Lecture 9 Generative Models
Generative Adversarial Network

Tian Sheuan Chang

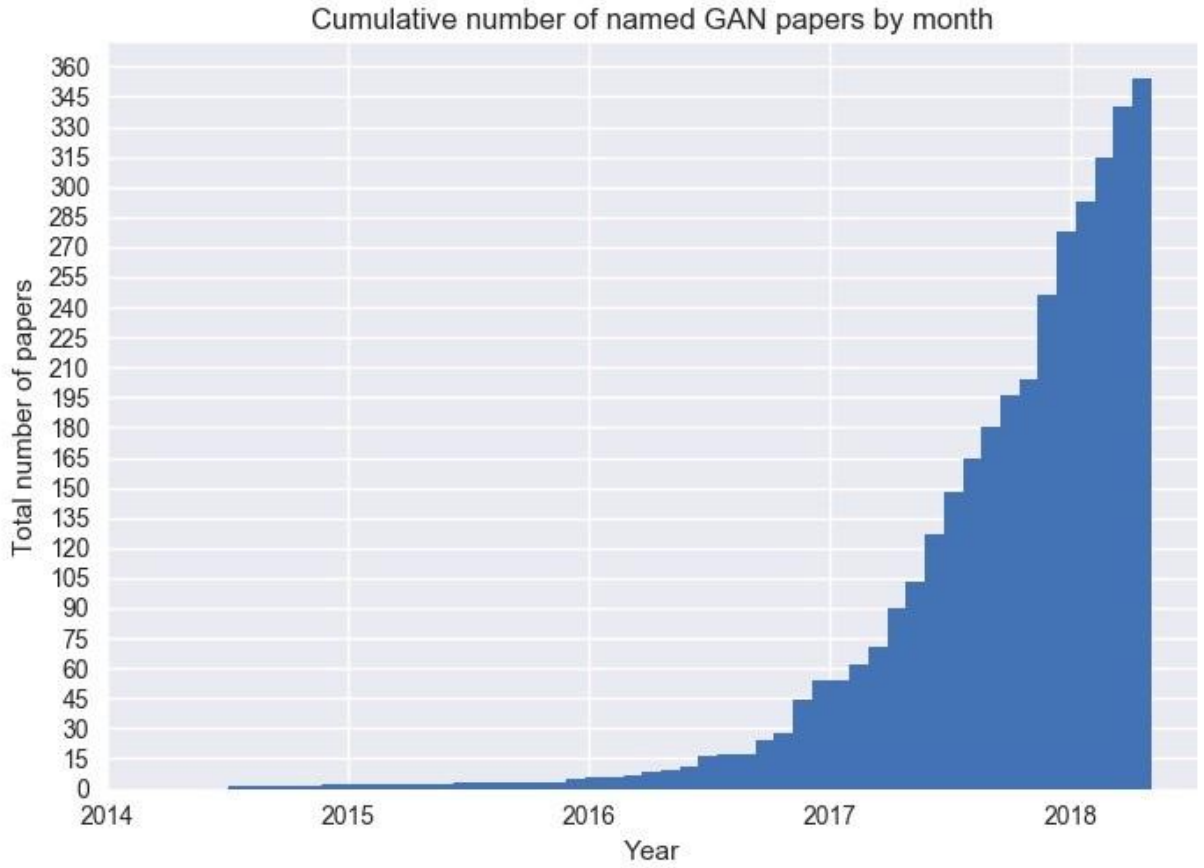
沒有什麼問題是不能用一個神經網路來解決的。
如果有，就用兩個。

Outline

- Fast progress of GAN
- GAN introduction
- 生成對抗網路的進展
  - Architecture variants: DCGAN, SA-GAN, BigGAN,
  - Latent space: cGAN and related applications
- 如何更有效的訓練GAN
  - Vanished gradient, mode collapse
  - WGAN, WGAN-GP, spectral gradient

FAST PROGRESS OF GAN

Track updates at the GAN Zoo

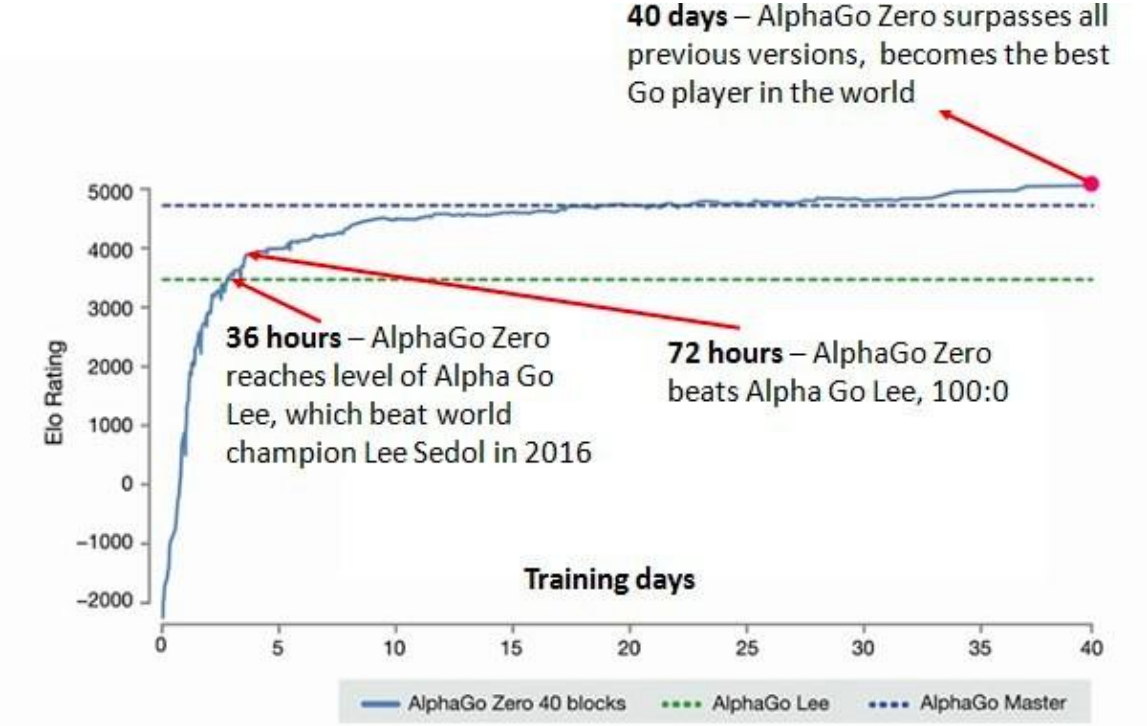


https://github.com/hindupuravinash/the-gan-zoo

(Goodfellow 2018)

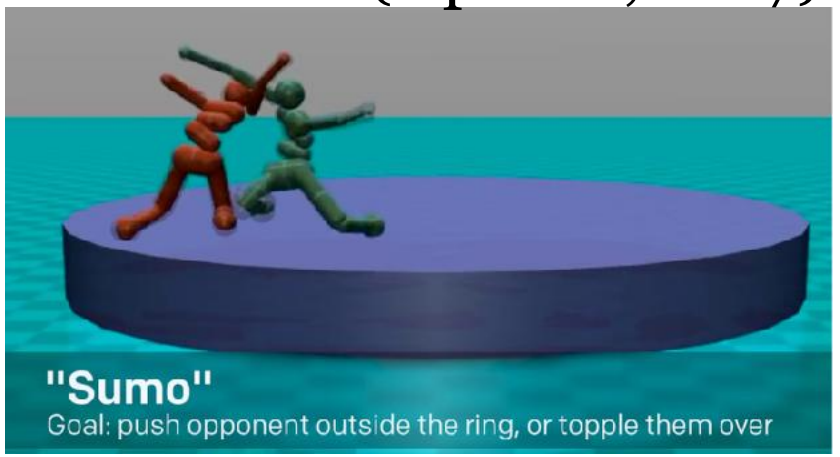
Self-Play

1959: Arthur Samuel's checkers agent



(Silver et al, 2017)

(OpenAI, 2017)



(Bansal et al, 2017)

(Goodfellow 2018)

3.5 Years of Progress on Faces

2014

2015

2016

2017

(Brundage et al, 2018)

<2 Years of Progress on ImageNet

Odena et al 2016

Miyato et al 2017

Zhang et al 2018

(Goodfellow 2018)

Self-Attention GAN

- State of the art FID on ImageNet: 1000 categories, 128x128 pixels

Goldfish

Redshank

Broccoli

Tiger Cat

Geyser

Indigo Bunting

Stone Wall

Saint Bernard

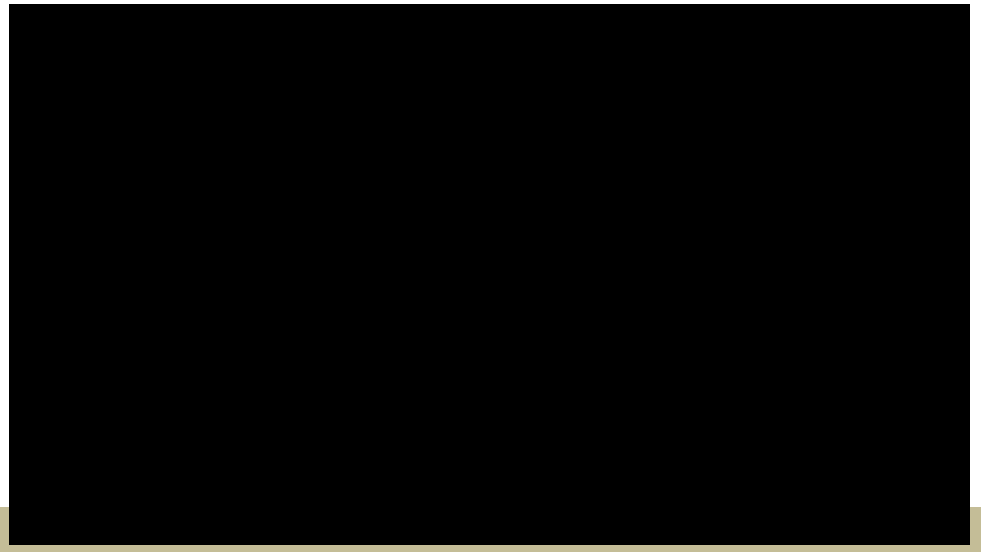
(Zhang et al., 2018)

BigGAN


"GANs are the most interesting idea in the last 10 years in ML"
- Yann LeCun

Demo:「有圖有真相」？

- Realistic image generation



Pix2pix 隨手畫畫變街景

Pix2pix 把衛星圖變地圖

input

output

INTRODUCTION TO GENERATIVE ADVERSARIAL NETWORK

分類

f()=5

f()=4

f()=1

生成模型

f(5) = 

f(4) = 

f(1) = 

Generative Modeling: Sample Generation

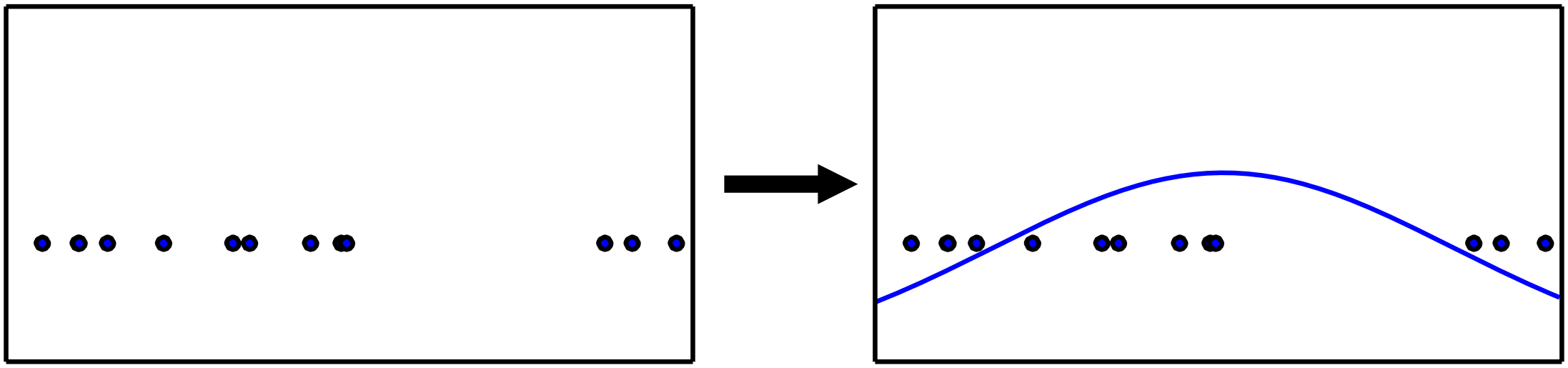
Training Data
(CelebA)

