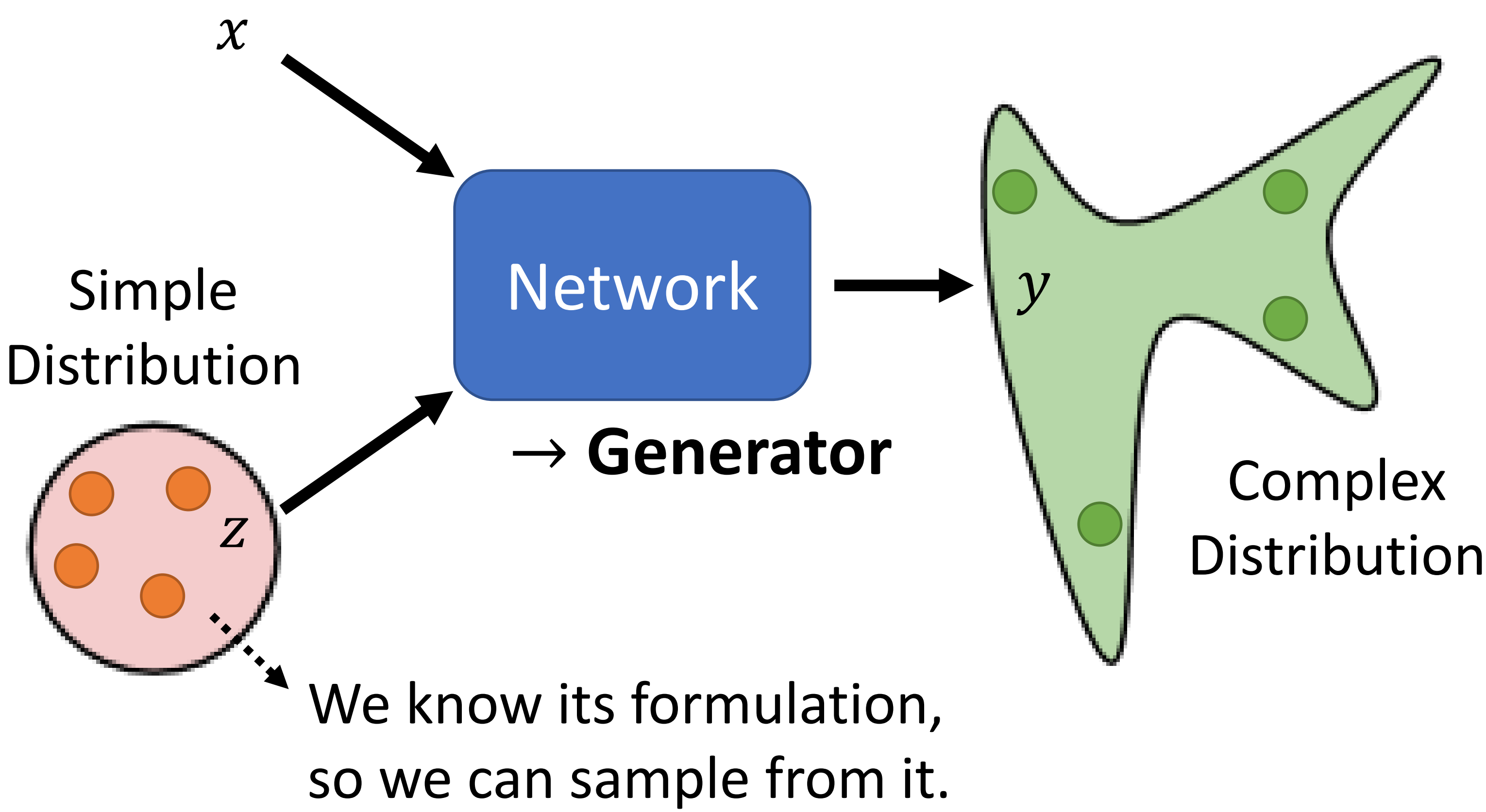
Generation

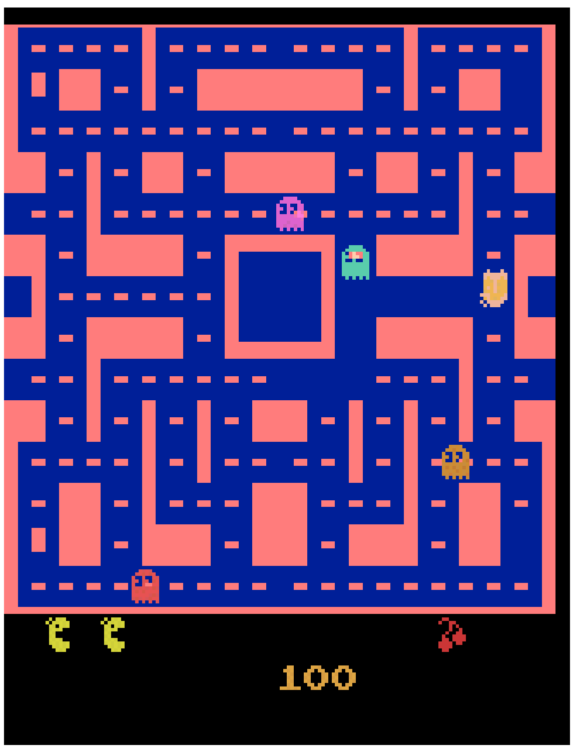
Hung-yi Lee 李宏毅

Network as Generator



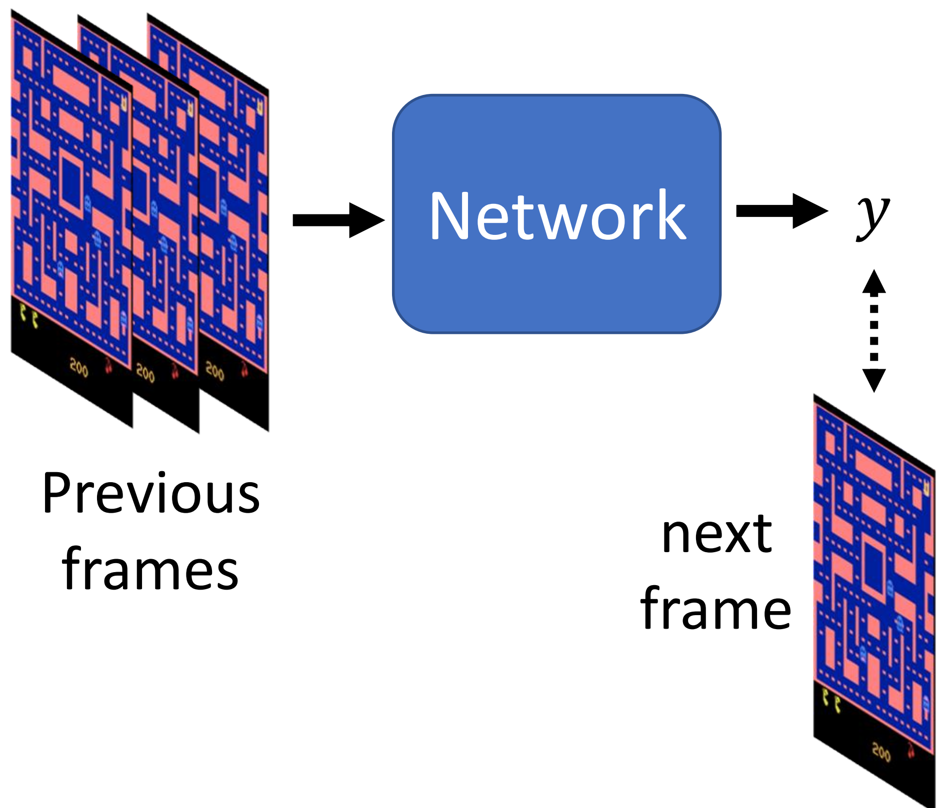
Source: https://github.com/dyelax/Adversarial_Video_Generation

Why distribution?



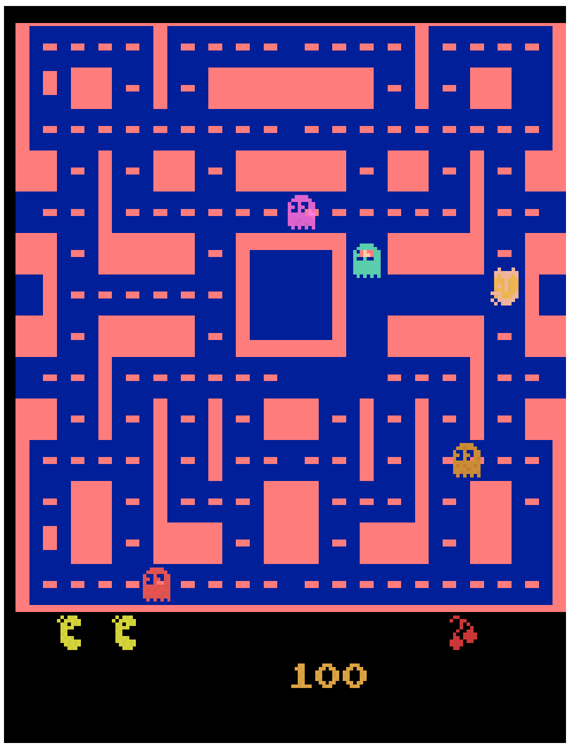
Real Video

Video Prediction



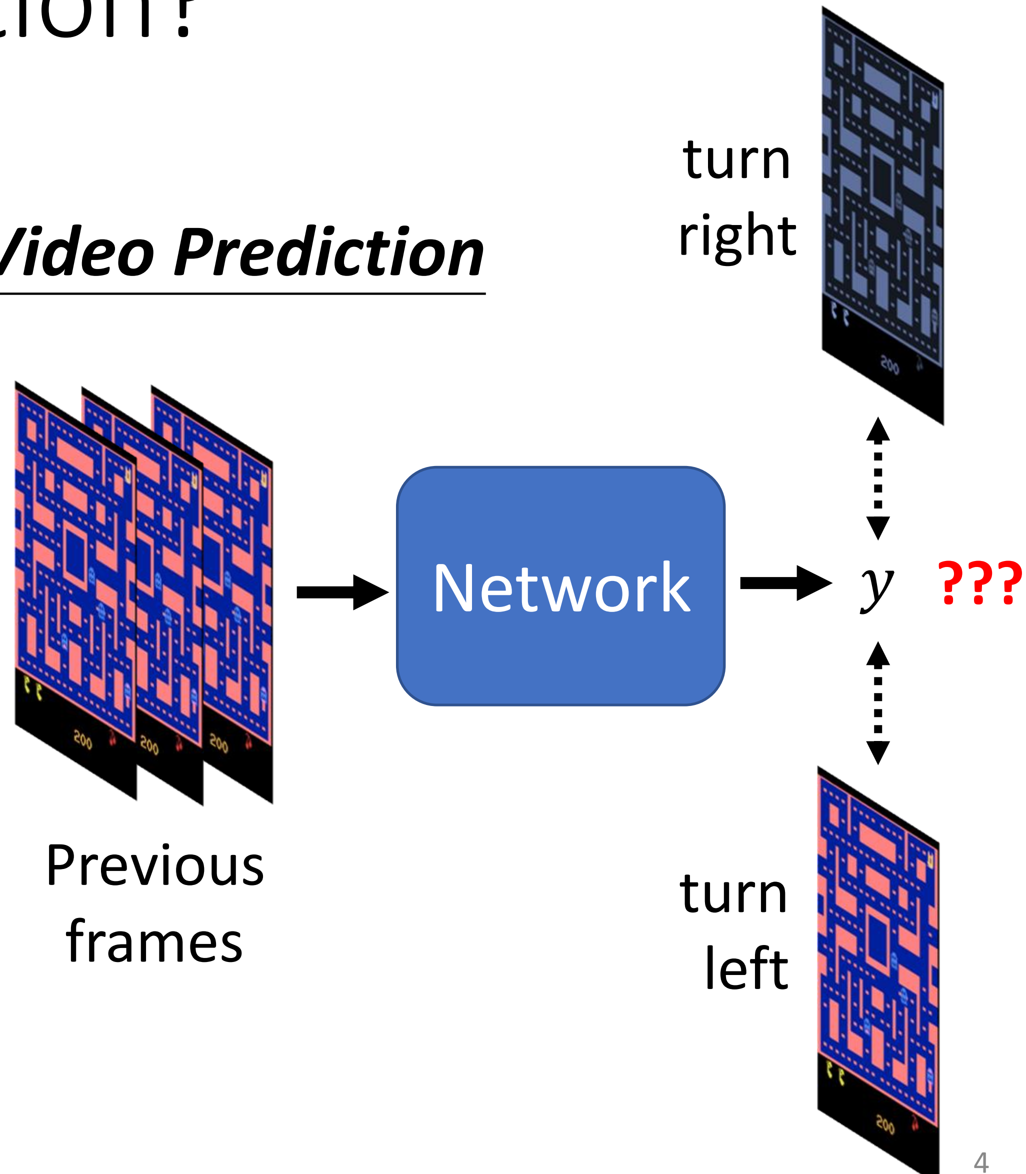
Source: https://github.com/dyelax/Adversarial_Video_Generation

Why distribution?



