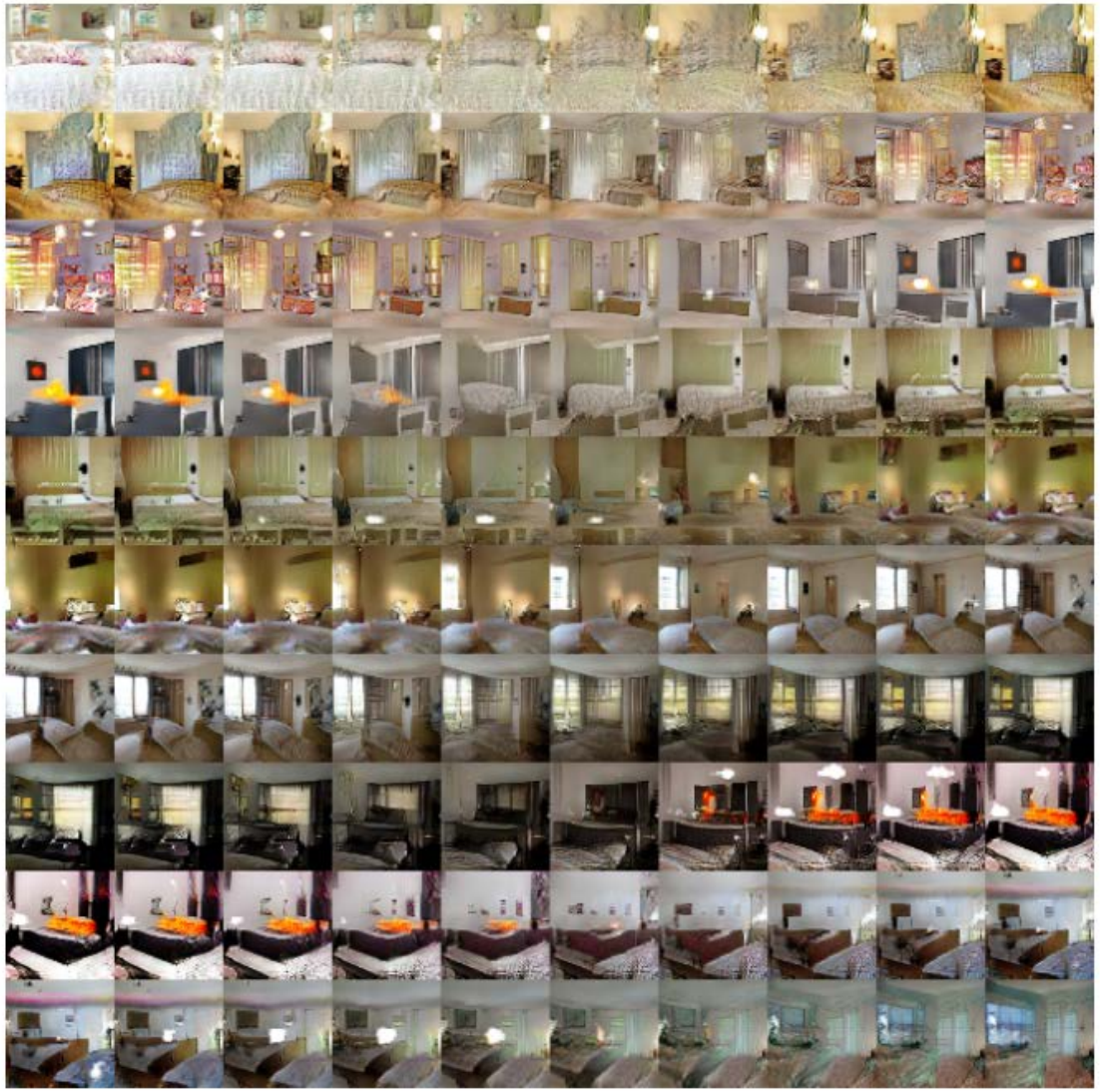
* Figure adopted from DCGAN, Alec Radford et al. 2016 (link)

# RESULTS

## What can GAN do?

"Walking in the latent space"

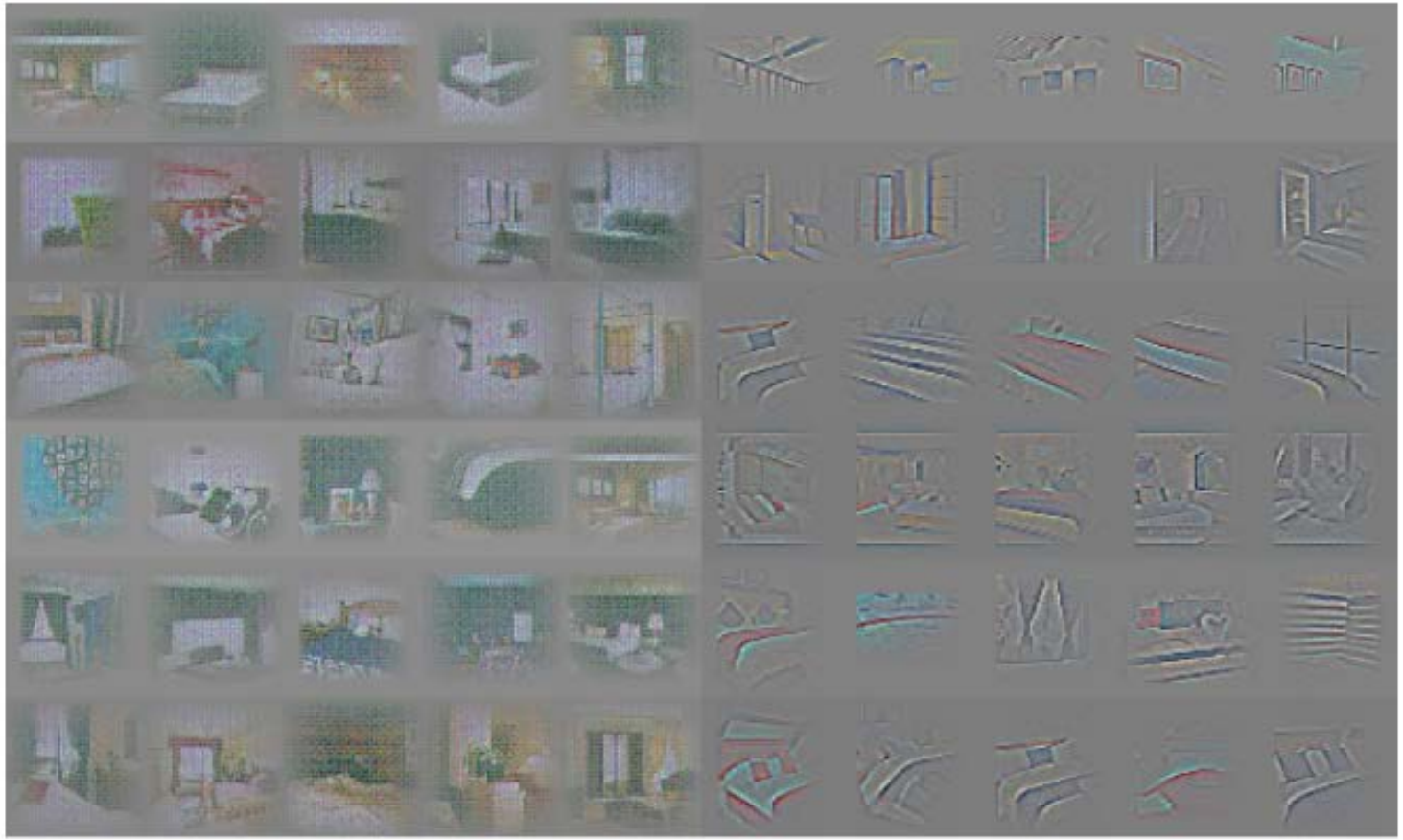


* Figure adopted from DCGAN, Alec Radford et al. 2016 (link)

# RESULTS

## What can GAN do?



Random filters | Trained filters

* Figure adopted from DCGAN, Alec Radford et al. 2016 (link)

# RESULTS

## What can GAN do?

"**Forgetting the feature it learned**"

* Figure adopted from DCGAN, Alec Radford et al. 2016 (link)

# RESULTS

## What can GAN do?

"**Vector arithmetic**"
(e.g. word2vec)

$$KING\,(\text{왕}) - MAN\,(\text{남자}) + WOMAN\,(\text{여자})$$

# RESULTS

## What can GAN do?

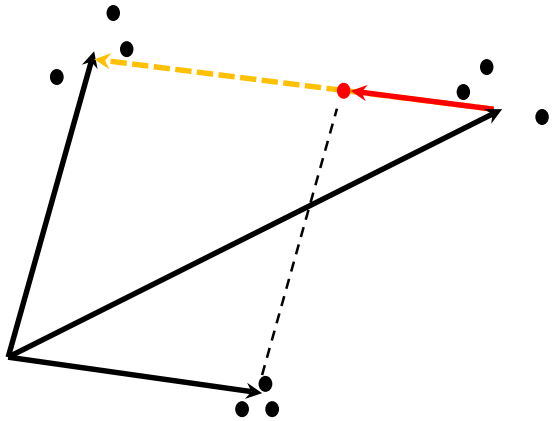
"**Vector arithmetic**"
(e.g. word2vec)

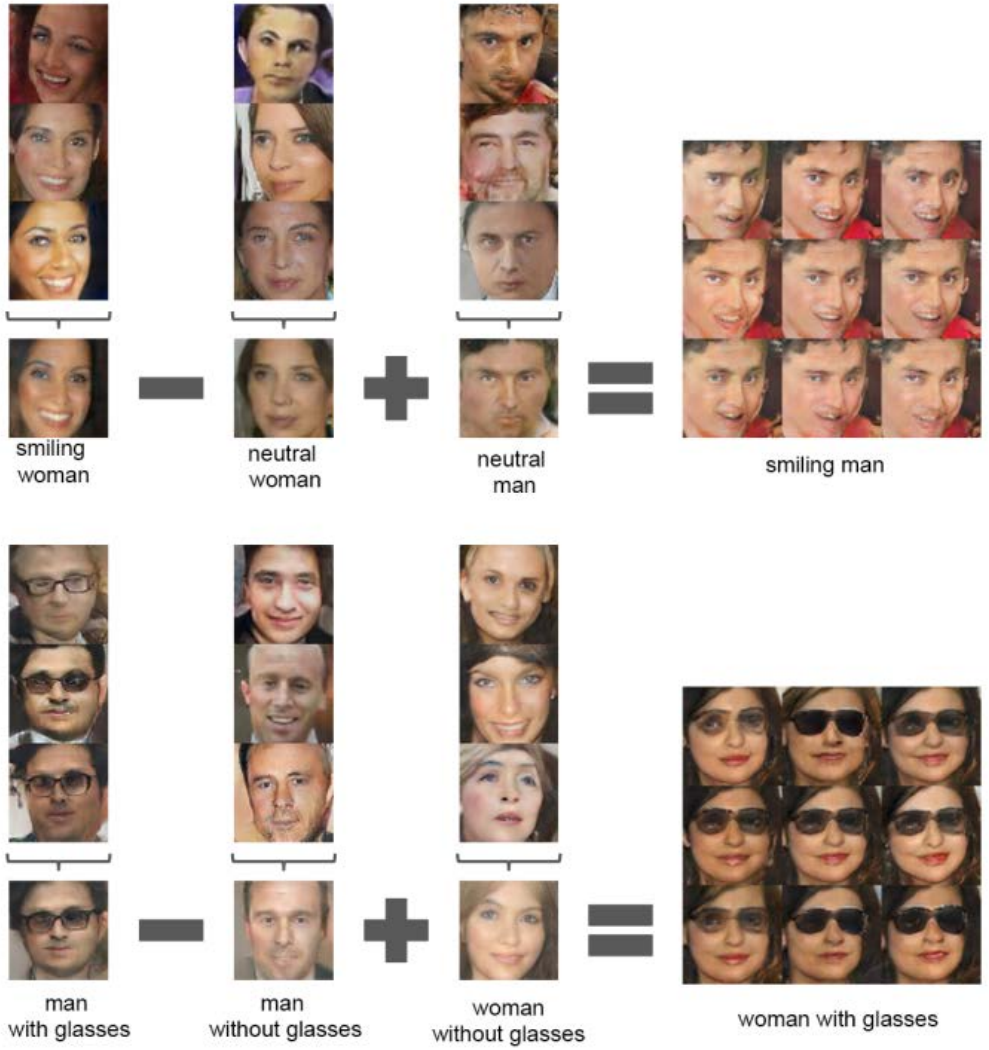
$$QUEEN\ (\text{여왕})$$

# RESULTS

## What can GAN do?

"**Vector arithmetic**"
(e.g. word2vec)






* Figure adopted from DCGAN, Alec Radford et al. 2016 (link)

# RESULTS

## What can GAN do?

"**Understand the meaning of the data**"
(e.g. code: rotation, category, and etc.)

**Neural network understanding "Rotation"**

* Figure adopted from DCGAN, Alec Radford et al. 2016 (link)

# SUMMARY

1. Guideline for stable learning
2. Good analysis on the results

- Show that the generator DOES NOT MEMORIZED the images
- Show that there are NO SHARP TRANSITION while walking in the latent space
- Show that the generator UNDERSTANDS the feature of the data

# Unrolled GAN

**LET'S GIVE EXTRA INFORMATION TO THE NETWORK**
**(allow it to 'see into the future')**

# MOTIVATION

## Convergence of the proposed algorithm

If $G$ and $D$ have enough capacity, and at each step of Algorithm 1, the discriminator is allowed to reach its optimum given G, and $p_g$ is updated so as to improve the criterion

$$\mathbb{E}_{x\sim p_{data}}\left[logD_G^*(x)\right] + \mathbb{E}_{x\sim p_g}\left[log(1-D_G^*(x))\right]$$

then $p_g$ converges to $p_{data}$.

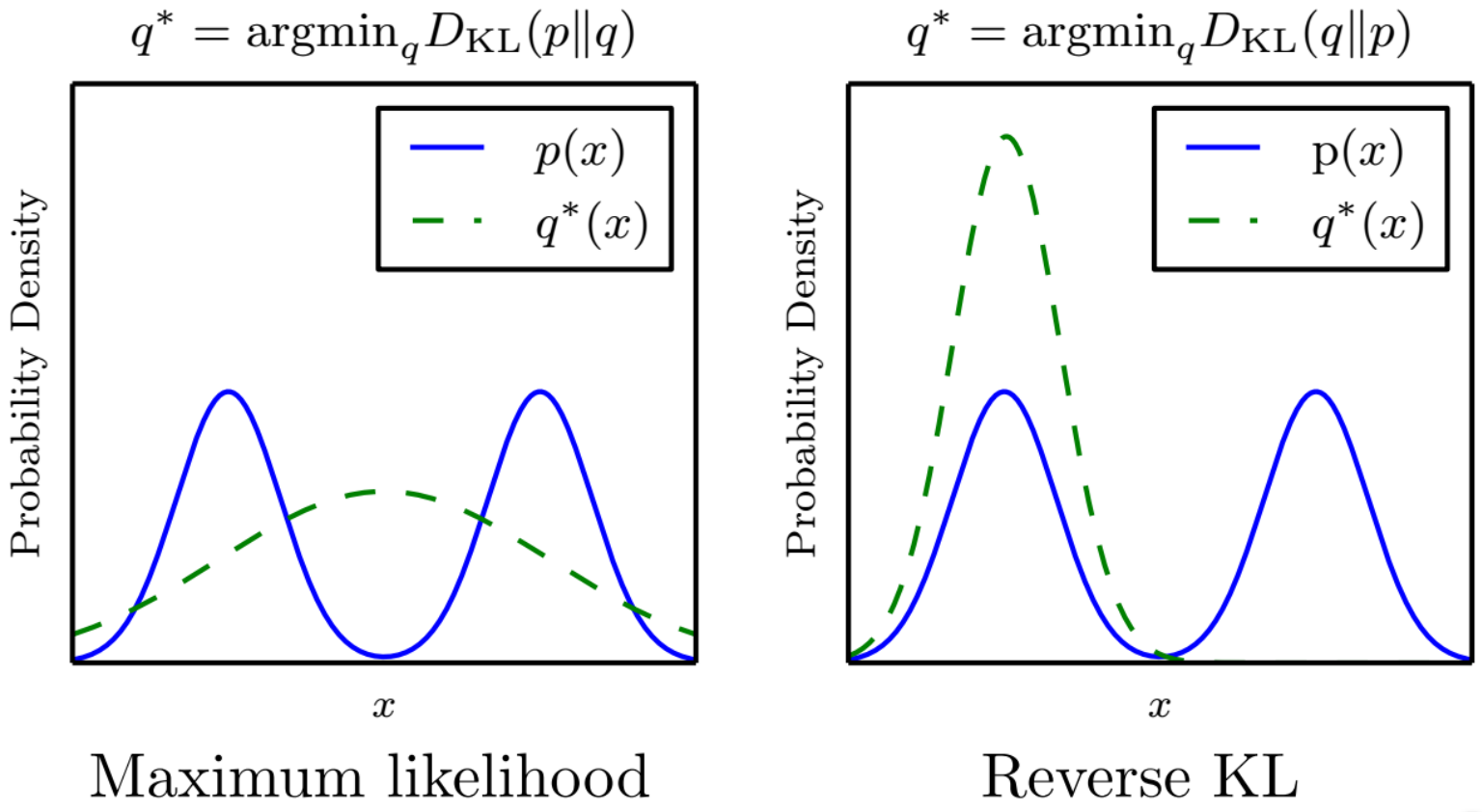
For $G$ fixed, the optimal discriminator $D$ is

$$D_G^*(x) = \frac{p_{data}\ (x)}{p_{data}\ (x) + p_g(x)}.$$

**Impossible to achieve in practice**

# MOTIVATION

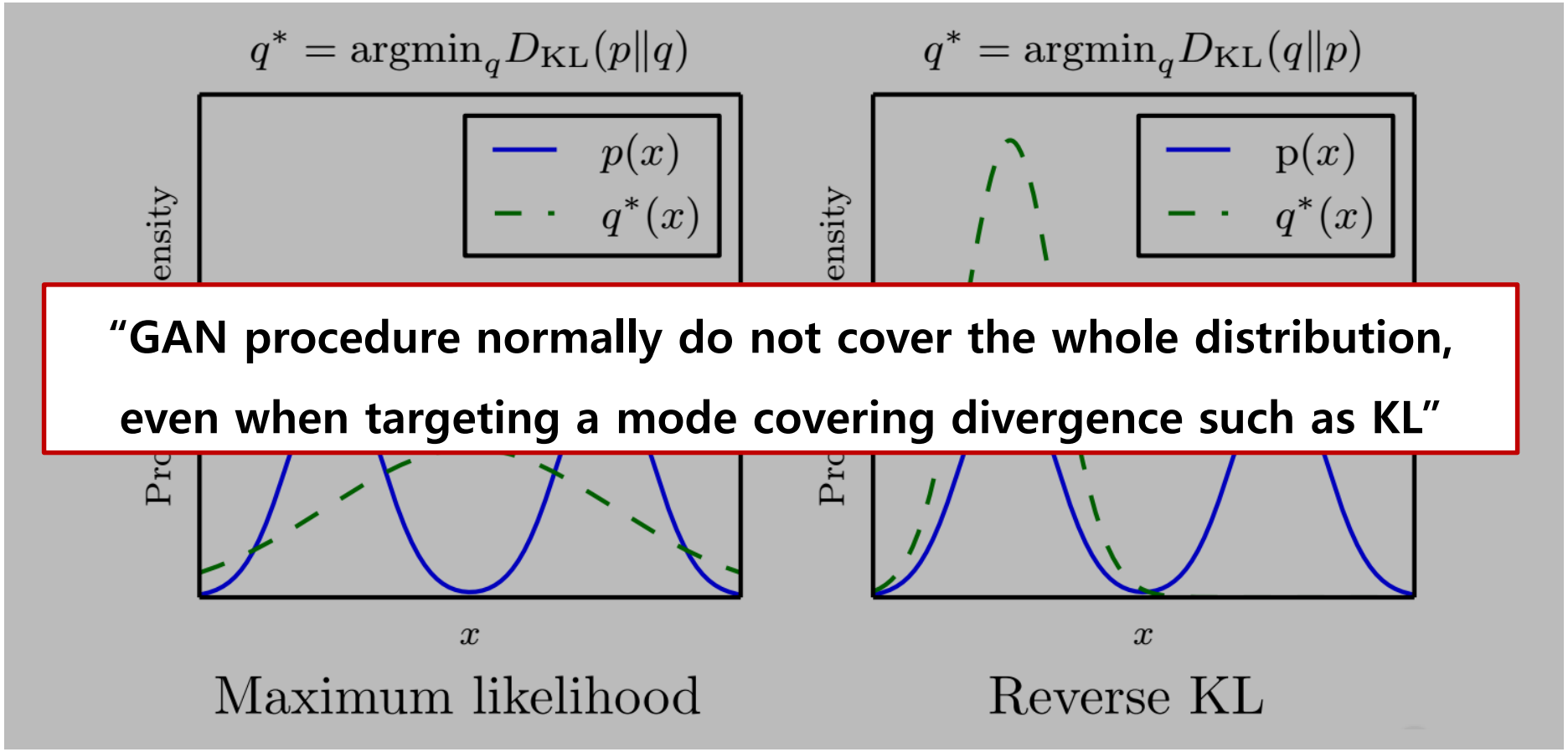
## WHAT HAPPENS?




* Figure adopted from NIPS 2016 Tutorial GAN, Ian Goodfellow 2016

# MOTIVATION

## WHAT HAPPENS?




**"GAN procedure normally do not cover the whole distribution, even when targeting a mode covering divergence such as KL"**

* Figure adopted from NIPS 2016 Tutorial GAN, Ian Goodfellow 2016

# MOTIVATION

## WHAT HAPPENS?

$$G^* = \min_G \max_D V(G, D)$$

*VS*

$$G^* = \max_D \min_G V(G, D)$$

# MOTIVATION

## WHAT HAPPENS?

$$G^* = \min_G \max_D V(G, D)$$

*VS*

$$G^* = \max_D \min_G V(G, D)$$

* A single output which can fool the current discriminator most

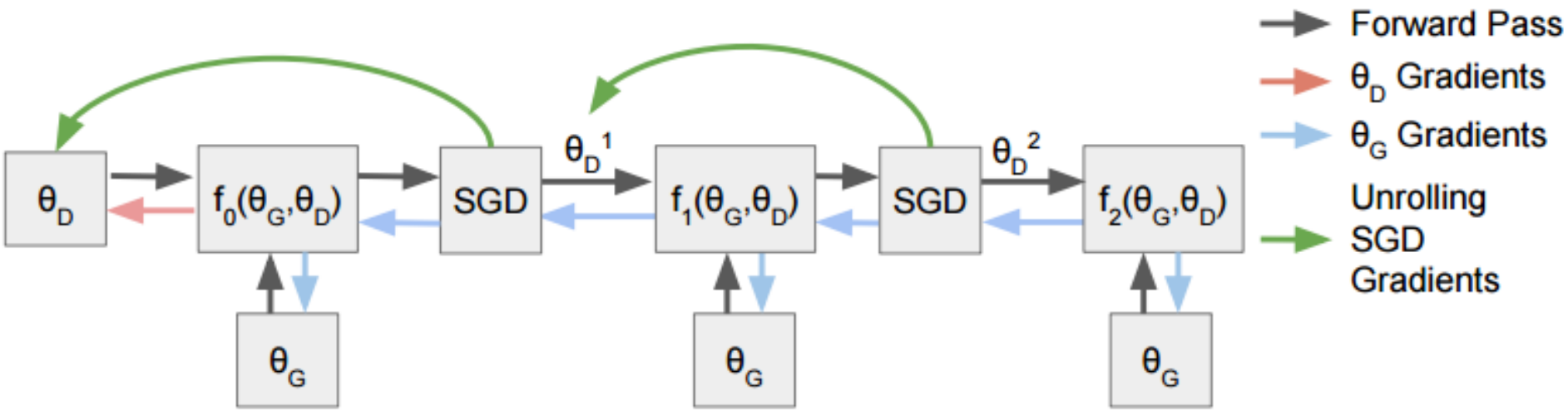
# SCHEMATIC OVERVIEW

**“Adversarial games are not guaranteed to converge using gradient descent, e.g. rock, scissor, paper.”**

* https://www.youtube.com/watch?v=JmON4S0kI04

# UNROLLED GAN

## Let's "UNROLL" the discriminator to see several modes!




$$\theta_D^0 = \theta_D$$

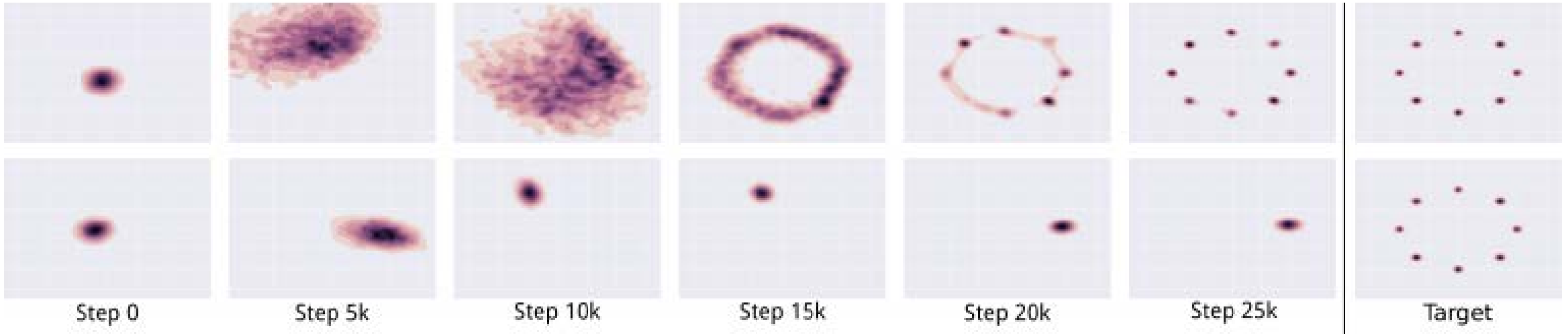
$$\theta_D^{k+1} = \theta_D^k + \eta^k \frac{\mathrm{d}f\left(\theta_G, \theta_D^k\right)}{\mathrm{d}\theta_D^k}$$

$$f_K\left(\theta_G, \theta_D\right) = f\left(\theta_G, \theta_D^K\left(\theta_G, \theta_D\right)\right)$$

$$\theta_D^* = \lim_{k \to \infty} \theta_D^k,$$

* Figure adopted from Unrolled GAN, Luke Metz et al. 2016

# UNROLLED GAN

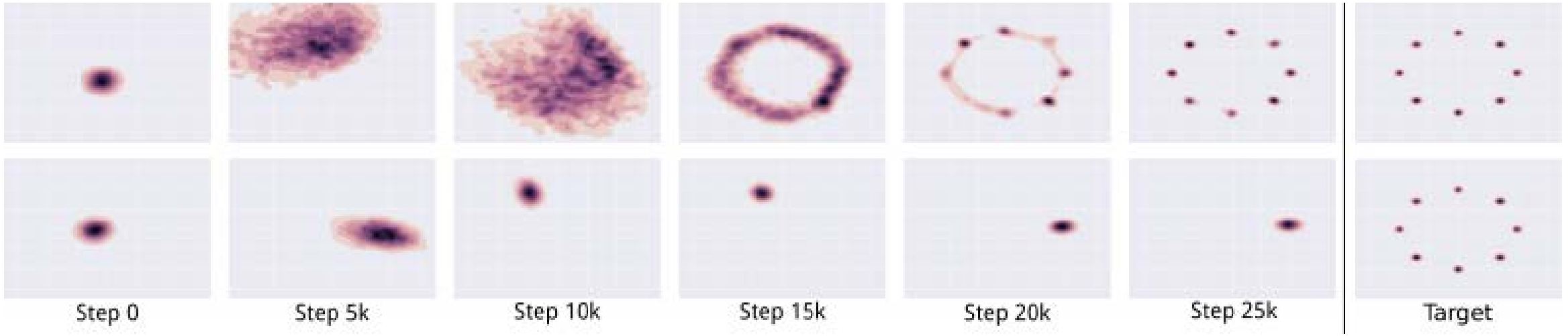



$$\theta_G \leftarrow \theta_G - \eta \frac{\mathrm{d} f_K\left(\theta_G, \theta_D\right)}{\mathrm{d} \theta_G}$$

$$\theta_D \leftarrow \theta_D + \eta \frac{\mathrm{d} f\left(\theta_G, \theta_D\right)}{\mathrm{d} \theta_D}.$$