Sample Generator
(Karras et al, 2017)

希望產生的結果很像真的資料
(但是不能只是記住原來訓練資料)

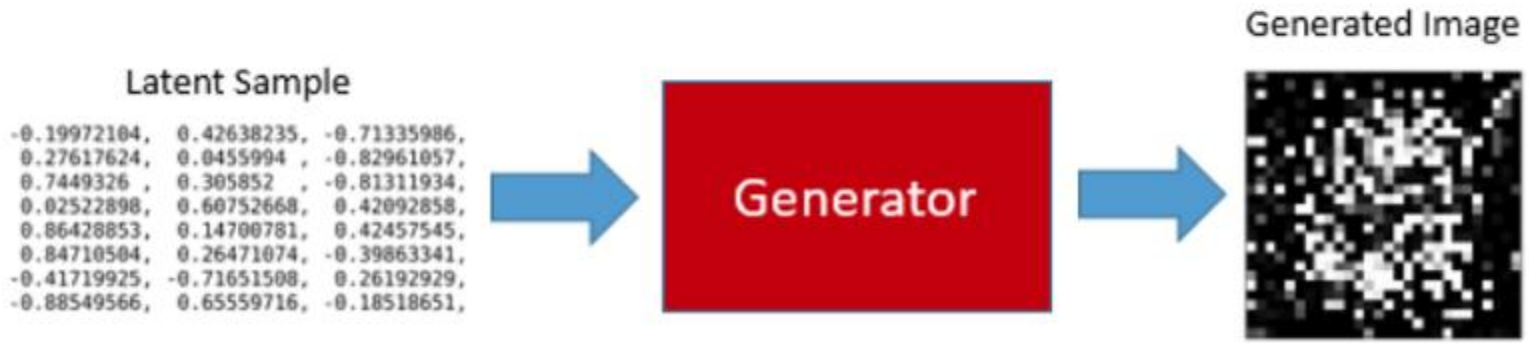
(Goodfellow 2018)

Generative Modeling: Density Estimation



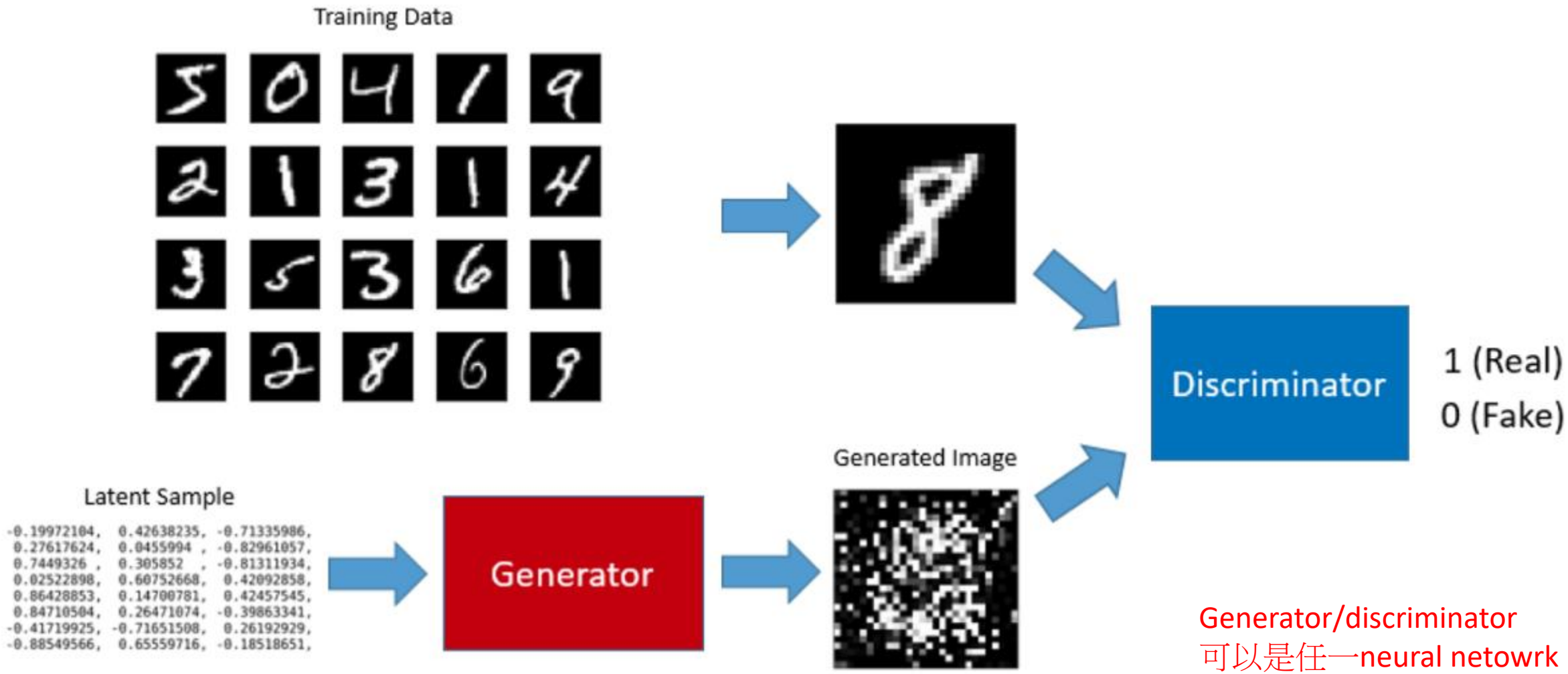
- 生成式模型
  - 通過觀測資料，學到真的樣本的機率分佈，
  - 根據學到的機率分布就可以產生很像但不一樣的資料，怎麼學 => Neural network

Generative Adversarial Network (GAN)



Generator/discriminator
可以是任一neural netowrk

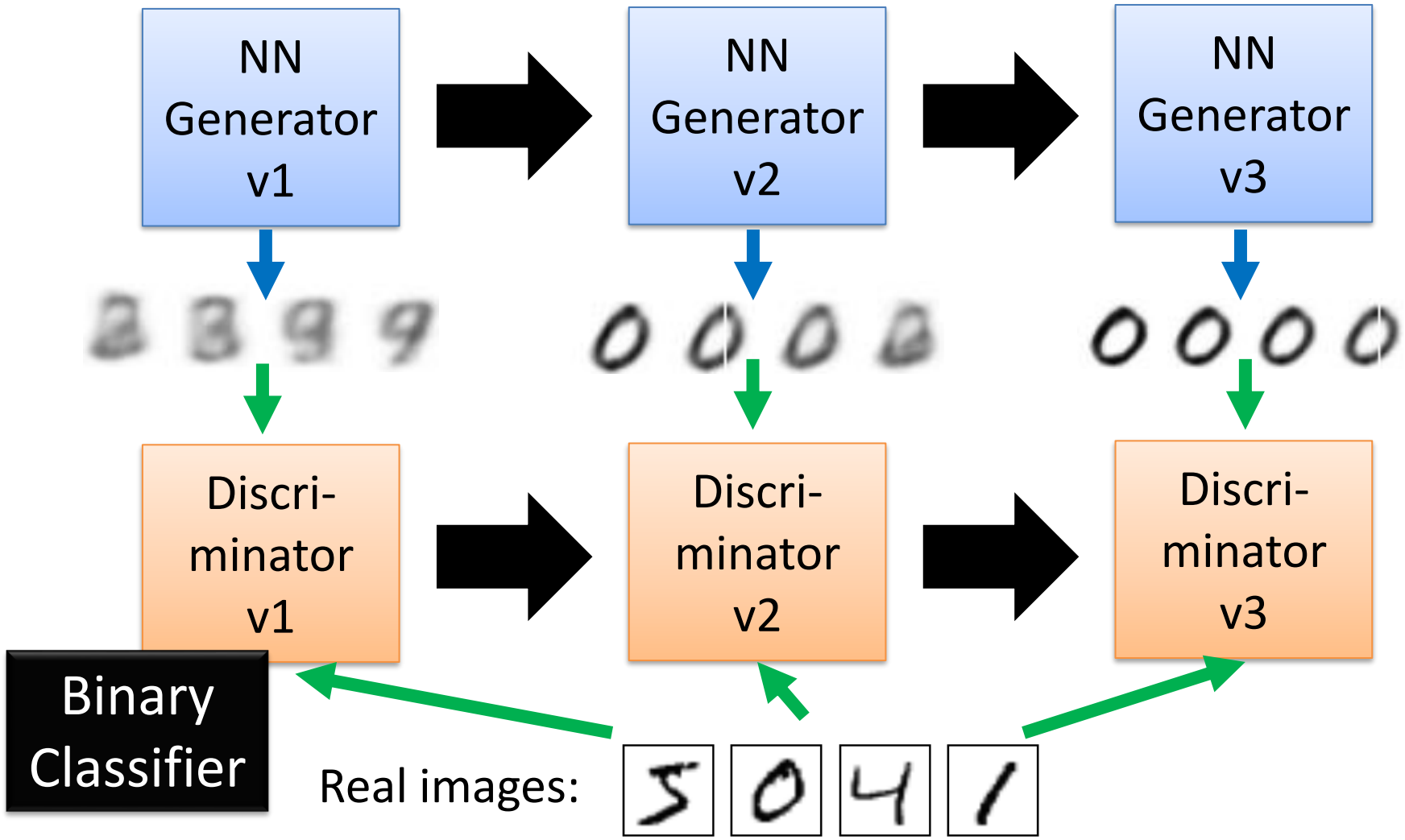
Generative Adversarial Network (GAN)



Generator/discriminator
可以是任一neural netowrk

https://towardsdatascience.com/understanding-generative-adversarial-networks-4dafc963f2ef