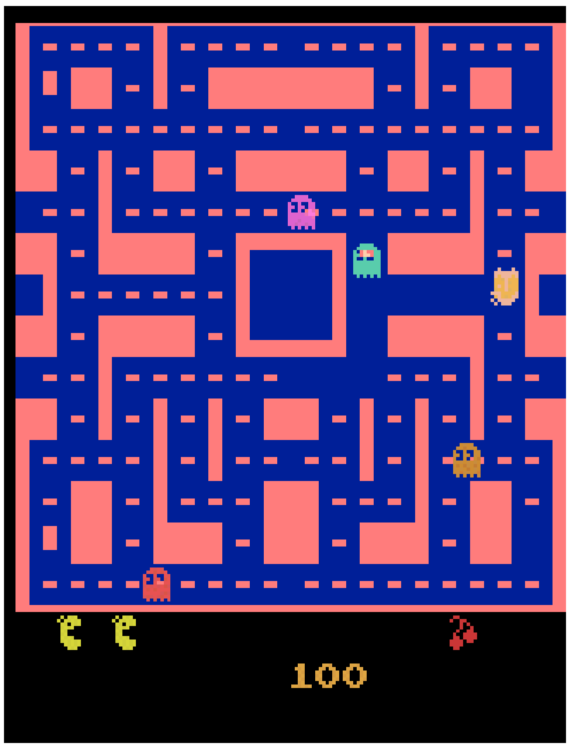
Prediction

Video Prediction



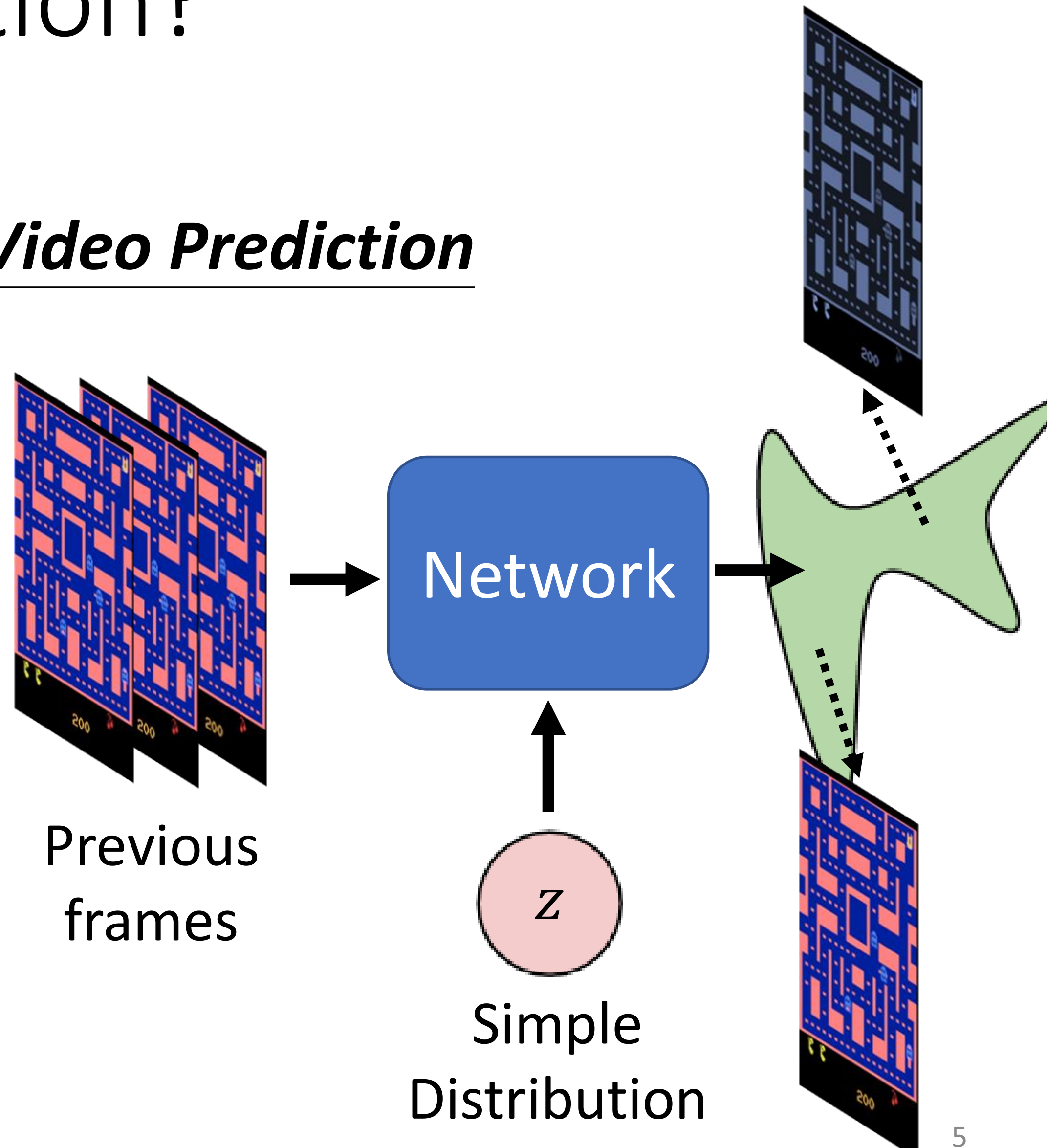
Source: https://github.com/dyelax/Adversarial_Video_Generation

Why distribution?



Prediction

<u>Video Prediction</u>



Why distribution?

(The same input has different outputs.)

- Especially for the tasks needs “*creativity*”

<u>Drawing</u>

Character
with red eyes → Network →


<u>Chatbot</u>

你知道輝夜是
誰嗎? → Network →

她是秀知院學生會 …

她開創了忍者時代 …

Generative Adversarial Network (GAN)

GAN

- How to pronounce “GAN”?

Google 小姐

All Kinds of GAN …

https://github.com/hindupuravinash/the-gan-zoo

GAN

ACGAN

BGAN

CGAN

DCGAN

EBGAN

fGAN

GoGAN

⋮

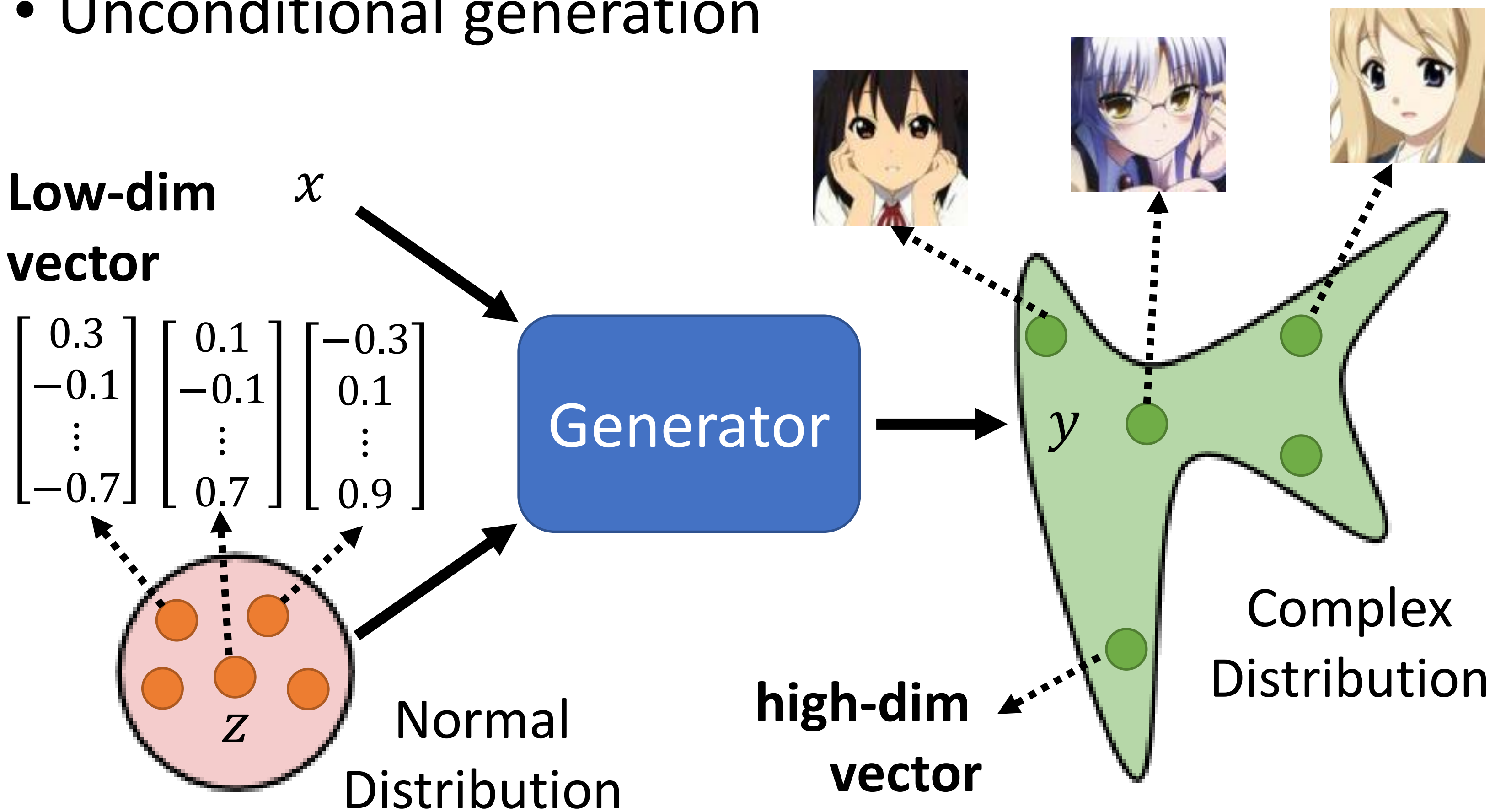
- SeUDA - Semantic-Aware Generative Adversarial Nets for Unsupervised Domain Adapt
Segmentation
- SG-GAN - Semantic-aware Grad-GAN for Virtual-to-Real Urban Scene Adaption (githu
- SG-GAN - Sparsely Grouped Multi-task Generative Adversarial Networks for Facial Attr
- SGAN - Texture Synthesis with Spatial Generative Adversarial Networks
- SGAN - Stacked Generative Adversarial Networks (github)
- SGAN - Steganographic Generative Adversarial Networks
- SGAN - SGAN: An Alternative Training of Generative Adversarial Networks
- SGAN - CT Image Enhancement Using Stacked Generative Adversarial Networks and Tr
Segmentation Improvement
- sGAN - Generative Adversarial Training for MRA Image Synthesis Using Multi-Contrast
- SiftingGAN - SiftingGAN: Generating and Sifting Labeled Samples to Improve the Rem
Classification Baseline in vitro
- SiGAN - SiGAN: Siamese Generative Adversarial Network for Identity-Preserving Face H
- SimGAN - Learning from Simulated and Unsupervised Images through Adversarial Trai
- SisGAN - Semantic Image Synthesis via Adversarial Learning

Mihaela Rosca, Balaji Lakshminarayanan, David Warde-Farley, Shakir Mohamed, "Variational Approaches for Auto-Encoding Generative Adversarial Networks", arXiv, 2017

[2]We use the Greek α prefix for α-GAN, as AEGAN and most other Latin prefixes seem to have been taken https://deephunt.in/the-gan-zoo-79597dc8c347.

Anime Face Generation

- Unconditional generation



Discriminator

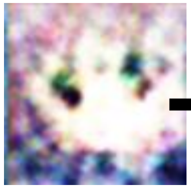
It is a neural network (that is, a function).

image → Discri-minator → **Scalar**: Larger means real, smaller value fake.

Discri-minator → 1.0

Discri-minator → 1.0

Discri-minator → 0.1



Discri-minator → 0.1

Basic Idea of GAN

Generator

Brown

veins

Butterflies are not brown

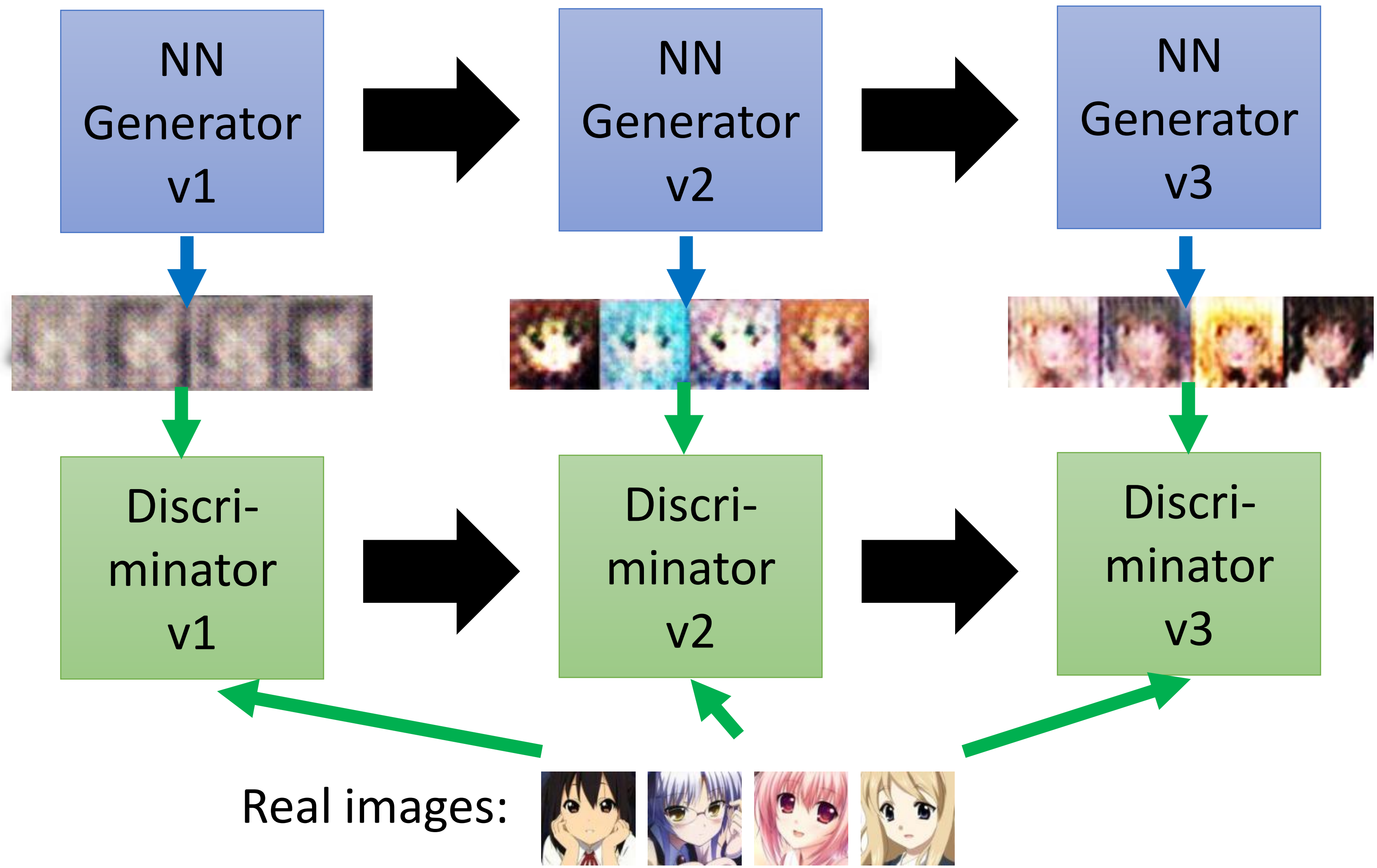
Butterflies do not have veins

……..

Discriminator

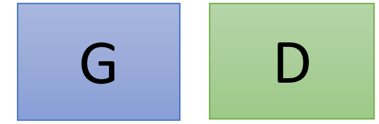
Basic Idea of GAN

This is where the term "***adversarial***" comes from.