* Figure adopted from Unrolled GAN, Luke Metz et al. 2016

# UNROLLED GAN




$$\theta_G \leftarrow \theta_G - \eta \frac{\mathrm{d} f_K\left(\theta_G, \theta_D\right)}{\mathrm{d}\theta_G}$$

$$\theta_D \leftarrow \theta_D + \eta \frac{\mathrm{d} f\left(\theta_G, \theta_D\right)}{\mathrm{d}\theta_D}$$

*** Here, only the "G" part is unrolled**
(∵In practice, "D" usually over-powers "G")

* Figure adopted from Unrolled GAN, Luke Metz et al. 2016

# UNROLLED GAN

## The Missing Gradient Term

$$\frac{\mathrm{d}f_K\left(\theta_G,\theta_D\right)}{\mathrm{d}\theta_G}=\frac{\partial f\left(\theta_G,\theta_D^K\left(\theta_G,\theta_D\right)\right)}{\partial\theta_G}+\frac{\partial f\left(\theta_G,\theta_D^K\left(\theta_G,\theta_D\right)\right)}{\partial\theta_D^K\left(\theta_G,\theta_D\right)}\frac{\mathrm{d}\theta_D^K\left(\theta_G,\theta_D\right)}{\mathrm{d}\theta_G}$$

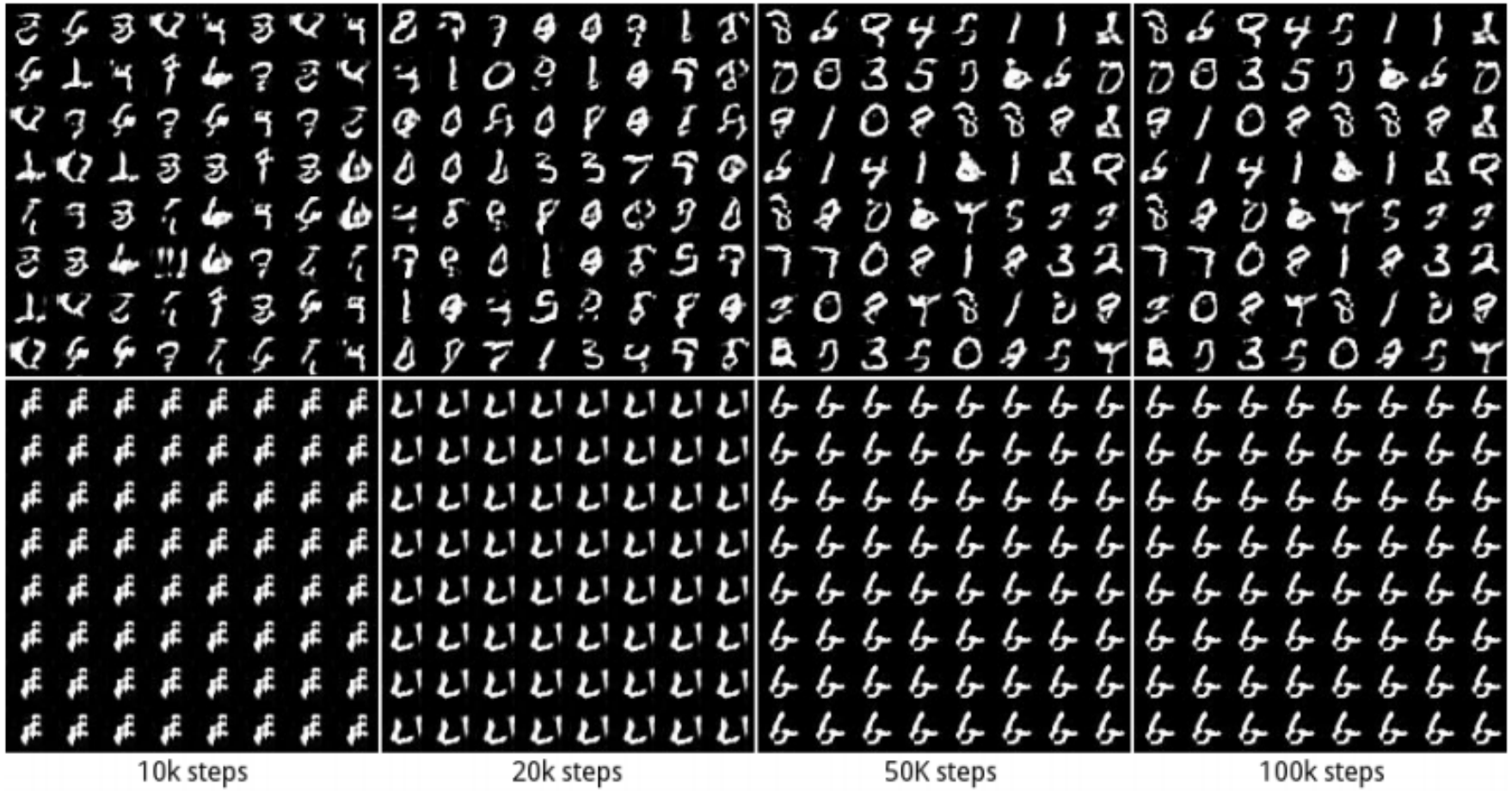
**"How the discriminator would react to a change in the generator."**

$K\to\infty$, $\theta_D^K$ goes to a local optimum of $f$, where $\frac{\partial f}{\partial\theta_D^K}=0$.

Trade off between $K\to\infty$, $K=0$ (두 가지 경우를 모두 생각해봅시다. )

# RESULTS

## Increased stability in terms of power balance




* Figure adopted from Unrolled GAN, Luke Metz et al. 2016

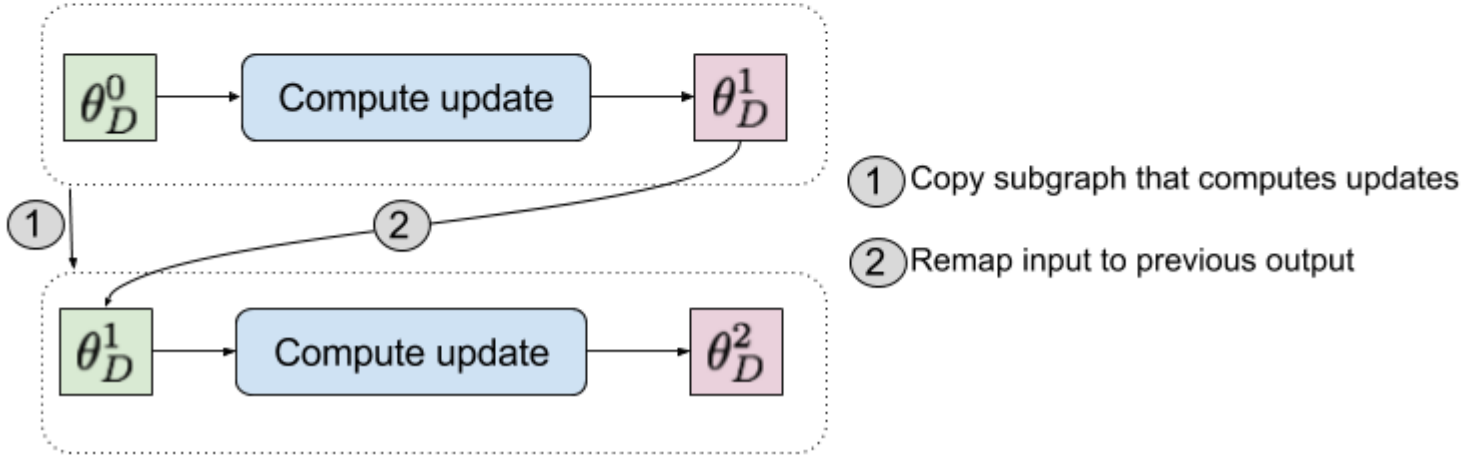
# SUMMARY

1. Address the mode collapsing problem
2. Unrolling the optimization problem
   - Make the discriminator optimal as possible as it can

# IMPLEMENTATION

## Implementation details

To backpropagate through the optimization process, we need to create a symbolic computational graph that includes all the operations from the initial weights to the optimized weights. TensorFlow's built-in optimizers use custom C++ code for efficiency, and do not construct a symbolic graph that is differentiable. For this notebook, we use the optimization routines from keras to compute updates. Next, we use `tf.contrib.graph_editor.graph_replace` to build a copy of the graph containing the mapping from initial weights to updated weights after one optimization iteration, but replacing the initial weights with the last iteration's weights:




* Codes from the jupyter notebook of Ben Poole (2nd Author)
: https://github.com/poolio/unrolled_gan

# IMPLEMENTATION

### Utility functions

In [3]:
```
def extract_update_dict(update_ops):
    """Extract variables and their new values from Assign and AssignAdd ops.

    Args:
        update_ops: list of Assign and AssignAdd ops, typically computed using Keras' opt.get_updates()

    Returns:
        dict mapping from variable values to their updated value
    """
    name_to_var = {v.name: v for v in tf.global_variables()}
    updates = OrderedDict()
    for update in update_ops:
        var_name = update.op.inputs[0].name
        var = name_to_var[var_name]
        value = update.op.inputs[1]
        if update.op.type == 'Assign':
            updates[var.value()] = value
        elif update.op.type == 'AssignAdd':
            updates[var.value()] = var + value
        else:
            raise ValueError("Update op type (%s) must be of type Assign or AssignAdd"%update_op.op.type)
    return updates
```

* Codes from the jupyter notebook of Ben Poole (2nd Author)
: https://github.com/poolio/unrolled_gan

# IMPLEMENTATION

```
# Saddle objective
loss = tf.reduce_mean(
    tf.nn.sigmoid_cross_entropy_with_logits(logits=real_score, labels=tf.ones_like(real_score)) +
    tf.nn.sigmoid_cross_entropy_with_logits(logits=fake_score, labels=tf.zeros_like(fake_score)))

gen_vars = tf.get_collection(tf.GraphKeys.TRAINABLE_VARIABLES, "generator")
disc_vars = tf.get_collection(tf.GraphKeys.TRAINABLE_VARIABLES, "discriminator")

# Vanilla discriminator update
d_opt = Adam(lr=params['disc_learning_rate'], beta_1=params['beta1'], epsilon=params['epsilon'])
updates = d_opt.get_updates(disc_vars, [], loss)
d_train_op = tf.group(*updates, name="d_train_op")

# Unroll optimization of the discrimiantor
if params['unrolling_steps'] > 0:
    # Get dictionary mapping from variables to their update value after one optimization step
    update_dict = extract_update_dict(updates)
    cur_update_dict = update_dict
    for i in xrange(params['unrolling_steps'] - 1):
        # Compute variable updates given the previous iteration's updated variable
        cur_update_dict = graph_replace(update_dict, cur_update_dict)
    # Final unrolled loss uses the parameters at the last time step
    unrolled_loss = graph_replace(loss, cur_update_dict)
else:
    unrolled_loss = loss

# Optimize the generator on the unrolled loss
g_train_opt = tf.train.AdamOptimizer(params['gen_learning_rate'], beta1=params['beta1'], epsilon=params['epsilon'])
g_train_op = g_train_opt.minimize(-unrolled_loss, var_list=gen_vars)
```

* Codes from the jupyter notebook of Ben Poole (2nd Author)
: https://github.com/poolio/unrolled_gan

# InfoGAN

LET'S USE ADDITIONAL CONSTRAINTS FOR THE GENERATOR

# MOTIVATION

**"We want to get a disentangled representation space EXPLICITLY."**

**Neural network understanding "Rotation"**

* Figure adopted from DCGAN paper (link)

# MOTIVATION

**"We want to get a disentangled representation space EXPLICITLY."**


**Neural network understanding "Digit Type"**

* Figure adopted from infoGAN paper (link)

# MOTIVATION

How can we achieve
unsupervised learning of disentangled representation?

In general, learned representation is entangled,
i.e. encoded in a data space in a complicated manner

When a representation is **disentangled**, it would be
more interpretable and easier to apply to tasks

**특징이 서로 얽혀 있어 해석이 불가능한 Physical space에서**
**해석이 용이하도록 서로 독립적인 Eigen space로 변환하는 것 처럼**

* Slide adopted from Takato Horii's slides in SlideShare (link)

# SCHEMATIC OVERVIEW

## infoGAN

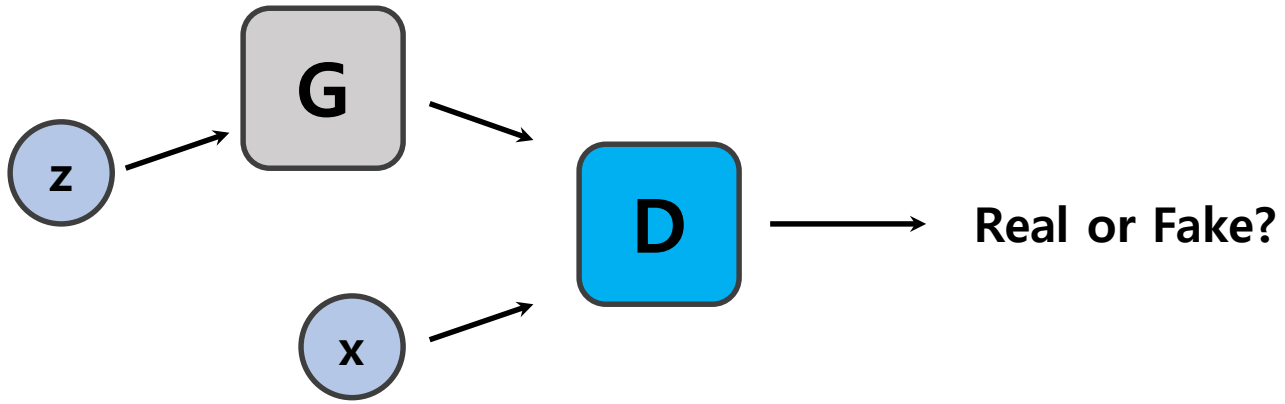
- When Generator studies data representations, infoGAN imposes an extra constraint to make NN learn the feature space in disentangled way.
- Unlike standard GAN, Generator takes a pair of variables as an input:

    1. Gaussian noise z (source of incompressible noise)
    2. latent code c (semantic feature of data distribution)

# SCHEMATIC OVERVIEW

## Diagram of Standard GAN

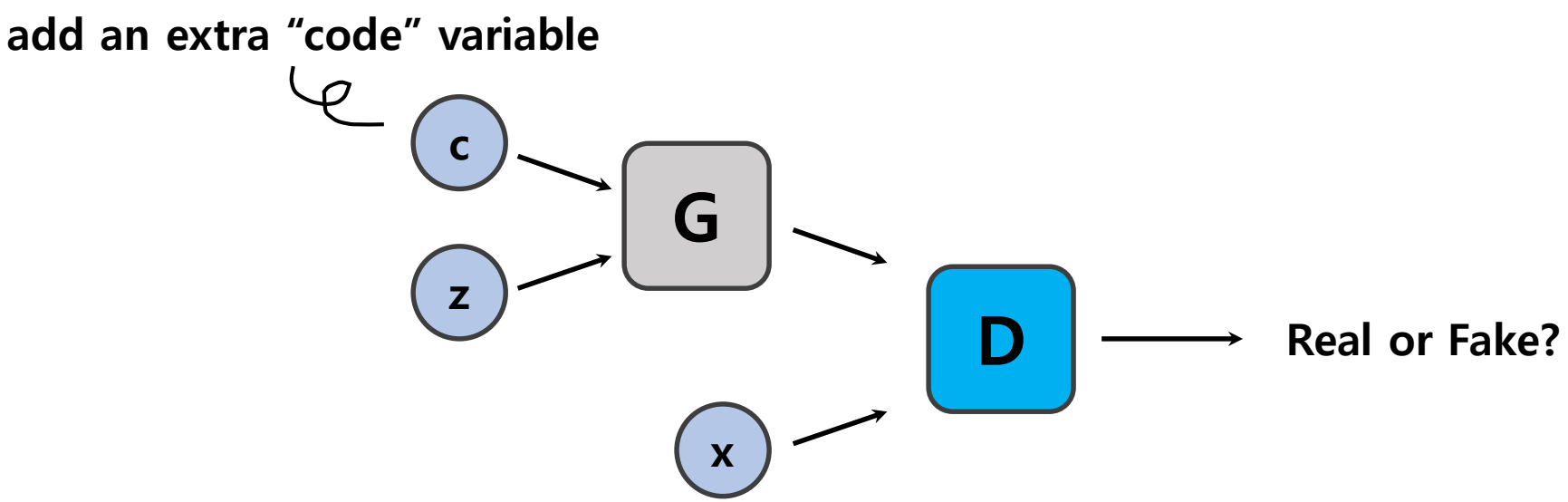
$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z \sim p_x(z)}[log(1 - D(G(z)))]$$



# SCHEMATIC OVERVIEW

**Diagram of infoGAN**

1. Gaussian noise z (source of incompressible noise)
2. latent code c (semantic feature of data distribution)

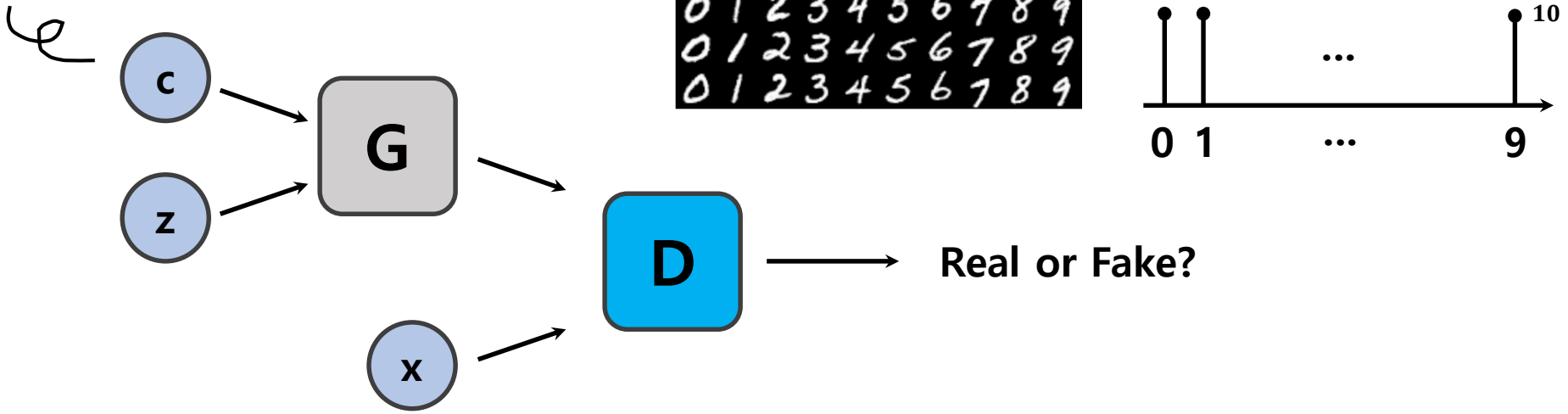


# SCHEMATIC OVERVIEW

## Diagram of infoGAN

1. Gaussian noise z (source of incompressible noise)
2. latent code c (semantic feature of data distribution)

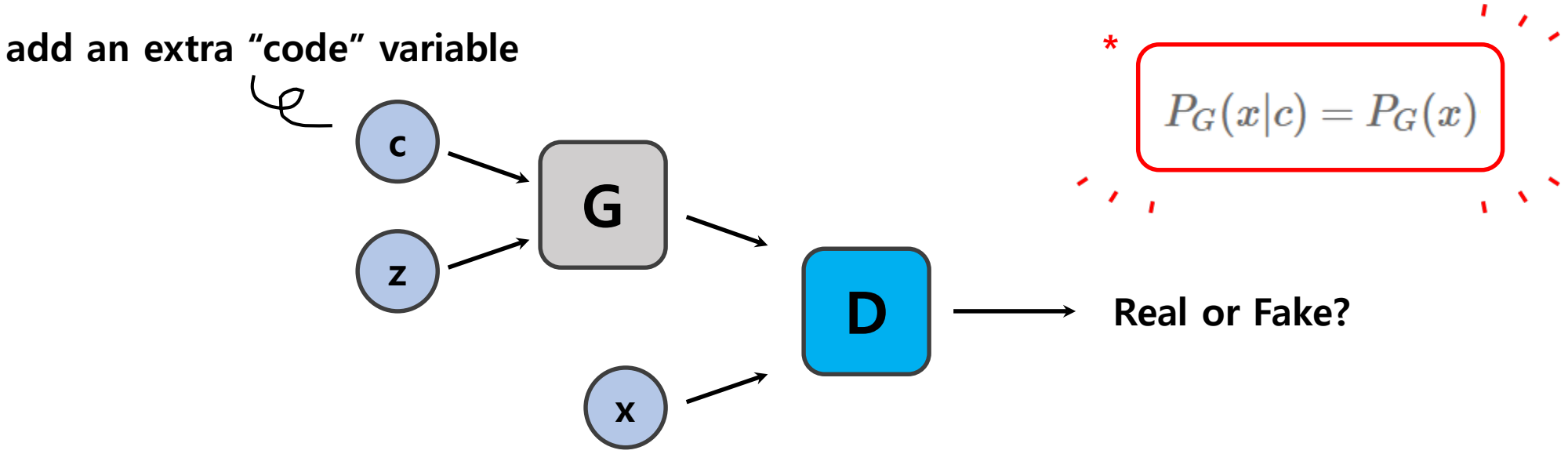
add an extra “code” variable



# SCHEMATIC OVERVIEW

**Diagram of infoGAN**

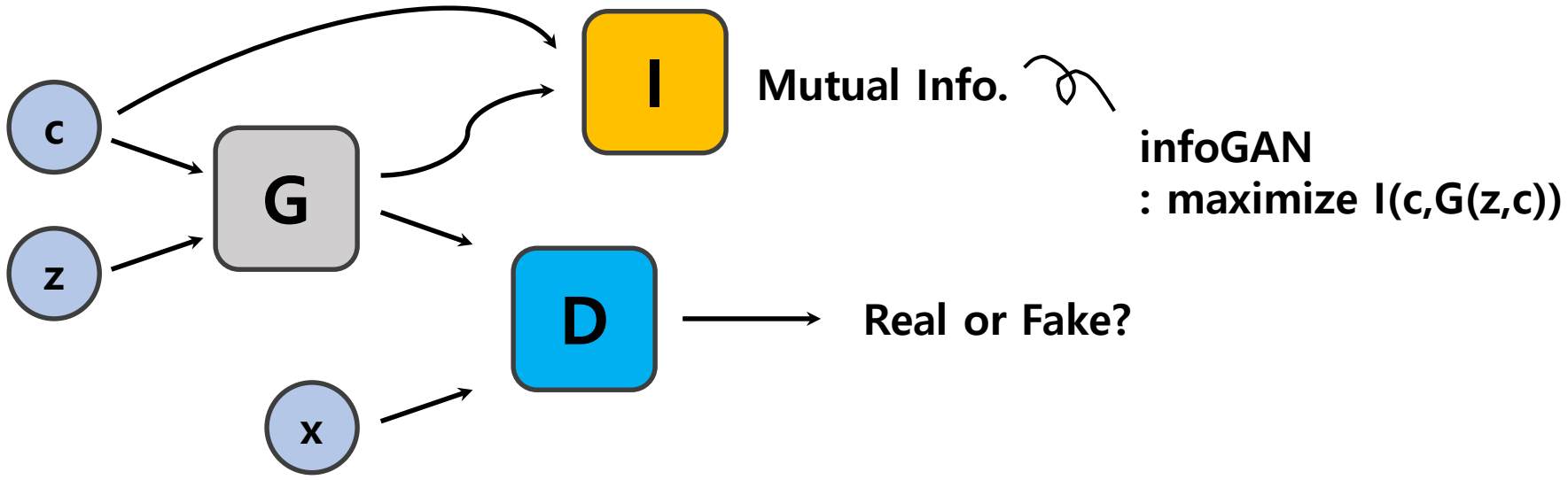
1. Gaussian noise z (source of incompressible noise)
2. latent code c (semantic feature of data distribution)



# SCHEMATIC OVERVIEW

## Diagram of infoGAN

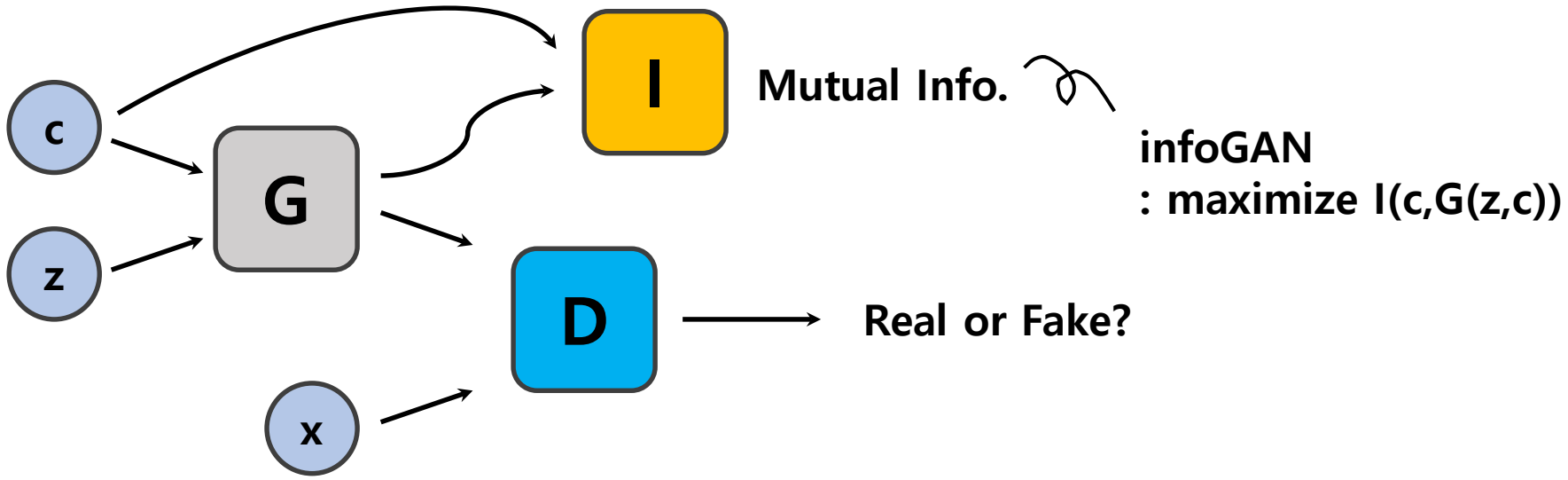
Impose an extra constraint to learn disentangled feature space



# SCHEMATIC OVERVIEW

## Diagram of infoGAN

Impose an extra constraint to learn disentangled feature space



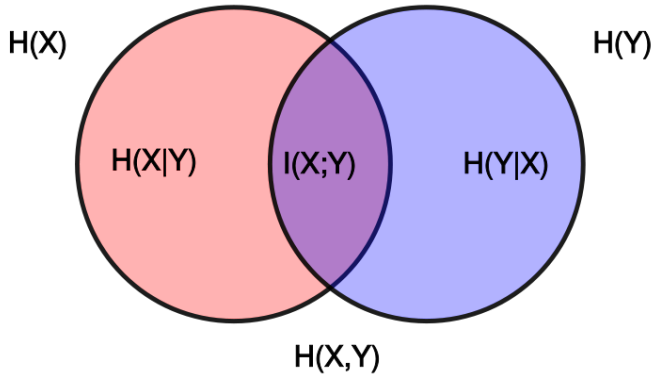

"The information in the latent code c should not be lost in the generation process."

# INFOGAN

**Changed Minimax problem:**

$$\min_G \max_D V_I(D,G) = V(D,G) - \lambda I(c; G(z,c))$$

**Mutual Information:**




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

$\therefore$ We need to minimize the entropy of $P_G(c|x)$ where $x \sim P_G(x)$.