The evolution of generation



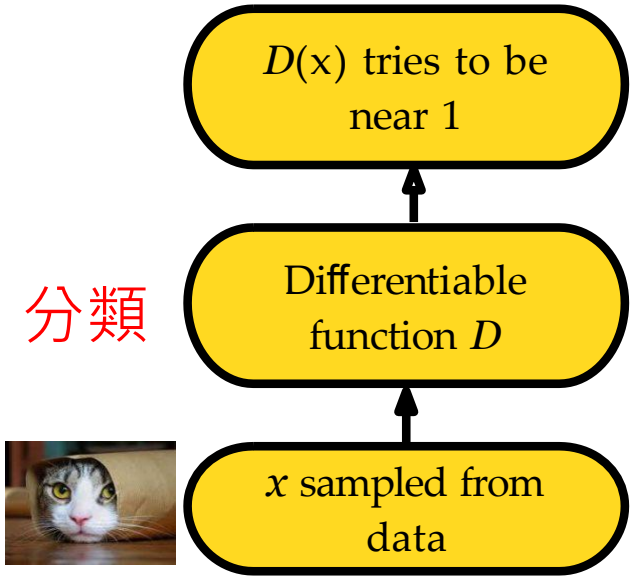
Source: Hung-Yi Lee, 2017

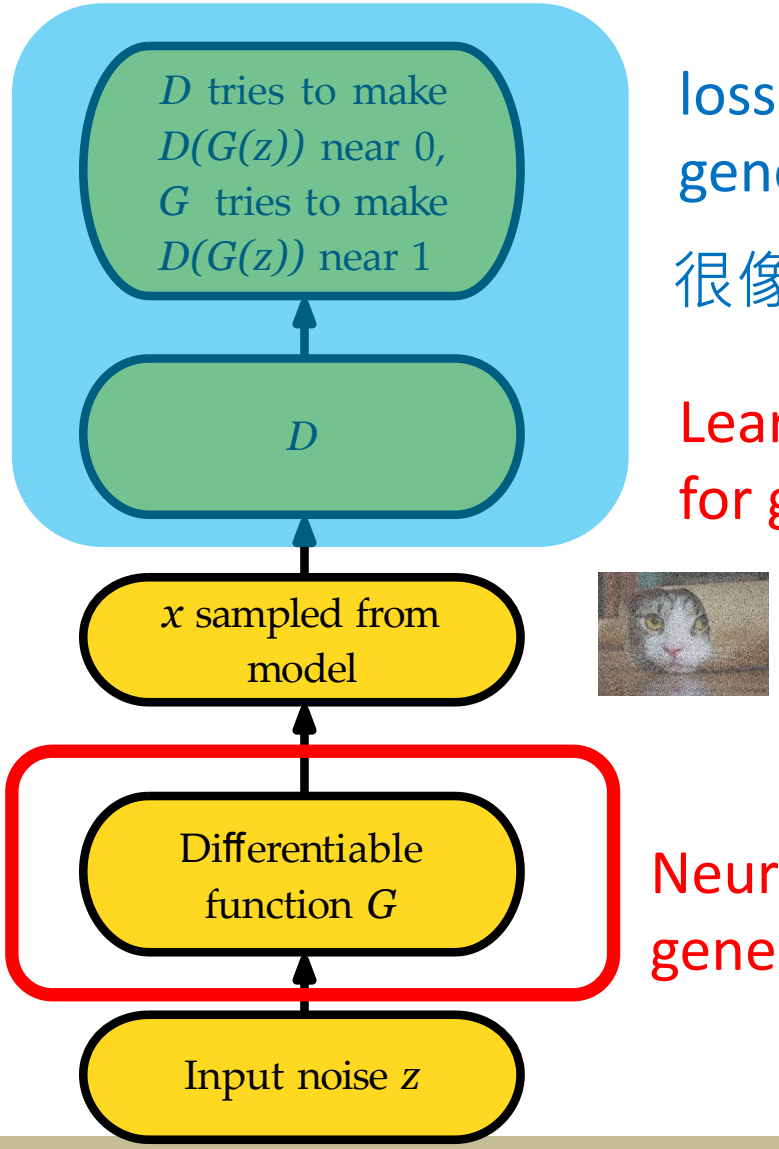
生成模型百百種 Why GAN?

- 可以平行生成
- Generator 的限制少
- 不需要 Markov chains
  - 高維度及計算量
- 生出的結果較好

- 缺點
  - 但比較不好訓練

GAN as Learned Loss Function?





loss function for generator G?

很像? True/false 二分類

Learned loss function for generator G

Neural network as generator G

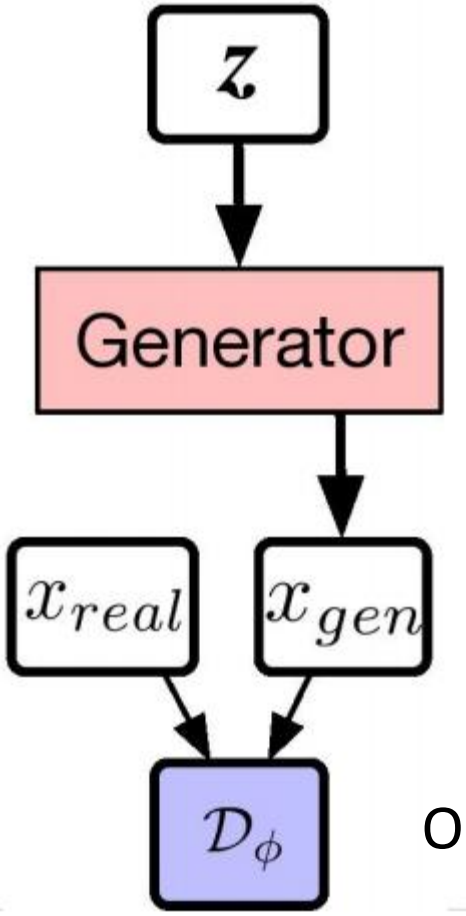
" … We'll learn about Generative Adversarial Networks (GANs). This is, at its heart, a different kind of loss function." By Jeremy Horward

(Goodfellow et al., 2014)

(Goodfellow 2018)

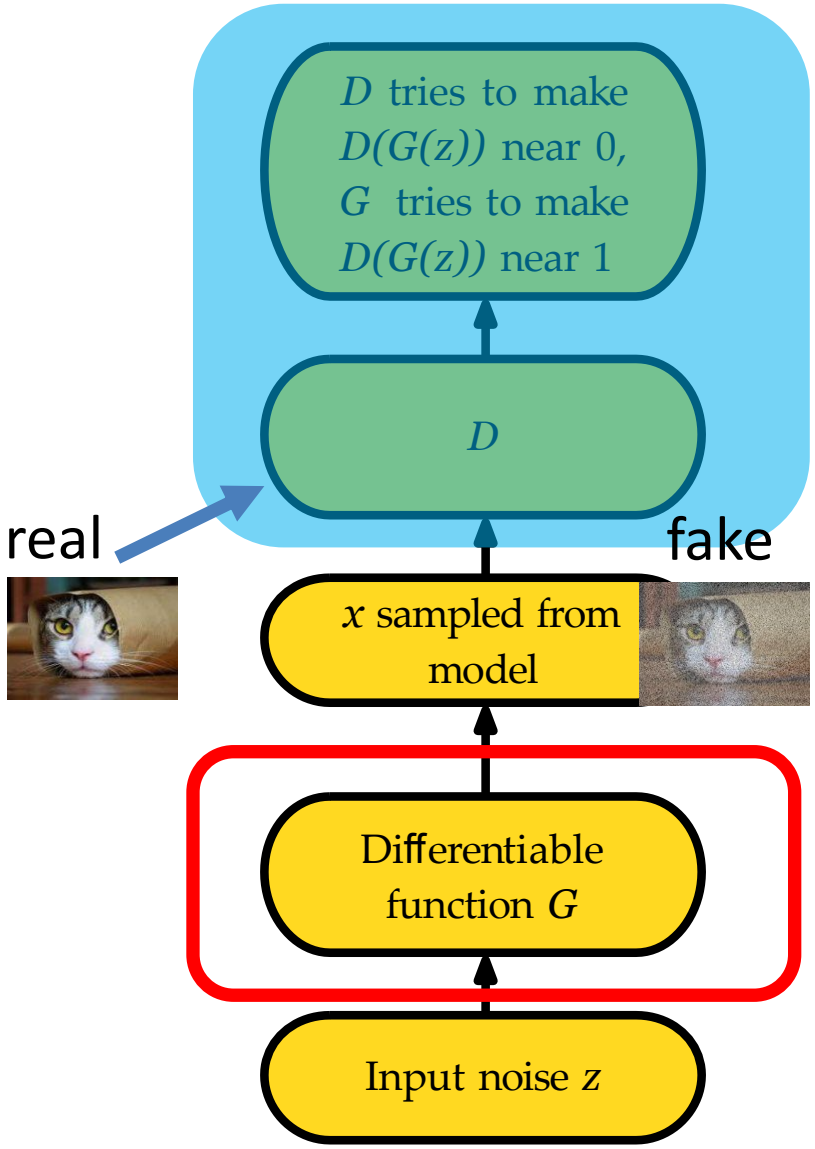
GAN as Learned Loss Function?

- Explicit loss function in conventional NN
- In GAN



Or Implicit loss function with NN?

Math Behind GAN



分類器 Discriminator loss,目標最小化L_D

$$L_D = -E_{x\sim P_r}[\log(D(x))] - E_{x\sim P_z}[\log(1 - D(G(z)))]$$

real　　fake

生成器 Generator loss, G的目標是與判別器D唱反調，既然D的目標是最小化 L_D ，那麼G的目標就設定為最小化 $-L_D$

$$L_G = E_{x\sim P_r}[\log(D(x))] + E_{x\sim P_z}[\log(1 - D(G(z)))]$$

合起來。minmax game

$$\min_G \max_D V(G,D) = E_{x\sim P_r}[\log(D(x))] + E_{x\sim P_z}[\log(1 - D(G(z)))]$$

(Goodfellow 2018)

Optimum Solution: equilibrium

對於固定的生成器G，我們總能找到一個最優的

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

把 $D_G^*(x)$ 代入 $V(D,G)$，有

$$\min_G \max_D \quad V(D,G) = \min_G C(G)$$

$$\begin{aligned} C(G) &= V(D_G^*, G) \\ &= \mathbb{E}_{x\sim P_r}\left[\log \frac{P_r(x)}{P_r(x)+P_f(x)}\right] + \mathbb{E}_{x\sim P_f}\left[\log \frac{P_f(x)}{P_r(x)+P_f(x)}\right] \\ &= -log(4) + KL\left(P_r || \frac{P_r+P_f}{2}\right) + KL\left(P_f || \frac{P_r+P_f}{2}\right) \\ &= -log(4) + 2JS(P_r||P_f). \end{aligned}$$

JS divergence

Optimum solution

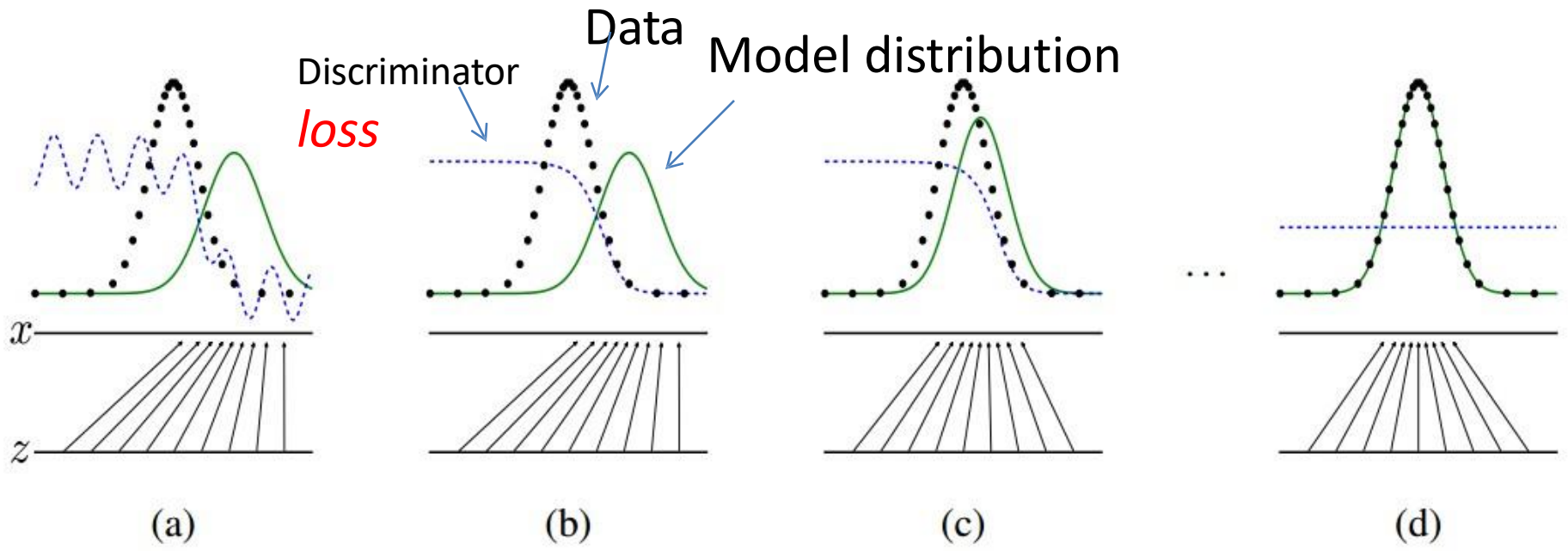
$P_{data}(x) = P_G(x) \ \forall x$

$D(x) = \frac{1}{2} \ \forall x$

Solution 最佳解

- Optimal *D* is always
  - when P_{model} == P_{data}, 完全學到真實資料的分布，就是最佳解

$$D(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{model}}(x)}$$



Algorithm 1 Minmax Game

1: **for** specified # of training iterations **do**

▷ TRAINING DISCRIMINATOR

2: **for** specified k steps* **do**

3: Draw minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\} \sim p_z$.