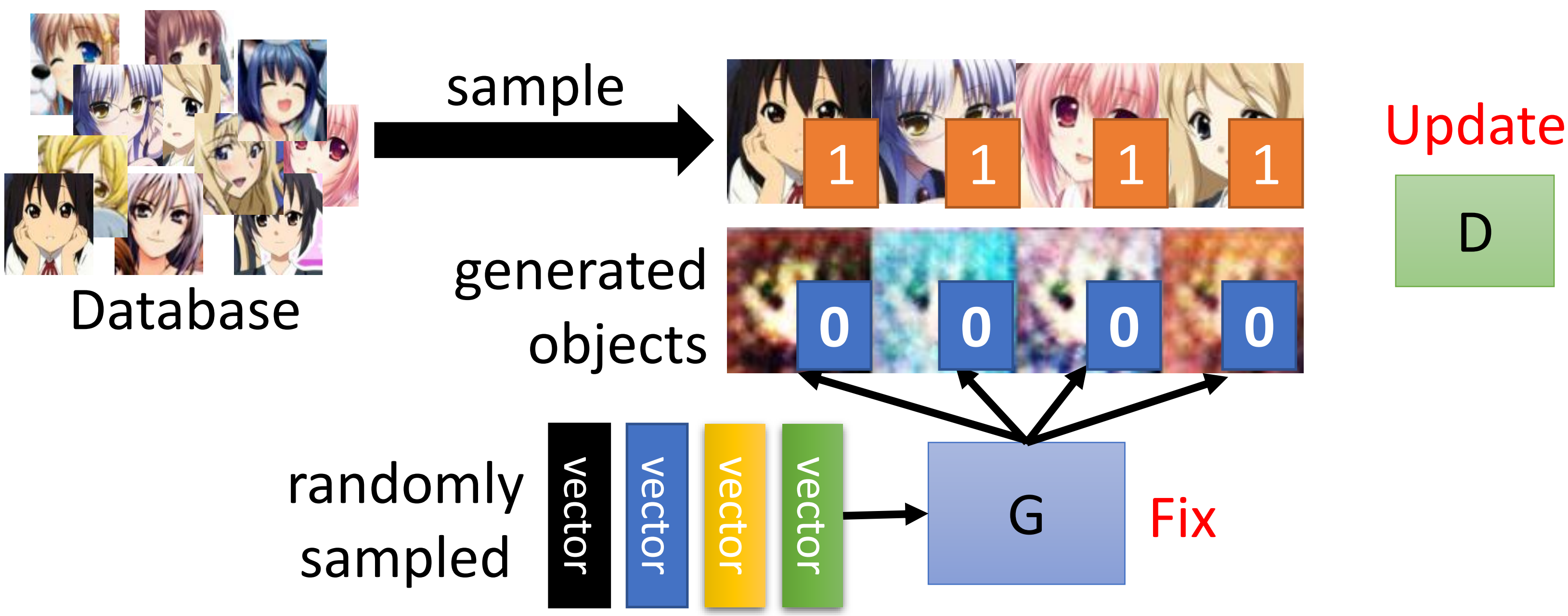
Algorithm

- Initialize generator and discriminator



- In each training iteration:

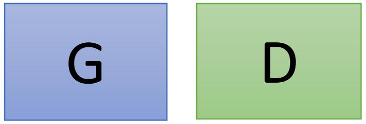
Step 1: Fix generator G, and update discriminator D



Discriminator learns to assign high scores to real objects and low scores to generated objects.

Algorithm

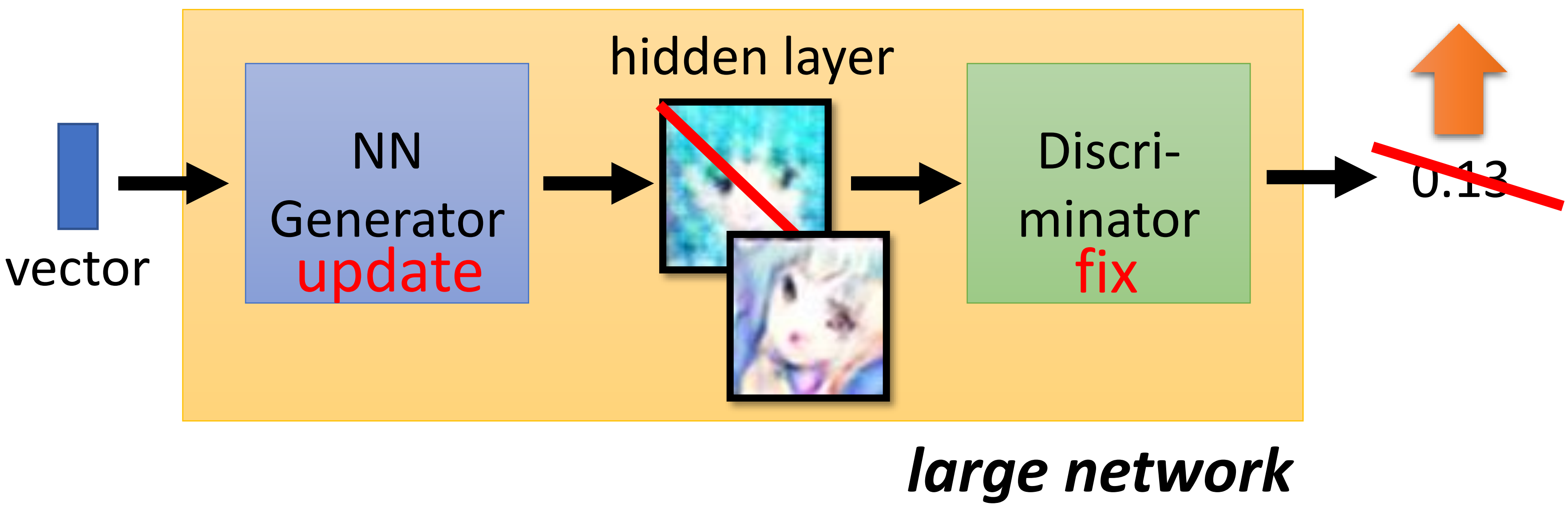
- Initialize generator and discriminator



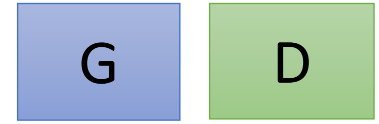
- In each training iteration:

Step 2: Fix discriminator D, and update generator G

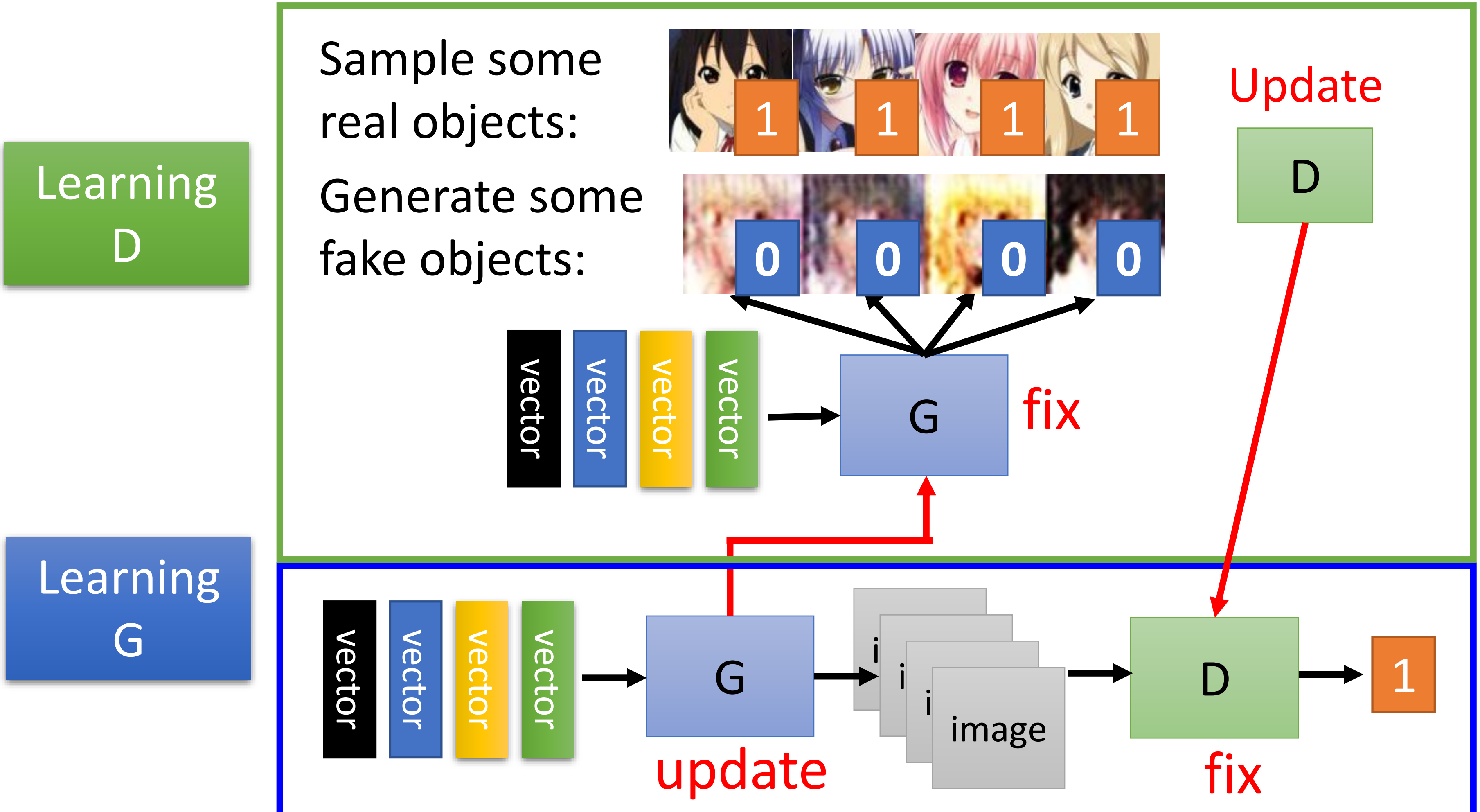
Generator learns to “fool” the discriminator



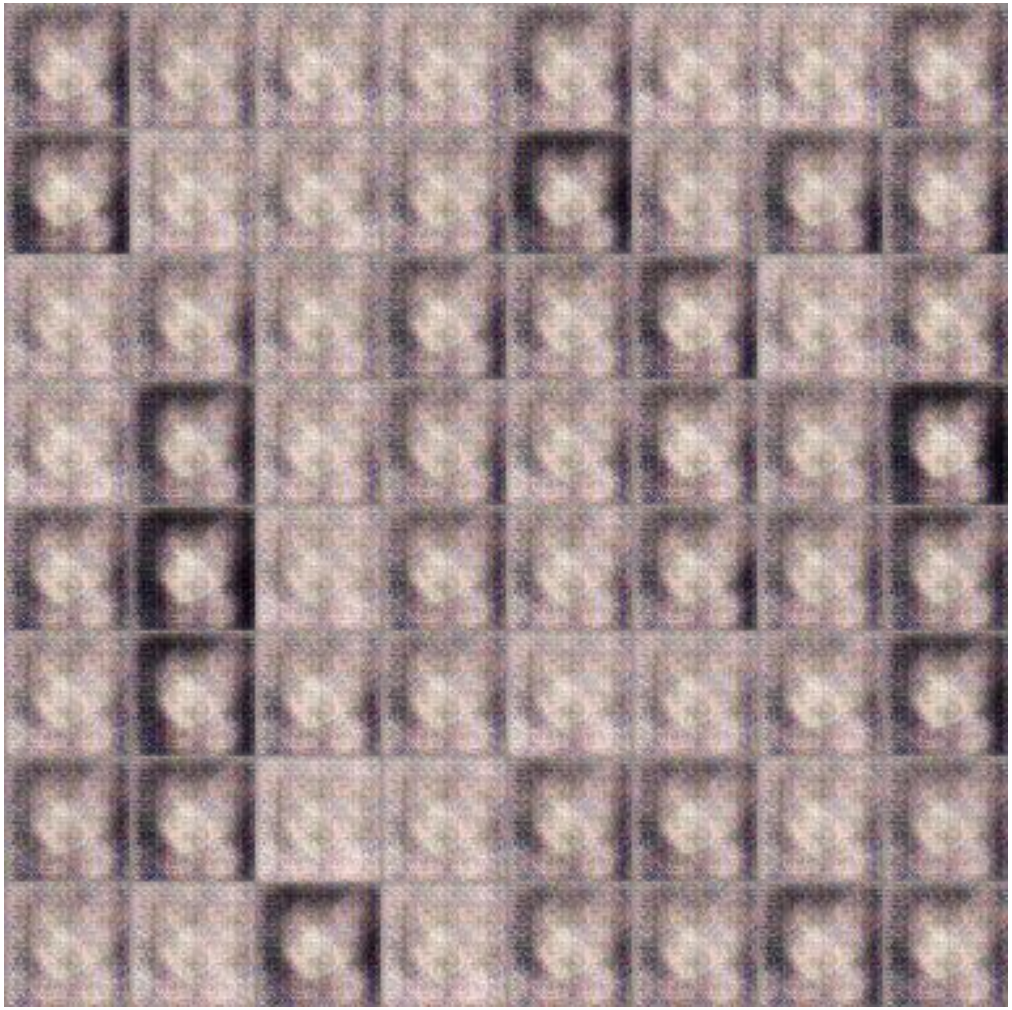
Algorithm

- Initialize generator and discriminator 

- In each training iteration:



Anime Face Generation



100 updates

Source of training data: https://zhuanlan.zhihu.com/p/24767059

Anime Face Generation

1000 updates

Anime Face Generation

2000 updates

Anime Face Generation

5000 updates

Anime Face Generation

10,000 updates

Anime Face Generation

20,000 updates

Anime Face Generation

50,000 updates

The faces generated by machine.