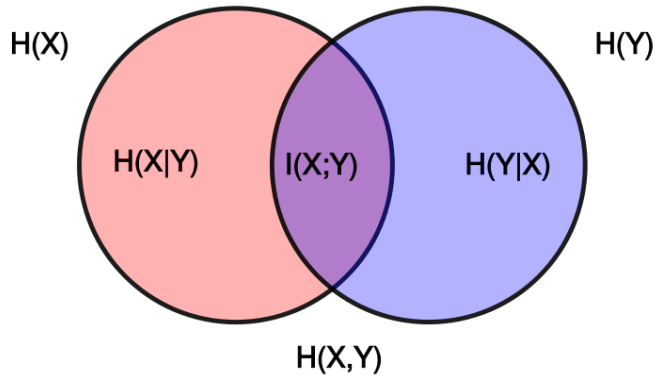
* Figure adopted from Wikipedia "Mutual Information"

# INFOGAN

**Changed Minimax problem:**

$$\min_G \max_D V_I(D,G) = V(D,G) - \lambda I(c; G(z,c))$$

**Mutual Information:**




**Uncertainty**

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

$\therefore$ We need to minimize the entropy of $P_G(c|x)$ where $x \sim P_G(x)$.

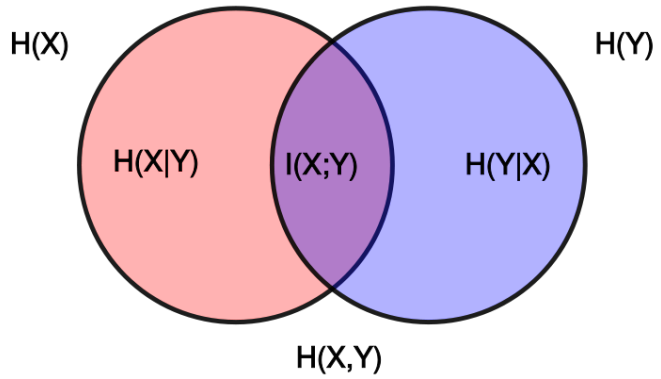
* Figure adopted from Wikipedia "Mutual Information"

# INFOGAN

**Changed Minimax problem:**

$$\min_G \max_D V_I(D,G) = V(D,G) - \lambda I(c; G(z,c))$$

**Mutual Information:**




**Uncertainty**

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

$\therefore$ We need to minimize the entropy of $P_G(c|x)$*

where $x \sim P_G(x)$.

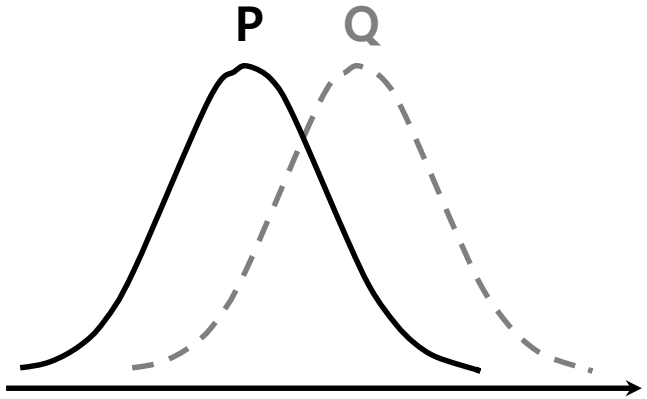
**intractable**

* Figure adopted from Wikipedia "Mutual Information"

# VARIATIONAL INFORMATION MAXIMIZATION

**Changed Minimax problem:**

$$\min_G \max_D V_I(D,G) = V(D,G) - \lambda I(c; G(z,c))$$




**Let's MAXIMIZE the LOWER BOUND which is tractable !**

$$\begin{aligned}
I(c; G(z,c)) &= H(c) - H(c|G(z,c)) \\
&= \mathbb{E}_{x\sim G(z,c)}[\mathbb{E}_{c'\sim P(c|x)}[logP(c'|x)]] + H(c) \\
&= \mathbb{E}_{x\sim G(z,c)}[\underbrace{D_{KL}(P(\cdot|x)||Q(\cdot|x))}_{\geq 0}] + \mathbb{E}_{c'\sim P(c|x)}[logQ(c'|x)]] + H(c) \\
&\geq \mathbb{E}_{x\sim G(z,c)}[\mathbb{E}_{c'\sim P(c|x)}[logQ(c'|x)]] + H(c) \\
&= L_I(G,Q).
\end{aligned}$$

# RESULTS

(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)

**MNIST dataset**

* Figure adopted from infoGAN paper (link)

# RESULTS

(a) Azimuth (pose)



(b) Elevation

(c) Lighting



(d) Wide or Narrow

**3D FACE dataset**

* Figure adopted from infoGAN paper (link)

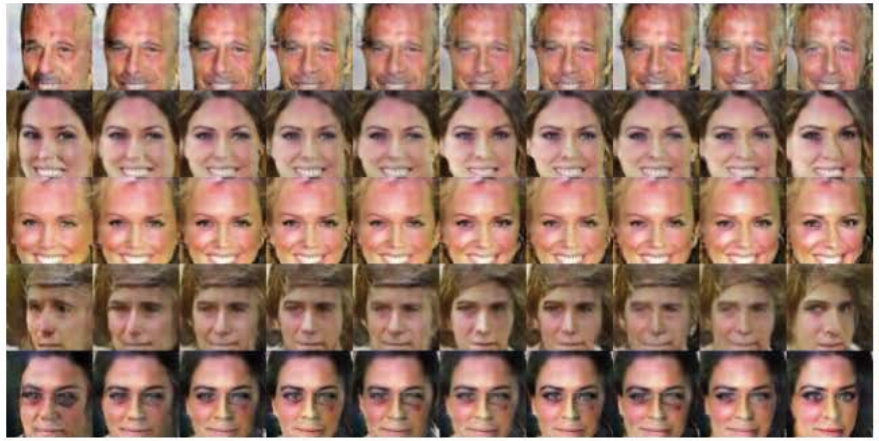
# RESULTS

(a) Rotation (b) Width

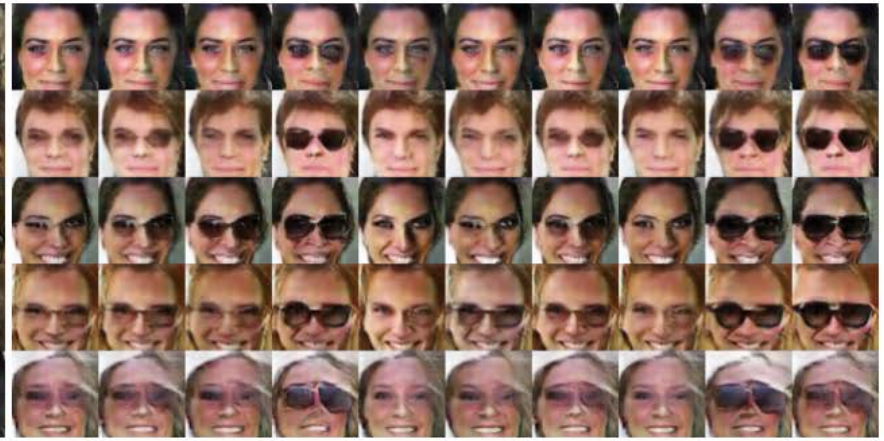
(a) Continuous variation: Lighting (b) Discrete variation: Plate Context

* Figure adopted from infoGAN paper (link)

# RESULTS



(a) Azimuth (pose)



(b) Presence or absence of glasses

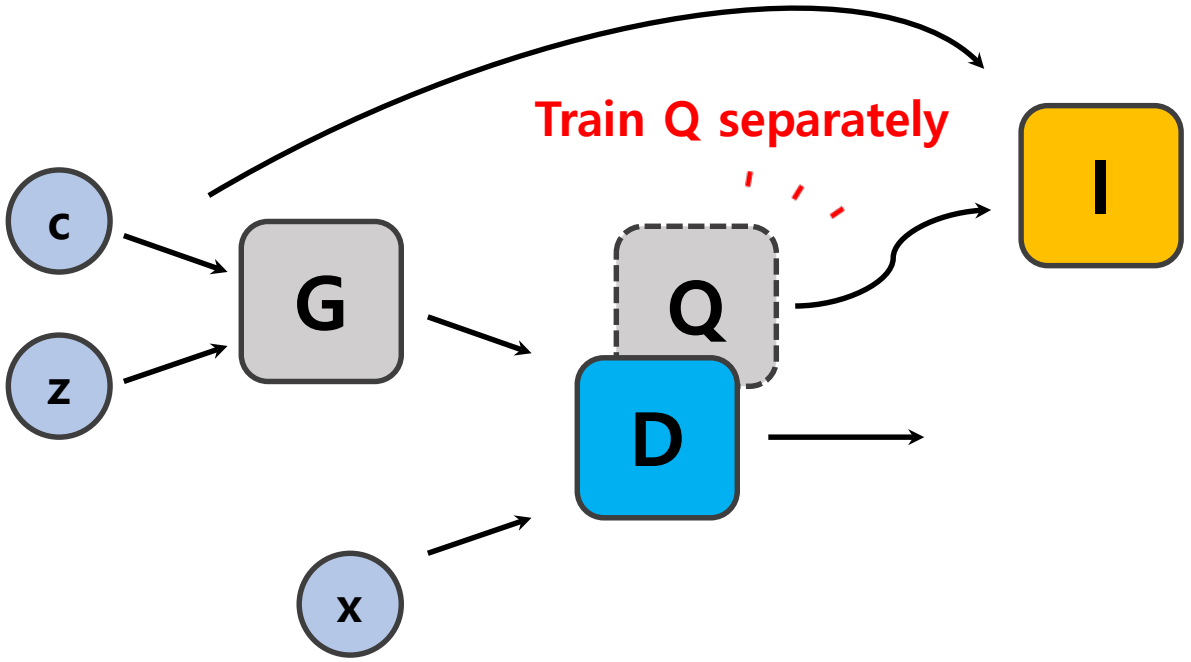
(c) Hair style

(d) Emotion

* Figure adopted from infoGAN paper (link)

# IMPLEMENTATION

Diagram of
infoGAN



# SUMMARY

1. Add an additional constraint to improve the performance
   - Mutual information
   - No adding on the computational cost
2. Learn better feature space
3. Unsupervised way to learn implicit features in the dataset
4. Variational method

# IMPLEMENTATION

```
X = tf.placeholder(tf.float32, shape=[None, 784])

D_W1 = tf.Variable(xavier_init([784, 128]))
D_b1 = tf.Variable(tf.zeros(shape=[128]))

D_W2 = tf.Variable(xavier_init([128, 1]))
D_b2 = tf.Variable(tf.zeros(shape=[1]))

theta_D = [D_W1, D_W2, D_b1, D_b2]


Z = tf.placeholder(tf.float32, shape=[None, 16])
c = tf.placeholder(tf.float32, shape=[None, 10])

G_W1 = tf.Variable(xavier_init([26, 256]))
G_b1 = tf.Variable(tf.zeros(shape=[256]))

G_W2 = tf.Variable(xavier_init([256, 784]))
G_b2 = tf.Variable(tf.zeros(shape=[784]))

theta_G = [G_W1, G_W2, G_b1, G_b2]


Q_W1 = tf.Variable(xavier_init([784, 128]))
Q_b1 = tf.Variable(tf.zeros(shape=[128]))

Q_W2 = tf.Variable(xavier_init([128, 10]))
Q_b2 = tf.Variable(tf.zeros(shape=[10]))

theta_Q = [Q_W1, Q_W2, Q_b1, Q_b2]
```

```
def sample_Z(m, n):
    return np.random.uniform(-1., 1., size=[m, n])


def sample_c(m):
    return np.random.multinomial(1, 10*[0.1], size=m)


def generator(z, c):
    inputs = tf.concat(axis=1, values=[z, c])
    G_h1 = tf.nn.relu(tf.matmul(inputs, G_W1) + G_b1)
    G_log_prob = tf.matmul(G_h1, G_W2) + G_b2
    G_prob = tf.nn.sigmoid(G_log_prob)

    return G_prob


def discriminator(x):
    D_h1 = tf.nn.relu(tf.matmul(x, D_W1) + D_b1)
    D_logit = tf.matmul(D_h1, D_W2) + D_b2
    D_prob = tf.nn.sigmoid(D_logit)

    return D_prob


def Q(x):
    Q_h1 = tf.nn.relu(tf.matmul(x, Q_W1) + Q_b1)
    Q_prob = tf.nn.softmax(tf.matmul(Q_h1, Q_W2) + Q_b2)

    return Q_prob
```

# IMPLEMENTATION

```
G_sample = generator(Z, c)
D_real = discriminator(X)
D_fake = discriminator(G_sample)
Q_c_given_x = Q(G_sample)

D_loss = -tf.reduce_mean(tf.log(D_real + 1e-8) + tf.log(1 - D_fake + 1e-8))
G_loss = -tf.reduce_mean(tf.log(D_fake + 1e-8))

cross_ent = tf.reduce_mean(-tf.reduce_sum(tf.log(Q_c_given_x + 1e-8) * c, 1))
ent = tf.reduce_mean(-tf.reduce_sum(tf.log(c + 1e-8) * c, 1))
Q_loss = cross_ent + ent

D_solver = tf.train.AdamOptimizer().minimize(D_loss, var_list=theta_D)
G_solver = tf.train.AdamOptimizer().minimize(G_loss, var_list=theta_G)
Q_solver = tf.train.AdamOptimizer().minimize(Q_loss, var_list=theta_G + theta_Q

mb_size = 32
Z_dim = 16

mnist = input_data.read_data_sets('../../MNIST_data', one_hot=True)
```

```
sess = tf.Session()
sess.run(tf.global_variables_initializer())

if not os.path.exists('out/'):
    os.makedirs('out/')

i = 0

for it in range(1000000):
    if it % 1000 == 0:
        Z_noise = sample_Z(16, Z_dim)

        idx = np.random.randint(0, 10)
        c_noise = np.zeros([16, 10])
        c_noise[range(16), idx] = 1

        samples = sess.run(G_sample,
                           feed_dict={Z: Z_noise, c: c_noise})

        fig = plot(samples)
        plt.savefig('out/{}.png'.format(str(i).zfill(3)), bbox_inches='tight')
        i += 1
        plt.close(fig)

    X_mb, _ = mnist.train.next_batch(mb_size)
    Z_noise = sample_Z(mb_size, Z_dim)
    c_noise = sample_c(mb_size)

    _, D_loss_curr = sess.run([D_solver, D_loss],
                              feed_dict={X: X_mb, Z: Z_noise, c: c_noise})

    _, G_loss_curr = sess.run([G_solver, G_loss],
                              feed_dict={Z: Z_noise, c: c_noise})

    sess.run([Q_solver], feed_dict={Z: Z_noise, c: c_noise})
```

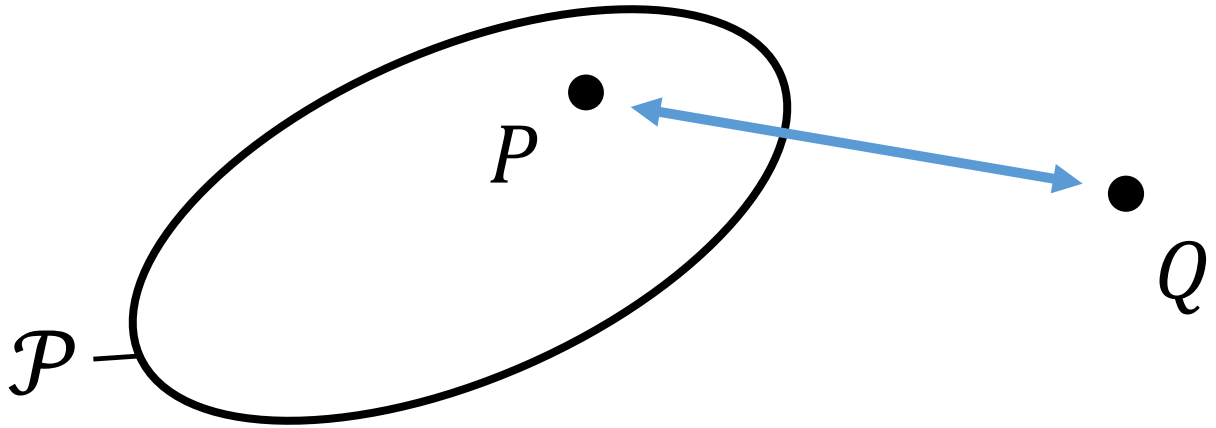
# *f*-GAN

## LET'S USE f-DIVERGENCE RATHER THAN FIXING A SINGLE ONE.

Here, I have **heavily reused** **the slides from S. Nowozin's (1st author) NIPS 2016 workshop** for GAN.
You can easily find the related information (slides) at:
http://www.nowozin.net/sebastian/blog/nips-2016-generative-adversarial-training-workshop-talk.html

# MOTIVATION

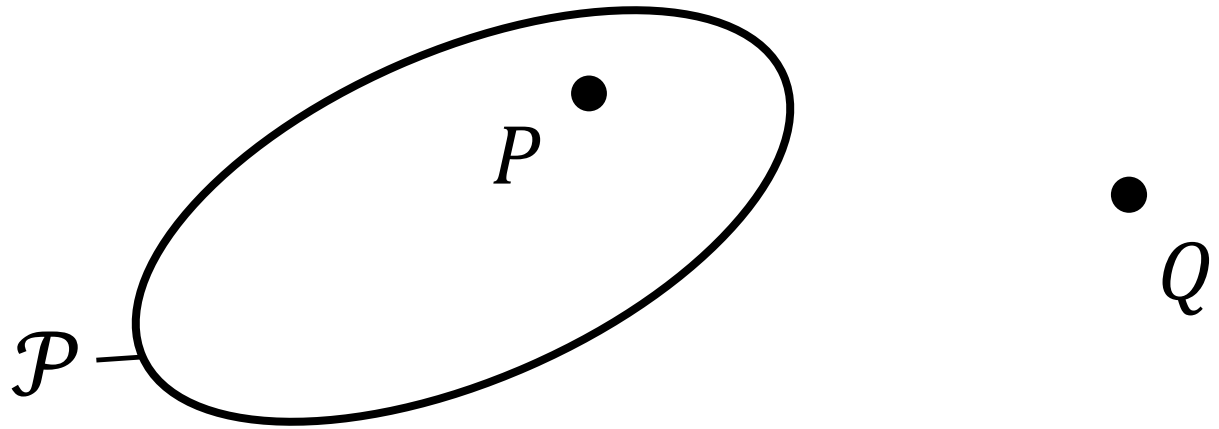
## LEARNING PROBABILISTIC MODELS




* Please note that in S. Nowozin's **slides**, Q represents the real distribution and P stands for the parametric model we set.

# MOTIVATION

## LEARNING PROBABILISTIC MODELS




Assumptions on $P$:

- tractable sampling
- tractable parameter gradient with respect to sample
- tractable likelihood function

* Please note that in S. Nowozin's **slides**, Q represents the real distribution and P stands for the parametric model we set.

# MOTIVATION

## Likelihood-free Model

[Goodfellow et al., 2014]

Random input

$z \sim \text{Uniform}_{100}$

Generator

$z \rightarrow \text{Lin}(100,1200) \rightarrow \text{ReLU}$
$\rightarrow \text{Lin}(1200,1200) \rightarrow \text{ReLU}$
$\rightarrow \text{Lin}(1200,784) \rightarrow \text{Sigmoid}$

Output

# SCHEMATIC OVERVIEW

## LEARNING PROBABILISTIC MODELS

Integral Probability Metrics
[Müller, 1997]
[Sriperumbudur et al., 2010]

$$\gamma_{\mathcal{F}}(P,Q) = \sup_{f\in\mathcal{F}} \left| \int f \mathrm{d}P - \int f \mathrm{d}Q \right|$$

- P: Expectation
- Q: Expectation
- Structure in $\mathcal{F}$
- Examples:
  - Energy statistic [Szekely, 1997]
  - Kernel MMD [Gretton et al., 2012], [Smola et al., 2007]
  - Wasserstein distance [Cuturi, 2013]
  - DISCO Nets [Bouchacourt et al., 2016]

Proper scoring rules
[Gneiting and Raftery, 2007]

$$S(P,Q) = \int S(P,x)\mathrm{d}Q(x)$$

- P: Distribution
- Q: Expectation
- Examples:
  - Log-likelihood [Fisher, 1922], [Good, 1952]
  - Quadratic score [Bernardo, 1979]

*f*-divergences
[Ali and Silvey, 1966]

$$D_f(P \parallel Q) = \int q(x) f\left(\frac{p(x)}{q(x)}\right) \mathrm{d}x$$

- P: Distribution
- Q: Distribution
- Examples:
  - Kullback-Leibler divergence [Kullback and Leibler, 1952]
  - Jensen-Shannon divergence
  - Total variation
  - Pearson $\chi^2$

# SCHEMATIC OVERVIEW

## LEARNING PROBABILISTIC MODELS

[Nguyen et al., 2010], [Reid and Williamson, 2011], [Goodfellow et al., 2014]

Variational representation of divergences

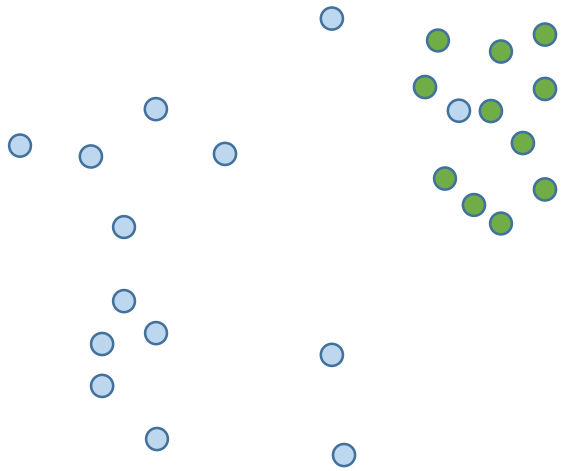
- $P$: Expectation
- $Q$: Expectation

- $P$: Distribution
- $Q$: Expectation

- $P$: Distribution
- $Q$: Distribution

# SCHEMATIC OVERVIEW

## TRAINING NEURAL SAMPLERS



Neural Sampler samples

Training samples

**How do we measure the distance only based on empirical samples from $P_\theta(x)$ and $\mathrm{Q}(x)$?**

* Please note that in S. Nowozin's **slides**, Q represents the real distribution and P stands for the parametric model we set.

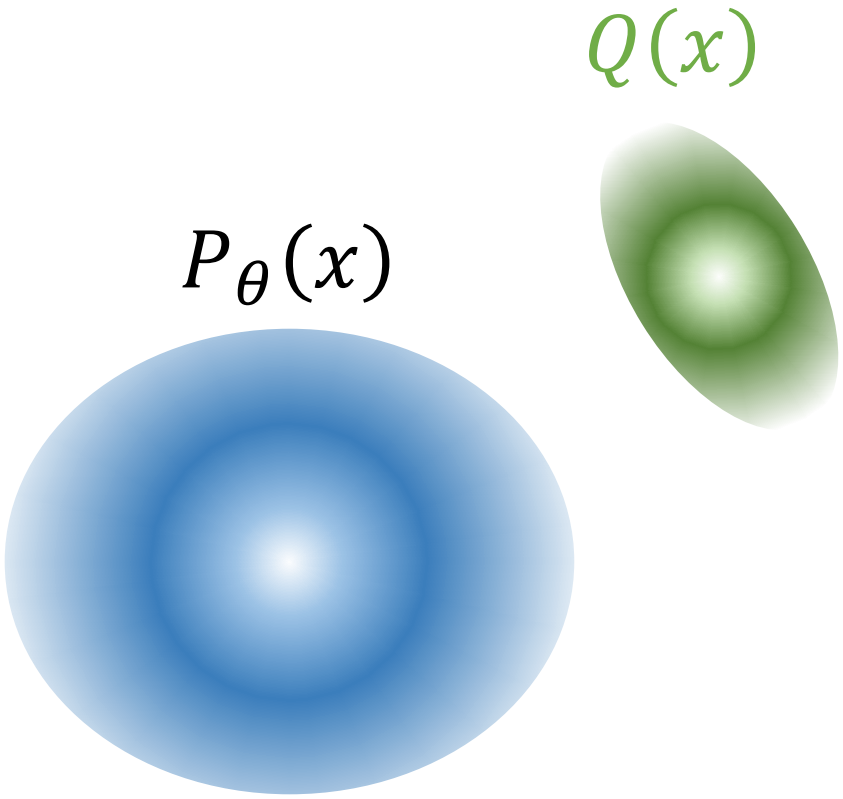
# SCHEMATIC OVERVIEW

**"Show that the GAN approach is a special case of an existing more general variational divergence estimation approach."**

**Let's generalize the GAN objective to arbitrary $f$-divergences!**

# SCHEMATIC OVERVIEW

## TRAINING NEURAL SAMPLERS




Neural Sampler distribution

True distribution

**We can minimize some distance (divergence) between the distributions if we had $P_\theta(x)$ and $Q(x)$**

* Please note that in S. Nowozin's **slides**, Q represents the real distribution and P stands for the parametric model we set.

# $f$-DIVERGENCE

[Ali and Silvey, 1966]

- Divergence between two distributions

$$D_f(Q \parallel P) = \int_x p(x)\, f\left(\frac{q(x)}{p(x)}\right) dx$$

- $f$ : generator function, convex, $f(1) = 0$

# $f$-DIVERGENCE

## TRY WHAT YOU WANT! (골라먹는 재미)

| Name | $D_f(P\|Q)$ | Generator $f(u)$ | $T^*(x)$ |
|---|---|---|---|
| Total variation | $\frac{1}{2}\int \lvert p(x)-q(x)\rvert \,\mathrm{d}x$ | $\frac{1}{2}\lvert u-1\rvert$ | $\frac{1}{2}\mathrm{sign}(\frac{p(x)}{q(x)}-1)$ |
| 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 Kullback-Leibler | $\int q(x)\log\frac{q(x)}{p(x)}\,\mathrm{d}x$ | $-\log u$ | $-\frac{q(x)}{p(x)}$ |
| Pearson $\chi^2$ | $\int \frac{(q(x)-p(x))^2}{p(x)}\,\mathrm{d}x$ | $(u-1)^2$ | $2(\frac{p(x)}{q(x)}-1)$ |
| Neyman $\chi^2$ | $\int \frac{(p(x)-q(x))^2}{q(x)}\,\mathrm{d}x$ | $\frac{(1-u)^2}{u}$ | $1-\left[\frac{q(x)}{p(x)}\right]^2$ |
| Squared Hellinger | $\int \left(\sqrt{p(x)}-\sqrt{q(x)}\right)^2\,\mathrm{d}x$ | $(\sqrt{u}-1)^2$ | $(\sqrt{\frac{p(x)}{q(x)}}-1)\cdot\sqrt{\frac{q(x)}{p(x)}}$ |
| Jeffrey | $\int (p(x)-q(x))\log\left(\frac{p(x)}{q(x)}\right)\,\mathrm{d}x$ | $(u-1)\log u$ | $1+\log\frac{p(x)}{q(x)}-\frac{q(x)}{p(x)}$ |
| Jensen-Shannon | $\frac{1}{2}\int p(x)\log\frac{2p(x)}{p(x)+q(x)}+q(x)\log\frac{2q(x)}{p(x)+q(x)}\,\mathrm{d}x$ | $-(u+1)\log\frac{1+u}{2}+u\log u$ | $\log\frac{2p(x)}{p(x)+q(x)}$ |
| Jensen-Shannon-weighted | $\int p(x)\pi\log\frac{p(x)}{\pi p(x)+(1-\pi)q(x)}+(1-\pi)q(x)\log\frac{q(x)}{\pi p(x)+(1-\pi)q(x)}\,\mathrm{d}x$ | $\pi u\log u-(1-\pi+\pi u)\log(1-\pi+\pi u)$ | $\pi\log\frac{p(x)}{(1-\pi)q(x)+\pi p(x)}$ |
| GAN | $\int p(x)\log\frac{2p(x)}{p(x)+q(x)}+q(x)\log\frac{2q(x)}{p(x)+q(x)}\,\mathrm{d}x-\log(4)$ | $u\log u-(u+1)\log(u+1)$ | $\log\frac{p(x)}{p(x)+q(x)}$ |
| $\alpha$-divergence ($\alpha\notin\{0,1\}$) | $\frac{1}{\alpha(\alpha-1)}\int\left(p(x)\left[\left(\frac{q(x)}{p(x)}\right)^\alpha-1\right]-\alpha(q(x)-p(x))\right)\mathrm{d}x$ | $\frac{1}{\alpha(\alpha-1)}\left(u^\alpha-1-\alpha(u-1)\right)$ | $\frac{1}{\alpha-1}\left[\left[\frac{p(x)}{q(x)}\right]^{\alpha-1}-1\right]$ |

**Table 5:** List of $f$-divergences $D_f(P\|Q)$ together with generator functions and the optimal variational functions.