4: Draw minibatch of m data samples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\} \sim p_{data}$.

5: Update D's parameters by gradient ascent:

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

6: **end for**

▷ TRAINING GENERATOR

7: Draw minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\} \sim p_z$.

8: Update G's parameters by gradient descent:

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

9: **end for**

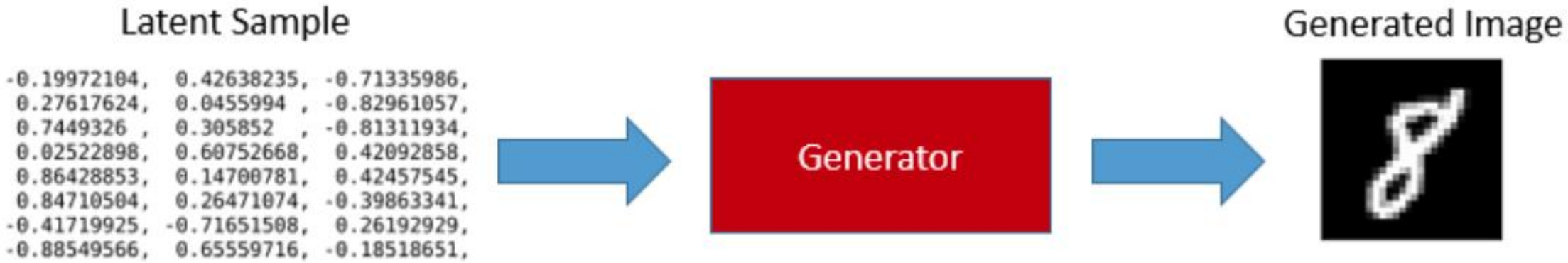
*k is a tunable hyperparameter which is usually set to 1 to lower the training cost of each iteration.

https://github.com/ericjang/genadv_tutorial/blob/master/genadv1.ipynb

ORIGINAL GAN IN KERAS

https://github.com/naokishibuya/deep-learning/blob/master/python/gan_mnist.ipynb

GAN in Keras: Generator

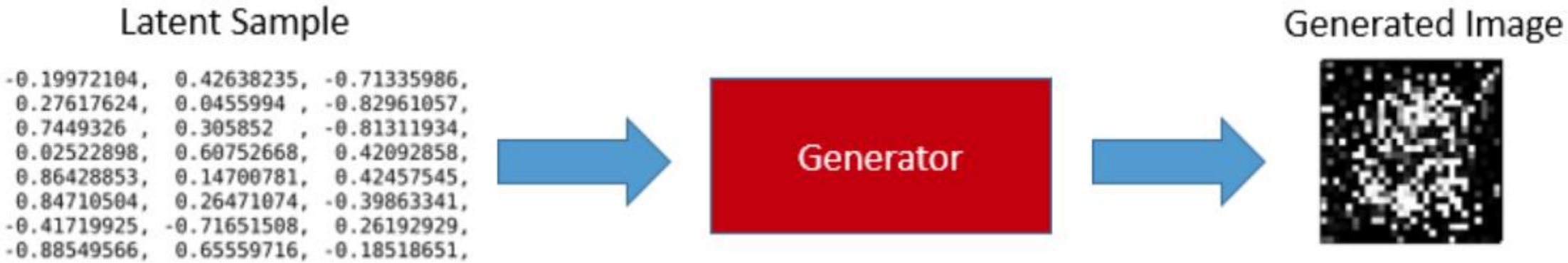



```
generator = Sequential([
    Dense(128, input_shape=(100,)),
    LeakyReLU(alpha=0.01),
    Dense(784),
    Activation('tanh')
], name='generator')
```

- NN輸出建議使用tanh
  - It also means that we need to rescale the MNIST images to be between -1 and 1

Initial Generator

- Without training, the generator produces garbage images only.



GAN in Keras: Discriminator

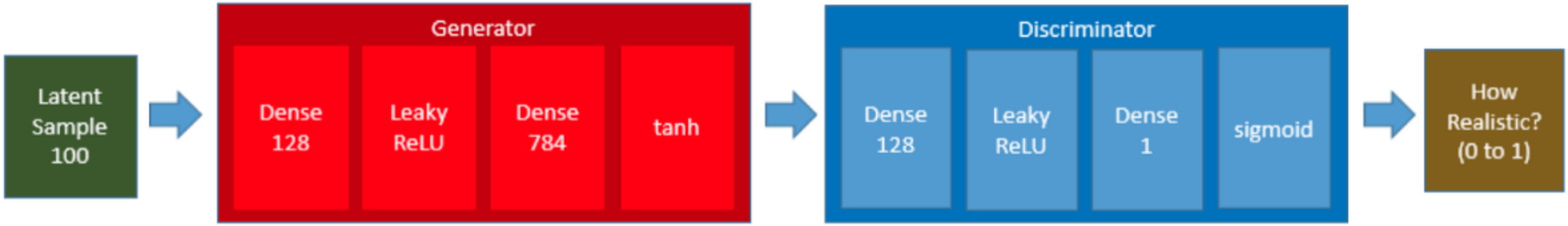
- 就是一般的分類器

- NN輸出建議使用sigmoid，得到多像的機率，方便訓練
  - tell us the probability of whether the input image is real or not. So, the output can be any value between 0 and 1

```
discriminator = Sequential([
    Dense(128, input_shape=(784,)),
    LeakyReLU(alpha=0.01),
    Dense(1),
    Activation('sigmoid')
], name='discriminator')
```

G+D = GAN

- connect the generator and the discriminator to produce a GAN

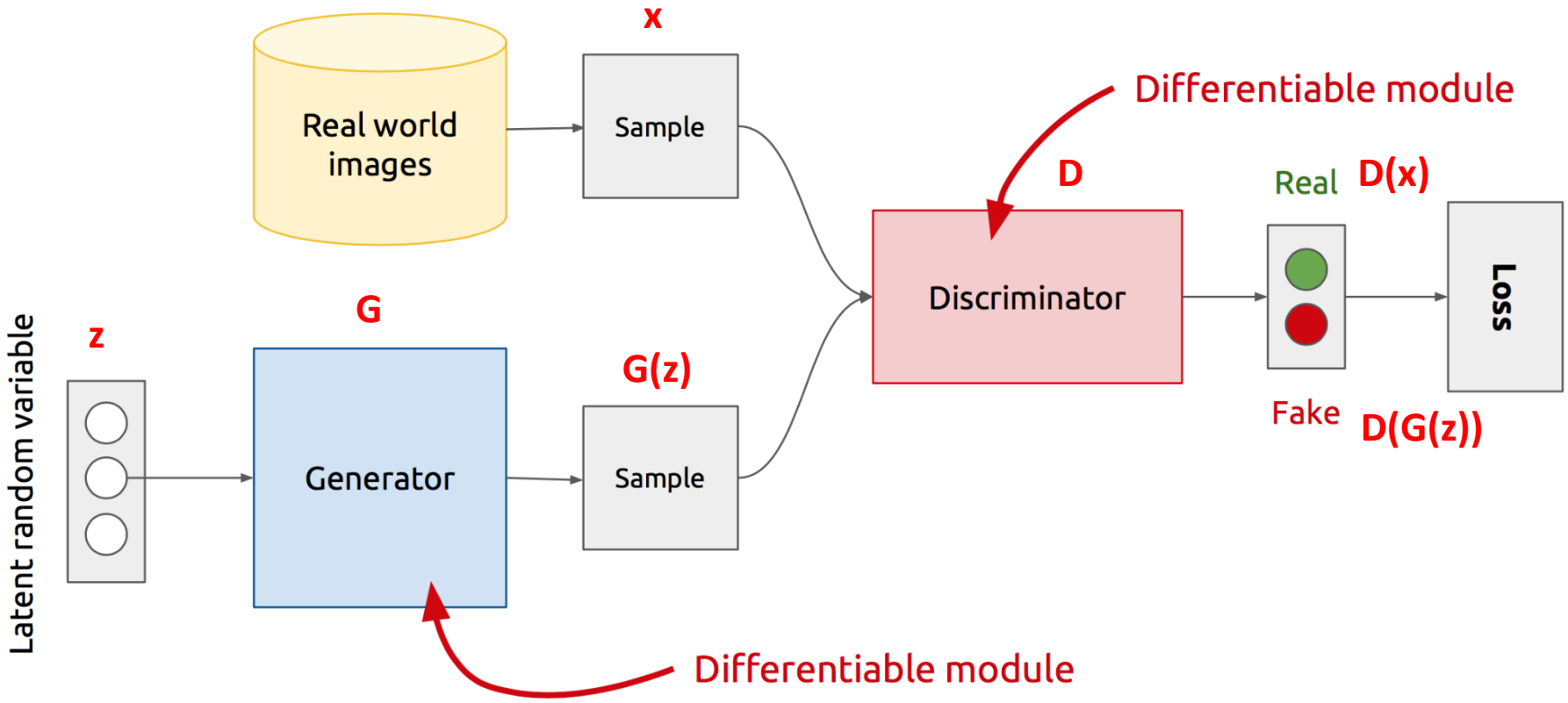



```
gan = Sequential([
    generator,
    discriminator
])
```

Training Loop

- 1. Train discriminator
  - Set the discriminator trainable
  - Set the generator non-trainable
  - Train the discriminator with the real MNIST digit images and the images generated by the generator to classify the real and fake images.
- 2. Train generator
  - Set the discriminator non-trainable
  - Set the generator trainable
  - Train the generator as part of the GAN.
    - We feed latent samples into the GAN and let the generator to produce digit images and use the discriminator to classify the image.