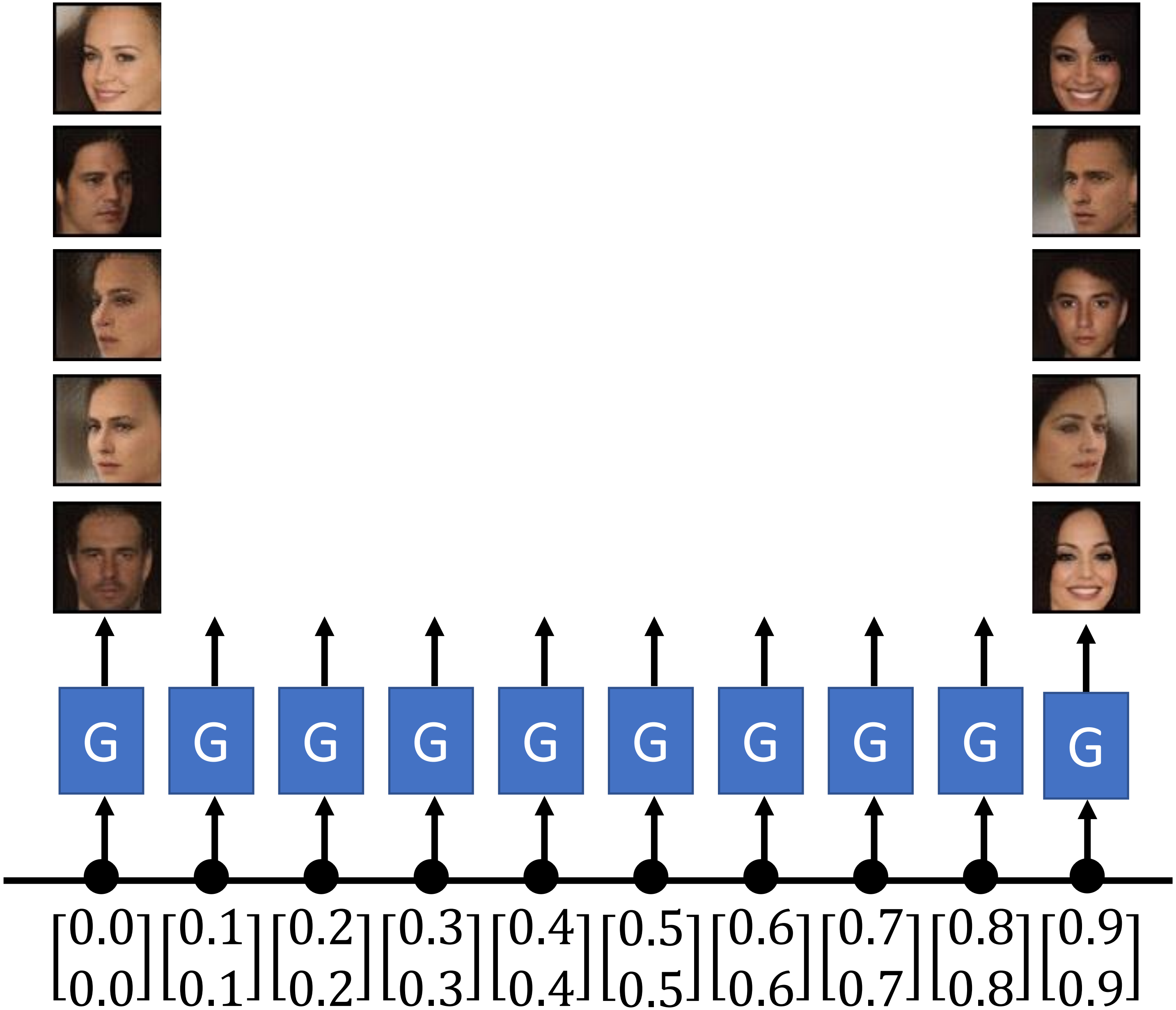
圖片生成：
吳宗翰、謝濬丞、
陳延昊、錢柏均

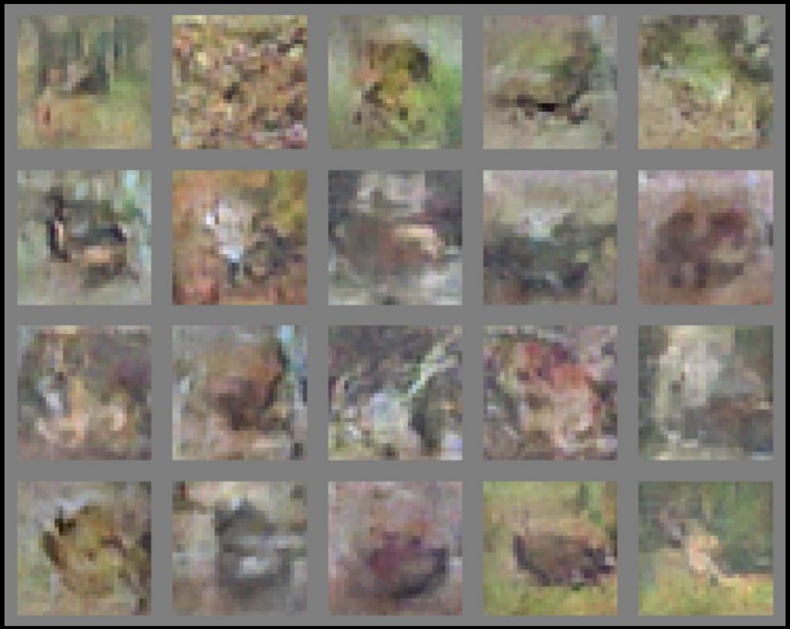
In 2019, with StyleGAN ……

Source of video:
https://www.gwern.net/Faces

Progressive GAN

https://arxiv.org/abs/1710.10196


G G G G G G G G G G
$\begin{bmatrix}0.0\\0.0\end{bmatrix}$ $\begin{bmatrix}0.1\\0.1\end{bmatrix}$ $\begin{bmatrix}0.2\\0.2\end{bmatrix}$ $\begin{bmatrix}0.3\\0.3\end{bmatrix}$ $\begin{bmatrix}0.4\\0.4\end{bmatrix}$ $\begin{bmatrix}0.5\\0.5\end{bmatrix}$ $\begin{bmatrix}0.6\\0.6\end{bmatrix}$ $\begin{bmatrix}0.7\\0.7\end{bmatrix}$ $\begin{bmatrix}0.8\\0.8\end{bmatrix}$ $\begin{bmatrix}0.9\\0.9\end{bmatrix}$



The first GAN

https://arxiv.org/abs/1406.2661 (Ian J. Goodfellow)

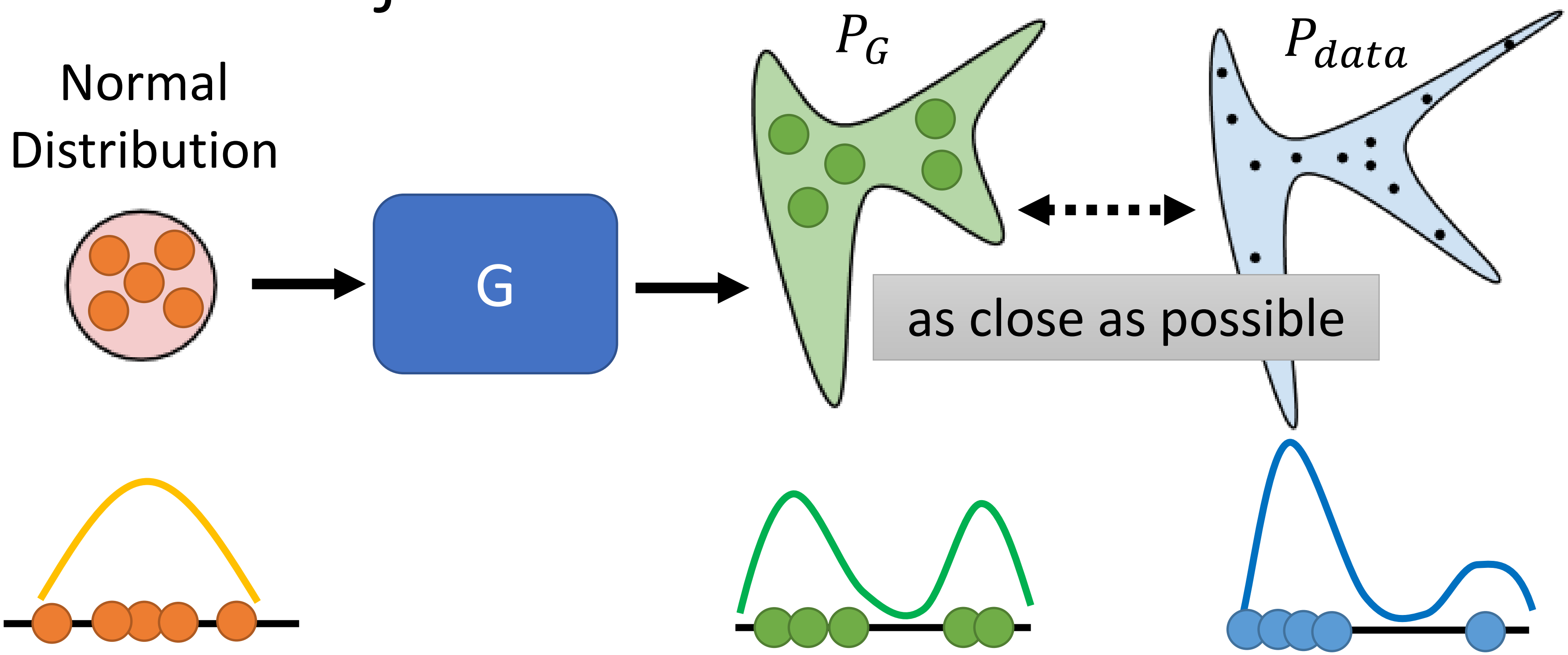
Today …… BigGAN

https://arxiv.org/abs/1809.11096

Theory behind GAN

c.f. $w^*, b^* = arg \min_{w,b} L$

Our Objective



$$G^* = arg \min_G \underline{Div(P_G, P_{data})}$$

Divergence between distributions P_G and P_{data}

How to compute the divergence?

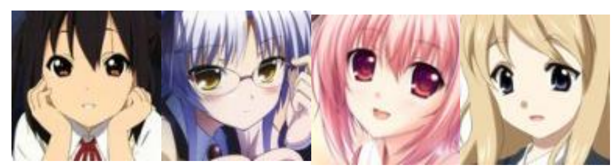
Sampling is good enough ……

$$G^* = arg\min_G Div(P_G, P_{data})$$

Although we do not know the distributions of P_G and P_{data}, we can sample from them.

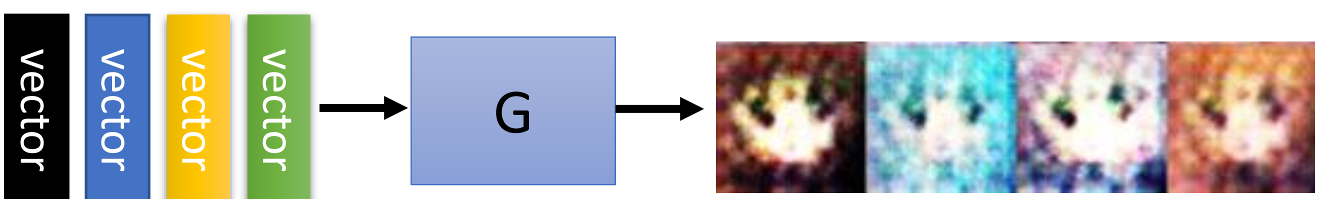
Database

sample



Sampling from P_{data}

sample from normal



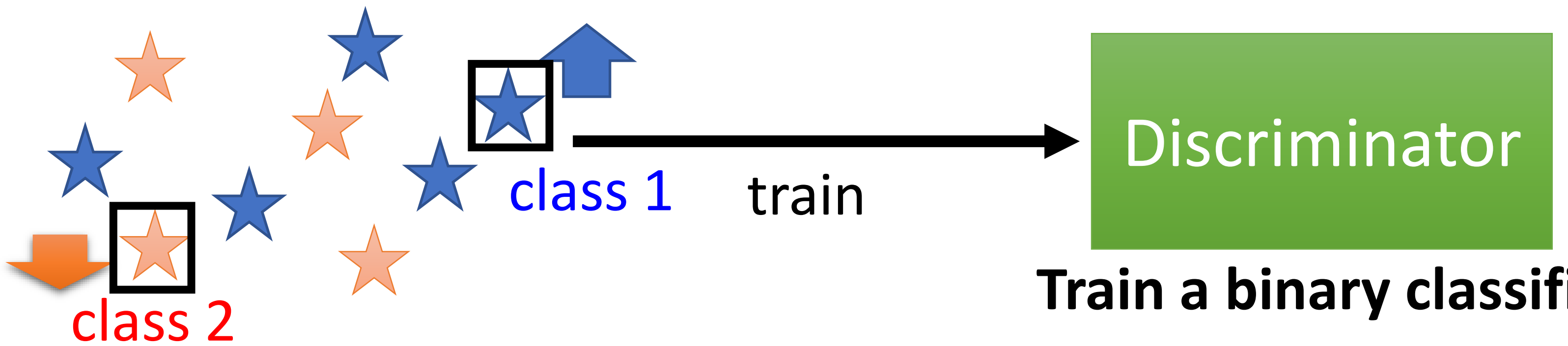
Sampling from P_G

https://arxiv.org/abs/1406.2661

Discriminator

$$G^* = arg \min_G Div(P_G, P_{data})$$

★ : data sampled from P_{data} ★ : data sampled from P_G



Train a binary classifier

Training: $D^* = arg \boxed{\max_D V(D, G)}$

The value is related to JS divergence.

Objective Function for D

$$V(G, D) = E_{y \sim P_{data}}[log D(y)] + E_{y \sim P_G}[log(1 - D(y))]$$

$D^* = arg \max_D \boxed{V(D, G)}$ = Training classifier: minimize cross entropy

negative cross entropy

Discriminator

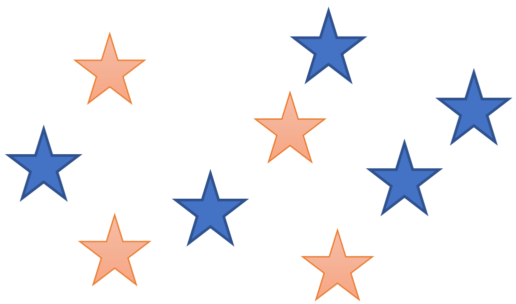
$$G^* = arg \min_G Div(P_G, P_{data})$$

★ : data sampled from P_{data}

☆ : data sampled from P_G

Training:

$$D^* = arg \boxed{\max_D V(D, G)}$$

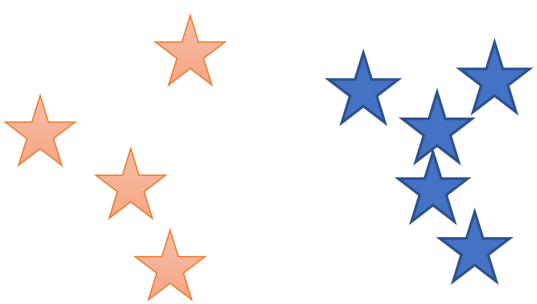


train → Discriminator

small divergence

hard to discriminate

Small $\boxed{\max_D V(D, G)}$



train → Discriminator

large divergence

easy to discriminate

$$G^* = arg \min_G \boxed{\max_D V(G, D)}$$

$$D^* = arg \boxed{\max_D V(D, G)}$$

The maximum objective value is related to JS divergence.

- Initialize generator and discriminator
- In each training iteration:

  Step 1: Fix generator G, and update discriminator D

  Step 2: Fix discriminator D, and update generator G

Can we use other divergence?

Name	$D_f(P\|Q)$	Generator $f(u)$
Total variation	$\frac{1}{2}\int \lvert p(x)-q(x)\rvert \,\mathrm{d}x$	$\frac{1}{2}\lvert u-1\rvert$
Kullback-Leibler	$\int p(x)\log\frac{p(x)}{q(x)}\,\mathrm{d}x$	$u\log u$
Reverse Kullback-Leibler	$\int q(x)\log\frac{q(x)}{p(x)}\,\mathrm{d}x$	$-\log u$
Pearson χ^2	$\int \frac{(q(x)-p(x))^2}{p(x)}\,\mathrm{d}x$	$(u-1)^2$
Neyman χ^2	$\int \frac{(p(x)-q(x))^2}{q(x)}\,\mathrm{d}x$	$\frac{(1-u)^2}{u}$
Squared Hellinger	$\int \left(\sqrt{p(x)}-\sqrt{q(x)}\right)^2\,\mathrm{d}x$	$(\sqrt{u}-1)^2$
Jeffrey	$\int (p(x)-q(x))\log\left(\frac{p(x)}{q(x)}\right)\,\mathrm{d}x$	$(u-1)\log u$
Jensen-Shannon	$\frac{1}{2}\int p(x)\log\frac{2p(x)}{p(x)+q(x)}+q(x)\log\frac{2q(x)}{p(x)+q(x)}\,\mathrm{d}x$	$-(u+1)\log\frac{1+u}{2}+u\log u$
Jensen-Shannon-weighted	$\int p(x)\pi\log\frac{p(x)}{\pi p(x)+(1-\pi)q(x)}+(1-\pi)q(x)\log\frac{q(x)}{\pi p(x)+(1-\pi)q(x)}\,\mathrm{d}x$	$\pi u\log u-(1-\pi+\pi u)\log(1-\pi+\pi u)$
GAN	$\int p(x)\log\frac{2p(x)}{p(x)+q(x)}+q(x)\log\frac{2q(x)}{p(x)+q(x)}\,\mathrm{d}x-\log(4)$	$u\log u-(u+1)\log(u+1)$

Name	Conjugate $f^*(t)$
Total variation	t
Kullback-Leibler (KL)	$\exp(t-1)$
Reverse KL	$-1-\log(-t)$
Pearson χ^2	$\frac{1}{4}t^2+t$
Neyman χ^2	$2-2\sqrt{1-t}$
Squared Hellinger	$\frac{t}{1-t}$
Jeffrey	$W(e^{1-t})+\frac{1}{W(e^{1-t})}+t-2$
Jensen-Shannon	$-\log(2-\exp(t))$
Jensen-Shannon-weighted	$(1-\pi)\log\frac{1-\pi}{1-\pi e^{t/\pi}}$
GAN	$-\log(1-\exp(t))$

Using the divergence you like ☺

https://arxiv.org/abs/1606.00709

GAN is difficult to train ……

(I found this joke from 陳柏文's facebook.)