# $f$-DIVERGENCE

**Estimating $f$-divergences from samples**

**[Nguyen, Wainwright, Jordan, 2010]**

- Divergence between two distributions

$$D_f(Q \parallel P) = \int_{\mathcal{X}} p(x)\, f\left(\frac{q(x)}{p(x)}\right) \mathrm{d}x$$

- Every convex function $f$ has a *Fenchel conjugate* $f^*$ so that

$$f(u) = \sup_{t \in \mathrm{dom}_{f^*}} \{tu - f^*(t)\}$$

"Any convex $f$ can be represented as point-wise max of *linear* functions"

# $f$-DIVERGENCE

## Estimating $f$-divergences from samples (cont)

$$
\begin{aligned}
D_f(Q \parallel P) &= \int_{\mathcal{X}} p(x)\, f\left(\frac{q(x)}{p(x)}\right) \mathrm{d}x \\
&= \int_{\mathcal{X}} p(x) \sup_{t \in \mathrm{dom}_{f^*}} \left\{ t\frac{q(x)}{p(x)} - f^*(t) \right\} \mathrm{d}x \\
&\geq \sup_{T \in \mathcal{T}} \left( \int_{\mathcal{X}} q(x)\, T(x)\, \mathrm{d}x - \int_{\mathcal{X}} p(x)\, f^*\big(T(x)\big)\, \mathrm{d}x \right) \\
&= \sup_{T \in \mathcal{T}} \Big( \underbrace{\mathbb{E}_{x \sim Q}[T(x)]}_{} - \underbrace{\mathbb{E}_{x \sim P}[f^*(T(x))]}_{} \Big)
\end{aligned}
$$

Approximate using: samples from $Q$ samples from $P$

* Please note that in S. Nowozin's **slides**, Q represents the real distribution and P stands for the parametric model we set.

# $f$-GAN

## $f$-GAN and GAN objectives

- GAN

$$\min_{\theta}\max_{\omega}\mathbb{E}_{x\sim Q}[\log D_{\omega}(x)]+\mathbb{E}_{x\sim P_{\theta}}[\log(1-D_{\omega}(x))]$$

- $f$-GAN

$$\min_{\theta}\max_{\omega}\left(\mathbb{E}_{x\sim Q}[T_{\omega}(x)]-\mathbb{E}_{x\sim P_{\theta}}[f^{*}(T_{\omega}(x))]\right)$$

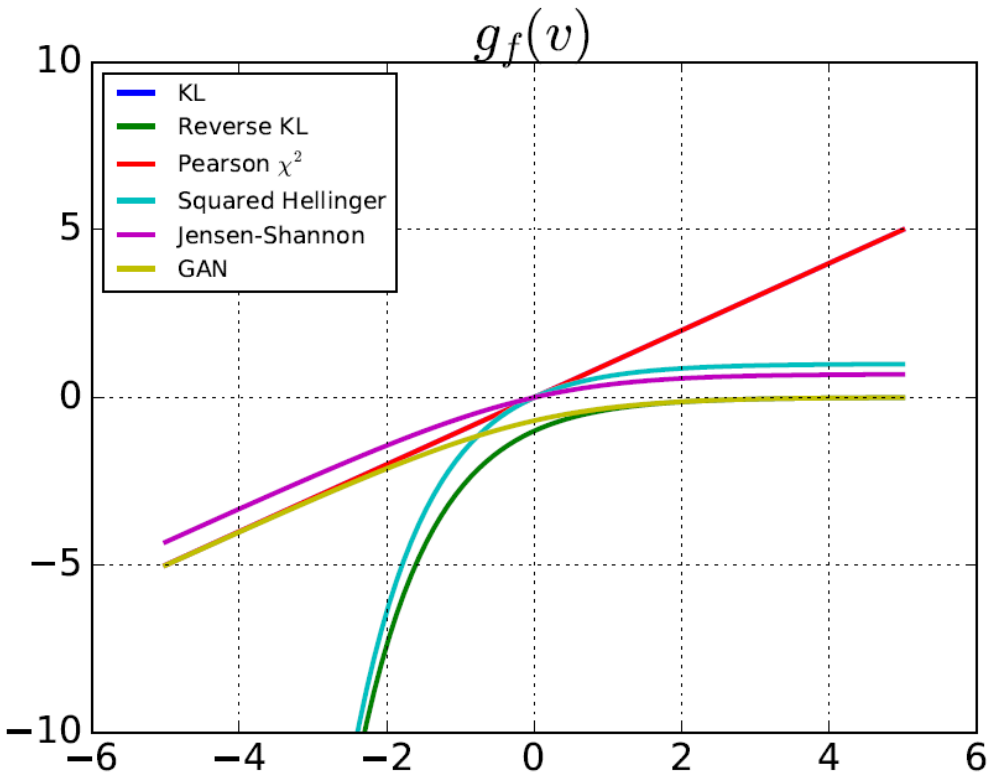
  - GAN discriminator-variational function correspondence: $\log D_{\omega}(x)=T_{\omega}(x)$
  - GAN minimizes the Jensen-Shannon divergence (which was also pointed out in Goodfellow et al., 2014)

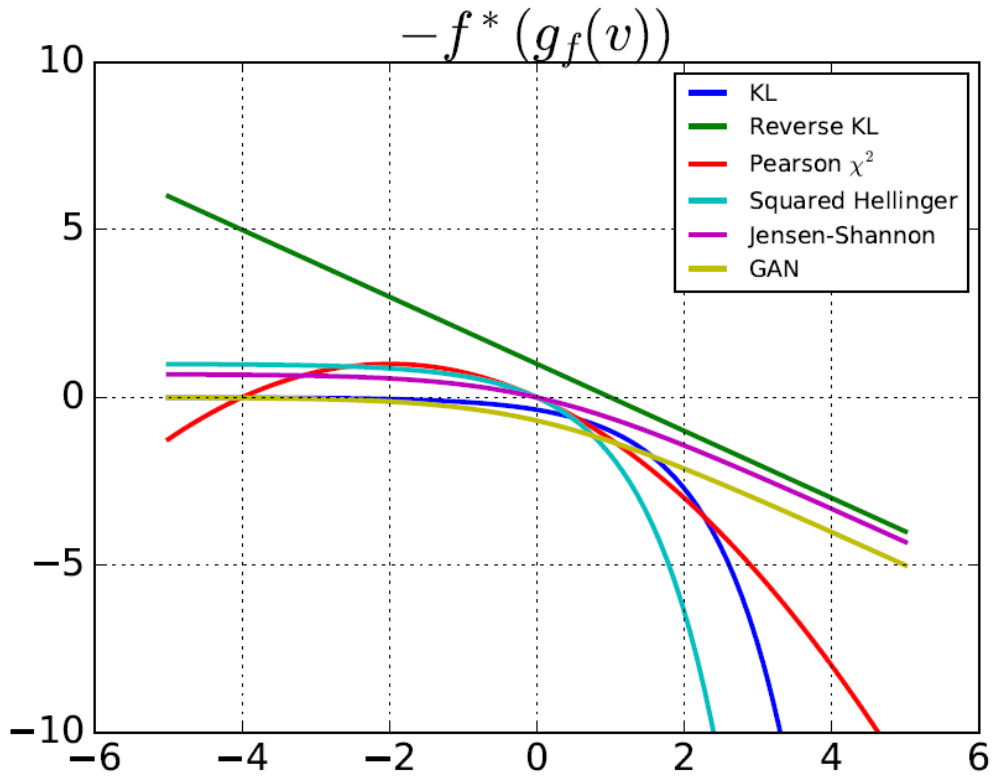
* Please note that in S. Nowozin's **slides**, Q represents the real distribution and P stands for the parametric model we set.

# $f$-GAN

## Comparison of the objectives

$$\min_{\theta}\max_{\omega}\left(\mathbb{E}_{x\sim Q}\left[g_f\left(V_\omega\ (x)\right)\right]+\mathbb{E}_{x\sim P_\theta}\left[-f^*\left(g_f\left(V_\omega(x)\right)\right)\right]\right)$$




* Please note that in S. Nowozin's **slides**, Q represents the real distribution and P stands for the parametric model we set.

# THEORETICAL RESULTS

**Algorithm: Double-Loop versus Single-Step**

- Double-loop algorithm [Goodfellow et al., 2014]
  - Algorithm:
    - Inner loop: tighten divergence lower bound
    - Outer loop: minimize generator loss
  - In practice the inner loop is run only for one step (two backprops)
  - Missing justification for this practice
- Single-step algorithm (proposed)
  - Algorithm: simultaneously take (one backprop)
    - a positive gradient step w.r.t. variational function $T_\omega(x)$
    - a negative gradient step w.r.t. generator function $P_\theta(x)$
  - Does this converge?

# f-GAN

## GENERAL ALGORITHM

dom$_{f^*}$ of the conjugate functions $f^*$. To this end, we assume that variational function $T_\omega$ is represented in the form $T_\omega(x) = g_f(V_\omega(x))$ and rewrite the saddle objective (6) as follows:

$$F(\theta, \omega) = \mathbb{E}_{x\sim P}\left[g_f(V_\omega(x))\right] + \mathbb{E}_{x\sim Q_\theta}\left[-f^*(g_f(V_\omega(x)))\right], \tag{7}$$

where $V_\omega : \mathcal{X} \to \mathbb{R}$ without any range constraints on the output, and $g_f : \mathbb{R} \to$ dom$_{f^*}$ is an *output activation function* specific to the $f$-divergence used. In Table 6 we propose suitable output activation

**Algorithm 1** Single-Step Gradient Method

1: **function** SINGLESTEPGRADIENTITERATION($P, \theta^t, \omega^t, B, \eta$)
2: Sample $X_P = \{x_1, \ldots, x_B\}$ and $X_Q = \{x'_1, \ldots, x'_B\}$, from $P$ and $Q_{\theta^t}$, respectively.
3: Update: $\omega^{t+1} = \omega^t + \eta \nabla_\omega F(\theta^t, \omega^t)$.
4: Update: $\theta^{t+1} = \theta^t - \eta \nabla_\theta F(\theta^t, \omega^t)$.
5: **end function**

* Please note that in S. Nowozin's **paper**, P represents the real distribution and $Q_\theta$ stands for the parametric model we set.

# THEORETICAL RESULTS

## Local convergence of the algorithm 1

**Analysis.** Here we show that Algorithm 1 geometrically converges to a saddle point $(\theta^*, \omega^*)$ if there is a neighborhood around the saddle point in which $F$ is strongly convex in $\theta$ and strongly concave in $\omega$. These conditions are similar to the assumptions made in [10] and can be formalized as follows:

$$\nabla_\theta F(\theta^*, \omega^*) = 0, \quad \nabla_\omega F(\theta^*, \omega^*) = 0, \quad \nabla^2_\theta F(\theta, \omega) \succeq \delta I, \quad \nabla^2_\omega F(\theta, \omega) \preceq -\delta I. \tag{9}$$

For convenience, let's define $\pi^t = (\theta^t, \omega^t)$. Now the convergence of Algorithm 1 can be stated as follows (the proof is given in the supplementary material):

**Theorem 1.** *Suppose that there is a saddle point $\pi^* = (\theta^*, \omega^*)$ with a neighborhood that satisfies conditions (9). Moreover, we define $J(\pi) = \frac{1}{2}\|\nabla F(\pi)\|_2^2$ and assume that in the above neighborhood, $F$ is sufficiently smooth so that there is a constant $L > 0$ such that $\|\nabla J(\pi') - \nabla J(\pi)\|_2 \leq L\|\pi' - \pi\|_2$ for any $\pi, \pi'$ in the neighborhood of $\pi^*$. Then using the step-size $\eta = \delta/L$ in Algorithm 1, we have*

$$J(\pi^t) \leq \left(1 - \frac{\delta^2}{2L}\right)^t J(\pi^0)$$

*That is, the squared norm of the gradient $\nabla F(\pi)$ decreases geometrically.*

# THEORETICAL RESULTS

**Local convergence of the algorithm 1**

- Assumptions
  - $F$ is locally (strongly) convex with respect to $\theta$
  - $F$ is (strongly) concave with respect to $\omega$
- Local convergence:

$$\nabla^2 F = \begin{bmatrix} \nabla_\theta^2 F & \nabla_\theta \nabla_\omega F \\ \nabla_\omega \nabla_\theta F & \nabla_\omega^2 F \end{bmatrix}$$

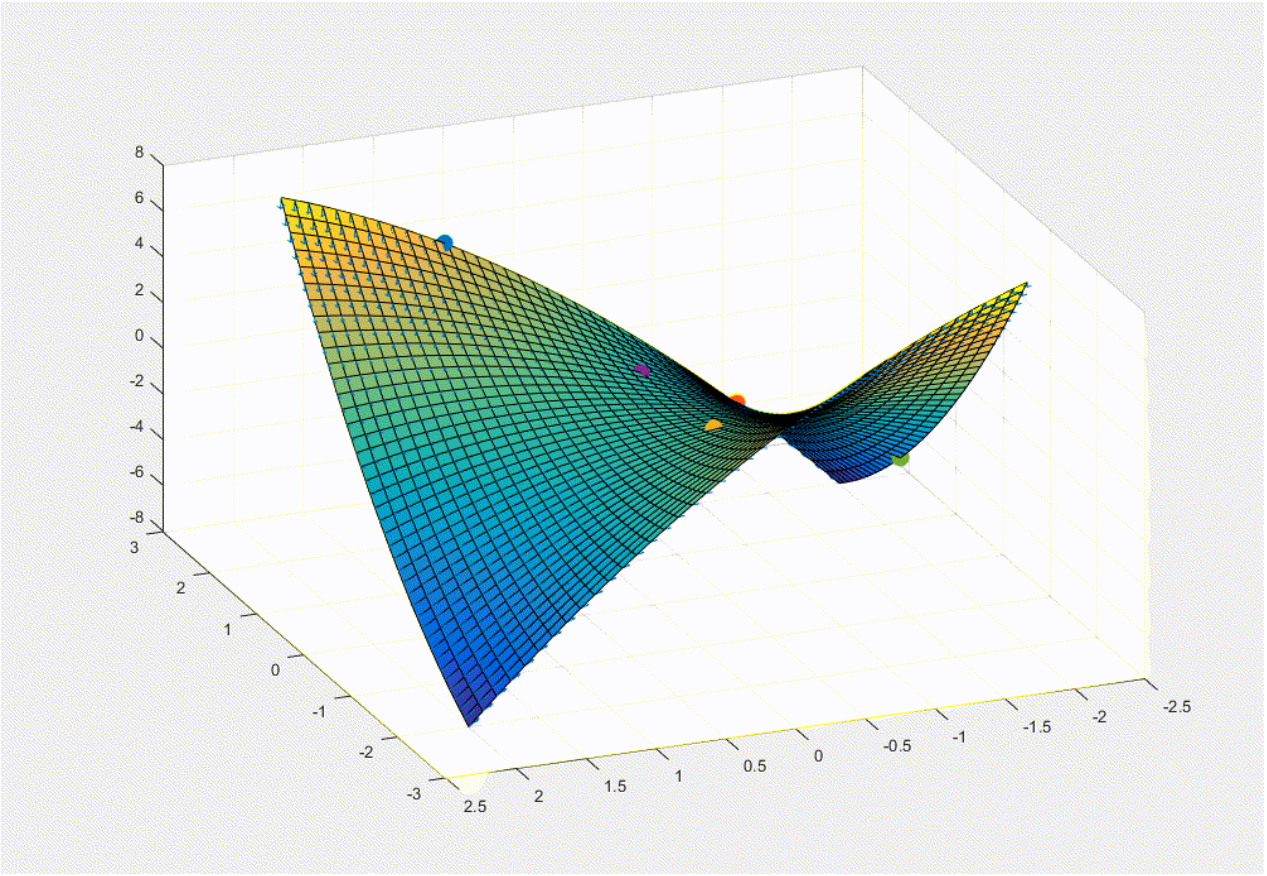
Define $J(\theta, \omega) = \frac{1}{2}\|\nabla_\theta F\|^2 + \frac{1}{2}\|\nabla_\omega F\|^2$, then $\nabla_\theta^2 F \succ 0, \quad \nabla_\omega^2 F \prec 0$

$$J(\theta^t, \omega^t) \leq \left(1 - \frac{\delta^2}{L}\right)^t J(\theta^0, \omega^0)$$

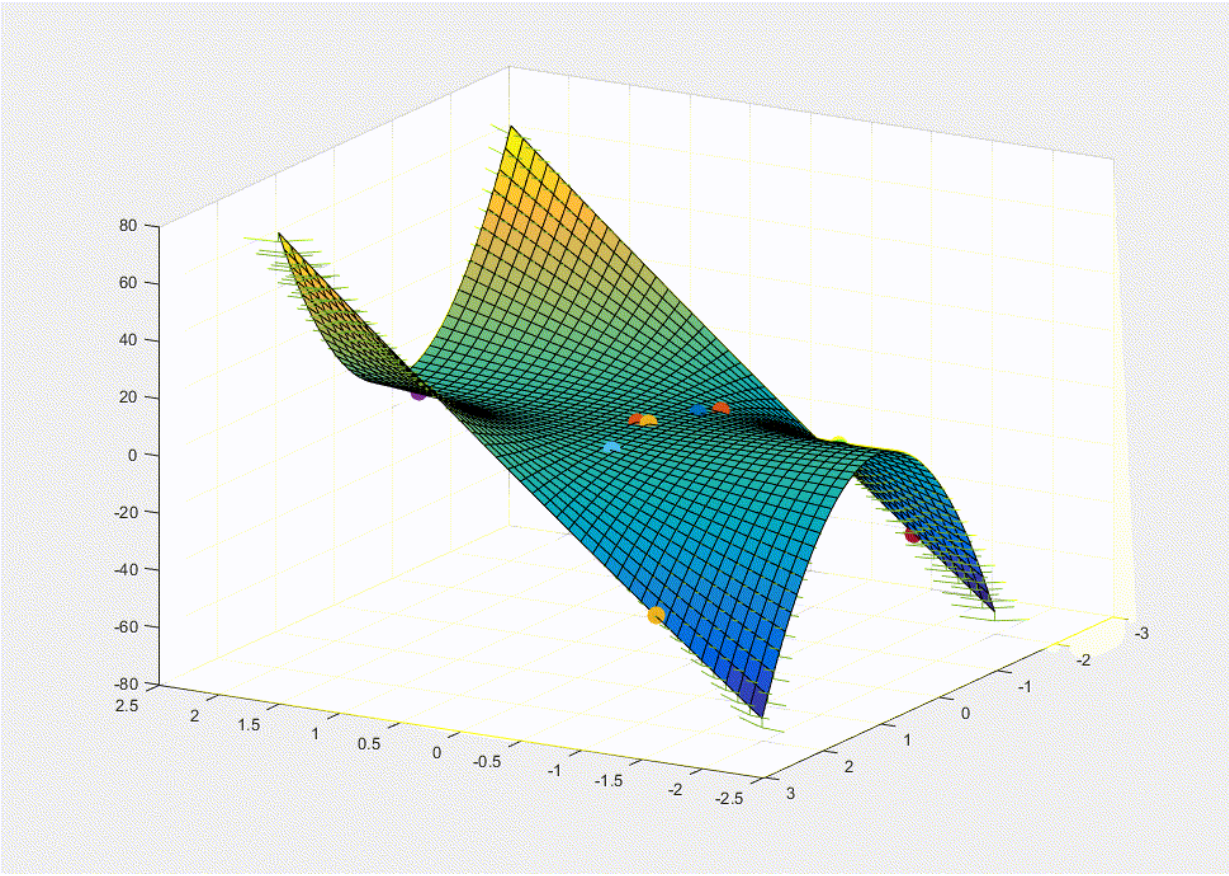
Geometric rate of convergence!

$\delta$: strong convexity parameter, $L$: smoothness parameter

# VIDEOS




$$V(x,y) = xy + \frac{\delta}{2}(x^2 - y^2)$$




$$V(x,y) = xy^2 + \frac{\delta}{2}(x^2 - y^2)$$

# RESULTS

## Synthetic 1D Univariate

Approximate a mixture of Gaussians by a Gaussian to
- Validate the approach
- Demonstrate the properties of different divergences [Minka, 2005]

Compare the exact optimization of the divergence with the GAN approach

| | KL | KL-rev | JS | Jeffrey | Pearson |
|---|---|---|---|---|---|
| $D_f(P\|\|Q_{\theta^*})$ | 0.2831 | 0.2480 | 0.1280 | 0.5705 | 0.6457 |
| $F(\hat{\omega}, \hat{\theta})$ | 0.2801 | 0.2415 | 0.1226 | 0.5151 | 0.6379 |
| $\mu^*$ | 1.0100 | 1.5782 | 1.3070 | 1.3218 | 0.5737 |
| $\hat{\mu}$ | 1.0335 | 1.5624 | 1.2854 | 1.2295 | 0.6157 |
| $\sigma^*$ | 1.8308 | 1.6319 | 1.7542 | 1.7034 | 1.9274 |
| $\hat{\sigma}$ | 1.8236 | 1.6403 | 1.7659 | 1.8087 | 1.9031 |

| train \ test | KL | KL-rev | JS | Jeffrey | Pearson |
|---|---|---|---|---|---|
| KL | **0.2808** | 0.3423 | 0.1314 | 0.5447 | 0.7345 |
| KL-rev | 0.3518 | **0.2414** | 0.1228 | 0.5794 | 1.3974 |
| JS | 0.2871 | 0.2760 | **0.1210** | 0.5260 | 0.92160 |
| Jeffrey | 0.2869 | 0.2975 | 0.1247 | **0.5236** | 0.8849 |
| Pearson | 0.2970 | 0.5466 | 0.1665 | 0.7085 | **0.648** |

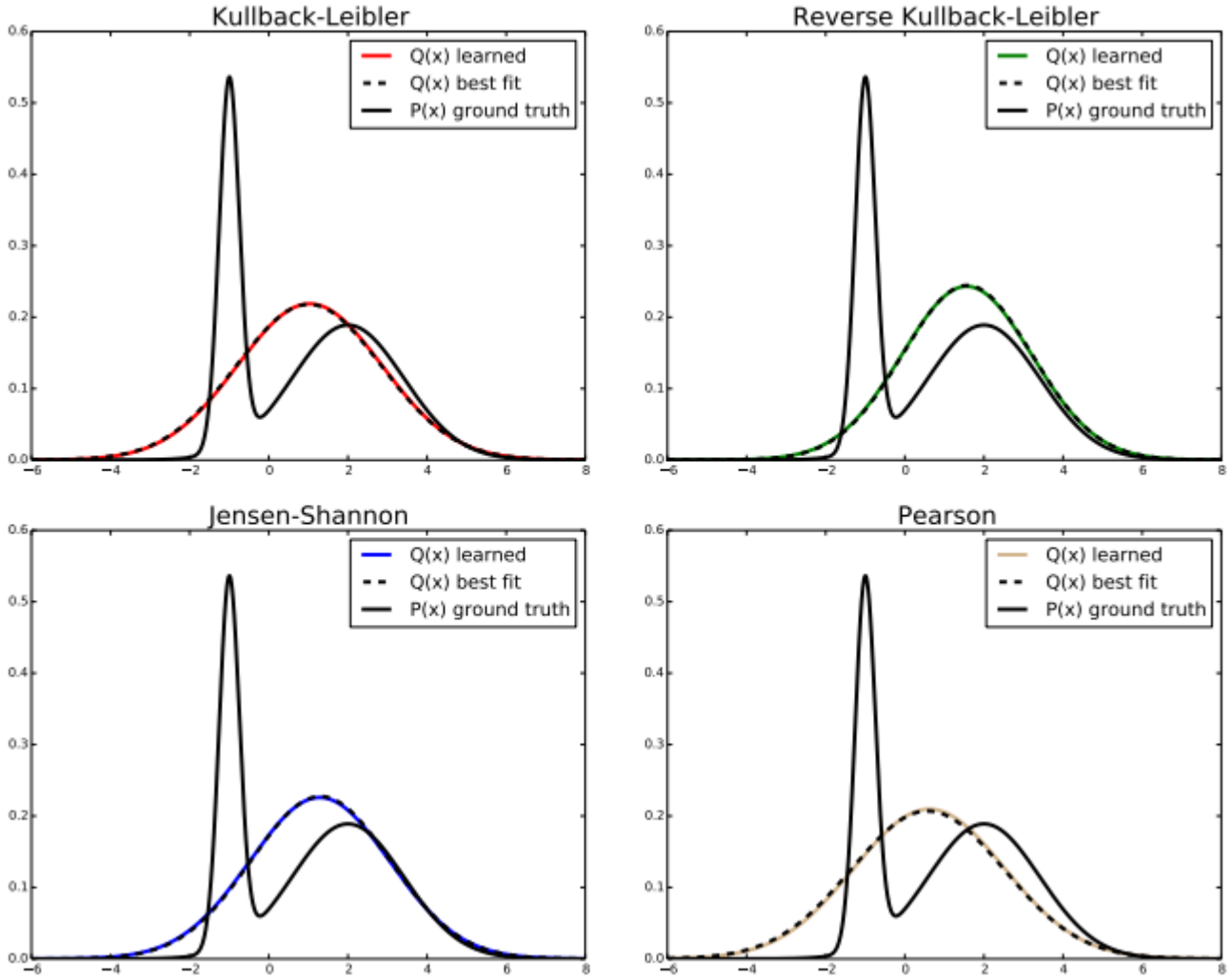
**Table 3:** Gaussian approximation of a mixture of Gaussians. Left: optimal objectives, and the learned mean and the standard deviation: $\hat{\theta} = (\hat{\mu}, \hat{\sigma})$ (learned) and $\theta^* = (\mu^*, \sigma^*)$ (best fit). Right: objective values to the true distribution for each trained model. For each divergence, the lowest objective function value is achieved by the model that was trained for this divergence.

* Please note that in S. Nowozin's **paper**, P represents the real distribution and $Q_\theta$ stands for the parametric model we set.

# RESULTS

## Synthetic 1D Univariate

* Figure adopted from f-GAN paper (link)




* Please note that in S. Nowozin's **paper**, P represents the real distribution and $Q_\theta$ stands for the parametric model we set.