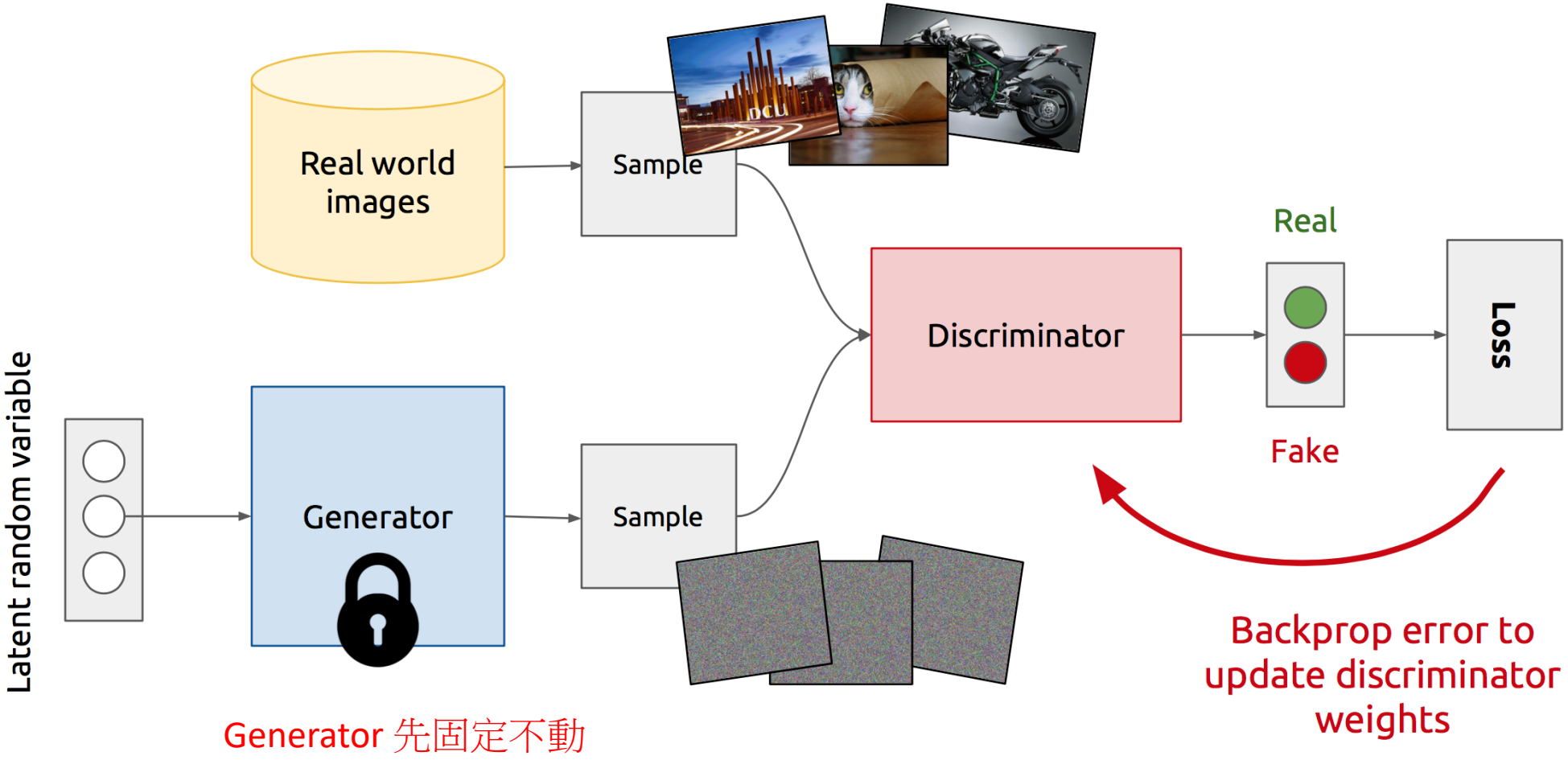
GAN's Architecture



- **Z** is some random noise (Gaussian/Uniform).
- **Z** can be thought as the latent representation of the image.

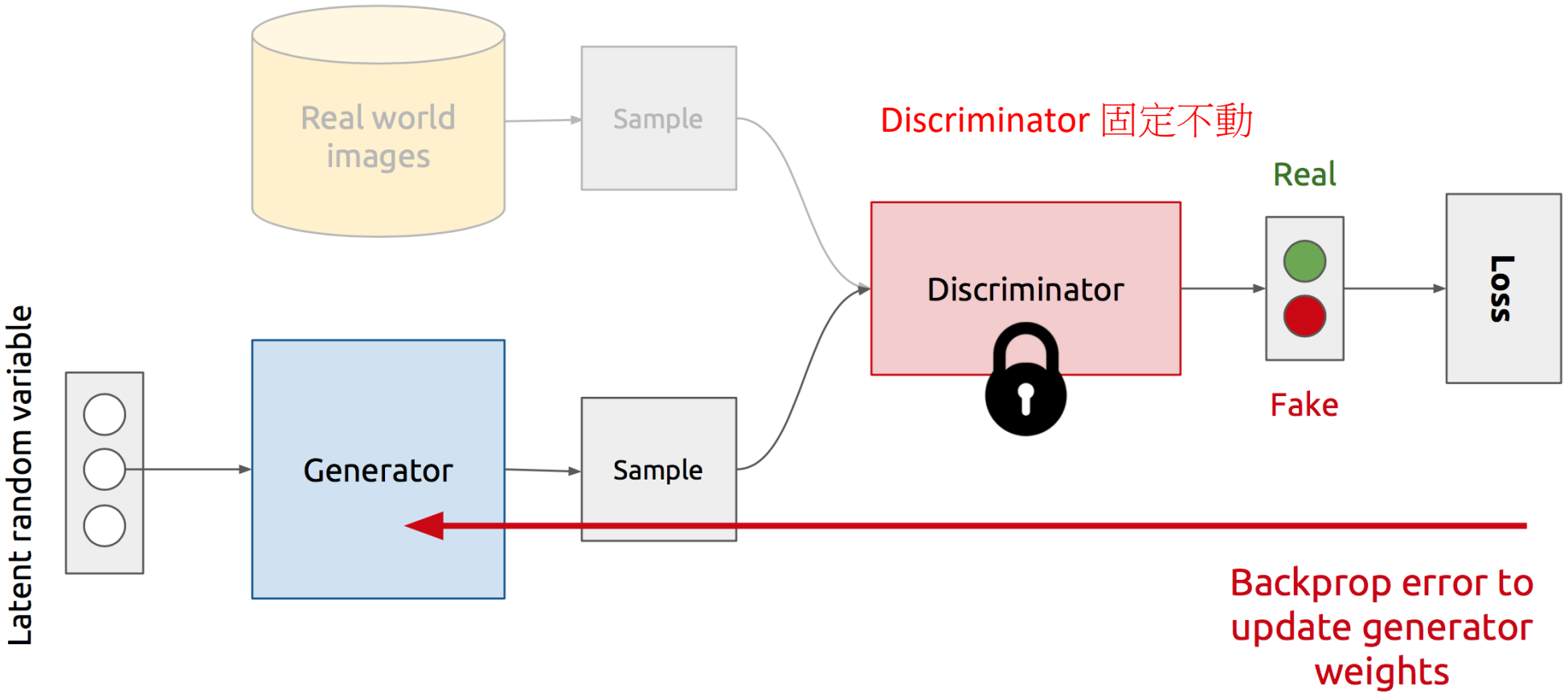
https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