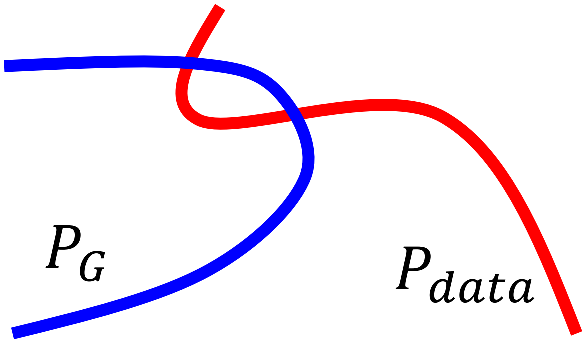
Tips for GAN

JS divergence is not suitable

- In most cases, P_G and P_{data} are not overlapped.
- 1. The nature of data

  Both P_{data} and P_G are low-dim manifold in high-dim space.

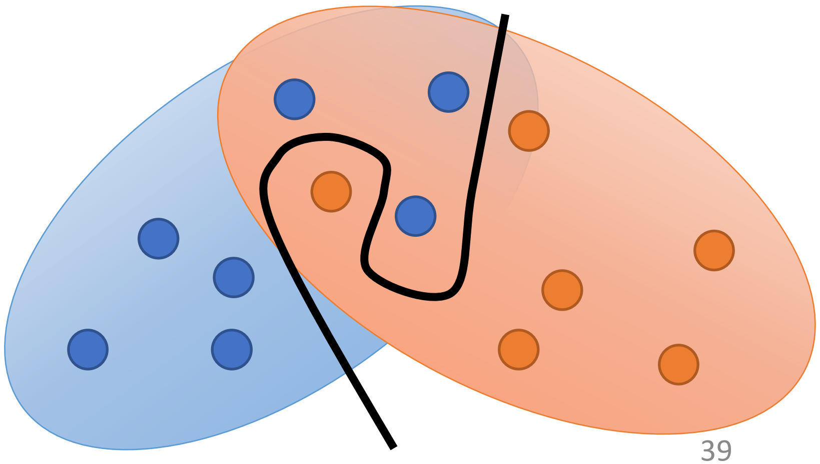
  The overlap can be ignored.



- 2. Sampling

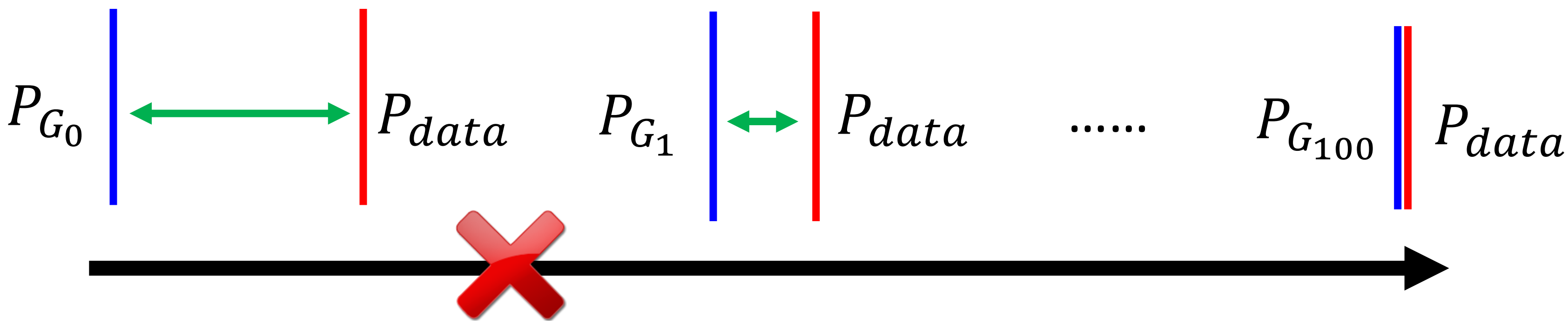
  Even though P_{data} and P_G have overlap.

  If you do not have enough sampling ……



What is the problem of JS divergence?

JS divergence is always log2 if two distributions do not overlap.



Equally bad

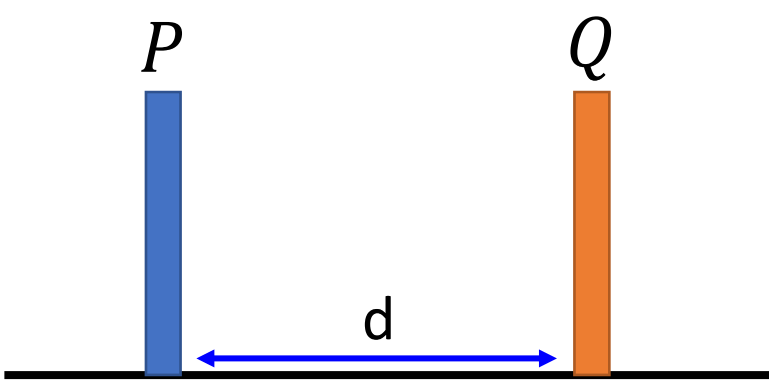
$JS(P_{G_0}, P_{data}) = log2$ $\quad$ $JS(P_{G_1}, P_{data}) = log2$ $\quad$ …… $\quad$ $JS(P_{G_{100}}, P_{data}) = 0$

Intuition: If two distributions do not overlap, binary classifier achieves 100% accuracy.

Its accuracy (or loss) means nothing during GAN training.

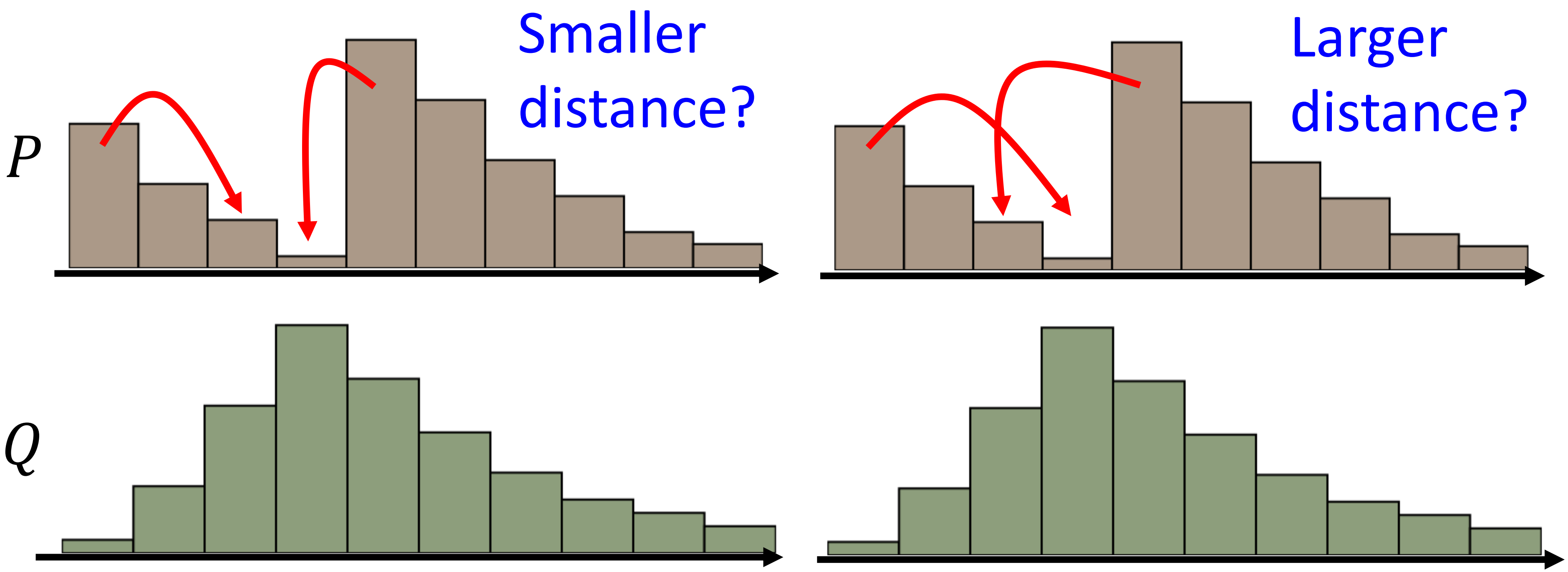
Wasserstein distance

- Considering one distribution P as a pile of earth, and another distribution Q as the target
- The average distance the earth mover has to move the earth.



$$W(P,Q) = d$$

Wasserstein distance

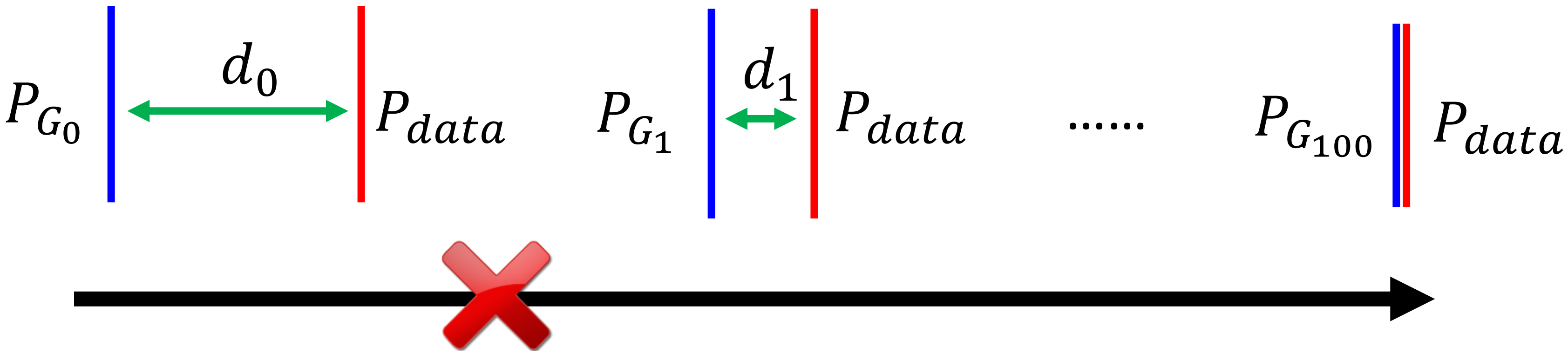


There are many possible "moving plans".

Using the "moving plan" with the smallest average distance to define the Wasserstein distance.

Source of image: https://vincentherrmann.github.io/blog/wasserstein/

What is the problem of JS divergence?

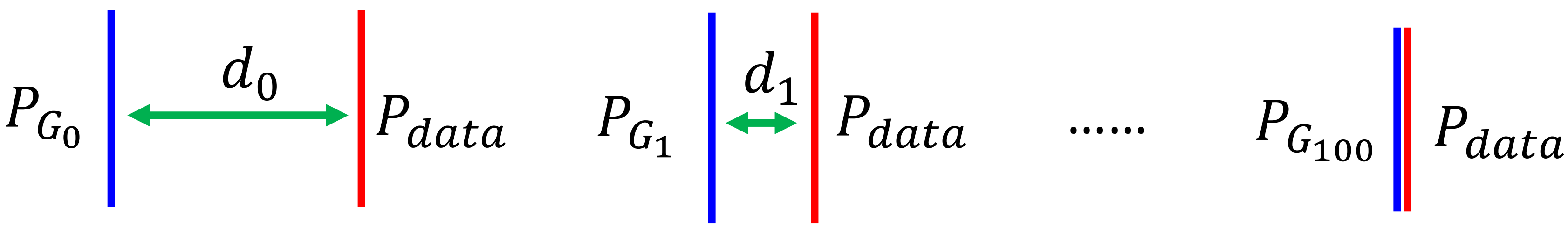


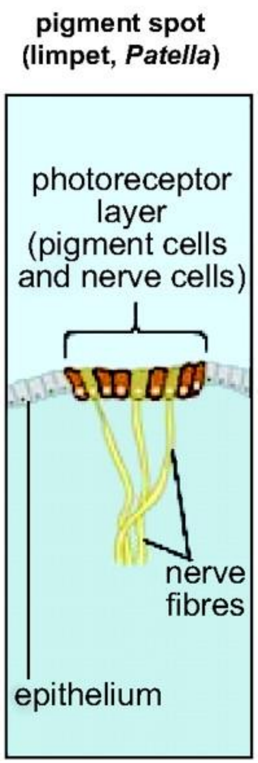
$JS(P_{G_0}, P_{data}) = log2$ $\quad$ $JS(P_{G_1}, P_{data}) = log2$ $\quad$ …… $\quad$ $JS(P_{G_{100}}, P_{data}) = 0$

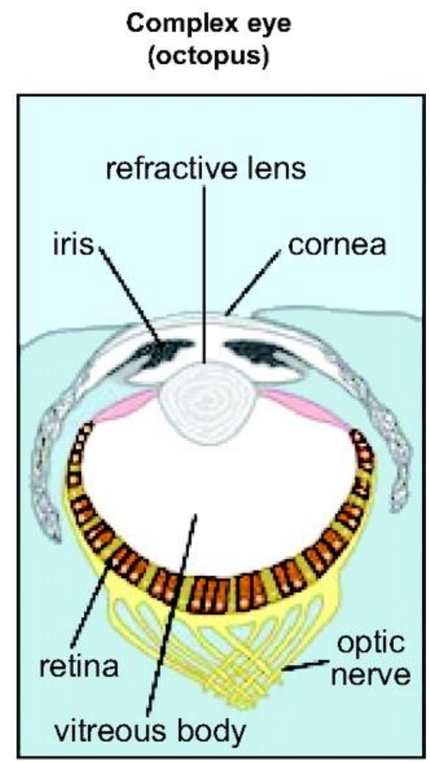
$W(P_{G_0}, P_{data}) = d_0$ $\quad$ $W(P_{G_1}, P_{data}) = d_1$ $\quad$ …… $\quad$ $W(P_{G_{100}}, P_{data}) = 0$

Better!