# RESULTS

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

**Figure 3:** Samples from three different divergences.

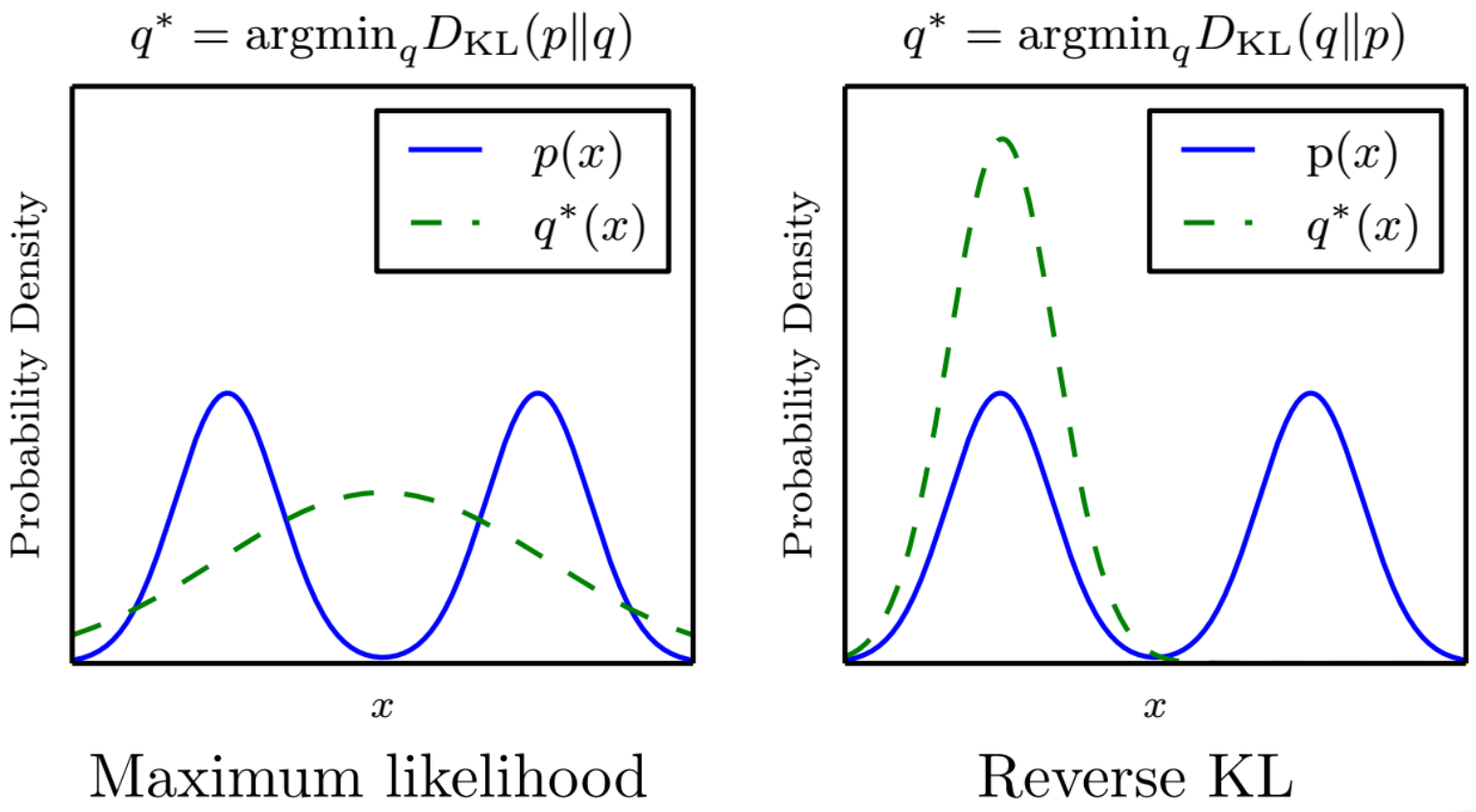
* Figure adopted from f-GAN paper (link)

# SUMMARY

- Generalize GAN objective to arbitrary $f$-divergences
- Simplify GAN algorithm + local convergence proof
- Demonstrate different divergences

# ETC.

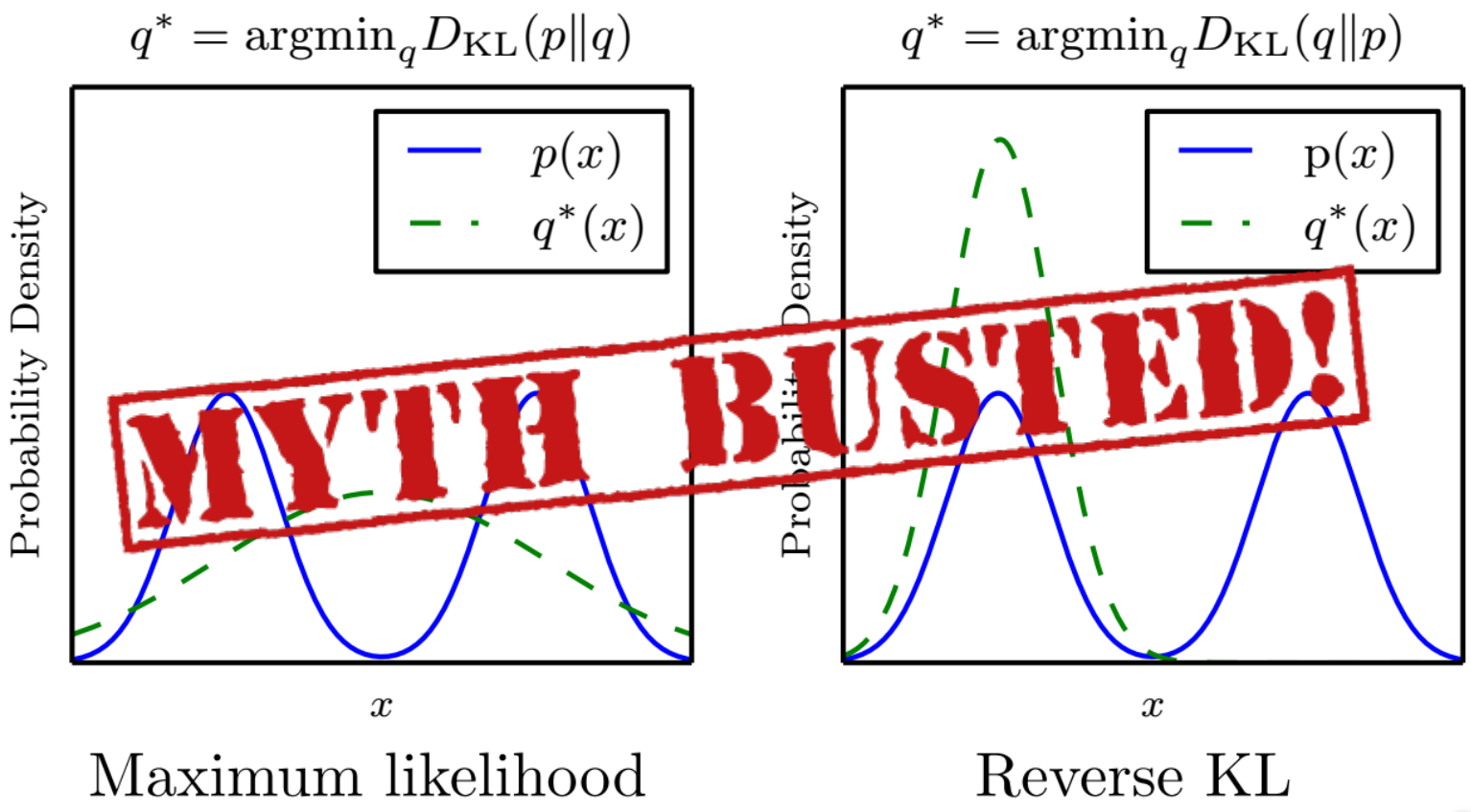
## WHY GAN GENERATES SHARPER IMAGES?




* Figure adopted from NIPS 2016 Tutorial GAN, Ian Goodfellow 2016

# ETC.

## WHY GAN GENERATES SHARPER IMAGES?

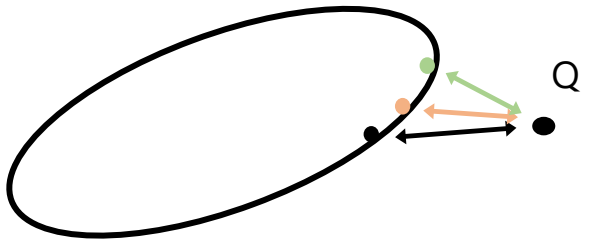



* Figure adopted from NIPS 2016 Tutorial GAN, Ian Goodfellow 2016

# ETC.

## DOES THE DIVERGENCE MATTER?

- LSUN experiment: No (visually)
- Empirical contradiction to intuition from [Theis et al., 2015], [Huszar, 2015]
- Why?
  - Intuition: strong inductive bias of model class



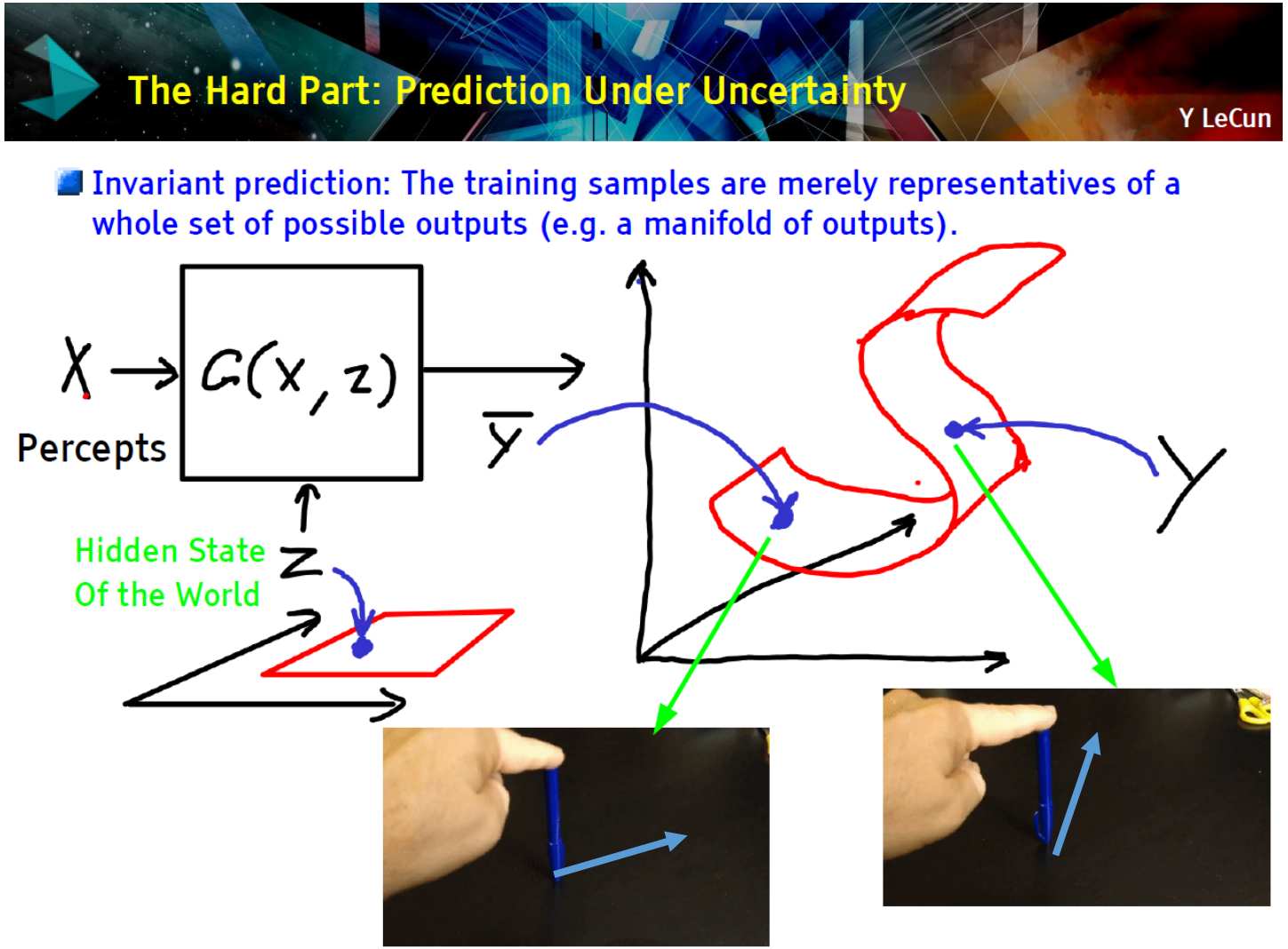
# EBGAN

LET'S USE ENERGY-BASED MODEL FOR THE DISCRIMINATOR

# MOTIVATION

**We want to learn**

**the data manifold!**



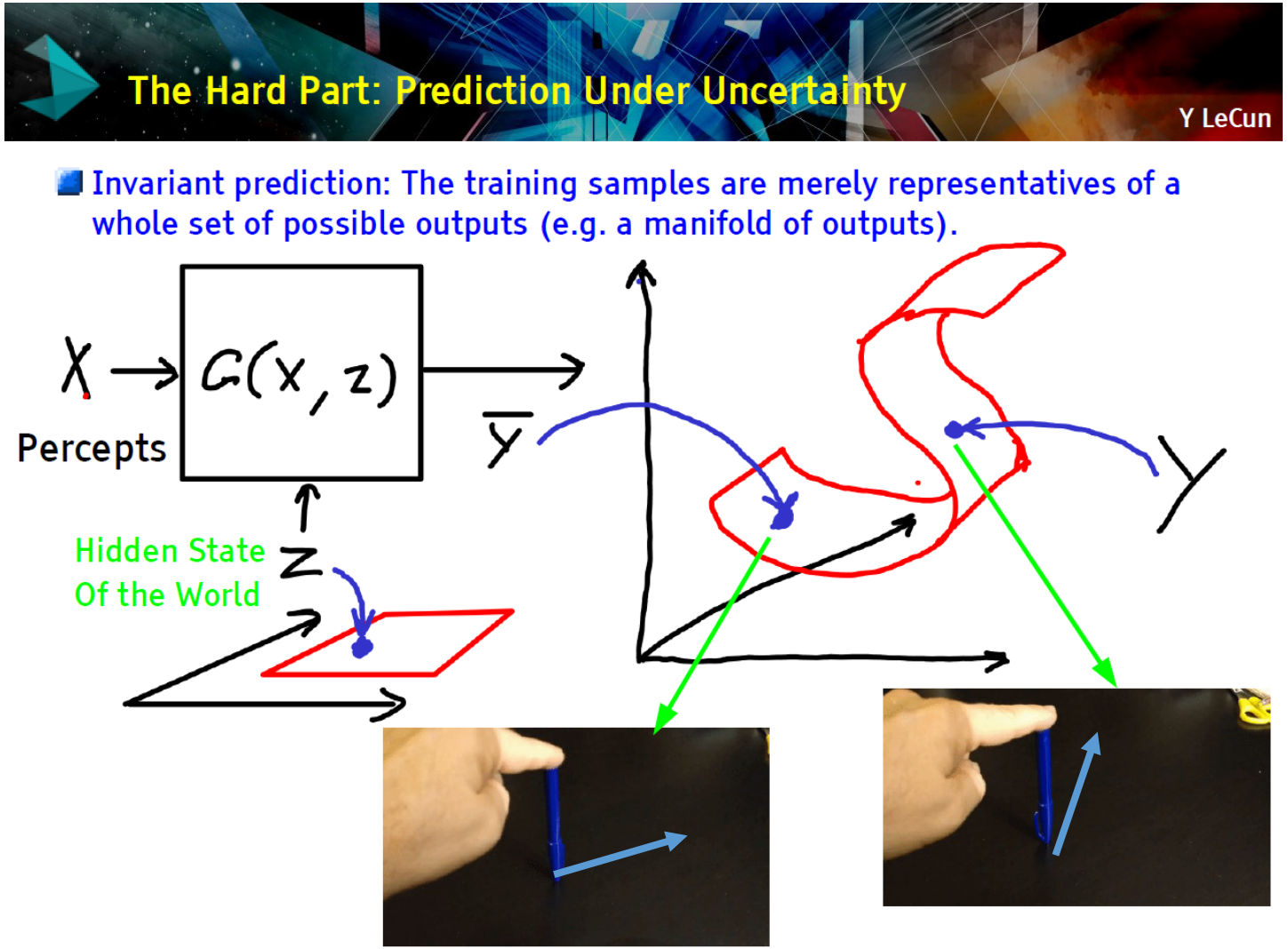

* Figure adopted from Yann Lecun's slides, NIPS 2016 (link)

# MOTIVATION

**We want to learn**

**the data manifold!**

**⇔ Do not want to give penalty to both $y$ and $\bar{y}$.**




* Figure adopted from Yann Lecun's slides, NIPS 2016 (link)

# MOTIVATION

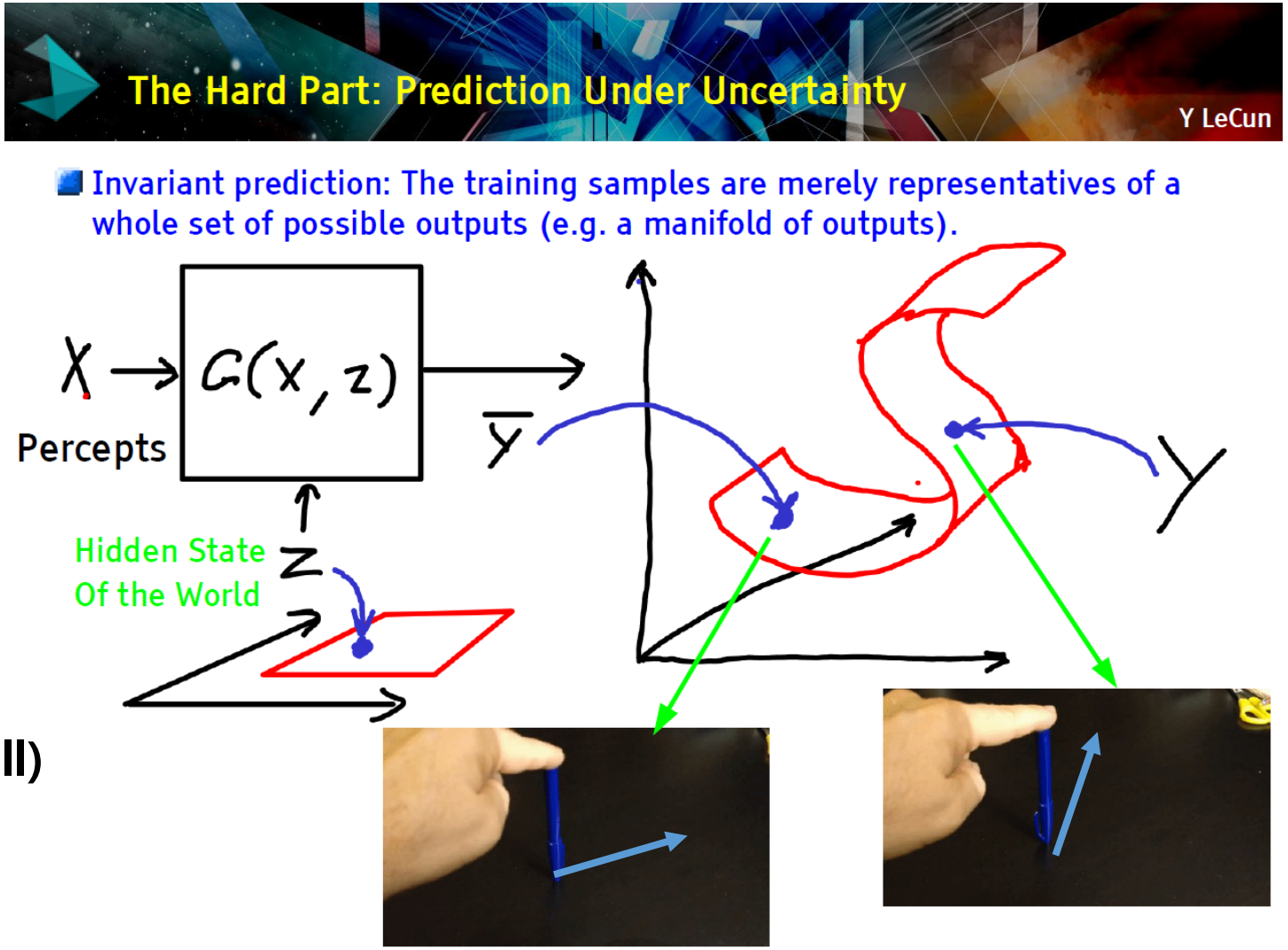
**We want to learn**

**the data manifold!**

**⇔ Do not want to give penalty to both $y$ and $\bar{y}$.**

**⇔ Pen can fall either way.**

**(there are no wrong way to fall)**




* Figure adopted from Yann Lecun's slides, NIPS 2016 (link)

# MOTIVATION

## ENERGY-BASED MODEL

**Energy based models capture dependencies between variables by associating a scalar *energy* (a measure of compatibility) to each configuration of the variables.**

***Inference*, i.e., making a prediction or decision, consists in setting the value of observed variables and finding values of the remaining variables that minimize the energy.**

***Learning* consists in finding an energy function that associates low energies to correct values of the remaining variables, and higer energies to incorrect values.**

**A *loss functional*, minimized during learning, is used to measure the quality of the available energy functions.**

* Quoted from "A Tutorial on Energy-Based Learning", Yann Lecun et al., 2006

# MOTIVATION

## ENERGY-BASED MODEL

Energy based models capture dependencies between variables by associating a scalar *energy* (a measure of compatibility) to each configuration of the variables.

*Inference*, i.e., making a prediction or decision, consists in setting the value of ob the en

*Le* to co

A *loss functional*, minimized during learning, is used to measure the quality of the available energy functions.

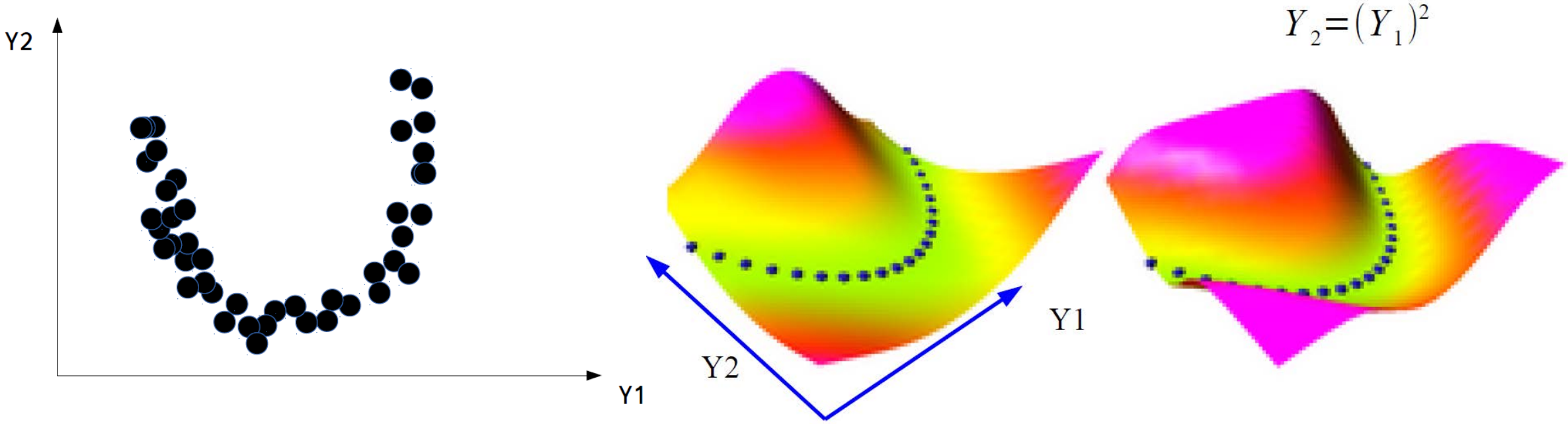
**WITHIN** the **COMMON** inference/learning **FRAMEWORK**, the wide choice of energy functions and loss functionals allows for the design of many types of statistical models, both probabilistic and non-probabilistic.

* Quoted from "A Tutorial on Energy-Based Learning", Yann Lecun et al., 2006

# MOTIVATION

## ENERGY-BASED MODEL

### "LET'S USE IT!"

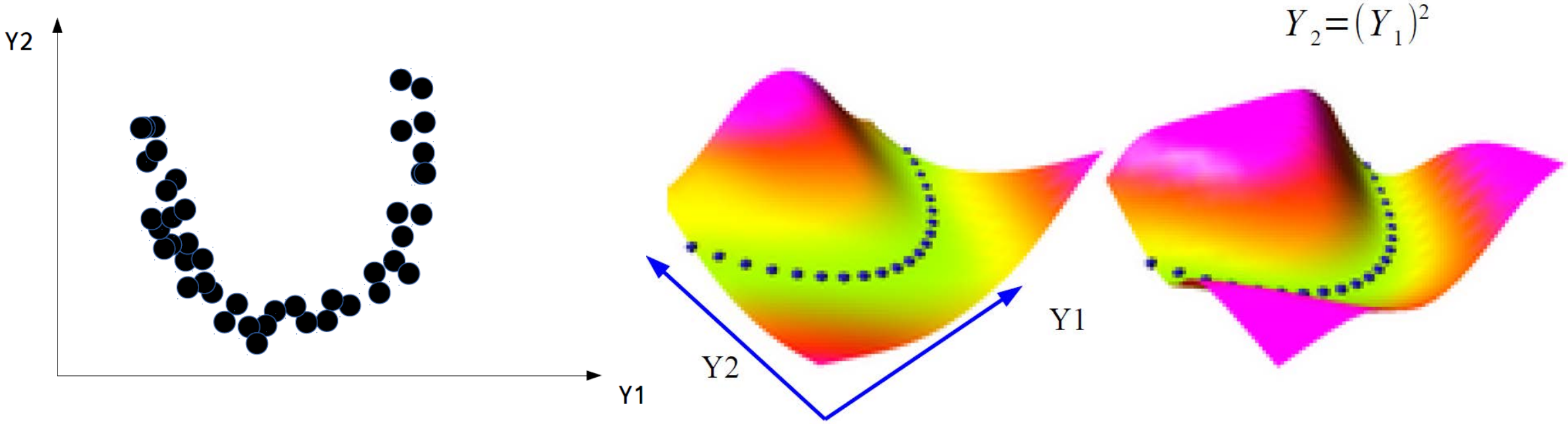



* Figure adopted from Yann Lecun's slides, NIPS 2016 (link)

# MOTIVATION

## ENERGY-BASED MODEL

**"BUT HOW do we choose where to push up?"**




* Figure adopted from Yann Lecun's slides, NIPS 2016 (link)

# MOTIVATION


- 1. build the machine so that the volume of low energy stuff is constant
  - PCA, K-means, GMM, square ICA
- 2. push down of the energy of data points, push up everywhere else
  - Max likelihood (needs tractable partition function)
- 3. push down of the energy of data points, push up on chosen locations
  - contrastive divergence, Ratio Matching, Noise Contrastive Estimation, Minimum Probability Flow
- 4. minimize the gradient and maximize the curvature around data points
  - score matching
- 5. train a dynamical system so that the dynamics goes to the manifold
  - denoising auto-encoder
- 6. use a regularizer that limits the volume of space that has low energy
  - Sparse coding, sparse auto-encoder, PSD
- 7. if E(Y) = ||Y - G(Y)||^2, make G(Y) as "constant" as possible.
  - Contracting auto-encoder, saturating auto-encoder

* Figure adopted from Yann Lecun's slides, NIPS 2016 (link)

# MOTIVATION


- 1. build the machine so that the volume of low energy stuff is constant
  - PCA, K-means, GMM, square ICA → **Limit the space**
- 2. push down of the energy of data points, push up everywhere else
  - Max likelihood (needs tractable partition function) → **Every interesting case is intractable**
- 3. push down of the energy of data points, push up on chosen locations
  - contrastive divergence, Ratio Matching, Noise Contrastive Estimation, Minimum Probability F → **How to pick the point to push up?**

**LIMITATIONS**

- 4. minimize the gradient and maximize the curvature around data points
  - score matching
- 5. train a dynamical system so that the dynamics goes to the manifold
  - denoising auto-encoder
- 6. use a regularizer that limits the volume of space that has low energy
  - Sparse coding, sparse auto-encoder, PSD

→ **Limit the model or space**

- 7. if E(Y) = ||Y - G(Y)||^2, make G(Y) as "constant" as possible.
  - Contracting auto-encoder, saturating auto-encoder

* Figure adopted from Yann Lecun's slides, NIPS 2016 (link)

# MOTIVATION

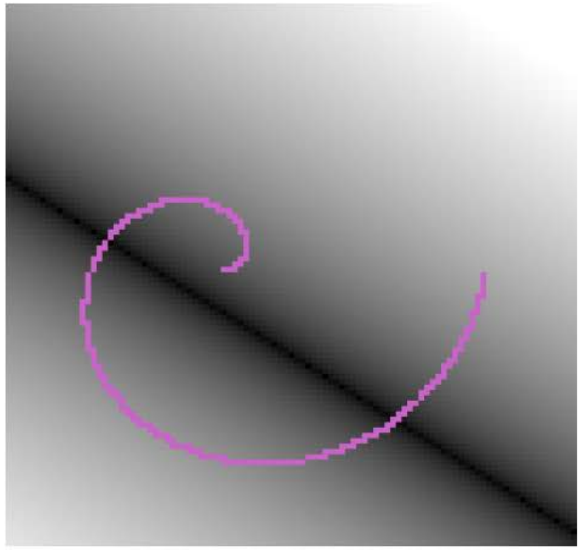
## #1: constant volume of low energy
## Energy surface for PCA and K-means

Y LeCun

- 1. build the machine so that the volume of low energy stuff is constant
  - PCA, K-means, GMM, square ICA...