1. Training Discriminator



https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

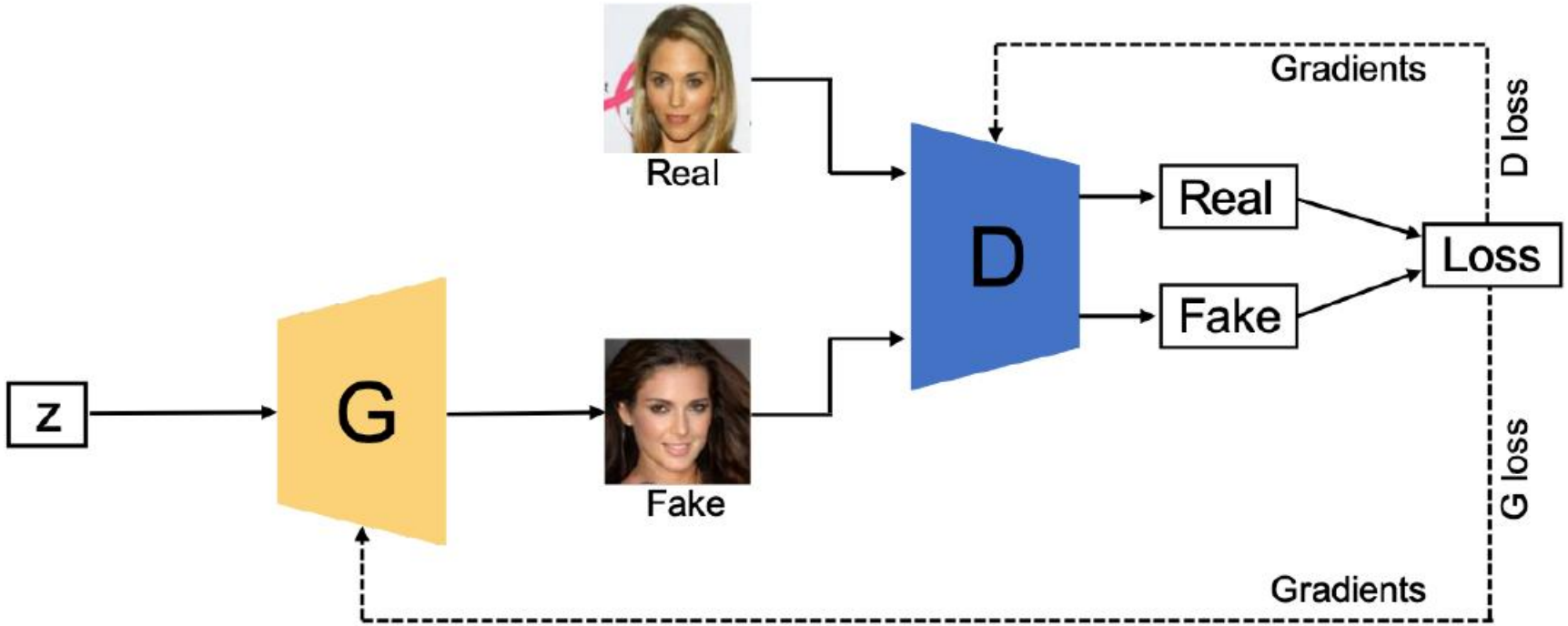
2. Training Generator



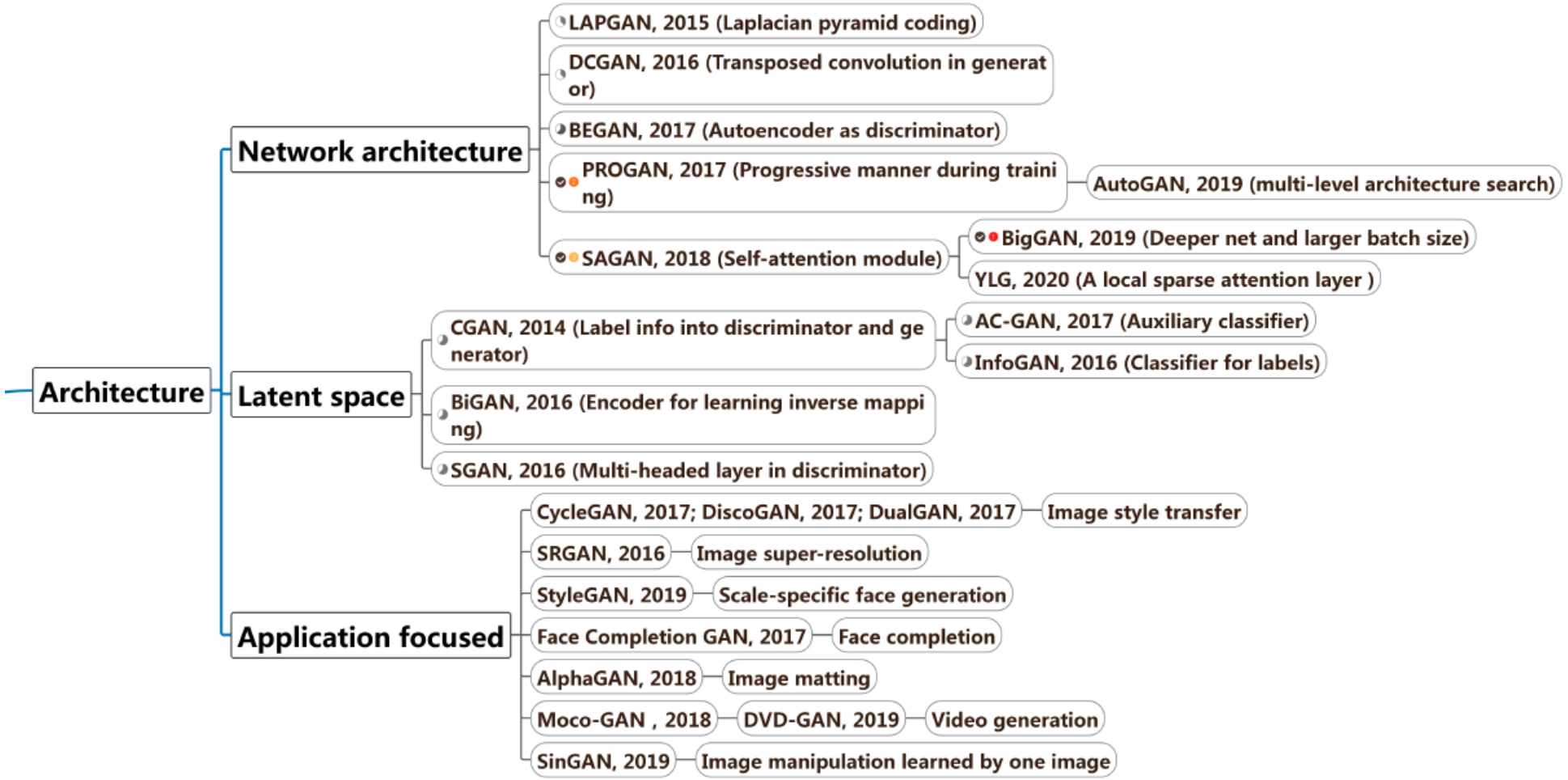
OVERVIEW OF GAN PROGRESS

GAN Improvement

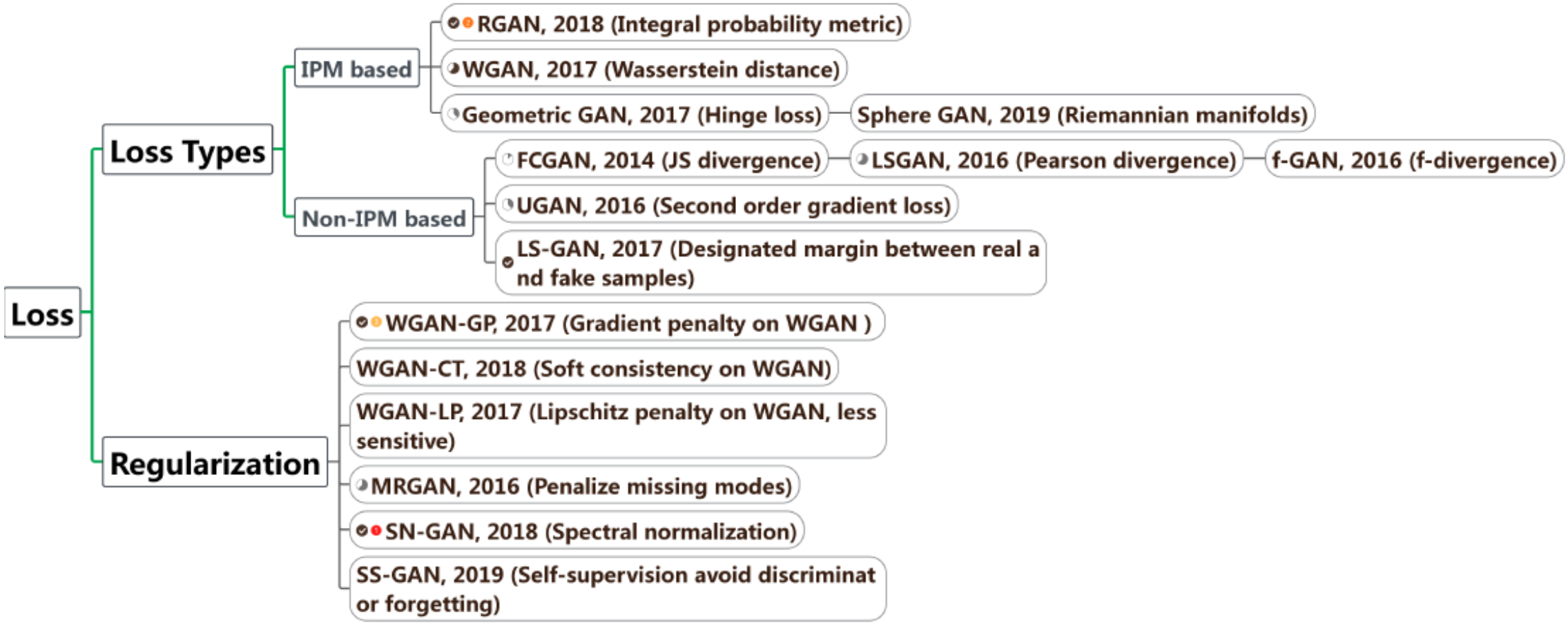
- image diversity (also known as mode diversity),
- image quality
- stable training



Taxonomy of the recent GANs



Taxonomy of the recent GANs



CHALLENGES OF GAN TRAINING
LOSS VARIANT GANS

Problem of GAN

- Vanished gradient 梯度消失
  - 損失函數對生成器和判別器的參數的梯度變為零，導致GAN無法收斂
- Mode collapse 模型坍塌
  - 生成器生成的圖像局限於某一小的區域，對輸入的噪聲不再敏感，失去了生成圖片的多樣性

- measure and minimize the distance between the two distributions p_r and p_g
  - 不能正確評估generator 到底好不好 (loss function problem)

$$V(D, \boldsymbol{\theta}^{(G)}) = 2JSD(p_r||p_g) - 2log2$$

$$V(D, \boldsymbol{\theta}^{(G)}) = 2JSD(p_r||p_g) - 2log2$$

- JS divergence stays constant (log2=0.693)
  - if there is no overlap between p_r and p_g

JS divergence 0.693 0.693 0.336

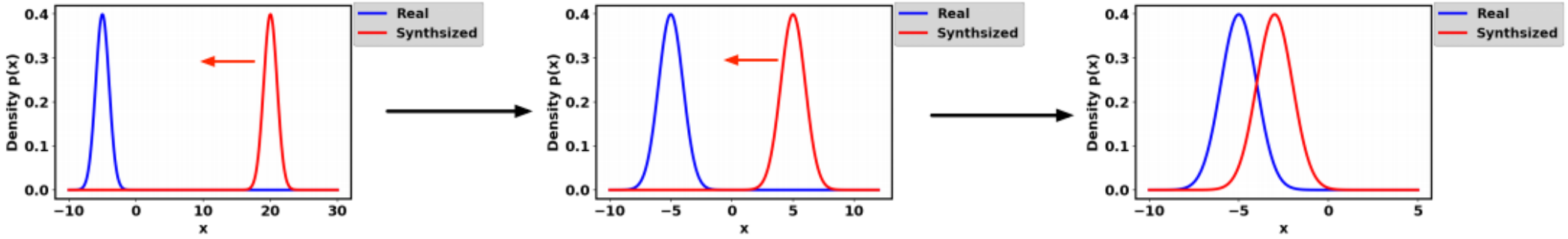
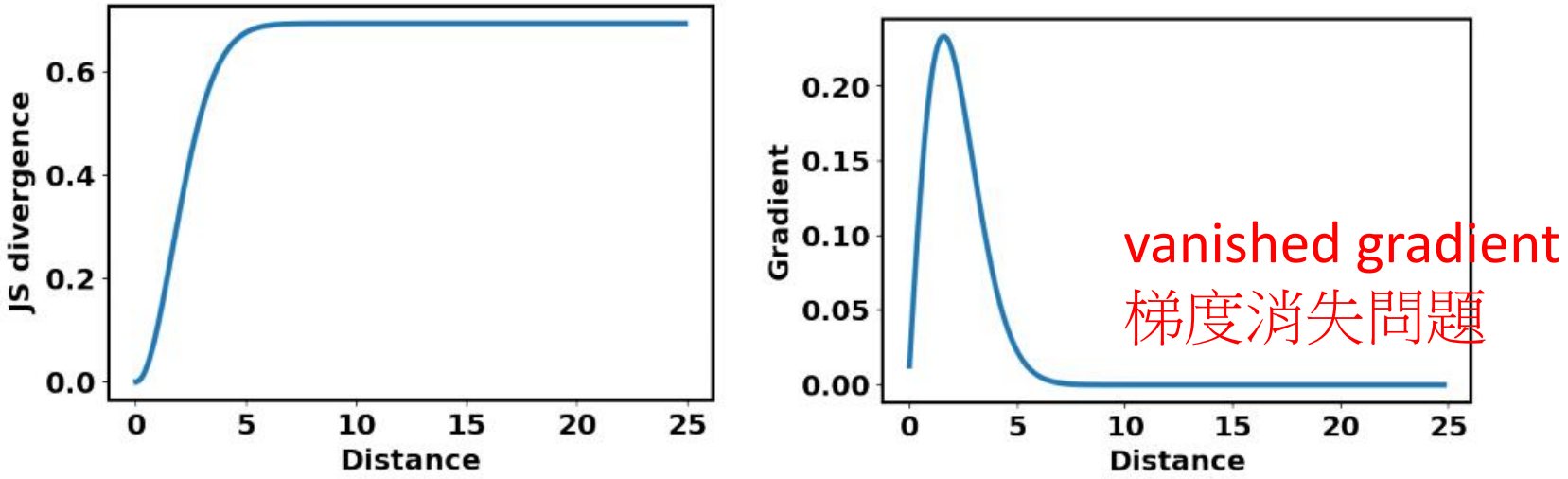


Fig. 19. Illustration of training progress for a GAN. Two normal distributions are used here for visualization. Given an optimal D, the objective of GANs is to update G in order to move the generated distribution p_g (red) towards the real distribution p_r (blue) (G is updated from left to right in this figure. Left: initial state, middle: during training, right: training converging). However, JS divergence for the left two figures are both 0.693 and the figure on the right is 0.336, indicating that JS divergence does not provide sufficient gradient at the initial state.

- JS divergence
  - is constant and its gradient is almost 0 when the distance is greater than 5,
    - training process does not have any effect on G.
  - Nonzero gradient only for substantial overlap, (very low probability)



(a) JS divergence changes with distance.

(b) Gradient JS divergence changes with distance.

Fig. 20. JS divergence and gradient change with the distance between p_r and p_g. The distance is the difference between two distribution means.