What is the problem of JS divergence?







https://www.pnas.org/content/104/suppl_1/8567.figures-only

https://arxiv.org/abs/1701.07875

WGAN

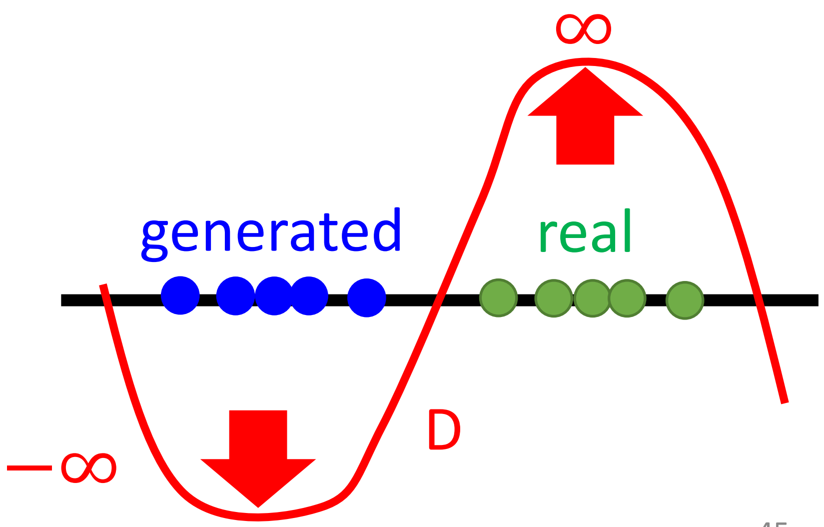
Evaluate Wasserstein distance between P_{data} and P_G

$$\max_{D \in 1-Lipschitz} \{E_{x \sim P_{data}}[D(x)] - E_{x \sim P_G}[D(x)]\}$$

D has to be smooth enough. How to fulfill this constraint?

Without the constraint, the training of D will not converge.

Keeping the D smooth forces D(x) become ∞ and $-\infty$



$$\max_{\boxed{D \in 1-Lipschitz}} \{E_{x \sim P_{data}}[D(x)] - E_{x \sim P_G}[D(x)]\}$$

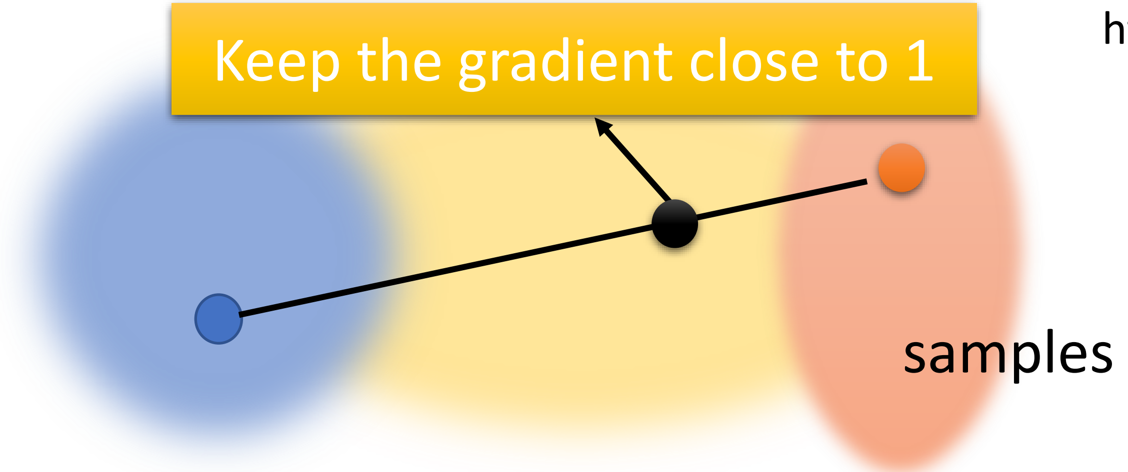
- Original WGAN → Weight

  Force the parameters w between c and -c

  After parameter update, if w > c, w = c; if w < -c, w = -c

- Improved WGAN → Gradient Penalty

https://arxiv.org/abs/1704.00028



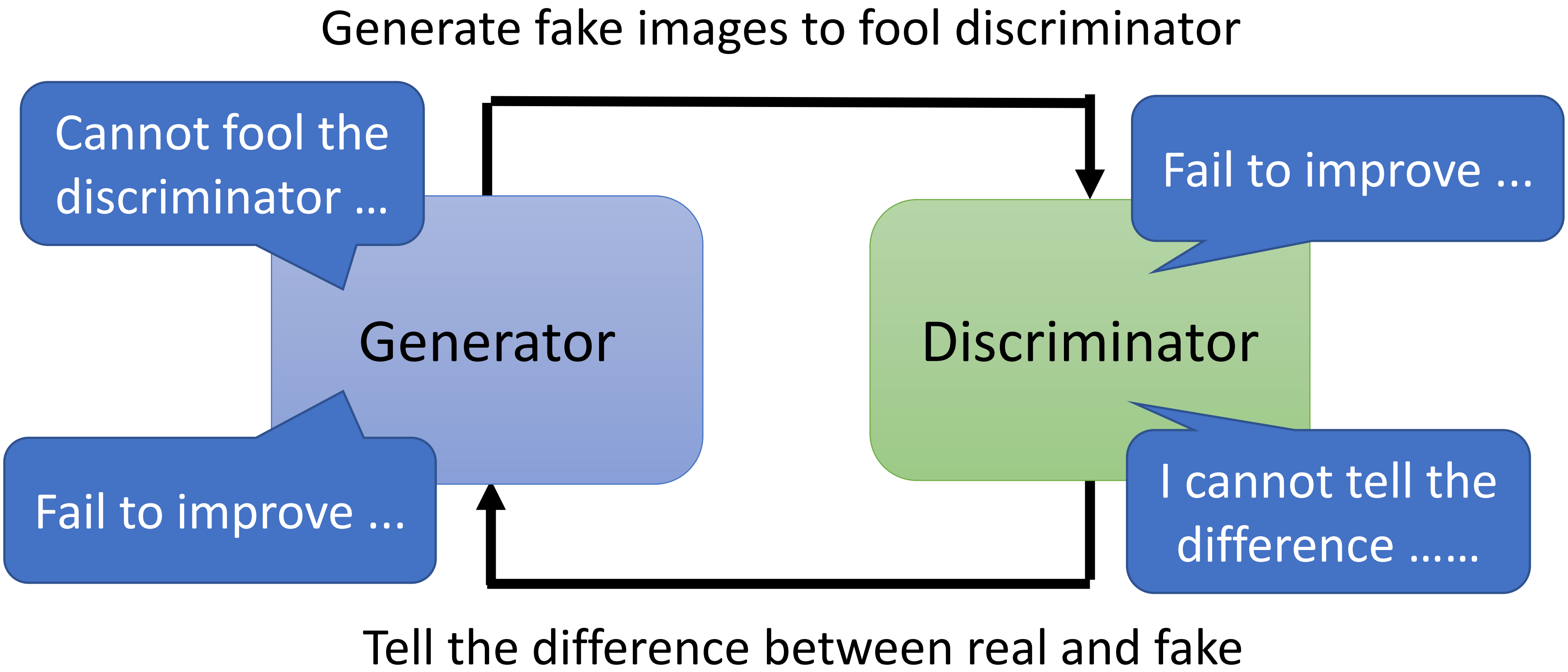
- Spectral Normalization → Keep gradient norm smaller than 1 everywhere

https://arxiv.org/abs/1802.05957

GAN is still challenging …

- Generator and Discriminator needs to match each other (棋逢敵手)

GAN is still challenging …

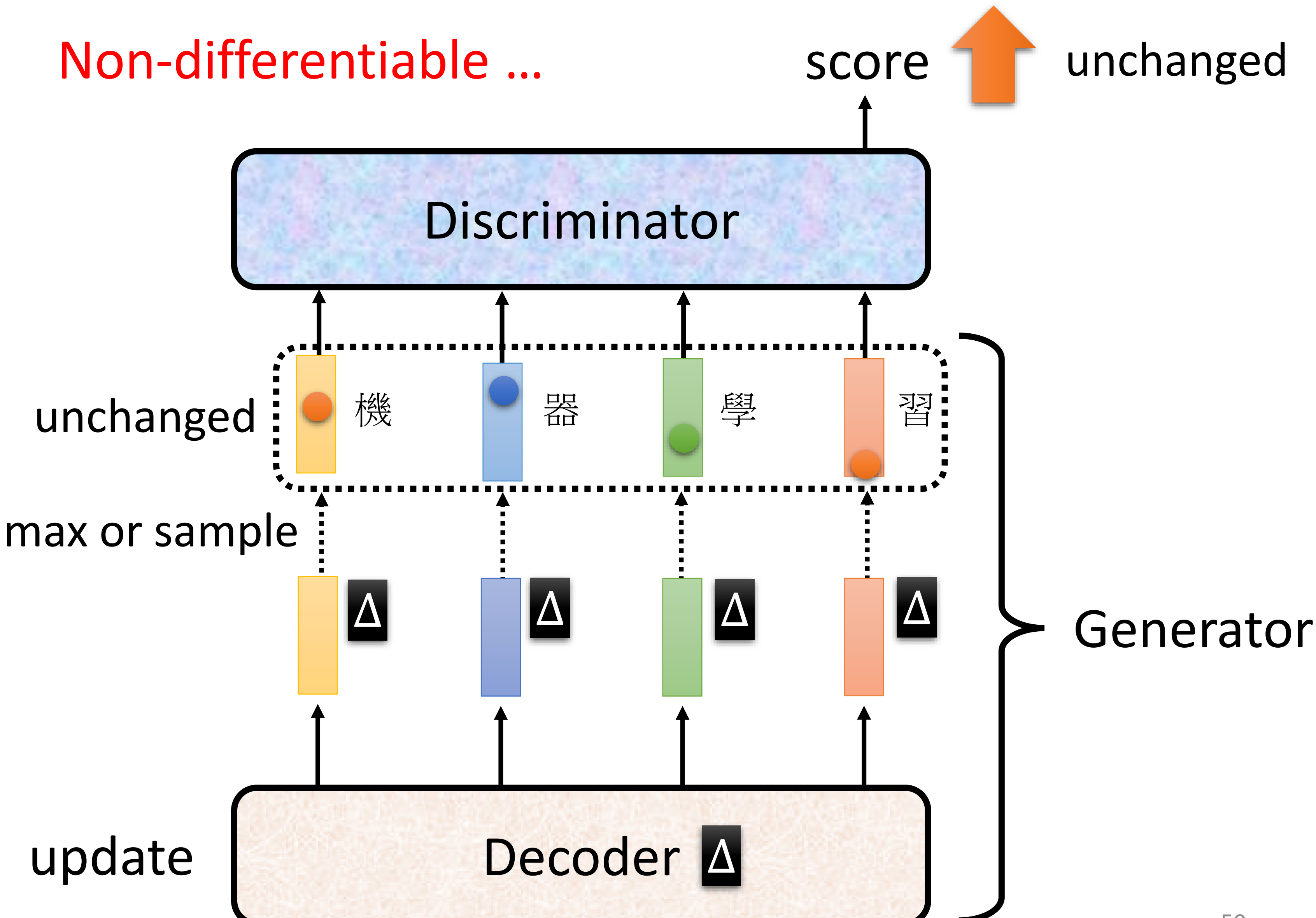


More Tips

- Tips from Soumith
  - https://github.com/soumith/ganhacks
- Tips in DCGAN: Guideline for network architecture design for image generation
  - https://arxiv.org/abs/1511.06434
- Improved techniques for training GANs
  - https://arxiv.org/abs/1606.03498
- Tips from BigGAN
  - https://arxiv.org/abs/1809.11096

GAN for Sequence Generation

Non-differentiable ...

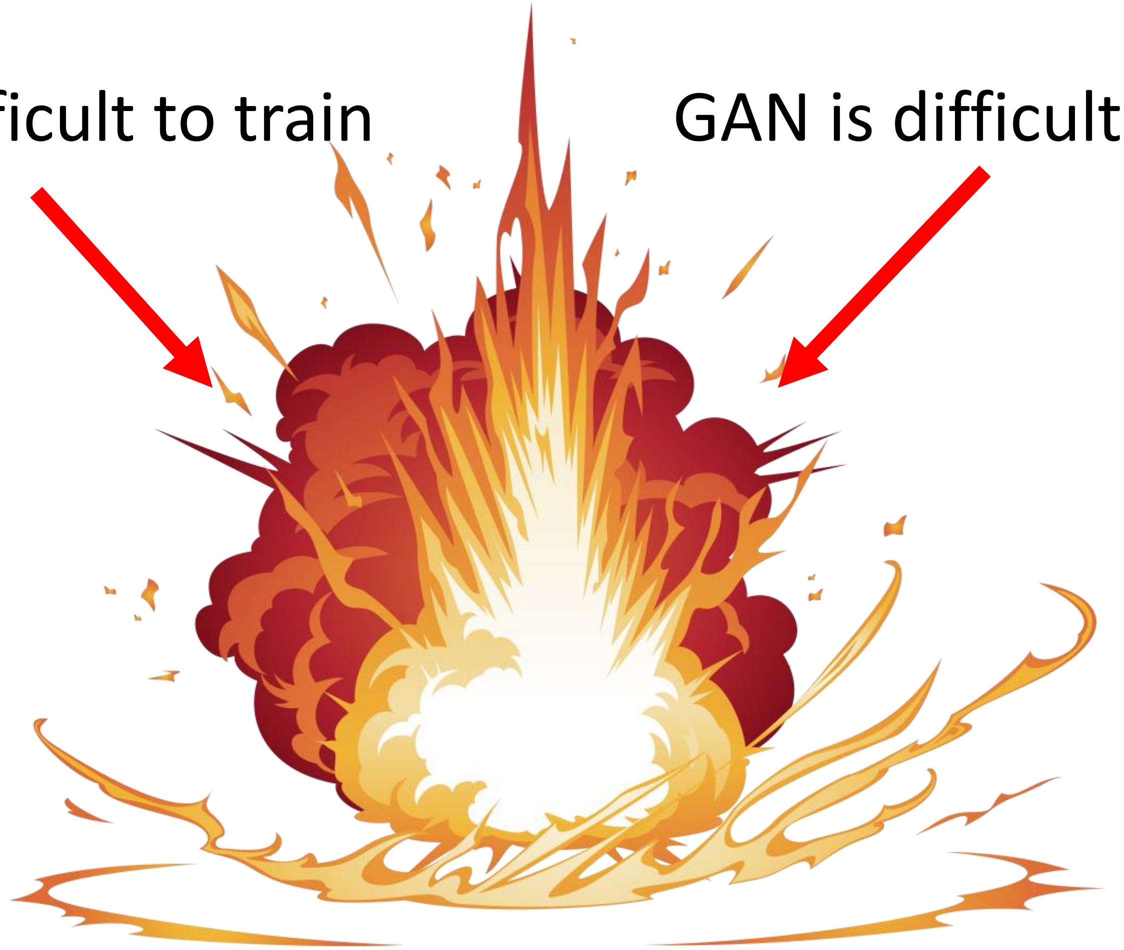


GAN for Sequence Generation

Reinforcement learning (RL) is involved ……

RL is difficult to train

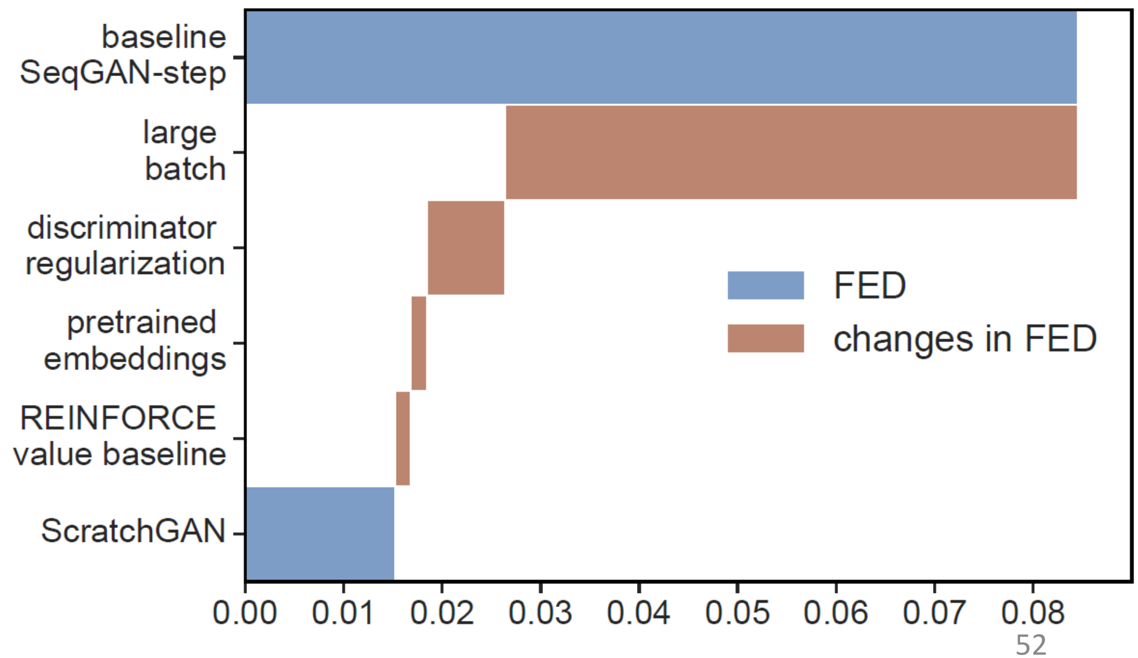
GAN is difficult to train



Sequence Generation GAN (RL+GAN)

GAN for Sequence Generation

- Usually, the generator are fine-tuned from a model learned by other approaches.
- However, with enough hyperparameter-tuning and tips, ScarchGAN can train from scratch.