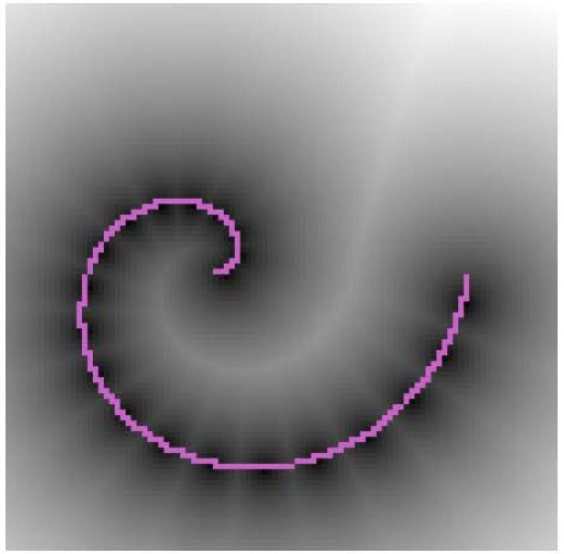
PCA

$$E(Y) = \|W^T WY - Y\|^2$$



K-Means,
Z constrained to 1-of-K code

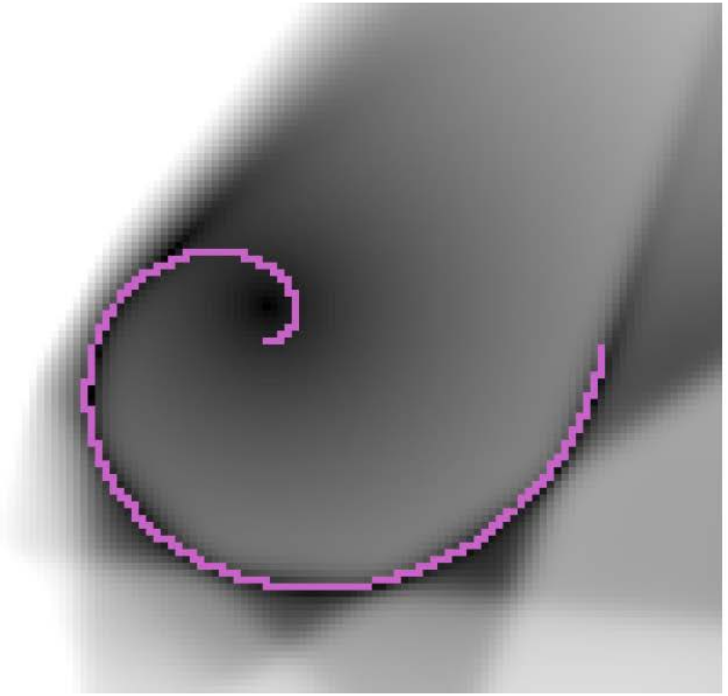
$$E(Y) = min_z \sum_i \|Y - W_i Z_i\|^2$$



* Figure adopted from Yann Lecun's slides, NIPS 2016 (link)

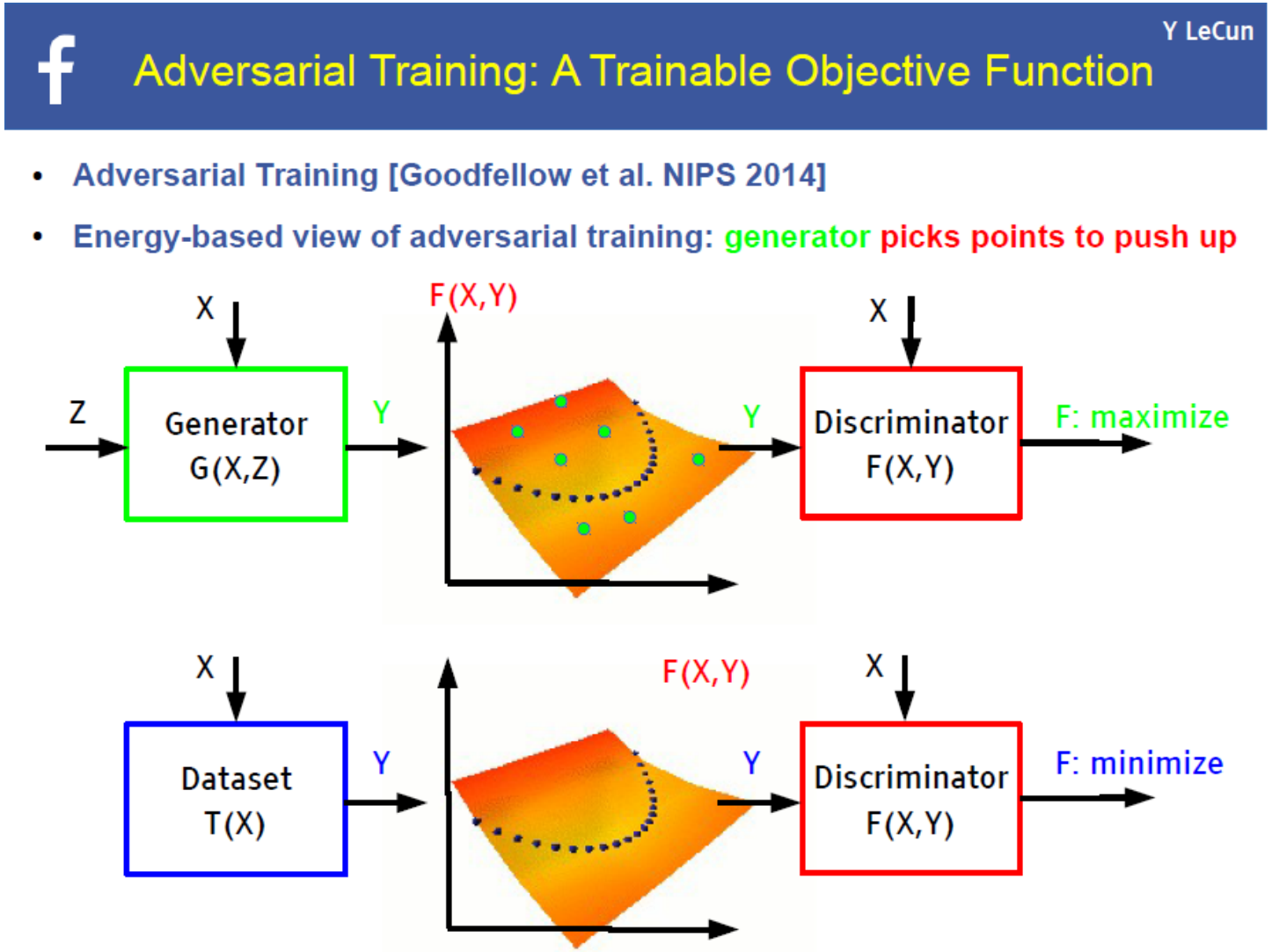
# MOTIVATION


- Sparse coding, sparse auto-encoder, Predictive Sparse Decomposition



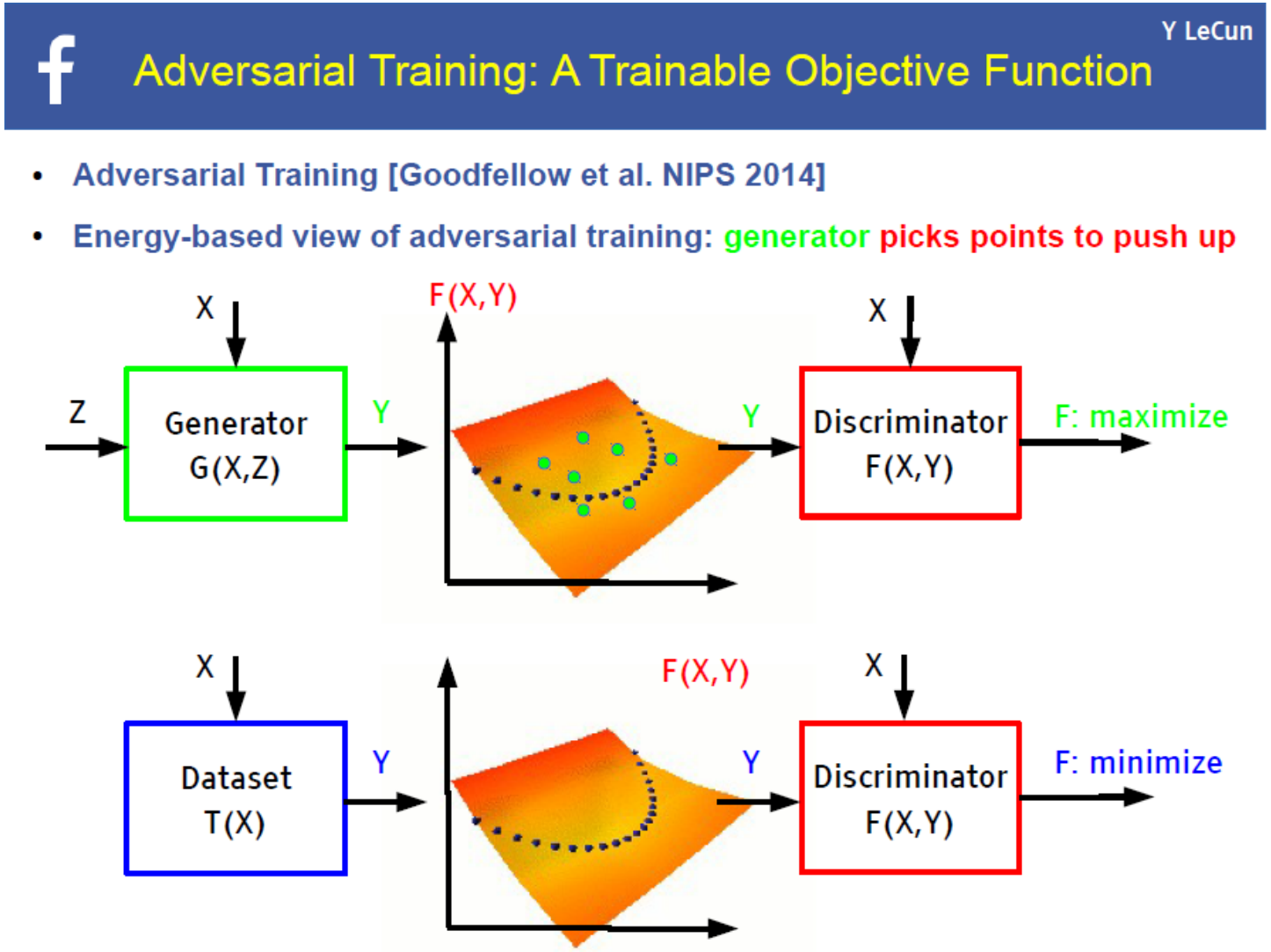
* Figure adopted from Yann Lecun's slides, NIPS 2016 (link)

# SCHEMATIC OVERVIEW



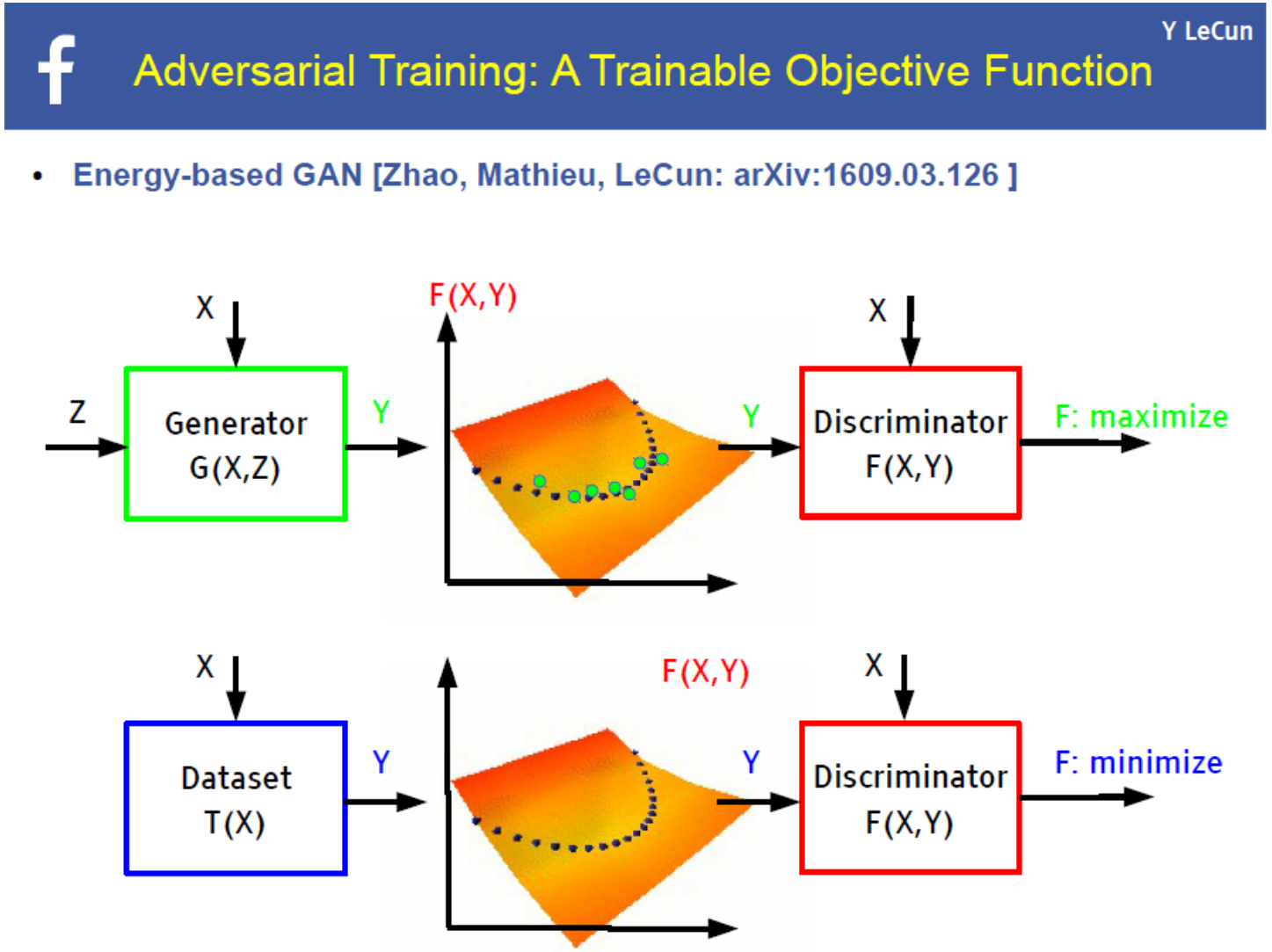

* Slides from Yann Lecun's talk in NIPS 2016 (link)

# SCHEMATIC OVERVIEW




* Slides from Yann Lecun's talk in NIPS 2016 (link)

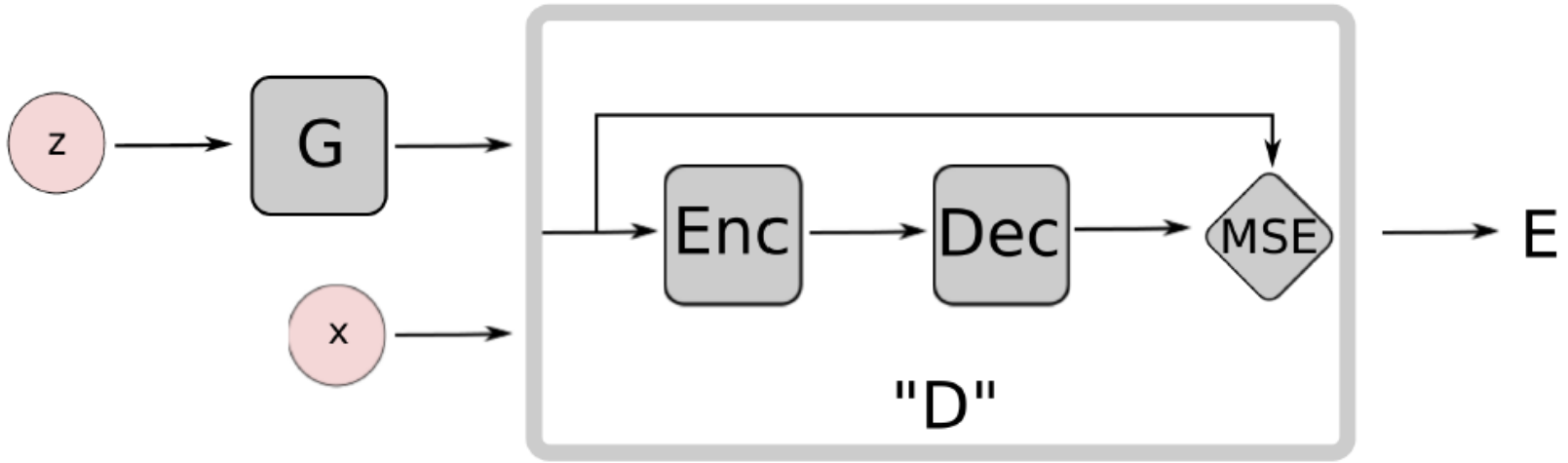
# SCHEMATIC OVERVIEW




* Slides from Yann Lecun's talk in NIPS 2016 (link)

# EBGAN

**Architecture: discriminator is an auto-encoder**




**Loss functions:**

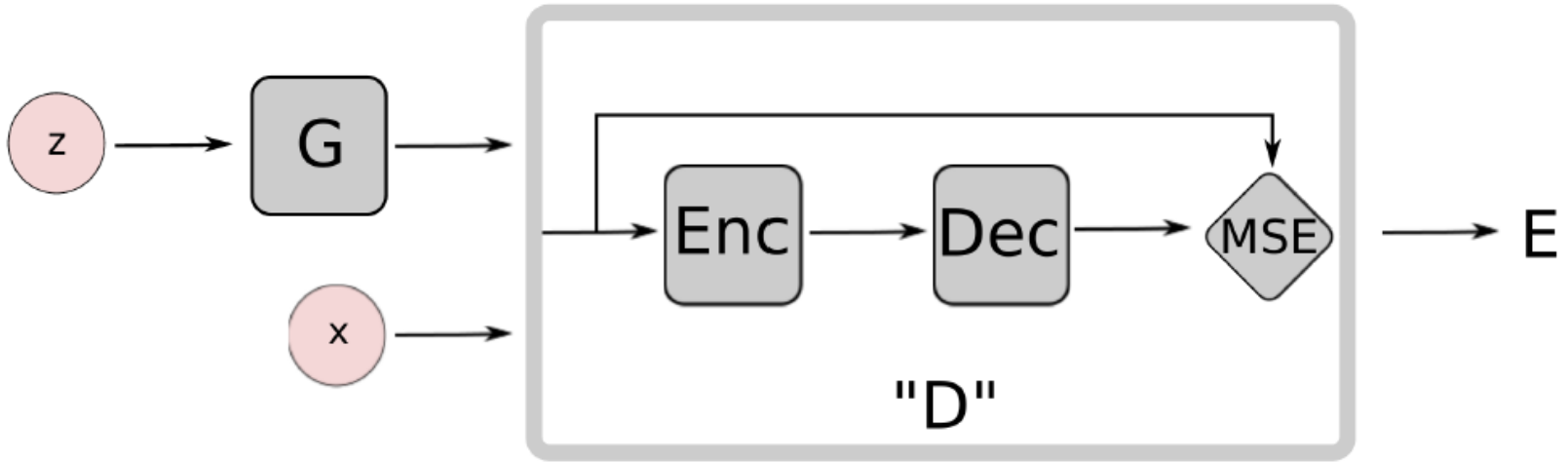
$$f_D(x,z) = D(x) + [m - D(G(x))]^+$$
$$= ||Dec(Enc(x)) - x|| + [m - ||Dec(Enc(G(z))) - G(z)||]^+$$

$$f_G(z) = ||D(G(z))||$$
$$= ||Dec(Enc(G(z))) - G(z)||$$

* Slides from Yann Lecun's talk in NIPS 2016 (link)

# EBGAN

**Architecture: discriminator is an auto-encoder**




**Loss functions:**

$$f_D(x, z) = D(x) + [m - D(G(x))]^+$$
$$= ||Dec(Enc(x)) - x|| + [m - ||Dec(Enc(G(z))) - G(z)||]^+$$

**hinge loss**

$$f_G(z) = ||D(G(z))||$$
$$= ||Dec(Enc(G(z))) - G(z)||$$

* Slides from Yann Lecun's talk in NIPS 2016 (link)

# THEORETICAL RESULTS

## EBGAN solutions are Nash Equilibria

Y LeCun

- **Loss functions for Discriminator and Generator**

$$\mathcal{L}_D(x, z) = D(x) + [m - D\big(G(z)\big)]^+$$
$$\mathcal{L}_G(z) = D\big(G(z)\big)$$

- **Nash Equilibrium**

We define $V(G, D) = \int_{x,z} \mathcal{L}_D(x, z) p_{data}(x) p_z(z) \mathrm{d}x\mathrm{d}z$ and $U(G, D) = \int_z \mathcal{L}_G(z) p_z(z) \mathrm{d}z$.

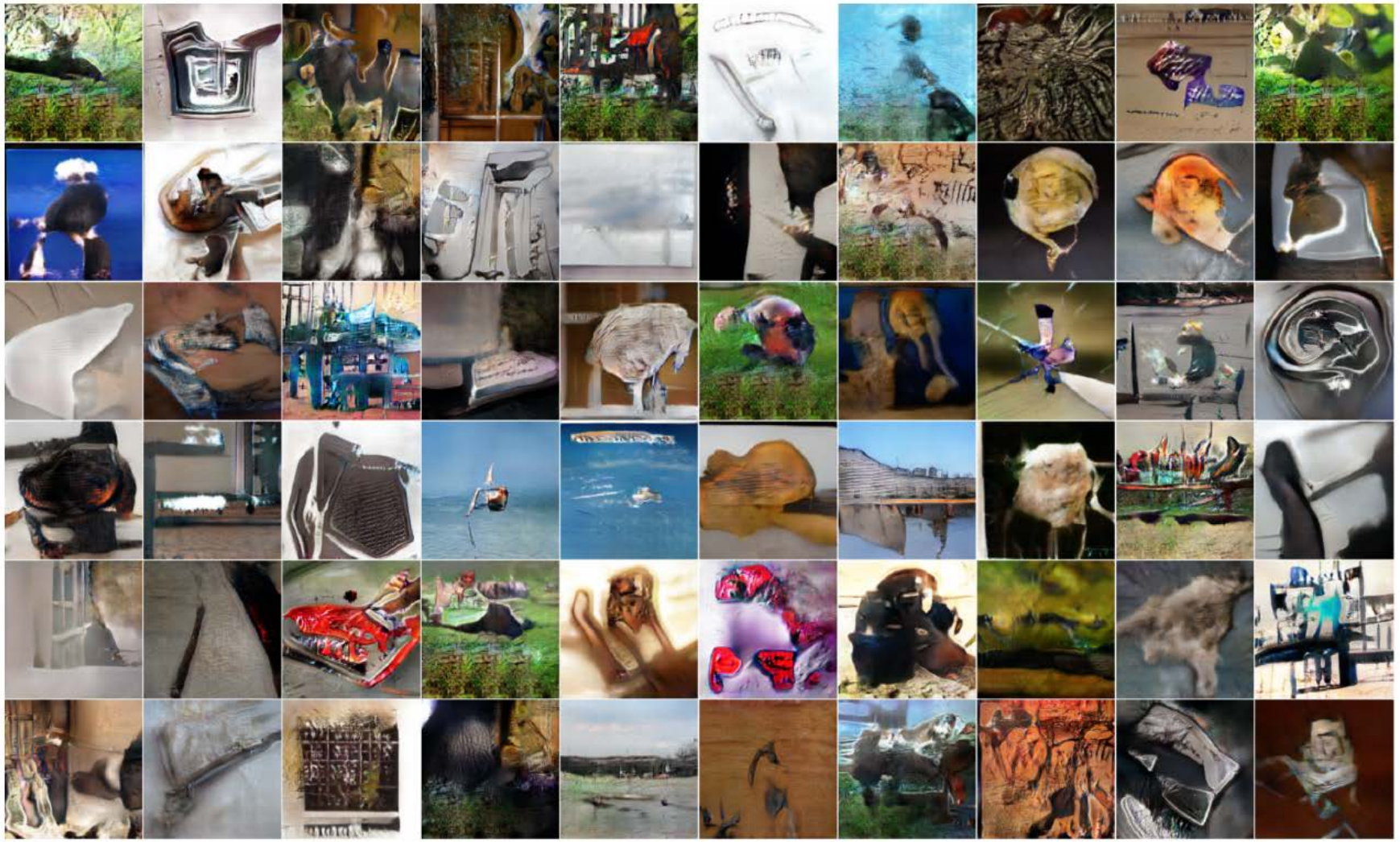
$$V(G^*, D^*) \leq V(G^*, D) \qquad \forall D$$
$$U(G^*, D^*) \leq U(G, D^*) \qquad \forall G$$

**Theorem 1.** *If $(D^*, G^*)$ is a Nash equilibrium of the system, then $p_{G^*} = p_{data}$ almost everywhere, and $V(D^*, G^*) = m$.*

**Theorem 2.** *Nash equilibrium of this system exists and is characterized by (a) $p_{G^*} = p_{data}$ (almost everywhere) and (b) there exists a constant $\gamma \in [0, m]$ such that $D^*(x) = \gamma$ (almost everywhere).*[1]

* Slides from Yann Lecun's talk in NIPS 2016 (link)

# RESULTS



* Slides from Yann Lecun's talk in NIPS 2016 (link)

# RESULTS

- Trained on dogs

* Slides from Yann Lecun's talk in NIPS 2016 (link)

# SUMMARY

1. New framework using an energy-based model
   - Discriminator as an energy function
   - Low values on the data manifold
   - Higher values everywhere else
   - Generator produce contrastive samples
2. Stable learning

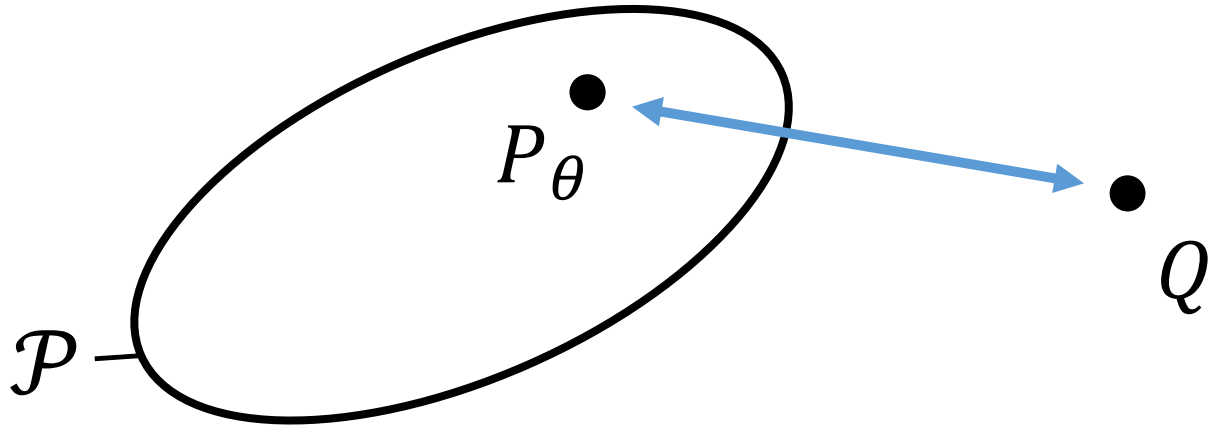
# WGAN

# MOTIVATION

**What does it mean to learn a probability model?**

**: we learned it from $f$-GAN!**

## LEARNING PROBABILISTIC MODELS



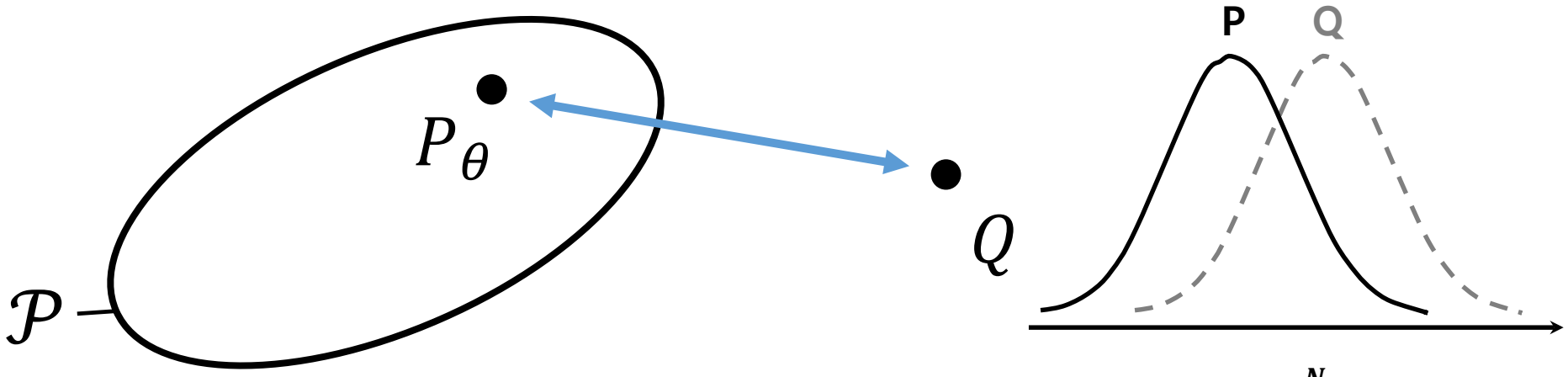

**Assumptions on $P$ :** tractable sampling, parameter gradient with respect to sample, likelihood function

# MOTIVATION

**What does it mean to learn a probability model?**

**: we learned it from $f$-GAN!**

## LEARNING PROBABILISTIC MODELS

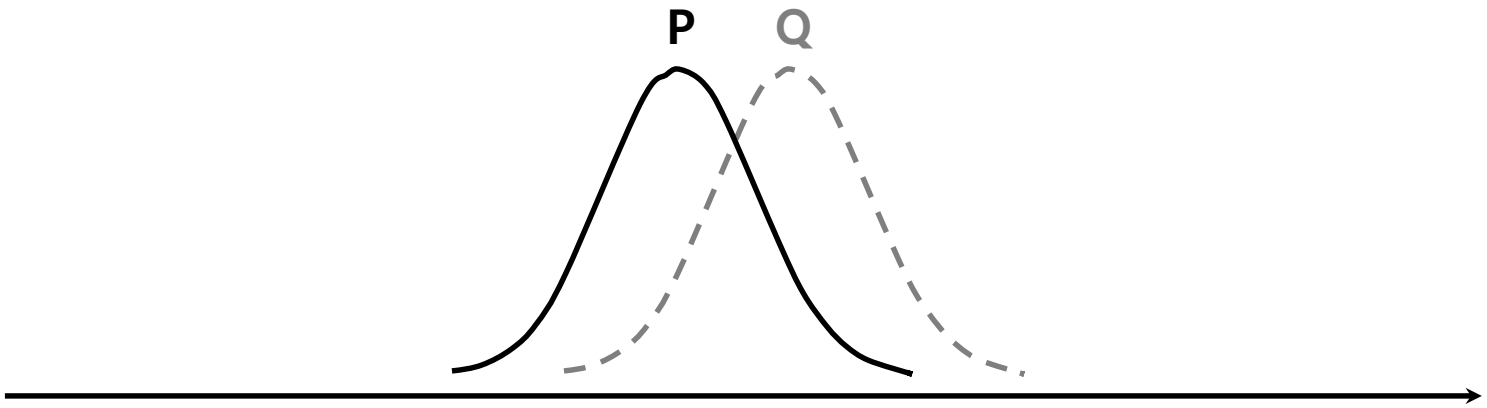



$$\dots \min_{\theta} KL[Q||P_\theta] \Leftrightarrow \max_{\theta} \sum_{i=1}^{N} log\, P(x_i|\theta)$$