Alternative Objective Function for Generator in Real Implementation

$$V = E_{x \sim P_{data}}[logD(x)]$$

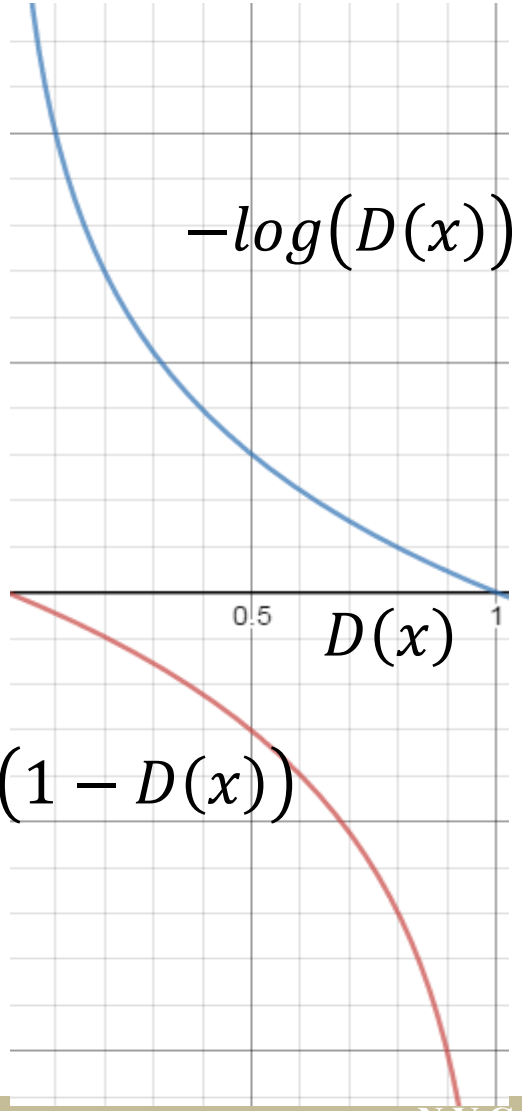
$$+E_{x \sim P_G}[log(1 - D(x))]$$

Slow at the beginning

$$V = E_{x \sim P_G}[-log(D(x))]$$

Real implementation:
label x from P_G as positive

logD trick

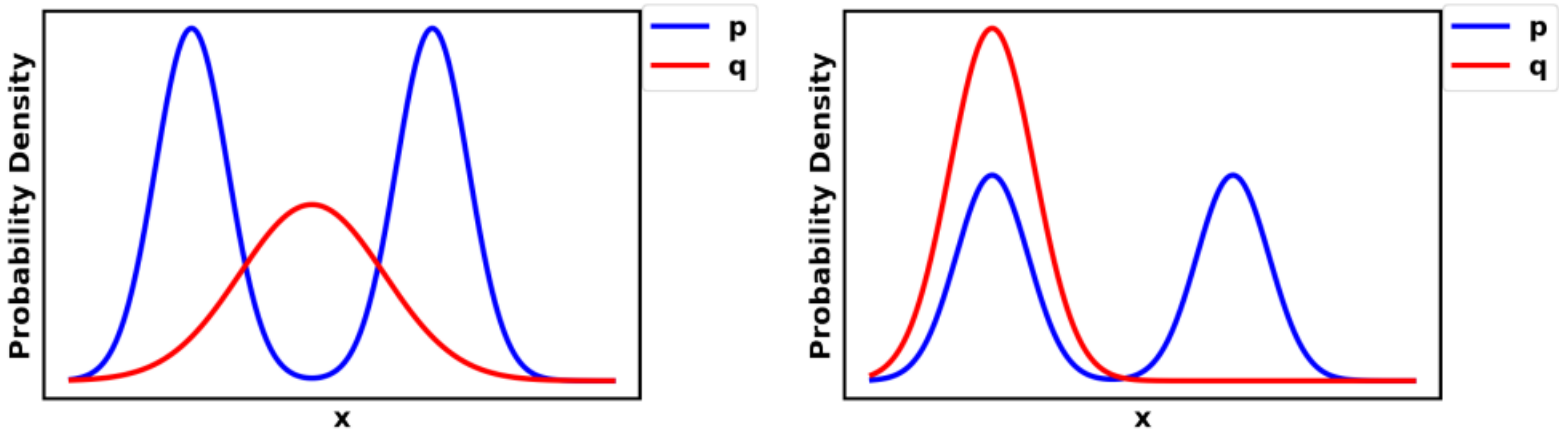


Source: Hung-Yi Lee, 2017

Problems of logD trick

$$
\begin{aligned}
&-\mathbb{E}_{\mathbf{x}\sim p_g}\log[D^*(\mathbf{x})] \\
&= KL(p_g\|p_r) - \mathbb{E}_{\mathbf{x}\sim p_g}\log[1-D^*(\mathbf{x})], \\
&= KL(p_g\|p_r) - 2\cdot JS(p_r\|p_g) + 2\cdot\log 2 + \mathbb{E}_{\mathbf{x}\sim p_x}\log[D^*(\mathbf{x})],
\end{aligned}
$$

reverse KL divergence



(a) KL divergence. (b) Reverse KL divergence.

Fig. 21. Optimized q (in red) when minimizing KL divergence $KL(p\|q)$ and reverse KL divergence $KL(q\|p)$

$$
\begin{aligned}
&When\ p_g(\mathbf{x}) \to 0,\ p_r(\mathbf{x}) \to 1,\ KL(p_g\|p_r) \to 0. \\
&When\ p_g(\mathbf{x}) \to 1,\ p_r(\mathbf{x}) \to 0,\ KL(p_g\|p_r) \to +\infty.
\end{aligned}
$$

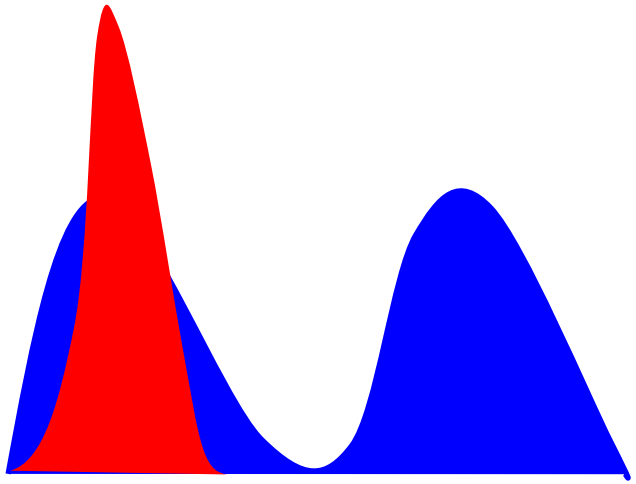
mode collapse
模型崩塌問題

G will generate *repeated* *but "safe"* samples

the mode collapse during training GANs

Mode Collapse模型崩潰

Converge to same faces



Sometimes, this is hard to tell since
one sees only what's generated, but not what's missed.

Source: Hung-Yi Lee, 2017

Why hard to train GAN?

WGAN前作:
Towards Principled methods for training generative adversarial networks. Martin Arjovsky

$$-\mathbb{E}_{x\sim P_r}[\log D(x)] - \mathbb{E}_{x\sim P_g}[\log(1-D(x))] \quad (公式1)$$

$$\mathbb{E}_{x\sim P_g}[\log(1-D(x))] \quad (公式2)$$

$$\mathbb{E}_{x\sim P_g}[-\log D(x)] \quad (公式3)$$

- **原始形式GAN(公式2)的問題:**

判別器訓練的太好，生成器梯度消失。判別器訓練的不好，生成器梯度不准， 四處亂跑。只有判別器訓練得不好不壞才行。但是這個火候很難把握。

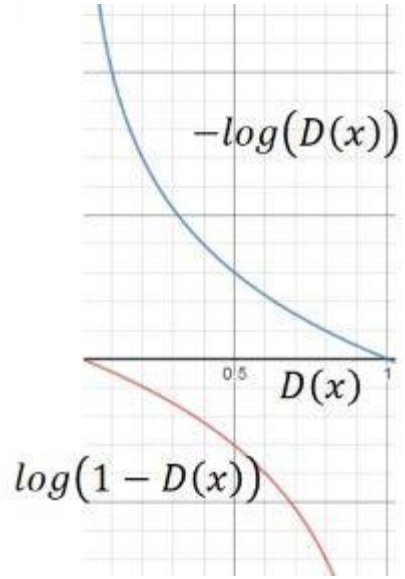
https://arxiv.org/pdf/1701.07875v1.pdf Wasserterin Gan.
https://openreview.net/pdf?id=Hk4_qw5xe Towards Principled methods for training generative adversarial networks

Why hard to train GAN? 原因在哪裡?

WGAN前作:
Towards Principled methods for training generative adversarial networks. Martin Arjovsky

$$-\mathbb{E}_{x\sim P_r}[\log D(x)] - \mathbb{E}_{x\sim P_g}[\log(1-D(x))] \quad (公式1)$$

$$\mathbb{E}_{x\sim P_g}[-\log D(x)] \quad (公式3)$$



-logD trick GAN的問題（從生成器的等價loss入手）

最小化公式3等價於最小化

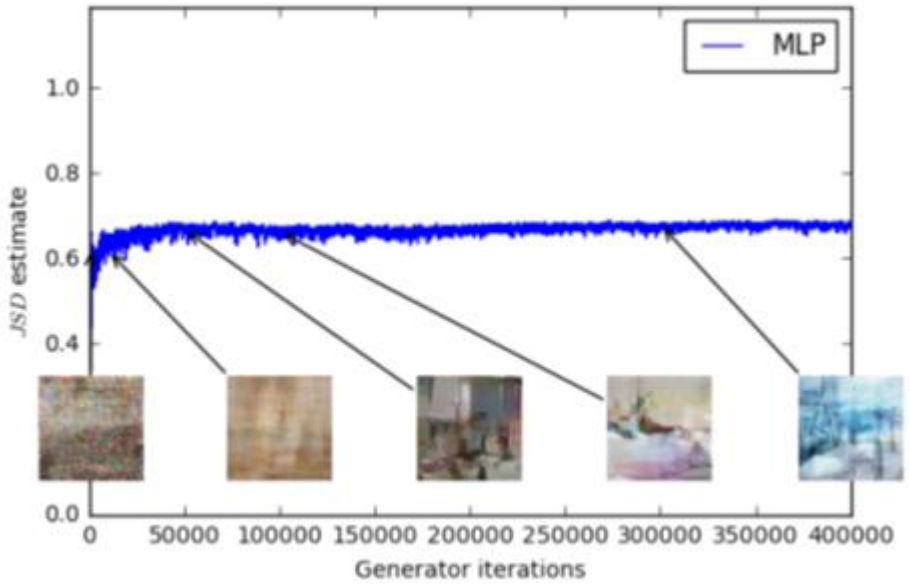
$$KL(P_g||P_r) - 2JS(P_r||P_g) \quad (公式11)$$

這個等價最小化目標存在兩個嚴重的問題。第一是它同時要最小化生成分佈與真實分佈的KL散度，卻又要最大化兩者的JS散度，一個要拉近，一個卻要推遠！這在直 觀上非常荒謬，在數值上則會導致梯度不穩定太好，容易出現mode collapse。

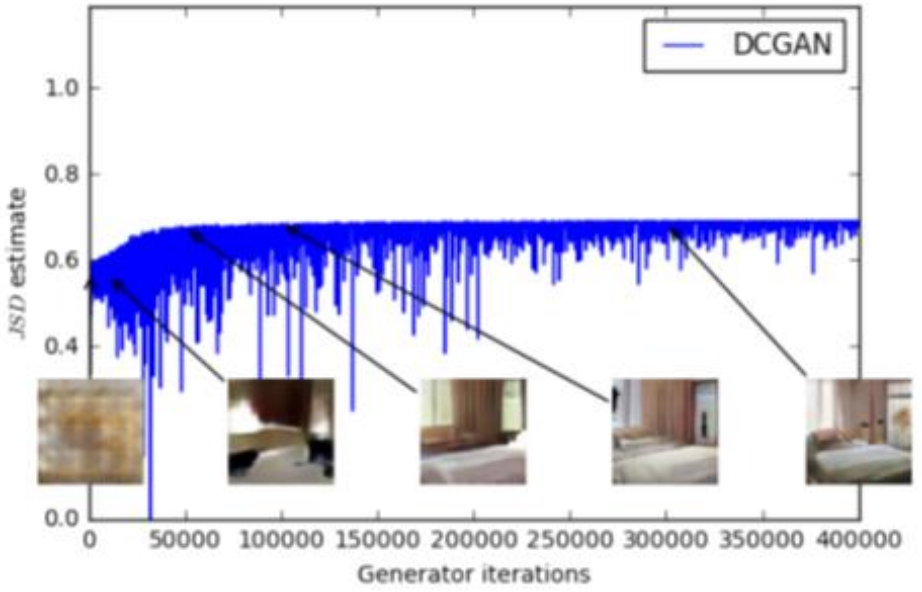
https://arxiv.org/pdf/1701.07875v1.pdf Wasserterin Gan.
https://openreview.net/pdf?id=Hk4_qw5xe Towards Principled methods for training generative adversarial networks

Problem of JS divergence (GAN Loss)