Training language GANs from Scratch

https://arxiv.org/abs/1905.09922

Generative Models

- This lecture: Generative Adversarial Network (GAN)

Full version

https://www.youtube.com/playlist?list=PLJV_el3uVTsMq
6JEFPW35BCiOQTsoqwNw

More Generative Models

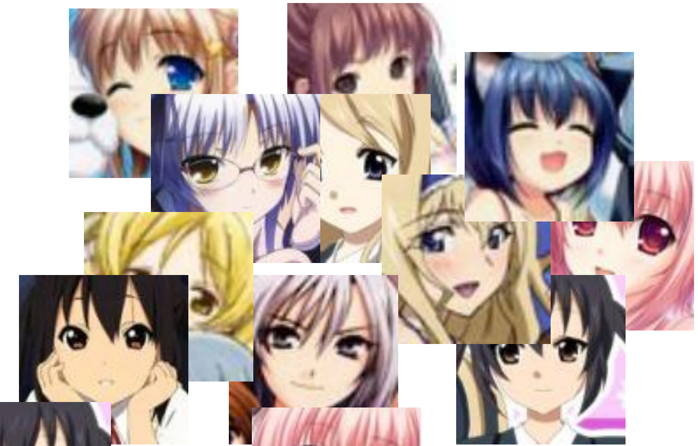
Variational Autoencoder (VAE)

https://youtu.be/8zomhgKrsmQ

FLOW-based Model

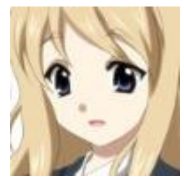
https://youtu.be/uXY18nzdSsM

Possible Solution?



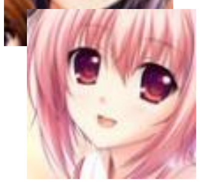
$\begin{bmatrix} 0.3 \\ -0.1 \\ \vdots \\ -0.7 \end{bmatrix}$

$\begin{bmatrix} 0.1 \\ -0.1 \\ \vdots \\ 0.7 \end{bmatrix}$



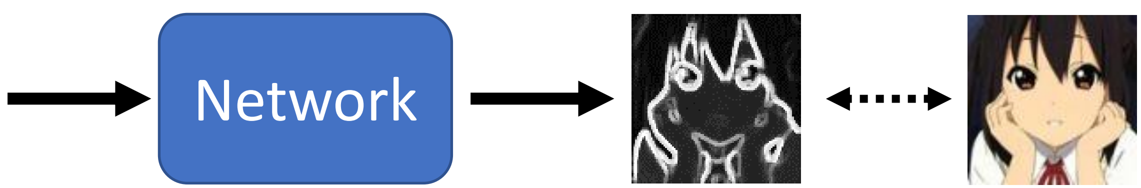
$\begin{bmatrix} -0.3 \\ 0.1 \\ \vdots \\ 0.9 \end{bmatrix}$

$\begin{bmatrix} 0.7 \\ 0.1 \\ \vdots \\ -0.9 \end{bmatrix}$



$\begin{bmatrix} -0.1 \\ 0.8 \\ \vdots \\ 0.8 \end{bmatrix}$

$\begin{bmatrix} 0.3 \\ -0.1 \\ \vdots \\ -0.7 \end{bmatrix}$



Using typical learning approaches?

Generative Latent Optimization (GLO), https://arxiv.org/abs/1707.05776

Gradient Origin Networks, https://arxiv.org/abs/2007.02798

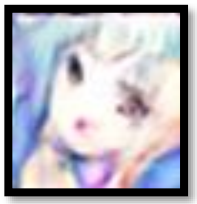
Evaluation of Generation

[Tim Salimans, et al., NIPS, 2016]

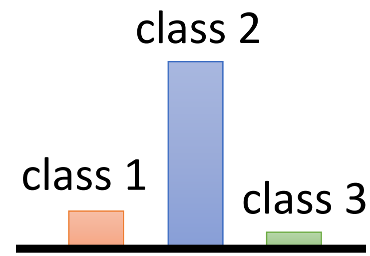
Inception Score

x: image

y: class (output of CNN)

x  → Off-the-shelf Image Classifier → $P(y|x)$

e.g. Inception net, VGG, etc.

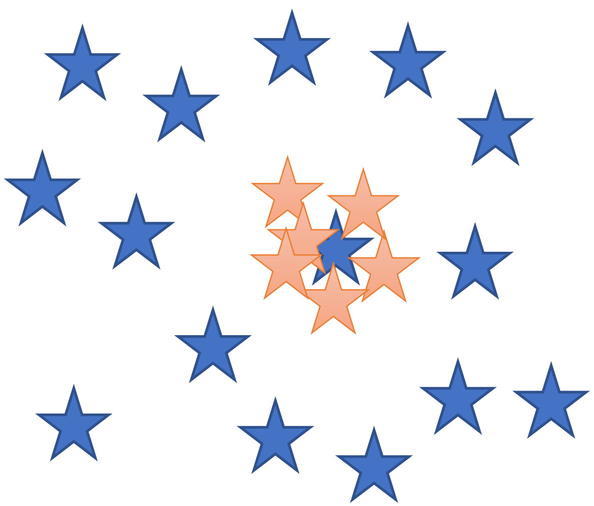


Concentrated distribution means higher visual quality

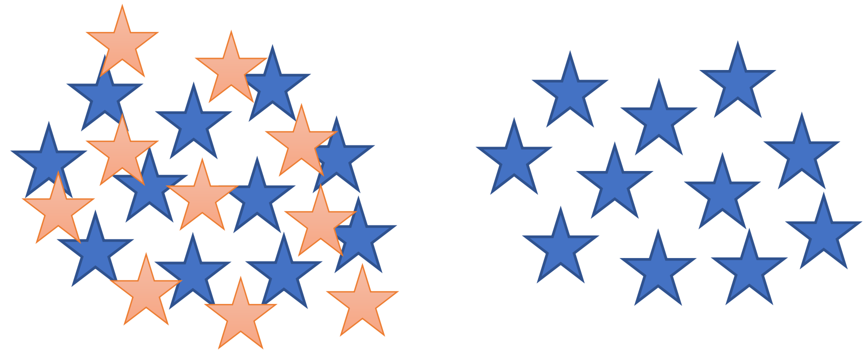
Mode Collapse

: real data

: generated data



Mode Dropping



Generator switches mode during training

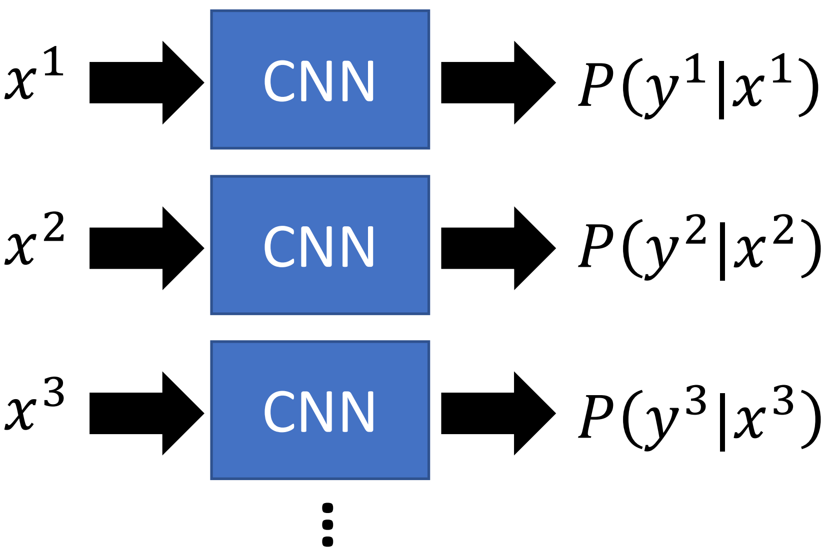
Generator
at iteration t

Generator
at iteration t+1

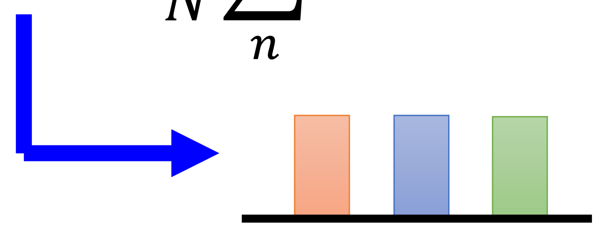
Generator
at iteration t+2

Mode
easy t

BEGAN on CelebA



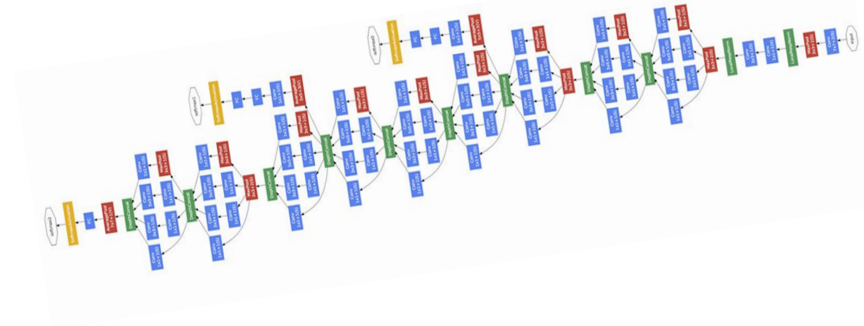
$$P(y) = \frac{1}{N}\sum_n P(y^n|x^n)$$



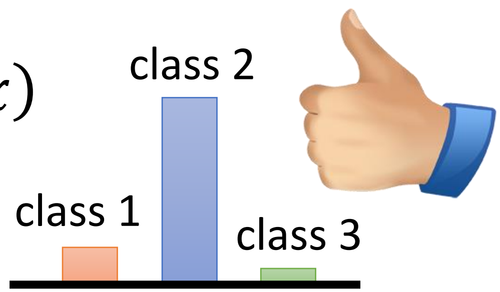
Uniform distribution means higher variety

60

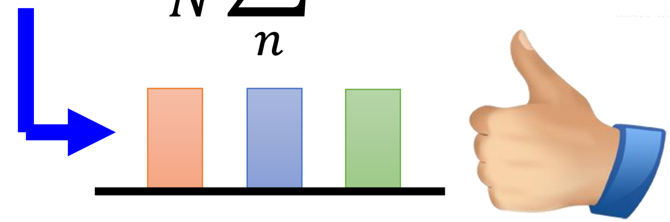
Inception Score



$P(y|x)$



$$P(y) = \frac{1}{N}\sum_{n} P(y^n|x^n)$$



Inception Score [Tim Salimans, et al., NIPS 2016]

$$= \sum_{x}\sum_{y} P(y|x)logP(y|x)$$

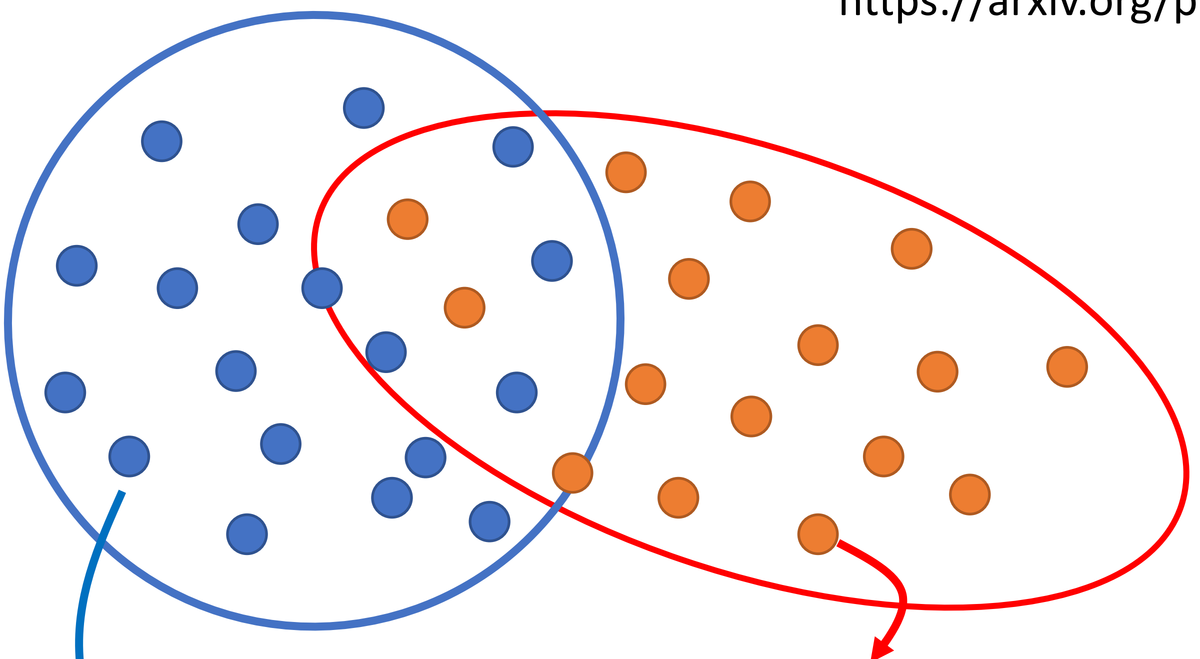
Negative entropy of P(y|x)

$$-\sum_{y} P(y)logP(y)$$

Entropy of P(y)