**Assumptions on $P$ :** tractable sampling, parameter gradient with respect to sample, likelihood function

# MOTIVATION

**What if the supports of two distribution does not overlap?**

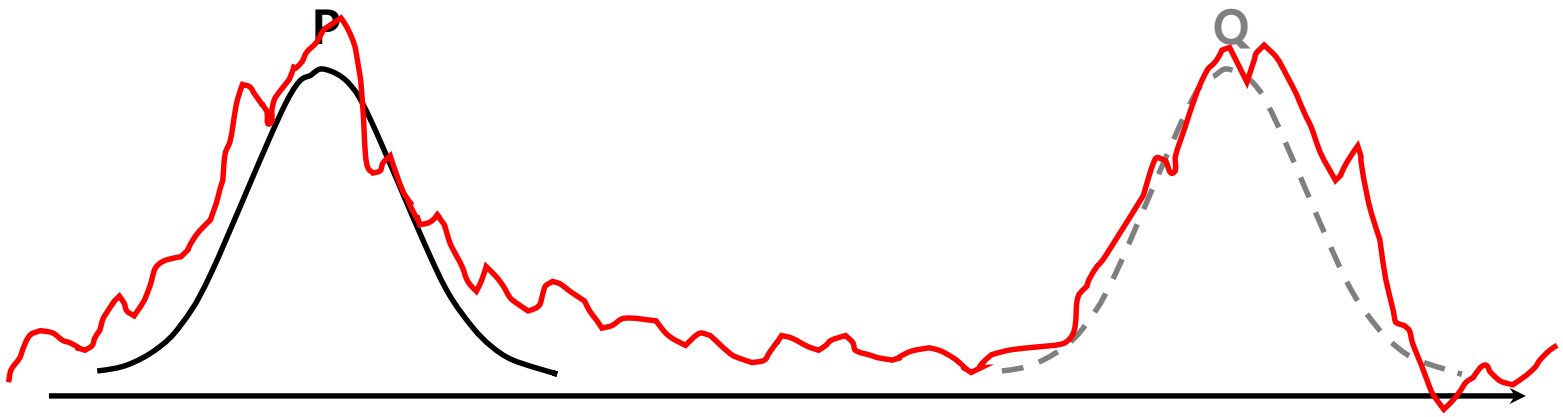



$$Is\ KL[Q||P_\theta]\ still\ defined?$$

# MOTIVATION

## What if the supports of two distribution does not overlap?

* Note that this is a very rough explanation




***Add a noise term to the model distribution***

# MOTIVATION

## What if the supports of two distribution does not overlap?

* Note that this is a very rough explanation


***Add a noise term to the model distribution***

# REVIEW!

## LEARNING PROBABILISTIC MODELS

Integral Probability Metrics
[Müller, 1997]
[Sriperumbudur et al., 2010]

$$\gamma_{\mathcal{F}}(P,Q) = \sup_{f \in \mathcal{F}} \left| \int f \mathrm{d}P - \int f \mathrm{d}Q \right|$$

- P: Expectation
- Q: Expectation
- Structure in $\mathcal{F}$
- Examples:
  - Energy statistic [Szekely, 1997]
  - Kernel MMD [Gretton et al., 2012], [Smola et al., 2007]
  - Wasserstein distance [Cuturi, 2013]
  - DISCO Nets [Bouchacourt et al., 2016]

Proper scoring rules
[Gneiting and Raftery, 2007]

$$S(P,Q) = \int S(P,x) \mathrm{d}Q(x)$$

- P: Distribution
- Q: Expectation
- Examples:
  - Log-likelihood [Fisher, 1922], [Good, 1952]
  - Quadratic score [Bernardo, 1979]

*f*-divergences
[Ali and Silvey, 1966]

$$D_f(P \parallel Q) = \int q(x) f\left(\frac{p(x)}{q(x)}\right) \mathrm{d}x$$

- P: Distribution
- Q: Distribution
- Examples:
  - Kullback-Leibler divergence [Kullback and Leibler, 1952]
  - Jensen-Shannon divergence
  - Total variation
  - Pearson $\chi^2$

# REVIEW!

## Likelihood-free Model

[Goodfellow et al., 2014]

Random input

$z \sim \text{Uniform}_{100}$

Generator

$$z \rightarrow \text{Lin}(100,1200) \rightarrow \text{ReLU}$$
$$\rightarrow \text{Lin}(1200,1200) \rightarrow \text{ReLU}$$
$$\rightarrow \text{Lin}(1200,784) \rightarrow \text{Sigmoid}$$

Output

# REVIEW!

## Likelihood-free Model

[Goodfellow et al., 2014]

Random input

Generator

Output

**WELL KNOWN FOR BEING DELICATE AND UNSTABLE FOR TRAINING**

$\rightarrow \mathrm{Lin}(1200,1200) \rightarrow \mathrm{ReLU}$

$\rightarrow \mathrm{Lin}(1200,784) \rightarrow \mathrm{Sigmoid}$

$z \sim \mathrm{Uniform}_{100}$

# REVIEW!

## LEARNING PROBABILISTIC MODELS

### Integral Probability Metrics
[Müller, 1997]
[Sriperumbudur et al., 2010]

$$\gamma_{\mathcal{F}}(P,Q) = \sup_{f \in \mathcal{F}} \left| \int f \mathrm{d}P - \int f \mathrm{d}Q \right|$$

- P: Expectation
- Q: Expectation
- Structure in $\mathcal{F}$
- Examples:
  - Energy statistic [Szekely, 1997]
  - Kernel MMD [Gretton et al., 2012], [Smola et al., 2007]
  - Wasserstein distance [Cuturi, 2013]
  - DISCO Nets [Bouchacourt et al., 2016]

### Proper scoring rules
[Gneiting and Raftery, 2007]

$$S(P,Q) = \int S(P,x) \mathrm{d}Q(x)$$

- P: Distribution
- Q: Expectation
- Examples:
  - Log-likelihood [Fisher, 1922], [Good, 1952]
  - Quadratic score [Bernardo, 1979]

### $f$-divergences
[Ali and Silvey, 1966]

$$D_f(P \parallel Q) = \int q(x) f\left(\frac{p(x)}{q(x)}\right) \mathrm{d}x$$

- P: Distribution
- Q: Distribution
- Examples:
  - Kullback-Leibler divergence [Kullback and Leibler, 1952]
  - Jensen-Shannon divergence
  - Total variation
  - Pearson $\chi^2$

# THEORETIC RESULTS

**Example 1** (Learning parallel lines)**.** Let $Z \sim U[0,1]$ the uniform distribution on the unit interval. Let $\mathbb{P}_0$ be the distribution of $(0, Z) \in \mathbb{R}^2$ (a 0 on the x-axis and the random variable $Z$ on the y-axis), uniform on a straight vertical line passing through the origin. Now let $g_\theta(z) = (\theta, z)$ with $\theta$ a single real parameter. It is easy to see that in this case,

- $W(\mathbb{P}_0, \mathbb{P}_\theta) = |\theta|$,
- $JS(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} \log 2 & \text{if } \theta \neq 0\ , \\ 0 & \text{if } \theta = 0\ , \end{cases}$
- $KL(\mathbb{P}_\theta \| \mathbb{P}_0) = KL(\mathbb{P}_0 \| \mathbb{P}_\theta) = \begin{cases} +\infty & \text{if } \theta \neq 0\ , \\ 0 & \text{if } \theta = 0\ , \end{cases}$
- and $\delta(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} 1 & \text{if } \theta \neq 0\ , \\ 0 & \text{if } \theta = 0\ . \end{cases}$

When $\theta_t \to 0$, the sequence $(\mathbb{P}_{\theta_t})_{t \in \mathbb{N}}$ converges to $\mathbb{P}_0$ under the EM distance, but does not converge at all under either the JS, KL, reverse KL, or TV divergences. Figure 1 illustrates this for the case of the EM and JS distances.

# THEORETIC RESULTS

## Different distances

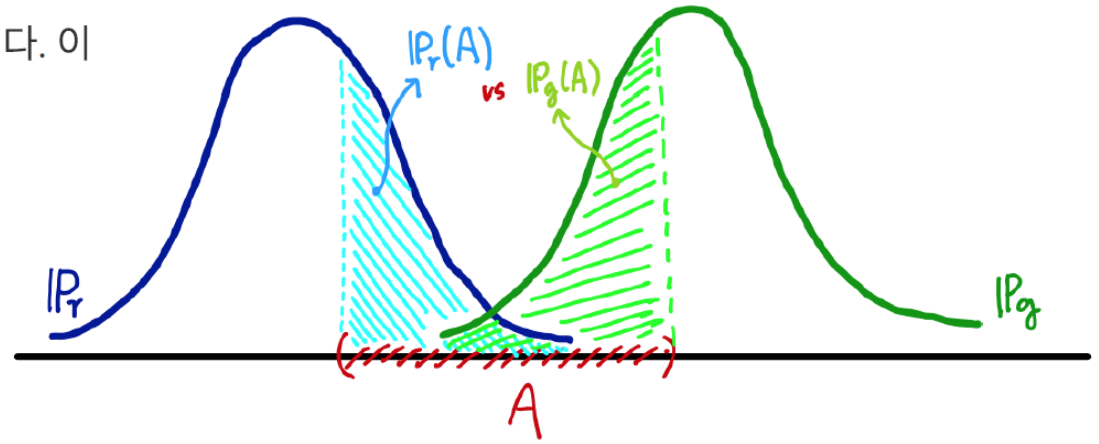
### Total Variation (TV)

$$\delta(\mathbb{P}_r,\mathbb{P}_g) = \sup_{A\in\Sigma} |\mathbb{P}_r(A) - \mathbb{P}_g(A)|$$

Total Variation 은 두 확률측도의 측정값이 벌어질 수 있는 값 중 **가장 큰 값** (또는 앞에서 설명한 supremum) 을 말합니다

같은 집합 $A$ 라 하더라도 두 확률분포가 측정하는 값은 다를 수 있습니다. 이 때 TV 는 모든 $A \in \Sigma$ 에 대해 가장 큰 값을 거리로 정의한 겁니다

$$\delta(\mathbb{P}_r,\mathbb{P}_g) = \sup_{A\in\Sigma} |\mathbb{P}_r(A) - \mathbb{P}_g(A)|$$



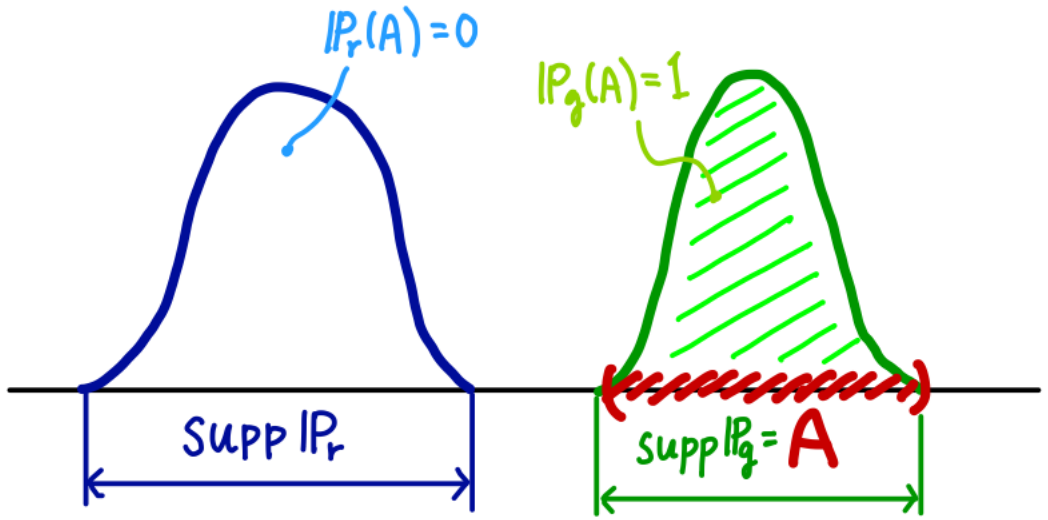

Slide courtesy of **Sungbin Lim**, DeepBio, 2017

# THEORETIC RESULTS

## Different distances

만약 두 확률분포의 확률밀도함수가 서로 겹치지 않는다면, 다시 말해 확률분포의 support 의 교집합이 공집합이라면 TV 는 무조건 1입니다!

$$\delta(\mathbb{P}_r, \mathbb{P}_g) = |0 - 1| = 1$$



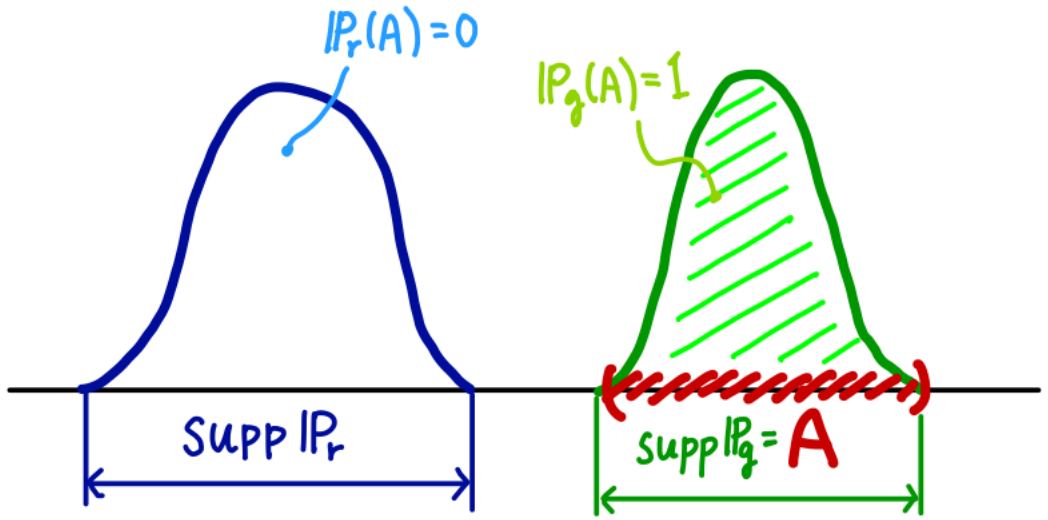

Slide courtesy of **Sungbin Lim**, DeepBio, 2017

# THEORETIC RESULTS

## Different distances

만약 두 확률분포의 확률밀도함수가 서로 겹치지 않는다면, 다시 말해 확률분포의 support 의 교집합이 공집합이라면 TV 는 무조건 1입니다!

$$\delta(\mathbb{P}_r, \mathbb{P}_g) = |0 - 1| = 1$$



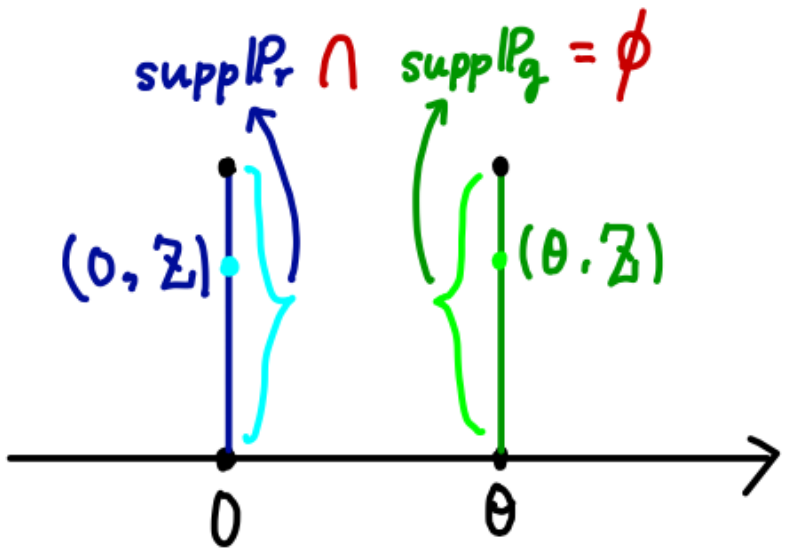

Slide courtesy of **Sungbin Lim**, DeepBio, 2017

# THEORETIC RESULTS

## Different distances

그러므로 Example 1 에서 $\theta \neq 0$ 인 경우엔 $\mathbb{P}_0$ 와 $\mathbb{P}_\theta$ 는 서로 겹치지 않은 확률분포이므로 TV 가 1이 되는 것입니다 😄

$$\delta(\mathbb{P}_r, \mathbb{P}_g) = |0 - 1| = 1$$




Slide courtesy of **Sungbin Lim**, DeepBio, 2017

# THEORETIC RESULTS

Kullback-Leibler & Jensen-Shannon divergence

$$\mathrm{KL}(\mathbb{P}_r \| \mathbb{P}_g) = \int \log\left(\frac{P_r(x)}{P_g(x)}\right) P_r(x)\mu(dx)$$

$$\mathrm{JS}(\mathbb{P}_r, \mathbb{P}_g) = \frac{1}{2}\mathrm{KL}\left(\mathbb{P}_r \| \frac{\mathbb{P}_r + \mathbb{P}_g}{2}\right) + \frac{1}{2}\mathrm{KL}\left(\mathbb{P}_g \| \frac{\mathbb{P}_r + \mathbb{P}_g}{2}\right)$$

KL 과 JS 는 워낙 익숙한 애들이죠? 😊

Slide courtesy of **Sungbin Lim**, DeepBio, 2017

# THEORETIC RESULTS

그런데 많이들 헷갈려 합니다만 사실 KL 은 metric 은 아닙니다. **대칭성** 과 **삼각부등식** 이 깨지기 때문이지요.

1. $d(x,y) \geq 0$
2. $d(x,y) = 0 \Leftrightarrow x = y$
3. $d(x,y) \neq d(y,x)$
4. $d(x,y) \not\leq d(x,z) + d(z,y)$

... 하지만 Premetric 이라서 괜찮은 성질들을 보유하고 있습니다! 😘

Slide courtesy of **Sungbin Lim**, DeepBio, 2017

# THEORETIC RESULTS

그런데 KL 은 TV 보다 strong 합니다! 😢

$$\mathrm{KL}(\mathbb{P}_n \| \mathbb{P}) \to 0 \ \text{ or } \ \mathrm{KL}(\mathbb{P} \| \mathbb{P}_n) \to 0 \quad \Rightarrow \quad \delta(\mathbb{P}_n, \mathbb{P}) \to 0$$

심지어 JS 는 TV 랑 equivalent 합니다! 😲

$$\mathrm{JS}(\mathbb{P}_n \| \mathbb{P}) \to 0 \quad \Leftrightarrow \quad \delta(\mathbb{P}_n, \mathbb{P}) \to 0$$

... 사실 오래전부터 알려진 것들입니다 (Rényi 의 1961년도 페이퍼 참조)

Slide courtesy of **Sungbin Lim**, DeepBio, 2017

# THEORETIC RESULTS

그 말은 역으로 TV 에서 수렴하지 않으면 KL 이나 JS 에서도 수렴하지 않는다는 얘기입니다 😢

$$\delta(\mathbb{P}_n, \mathbb{P}) \not\to 0 \quad \Rightarrow \quad \mathrm{KL}(\mathbb{P}_n \| \mathbb{P}) \not\to 0$$

이 사실은 Example 1 에서도 볼 수 있습니다

Slide courtesy of **Sungbin Lim**, DeepBio, 2017

# THEORETIC RESULTS

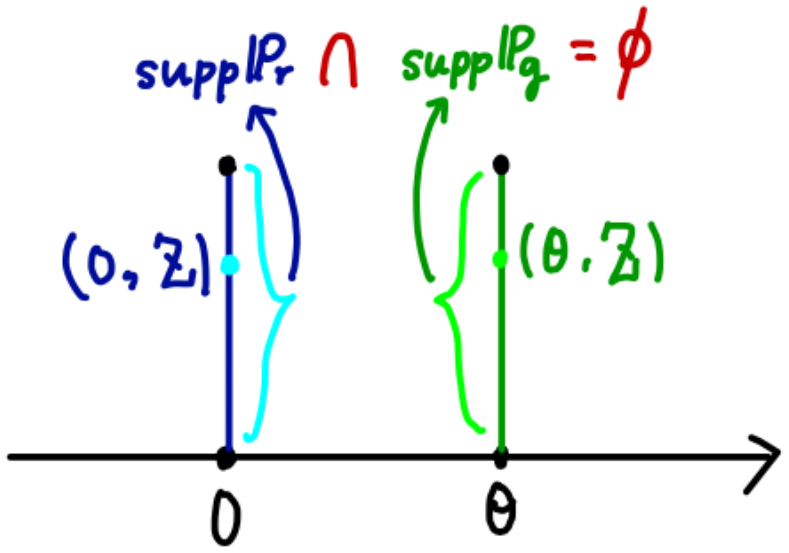
따라서 $P_\theta > 0$ 인 곳에서 log 값은 $\infty$ 가 됩니다

$$\log\left(\frac{P_\theta(x)}{P_0(x)}\right) = \infty$$

그러므로

$$\mathrm{KL}(\mathbb{P}_\theta \| \mathbb{P}_0) = \int_{\{x:P_\theta(x)\neq 0\}} \log\left(\frac{P_\theta(x)}{P_0(x)}\right) P_\theta(x)\mu(dx)$$

$$= \int \infty \cdot P_\theta(x)\mu(dx) = \infty$$




Slide courtesy of **Sungbin Lim**, DeepBio, 2017

# WASSERSTEIN DISTANCE

- The *Earth-Mover* (EM) distance or Wasserstein-1

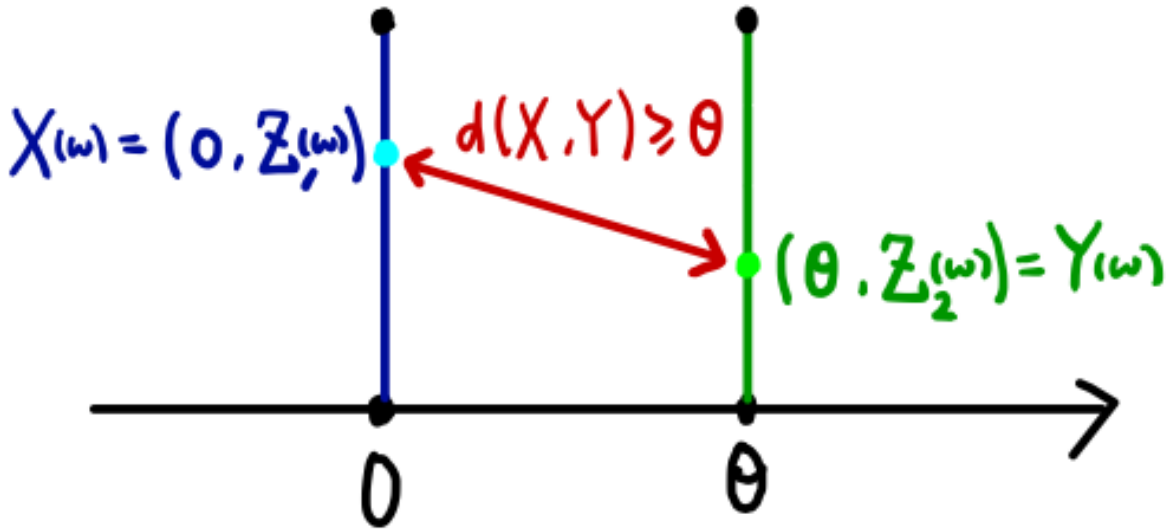
$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\big[\, \|x - y\| \,\big] \,, \tag{1}$$

where $\Pi(\mathbb{P}_r, \mathbb{P}_g)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are respectively $\mathbb{P}_r$ and $\mathbb{P}_g$. Intuitively, $\gamma(x, y)$ indicates how much "mass" must be transported from $x$ to $y$ in order to transform the distributions $\mathbb{P}_r$ into the distribution $\mathbb{P}_g$. The EM distance then is the "cost" of the optimal transport plan.

# WASSERSTEIN DISTANCE

이 때 두 점 사이의 거리는 다음과 같이 계산됩니다

$$d(X,Y) = \left(|\theta - 0|^2 + |Z_1(\omega) - Z_2(\omega)|\right)^{1/2} \geq |\theta|$$




Slide courtesy of **Sungbin Lim**, DeepBio, 2017

# WASSERSTEIN DISTANCE

이 때 두 점 사이의 거리는 다음과 같이 계산됩니다

$$d(X,Y) = \left(|\theta - 0|^2 + |Z_1(\omega) - Z_2(\omega)|\right)^{1/2} \geq |\theta|$$

## 정리 😎

- TV, KL, JS 는 $(\mathbb{P}_r, \mathbb{P}_g)$ 가 서로 겹치지 않는 상황에선 불연속이 됩니다
- EM(Wasserstein distance) 은 TV, KL, JS 보다 **약한**(weak) metric 으로 수렴을 판정하는데 무른(soft) 성질을 가집니다
- EM 은 분포수렴 과 동등합니다!

Slide courtesy of **Sungbin Lim**, DeepBio, 2017

# THEORETIC RESULTS

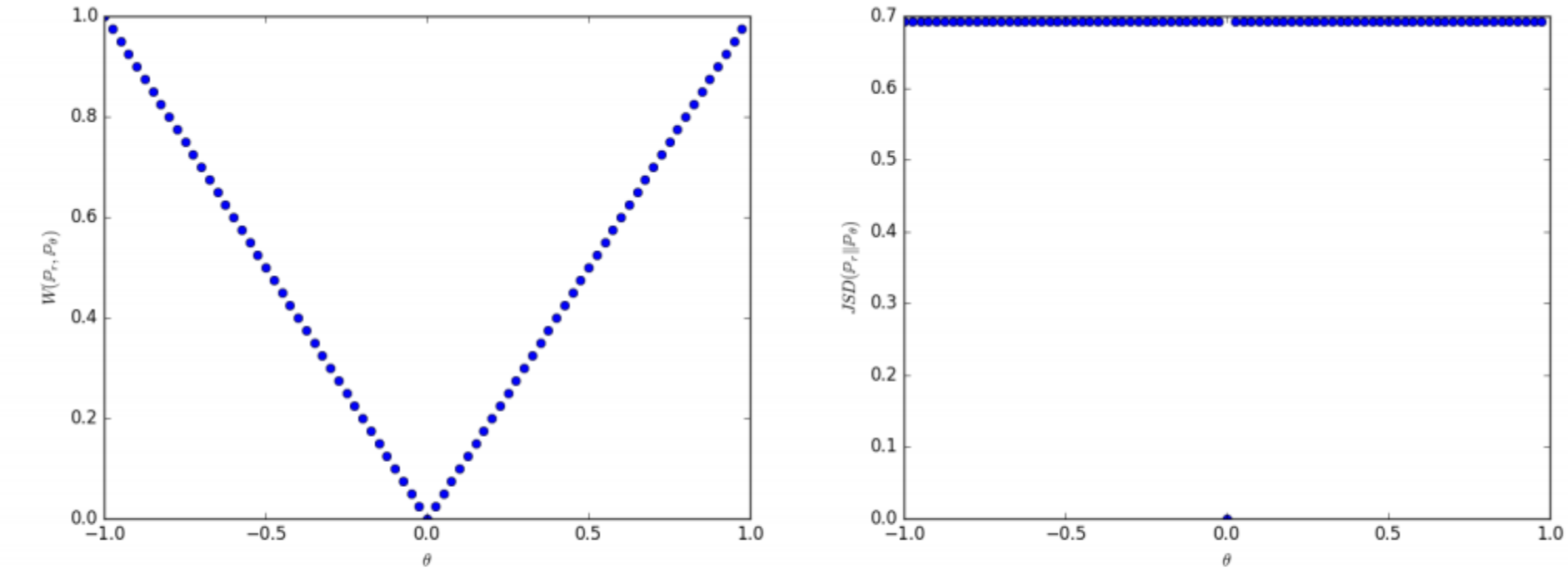



*Figure 1: These plots show $\rho(\mathbb{P}_\theta, \mathbb{P}_0)$ as a function of $\theta$ when $\rho$ is the EM distance (left plot) or the JS divergence (right plot). The EM plot is continuous and provides a usable gradient everywhere. The JS plot is not continuous and does not provide a usable gradient.*

* Figure adopted from WGAN paper (link)

# THEORETIC RESULTS

이런 일이 일어나는 이유는 TV 나 KL 이나 JS 는 두 확률분포 $\mathbb{P}_r, \mathbb{P}_g$ 가 서로 다른 영역에서 측정된 경우 **완전히 다르다** 고 판단을 내리게끔 metric 이 계산되기 때문입니다.

즉 두 확률분포의 차이를 명탐정처럼 깐깐하게(harsh) 본다는 것이죠 😤

Slide courtesy of **Sungbin Lim**, DeepBio, 2017

# THEORETIC RESULTS

그래서 GAN 의 학습에 맞게 조금 유연하면서도 **수렴** 에 포커스를 맞춘 다른 metric 이 필요한 것입니다 😉

Slide courtesy of **Sungbin Lim**, DeepBio, 2017

# THEORETIC RESULTS

Since the Wasserstein distance is much weaker than the JS distance[3], we can now ask whether $W(\mathbb{P}_r, \mathbb{P}_\theta)$ is a continuous loss function on $\theta$ under mild assumptions. This, and more, is true, as we now state and prove.

**Theorem 1.** *Let $\mathbb{P}_r$ be a fixed distribution over $\mathcal{X}$. Let $Z$ be a random variable (e.g Gaussian) over another space $\mathcal{Z}$. Let $g : \mathcal{Z} \times \mathbb{R}^d \to \mathcal{X}$ be a function, that will be denoted $g_\theta(z)$ with $z$ the first coordinate and $\theta$ the second. Let $\mathbb{P}_\theta$ denote the distribution of $g_\theta(Z)$. Then,*

1. *If $g$ is continuous in $\theta$, so is $W(\mathbb{P}_r, \mathbb{P}_\theta)$.*
2. *If $g$ is locally Lipschitz and satisfies regularity assumption 1, then $W(\mathbb{P}_r, \mathbb{P}_\theta)$ is continuous everywhere, and differentiable almost everywhere.*
3. *Statements 1-2 are false for the Jensen-Shannon divergence $JS(\mathbb{P}_r, \mathbb{P}_\theta)$ and all the KLs.*

**Wasserstein distance is a continuous function on $\theta$ under mild assumption!**

# THEORETIC RESULTS

**Theorem 2.** *Let $\mathbb{P}$ be a distribution on a compact space $\mathcal{X}$ and $(\mathbb{P}_n)_{n\in\mathbb{N}}$ be a sequence of distributions on $\mathcal{X}$. Then, considering all limits as $n \to \infty$,*

1. *The following statements are equivalent*
   - *$\delta(\mathbb{P}_n, \mathbb{P}) \to 0$ with $\delta$ the total variation distance.*
   - *$JS(\mathbb{P}_n, \mathbb{P}) \to 0$ with $JS$ the Jensen-Shannon divergence.*
2. *The following statements are equivalent*
   - *$W(\mathbb{P}_n, \mathbb{P}) \to 0$.*
   - *$\mathbb{P}_n \xrightarrow{\mathcal{D}} \mathbb{P}$ where $\xrightarrow{\mathcal{D}}$ represents convergence in distribution for random variables.*
3. *$KL(\mathbb{P}_n \| \mathbb{P}) \to 0$ or $KL(\mathbb{P} \| \mathbb{P}_n) \to 0$ imply the statements in (1).*
4. *The statements in (1) imply the statements in (2).*

**Wasserstein distance is the weakest one**

# THEORETIC RESULTS

- The *Earth-Mover* (EM) distance or Wasserstein-1

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\left[\, \|x - y\| \,\right], \tag{1}$$

where $\Pi(\mathbb{P}_r, \mathbb{P}_g)$ denotes the set of all joint dist arginals are respectively $\mathbb{P}_r$ and $\mathbb{P}_g$. Intuitively, $\gamma(x,$ "mass" must be transported from $x$ to $y$ in order to transform the distributions $\mathbb{P}_r$ into the distribution $\mathbb{P}_g$. The EM distance then is the "cost" of the optimal transport plan.

**Highly intractable!**
**(for all joint dist....)**

# THEORETIC RESULTS

Again, Theorem 2 points to the fact that $W(\mathbb{P}_r, \mathbb{P}_\theta)$ might have nicer properties when optimized than $JS(\mathbb{P}_r, \mathbb{P}_\theta)$. However, the infimum in (1) is highly intractable. On the other hand, the Kantorovich-Rubinstein duality [22] tells us that

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta}[f(x)] \tag{2}$$

where the supremum is over all the 1-Lipschitz functions $f : \mathcal{X} \to \mathbb{R}$. Note that if we replace $\|f\|_L \leq 1$ for $\|f\|_L \leq K$ (consider $K$-Lipschitz for some constant $K$), then we end up with $K \cdot W(\mathbb{P}_r, \mathbb{P}_g)$. Therefore, if we have a parameterized family of

**Change the problem with its dual problem which is tractable (somehow)!**

# THEORETIC RESULTS

**Theorem 3.** *Let $\mathbb{P}_r$ be any distribution. Let $\mathbb{P}_\theta$ be the distribution of $g_\theta(Z)$ with $Z$ a random variable with density $p$ and $g_\theta$ a function satisfying assumption 1. Then, there is a solution $f : \mathcal{X} \to \mathbb{R}$ to the problem*

$$\max_{\|f\|_L \leq 1} \mathbb{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta}[f(x)]$$

*and we have*

$$\nabla_\theta W(\mathbb{P}_r, \mathbb{P}_\theta) = -\mathbb{E}_{z \sim p(z)}[\nabla_\theta f(g_\theta(z))]$$

*when both terms are well-defined.*

**Okay! We can use the neural network!**
**(up to a constant)**

# IMPLEMENTATION

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

**Require:** : $\alpha$, the learning rate. $c$, the clipping parameter. $m$, the batch size.
$n_{\text{critic}}$, the number of iterations of the critic per generator iteration.
**Require:** : $w_0$, initial critic parameters. $\theta_0$, initial generator's parameters.

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

# RESULTS

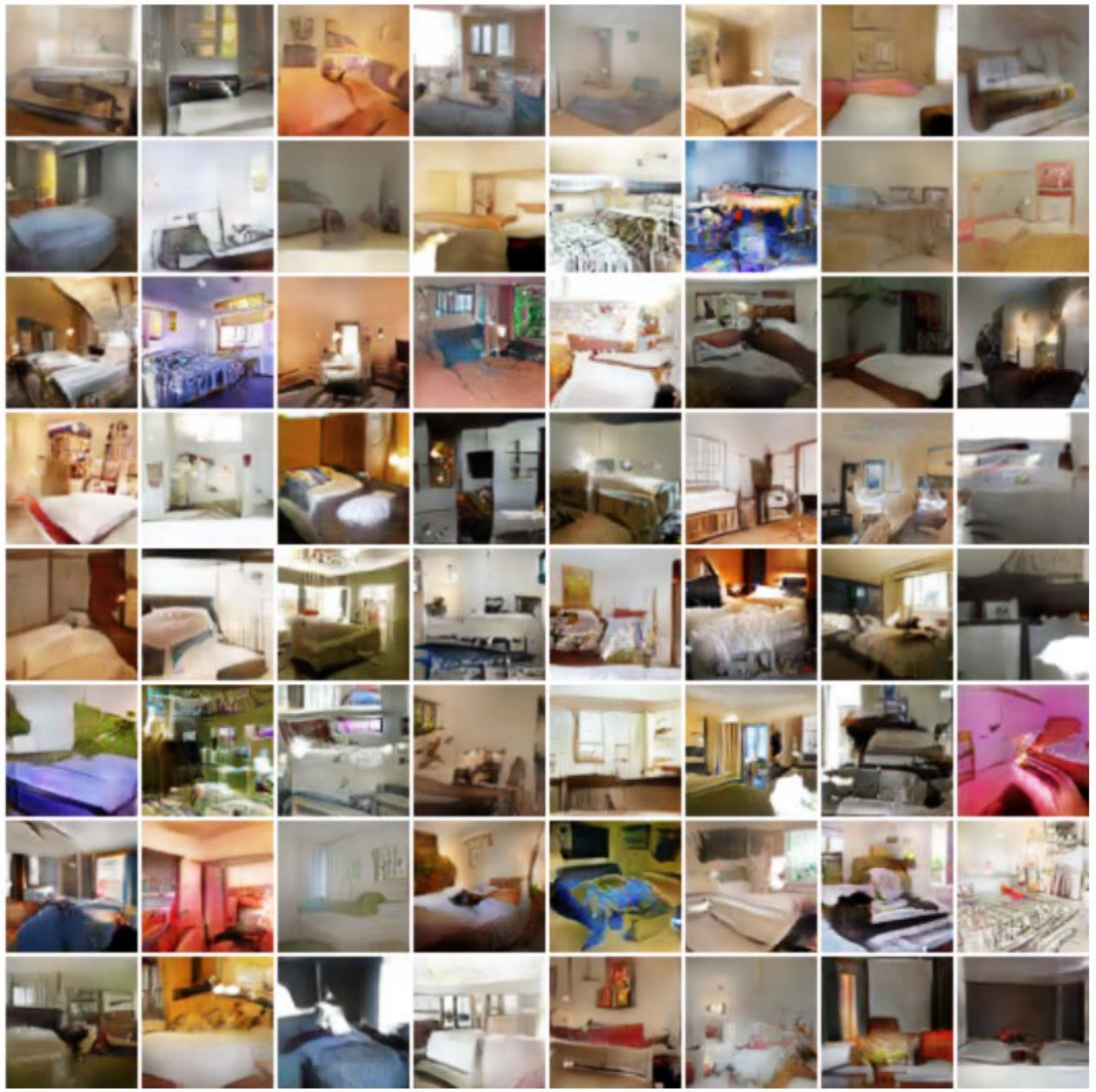


*Figure 9: WGAN algorithm: generator and critic are DCGANs.*

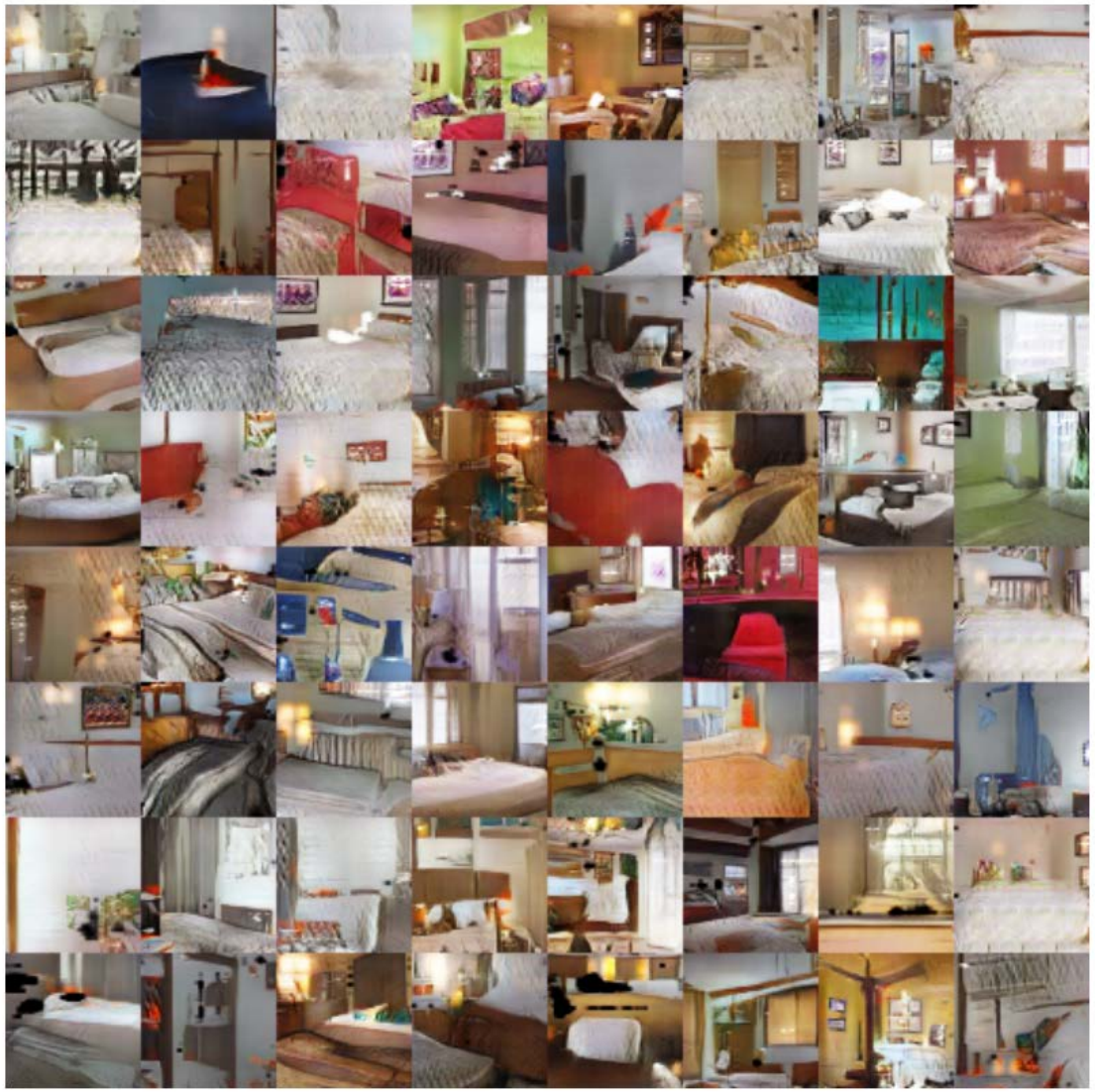
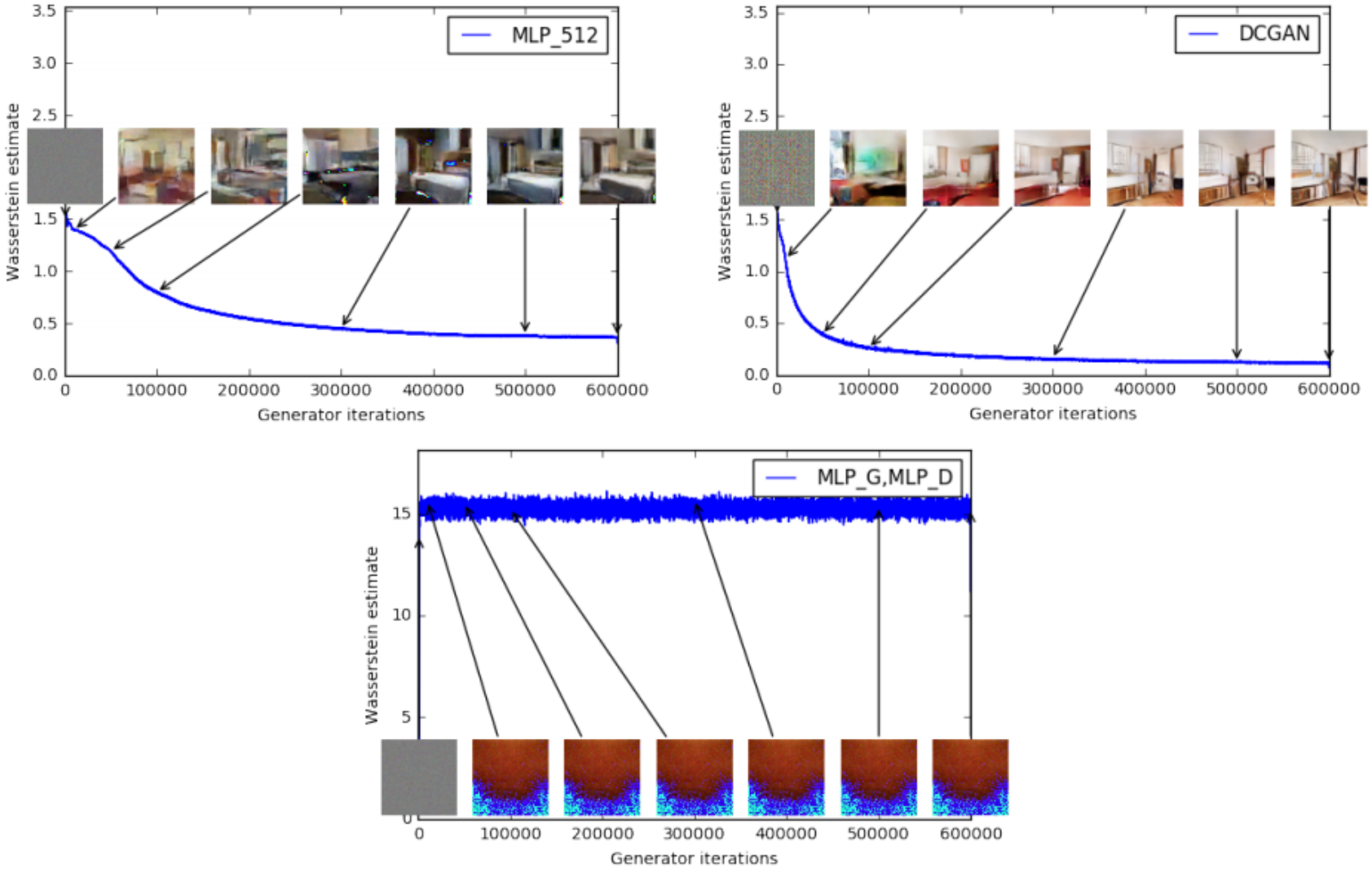


*Figure 10: Standard GAN procedure: generator and discriminator are DCGANs.*

* Figure adopted from WGAN paper (link)

# RESULTS




* Figure adopted from WGAN paper (link)

# RESULTS

## Stable without Batch normalization!

*Figure 6: Algorithms trained with a generator without batch normalization and constant number of filters at every layer (as opposed to duplicating them every time as in [18]). Aside from taking out batch normalization, the number of parameters is therefore reduced by a bit more than an order of magnitude. Left: WGAN algorithm. Right: standard GAN formulation. As we can see the standard GAN failed to learn while the WGAN still was able to produce samples.*

* Figure adopted from WGAN paper (link)

# RESULTS

## Stable without DCGAN structure (generator)!

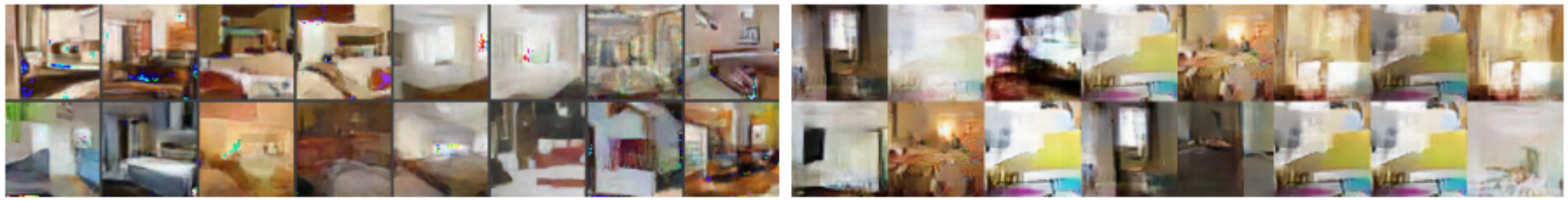


*Figure 7: Algorithms trained with an MLP generator with 4 layers and 512 units with ReLU nonlinearities. The number of parameters is similar to that of a DCGAN, but it lacks a strong inductive bias for image generation. Left: WGAN algorithm. Right: standard GAN formulation. The WGAN method still was able to produce samples, lower quality than the DCGAN, and of higher quality than the MLP of the standard GAN. Note the significant degree of mode collapse in the GAN MLP.*

* Figure adopted from WGAN paper (link)

# SUMMARY

1. Provide a comprehensive theoretical analysis of how the EM distance behaves in comparison to the others
2. Introduce Wasserstein-GAN that minimizes a reasonable and efficient approximation of the EM distance.
3. Empirically show that WGANs cure the main training problems of GANs (e.g. stability, power balance, mode-collapsing)
4. Evaluation criteria (learning curve)

# IMPLEMENTATION

```
G_sample = generator(z)
D_real = discriminator(X)
D_fake = discriminator(G_sample)

D_loss = tf.reduce_mean(D_real) - tf.reduce_mean(D_fake)
G_loss = -tf.reduce_mean(D_fake)

D_solver = (tf.train.RMSPropOptimizer(learning_rate=1e-4)
            .minimize(-D_loss, var_list=theta_D))
G_solver = (tf.train.RMSPropOptimizer(learning_rate=1e-4)
            .minimize(G_loss, var_list=theta_G))

clip_D = [p.assign(tf.clip_by_value(p, -0.01, 0.01)) for p in theta_D]
```

**THAT SIMPLE!**

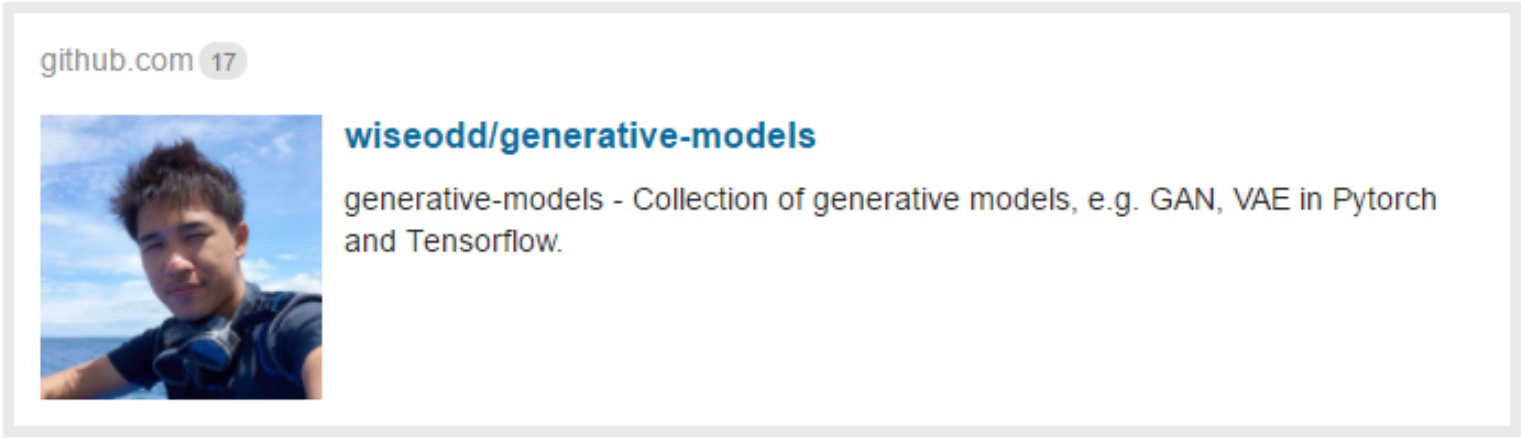
**(after all those mathematics…)**

```
sess = tf.Session()
sess.run(tf.global_variables_initializer())

if not os.path.exists('out/'):
    os.makedirs('out/')

i = 0

for it in range(1000000):
    for _ in range(5):
        X_mb, _ = mnist.train.next_batch(mb_size)

        _, D_loss_curr, _ = sess.run(
            [D_solver, D_loss, clip_D],
            feed_dict={X: X_mb, z: sample_z(mb_size, z_dim)}
        )

    _, G_loss_curr = sess.run(
        [G_solver, G_loss],
        feed_dict={z: sample_z(mb_size, z_dim)}
    )
```

# IMPLEMENTATION

- 거의 모든 GAN에 대한 구현이 Pytorch와 tensorflow 버전으로 구현되어있는 repo:

https://github.com/wiseodd/generative-models



# THANK YOU ☺

jaejun.yoo@kaist.ac.kr

# MLE & KL DIVERGENCE

Let $P(x_i|\theta)$ be the distribution for generating each data point $x_i$. We can define the model parameter distribution and the "empirical data distribution" as:

$$P_D(x) = \sum_{i=1}^{N} \frac{1}{N}\delta(x - x_i), \quad P_\theta(x) = P(x|\theta)$$

where $\delta$ is the Dirac delta function. We can verify this is a valid distribution by summing over all $x$ : $\sum_{j=1}^{N} P_D(x_j) = 1$. By using this empirical data distribution, we wish to calculate the follwoing KL distribution:

$$\begin{aligned} KL[P_D(x)||P_\theta(x)] &= \int P_D(x) \log \frac{P_D(x)}{P_\theta(x)} dx \\ &= \int P_D(x) \log P_D(x) dx - \int P_D(x) \log P_\theta(x) dx \\ &= -H[P_D(x)] - \int P_D(x) \log P_\theta(x) dx \\ &\propto -\mathbb{E}_{P_D(x)} [\log P_\theta(x)] \end{aligned}$$

$$\begin{aligned} \mathbb{E}_{P_D(x)}[\log P_\theta(x)] &= \sum_x P_D(x) \log P(x|\theta) \\ &= \sum_x \left[ \frac{1}{N} \sum_{i=1}^{N} \delta(x - x_i) \right] \log P(x|\theta) \\ &= \frac{1}{N} \sum_{i=1}^{N} \log P(x_i|\theta) \end{aligned}$$

# 분포수렴이란?

## 그나저나 분포수렴이 뭔가요? 😟

- 분포수렴은 확률분포 수렴 종류 중 하나로서 **제일 약한 수렴** 입니다
- 확률분포의 개별적인 특징보다 전체적인 모양을 중시하는 수렴입니다
- 중심극한정리(Central Limit Theorem) 에서 표본평균이 정규분포로 수렴하는 종류가 바로 분포수렴입니다
- $X_n$ 의 **모든** 모멘트 가 $X$ 의 모멘트로 수렴하면 분포수렴합니다
- $X_n$ 의 누적확률밀도함수 가 $X$ 의 누적확률밀도함수 중 연속인 모든 점에서 수렴하면 분포수렴합니다
- $X_n$ 의 Fourier transform 이 수렴하면 분포수렴합니다!