Fréchet Inception Distance (FID)

https://arxiv.org/pdf/1706.08500.pdf

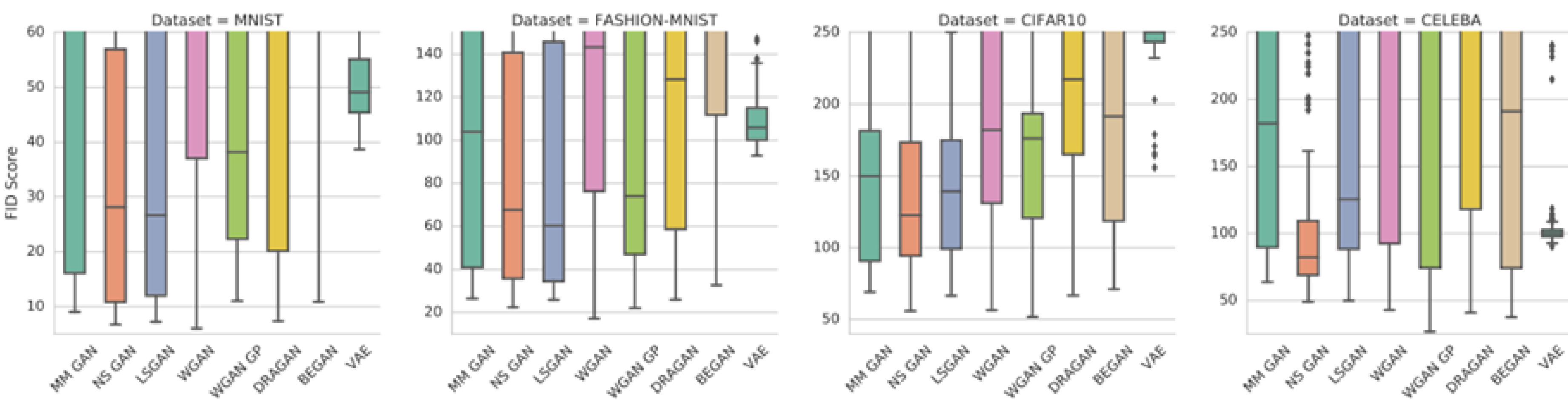


FID =
Fréchet distance
between the two
Gaussians

red points: latent representation of Inception net for the real images

blue points: latent representation of Inception net for the generated images

GAN	Discriminator Loss	Generator Loss
MM GAN	$\mathcal{L}_{\mathrm{D}}^{\mathrm{GAN}} = -\mathbb{E}_{x\sim p_d}[\log(D(x))] + \mathbb{E}_{\hat{x}\sim p_g}[\log(1-D(\hat{x}))]$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{GAN}} = -\mathcal{L}_{\mathrm{D}}^{\mathrm{GAN}}$
NS GAN	$\mathcal{L}_{\mathrm{D}}^{\mathrm{NSGAN}} = \mathcal{L}_{\mathrm{D}}^{\mathrm{GAN}}$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{NSGAN}} = \mathbb{E}_{\hat{x}\sim p_g}[\log(D(\hat{x}))]$
WGAN	$\mathcal{L}_{\mathrm{D}}^{\mathrm{WGAN}} = -\mathbb{E}_{x\sim p_d}[D(x)] + \mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})]$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{WGAN}} - = \mathcal{L}_{\mathrm{D}}^{\mathrm{WGAN}}$
WGAN GP	$\mathcal{L}_{\mathrm{D}}^{\mathrm{WGAN}} = \mathcal{L}_{\mathrm{D}}^{\mathrm{WGAN}} + \lambda\mathbb{E}_{\hat{x}\sim p_g}[(\|\|\nabla D(\alpha x + (1-\alpha\hat{x})\|\|_2 - 1)^2]$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{WGAN}} = -\mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})]$
LS GAN	$\mathcal{L}_{\mathrm{D}}^{\mathrm{LSGAN}} = -\mathbb{E}_{x\sim p_d}[(D(x)-1)^2] + \mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})^2]$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{LSGAN}} = -\mathbb{E}_{\hat{x}\sim p_g}[(D(\hat{x}-1)^2]$
DRAGAN	$\mathcal{L}_{\mathrm{D}}^{\mathrm{DRAGAN}} = \mathcal{L}_{\mathrm{D}}^{\mathrm{GAN}} + \lambda\mathbb{E}_{\hat{x}\sim p_d+\mathcal{N}(0,c)}[(\|\|\nabla D(\hat{x})\|\|_2 - 1)^2]$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{DRAGAN}} = -\mathcal{L}_{\mathrm{D}}^{\mathrm{NS\ GAN}}$
BEGAN	$\mathcal{L}_{\mathrm{D}}^{\mathrm{BEGAN}} = \mathbb{E}_{x\sim p_d}[\|\|x - \mathrm{AE}(x)\|\|_1] - k_t\mathbb{E}_{\hat{x}\sim p_g}[\|\|\hat{x} - \mathrm{AE}(\hat{x})\|\|_1]$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{BEGAN}} = \mathbb{E}_{\hat{x}\sim p_g}[\|\|\hat{x} - \mathrm{AE}(\hat{x})\|\|_1]$



FIT: Smaller is better

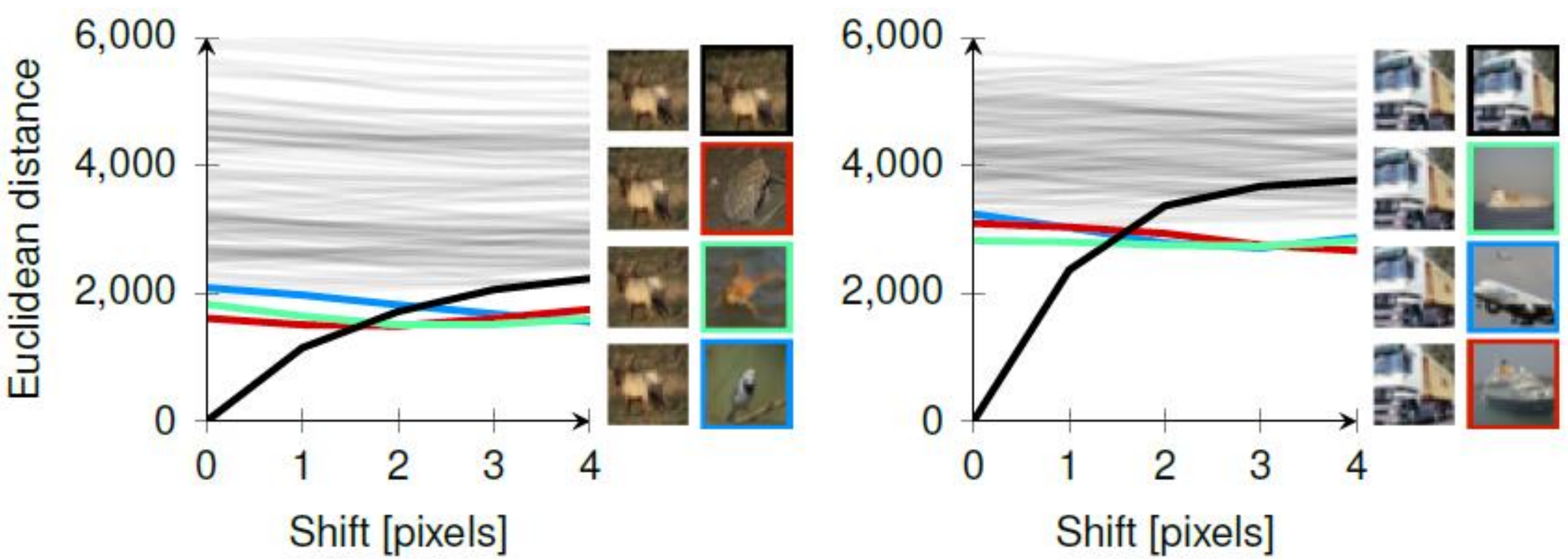
Are GANs Created Equal? A Large-Scale Study

https://arxiv.org/abs/1711.10337

https://arxiv.org/pdf/1511.01844.pdf

We don't want memory GAN.

- Using k-nearest neighbor to check whether the generator generates new objects



To learn more about evaluation …

	Measure	Description
Quantitative	1. Average Log-likelihood [18, 22]	• Log likelihood of explaining realworld held out/test data using a density estimated from the generated data (*e.g.* using KDE or Parzen window estimation). $L = \frac{1}{N}\sum_i \log P_{model}(\mathbf{x}_i)$
	2. Coverage Metric [33]	• The probability mass of the true data "covered" by the model distribution $C := P_{data}(dP_{model} > t)$ with t such that $P_{model}(dP_{model} > t) = 0.95$
	3. Inception Score (IS) [3]	• KLD between conditional and marginal label distributions over generated data. $\exp\left(\mathbb{E}_{\mathbf{x}}\left[\mathrm{KL}\left(p(y \mid \mathbf{x}) \parallel p(y)\right)\right]\right)$
	4. Modified Inception Score (m-IS) [34]	• Encourages diversity within images sampled from a particular category. $\exp(\mathbb{E}_{\mathbf{x}_i}[\mathbb{E}_{\mathbf{x}_j}[(\mathrm{KL}(P(y\mid\mathbf{x}_i)\|P(y\mid\mathbf{x}_j))]])$
	5. Mode Score (MS) [35]	• Similar to IS but also takes into account the prior distribution of the labels over real data. $\exp\left(\mathbb{E}_{\mathbf{x}}\left[\mathrm{KL}\left(p(y \mid \mathbf{x}) \parallel p(y^{train})\right)\right] - \mathrm{KL}\left(p(y) \parallel p(y^{train})\right)\right)$
	6. AM Score [36]	• Takes into account the KLD between distributions of training labels vs. predicted labels, as well as the entropy of predictions. $\mathrm{KL}(p(y^{train}) \parallel p(y)) + \mathbb{E}_{\mathbf{x}}[H(y\mid\mathbf{x})]$
	7. Fréchet Inception Distance (FID) [37]	• Wasserstein-2 distance between multi-variate Gaussians fitted to data embedded into a feature space $FID(r, g) = \lVert\mu_r - \mu_g\rVert_2^2 + Tr(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{\frac{1}{2}})$
	8. Maximum Mean Discrepancy (MMD) [38]	• Measures the dissimilarity between two probability distributions P_r and P_g using samples drawn independently from each distribution. $M_k(P_r, P_g) = \mathbb{E}_{\mathbf{x},\mathbf{x}' \sim P_r}[k(\mathbf{x}, \mathbf{x}')] - 2\mathbb{E}_{\mathbf{x}\sim P_r, \boldsymbol{y}\sim P_g}[k(\mathbf{x}, \boldsymbol{y})] + \mathbb{E}_{\boldsymbol{y},\boldsymbol{y}'\sim P_g}[k(\boldsymbol{y}, \boldsymbol{y}')]$
	9. The Wasserstein Critic [39]	• The critic (*e.g.* an NN) is trained to produce high values at real samples and low values at generated samples $\hat{W}(\mathbf{x}_{test}, \mathbf{x}_g) = \frac{1}{N}\sum_{i=1}^{N}\hat{f}(\mathbf{x}_{test}[i]) - \frac{1}{N}\sum_{i=1}^{N}\hat{f}(\mathbf{x}_g[i])$
	10. Birthday Paradox Test [27]	• Measures the support size of a discrete (continuous) distribution by counting the duplicates (near duplicates)
	11. Classifier Two Sample Test (C2ST) [40]	• Answers whether two samples are drawn from the same distribution (*e.g.* by training a binary classifier)
	12. Classification Performance [1, 15]	• An indirect technique for evaluating the quality of unsupervised representations (*e.g.* feature extraction; FCN score). See also the GAN Quality Index (GQI) [41].
	13. Boundary Distortion [42]	• Measures diversity of generated samples and covariate shift using classification methods.
	14. Number of Statistically-Different Bins (NDB) [43]	• Given two sets of samples from the same distribution, the number of samples that fall into a given bin should be the same up to sampling noise
	15. Image Retrieval Performance [44]	• Measures the distributions of distances to the nearest neighbors of some query images (*i.e.* diversity)
	16. Generative Adversarial Metric (GAM) [31]	• Compares two GANs by having them engaged in a battle against each other by swapping discriminators or generators. $p(\mathbf{x}\mid y = 1; M_1^{'})/p(\mathbf{x}\mid y = 1; M_2^{'}) = (p(y = 1\mid\mathbf{x}; D_1)p(\mathbf{x}; G_2))/(p(y = 1\mid\mathbf{x}; D_2)p(\mathbf{x}; G_1))$
	17. Tournament Win Rate and Skill Rating [45]	• Implements a tournament in which a player is either a discriminator that attempts to distinguish between real and fake data or a generator that attempts to fool the discriminators into accepting fake data as real.
	18. Normalized Relative Discriminative Score (NRDS) [32]	• Compares n GANs based on the idea that if the generated samples are closer to real ones, more epochs would be needed to distinguish them from real samples.
	19. Adversarial Accuracy and Divergence [46]	• Adversarial Accuracy. Computes the classification accuracies achieved by the two classifiers, one trained on real data and another on generated data, on a labeled validation set to approximate $P_g(y\mid\mathbf{x})$ and $P_r(y\mid\mathbf{x})$. Adversarial Divergence: Computes $\mathrm{KL}(P_g(y\mid\mathbf{x}), P_r(y\mid\mathbf{x}))$
	20. Geometry Score [47]	• Compares geometrical properties of the underlying data manifold between real and generated data.
	21. Reconstruction Error [48]	• Measures the reconstruction error (*e.g.* L_2 norm) between a test image and its closest generated image by optimizing for z (*i.e.* $min_{\mathbf{z}}\lVert G(\mathbf{z}) - \mathbf{x}^{(test)}\rVert^2$)
	22. Image Quality Measures [49, 50, 51]	• Evaluates the quality of generated images using measures such as SSIM, PSNR, and sharpness difference
	23. Low-level Image Statistics [52, 53]	• Evaluates how similar low-level statistics of generated images are to those of natural scenes in terms of mean power spectrum, distribution of random filter responses, contrast distribution, etc.
	24. Precision, Recall and F_1 score [23]	• These measures are used to quantify the degree of overfitting in GANs, often over toy datasets.
Qualitative	1. Nearest Neighbors	• To detect overfitting, generated samples are shown next to their nearest neighbors in the training set
	2. Rapid Scene Categorization [18]	• In these experiments, participants are asked to distinguish generated samples from real images in a short presentation time (*e.g.* 100 ms); *i.e.* real v.s fake
	3. Preference Judgment [54, 55, 56, 57]	• Participants are asked to rank models in terms of the fidelity of their generated images (*e.g.* pairs, triples)
	4. Mode Drop and Collapse [58, 59]	• Over datasets with known modes (*e.g.* a GMM or a labeled dataset), modes are computed as by measuring the distances of generated data to mode centers
	5. Network Internals [1, 60, 61, 62, 63, 64]	• Regards exploring and illustrating the internal representation and dynamics of models (*e.g.* space continuity) as well as visualizing learned features

Pros and cons of GAN evaluation measures

https://arxiv.org/abs/1802.03446

Concluding Remarks

- Introduction of Generative Models
- Generative Adversarial Network (GAN)
- Theory behind GAN
- Tips for GAN
- Evaluation of Generative Models


Q&A