- JS divergence estimated by discriminator telling little information



Weak Generator

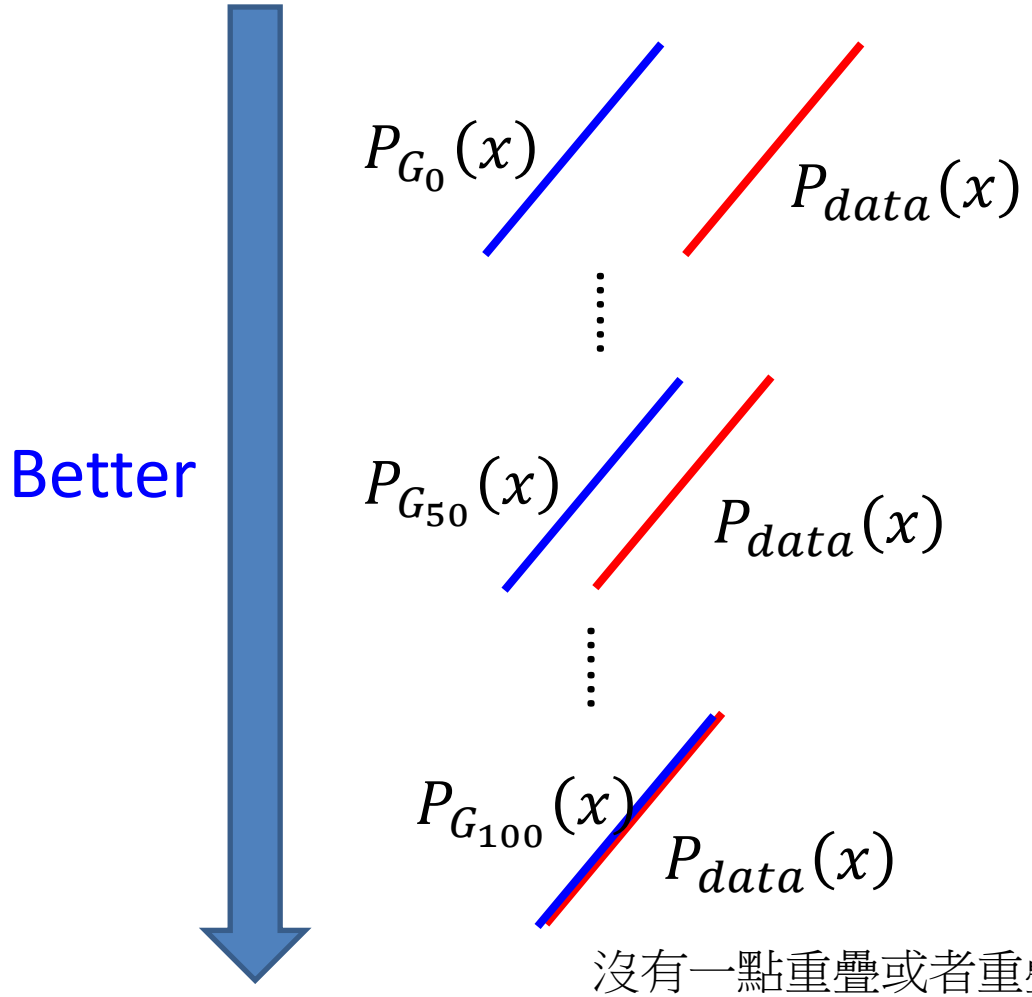


Strong Generator

Source: Hung-Yi Lee, 2017

https://arxiv.org/abs/1701.07875

Why GAN is hard to train?
Problem of JS distance



$JS(P_{G_1}||P_{data}) = log2$

$JS(P_{G_2}||P_{data}) = log2$

Not really better ……

可以不用機率，不限定值在[0, 1]，直接算距離嗎?

$JS(P_{G_2}||P_{data}) = 0$

沒有一點重疊或者重疊部分可忽略，JS散度就固定是常數log 2，而這對於梯度下降方法意味著——梯度為0

Source: Hung-Yi Lee, 2017

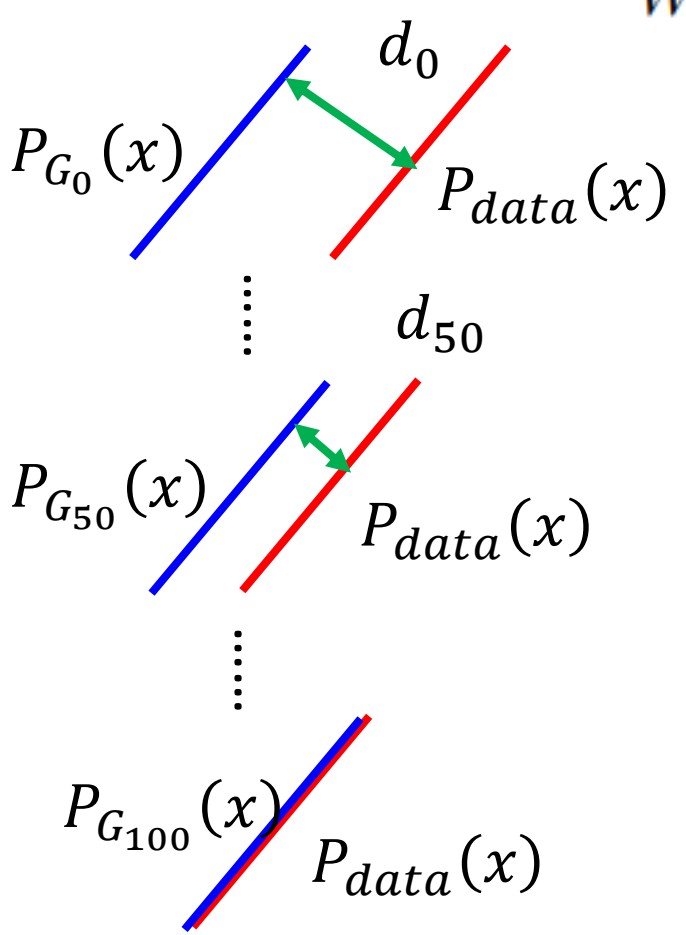
WGAN

Why GAN is hard to train?
Problem of JS distance

Using Wasserstein distance instead of JS divergence

$$W(p_r, p_g) = \inf_{\gamma \in \prod(p_r, p_g)} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\gamma} \|\mathbf{x} - \mathbf{y}\|,$$

Better



$W(P_{G_1}||P_{data}) = d_0$

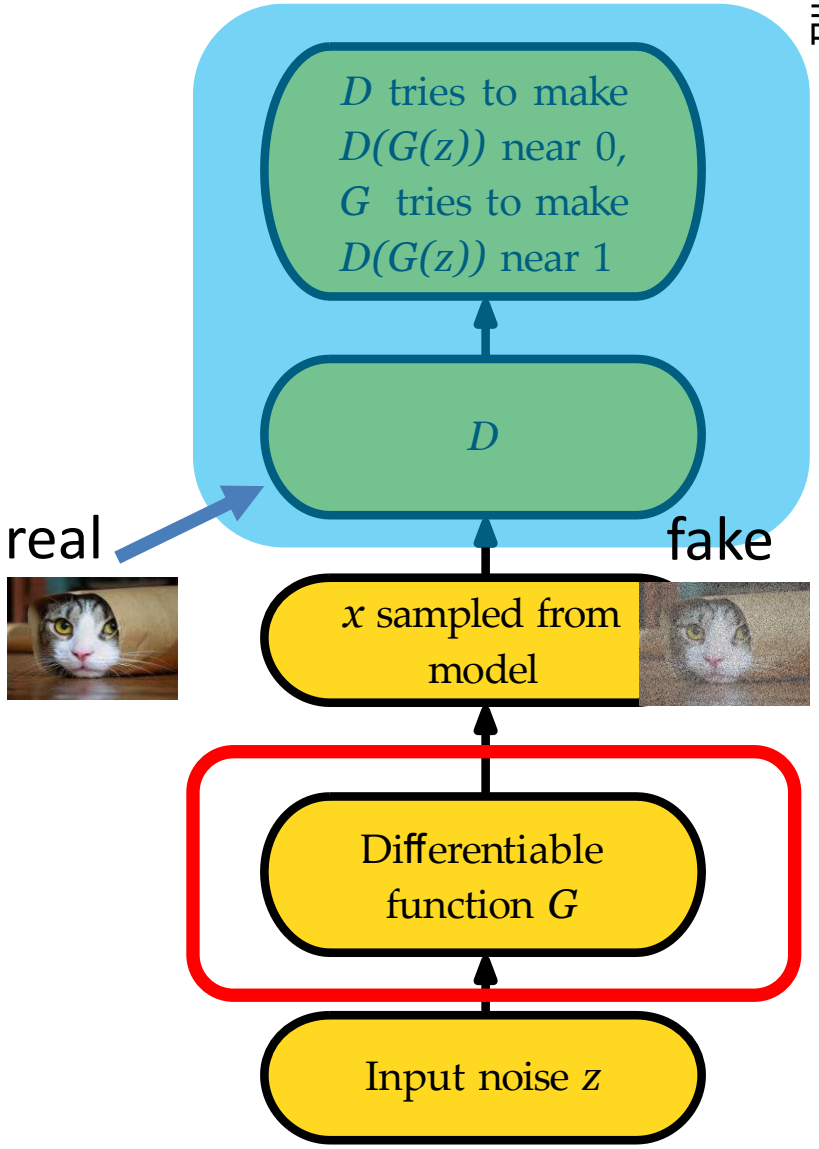
$W(P_{G_2}||P_{data}) = d_{50}$

$W(P_{G_2}||P_{data}) = 0$

Wasserstein距離 (直接看距離)，相比KL散度、JS散度的優越性在於，即便兩個分佈沒有重疊，Wasserstein距離仍然能夠反映它們的遠近。

Source: Hung-Yi Lee, 2017

Math Behind WGAN



讓D(x)描述的不再是x為真的概率，而描述的是x為真的程度

分類器 Discriminator loss,目標最小化L_D

生成器 Generator loss, G的目標是與判別器D唱反調，既然D的目標是最小化 L_D ，那麼G的目標就設定為最小化 $-L_D$

$$\min_G \max_D V(G,D) = E_{x\sim P_r}[\log(D(x))] + E_{x\sim P_z}[\log(1-D(G(z)))]$$

$$\min_G \max_D \mathbb{E}_{x\sim P_r}[D(x)] - \mathbb{E}_{x\sim P_z}[D(G(z))]$$

Lipschitz constraint (D(x) 輸出的變化量不超過輸入的變化量)

- Clipping (WGAN)
- Spectral normalization (SN-GAN)

(Goodfellow 2018)

WGAN

WGAN本作:
Wasserstein GAN. Martin Arjovsky

具體實現

- 判別器最後一層去掉sigmoid (因為算距離，是regression task)
- 生成器和判別器的loss不取log
- Weight clipping 每次更新判別器的參數之後把絕對值截斷到不超過一個固定常數c, (e.g. +/-1)
- 不要用基於動量的優化演算法（包括momentum和Adam，會不穩），推薦RMSProp，SGD 也行

https://arxiv.org/pdf/1701.07875v1.pdf Wasserterin Gan.
https://openreview.net/pdf?id=Hk4_qw5xe Towards Principled methods for training generative adversarial networks

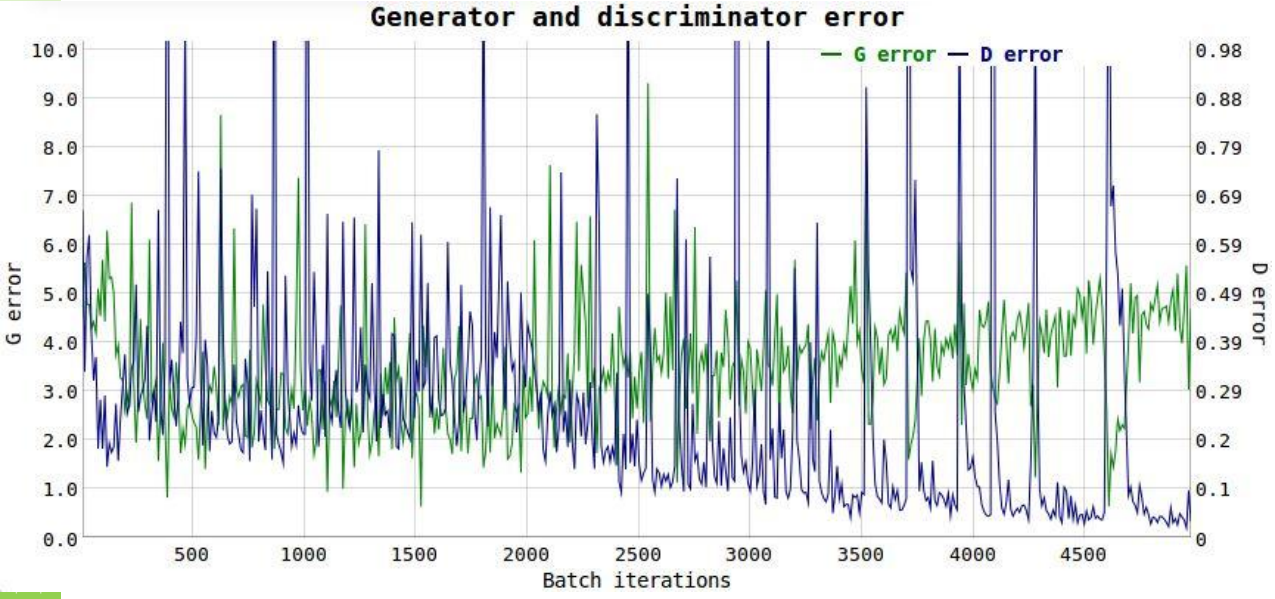
Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.

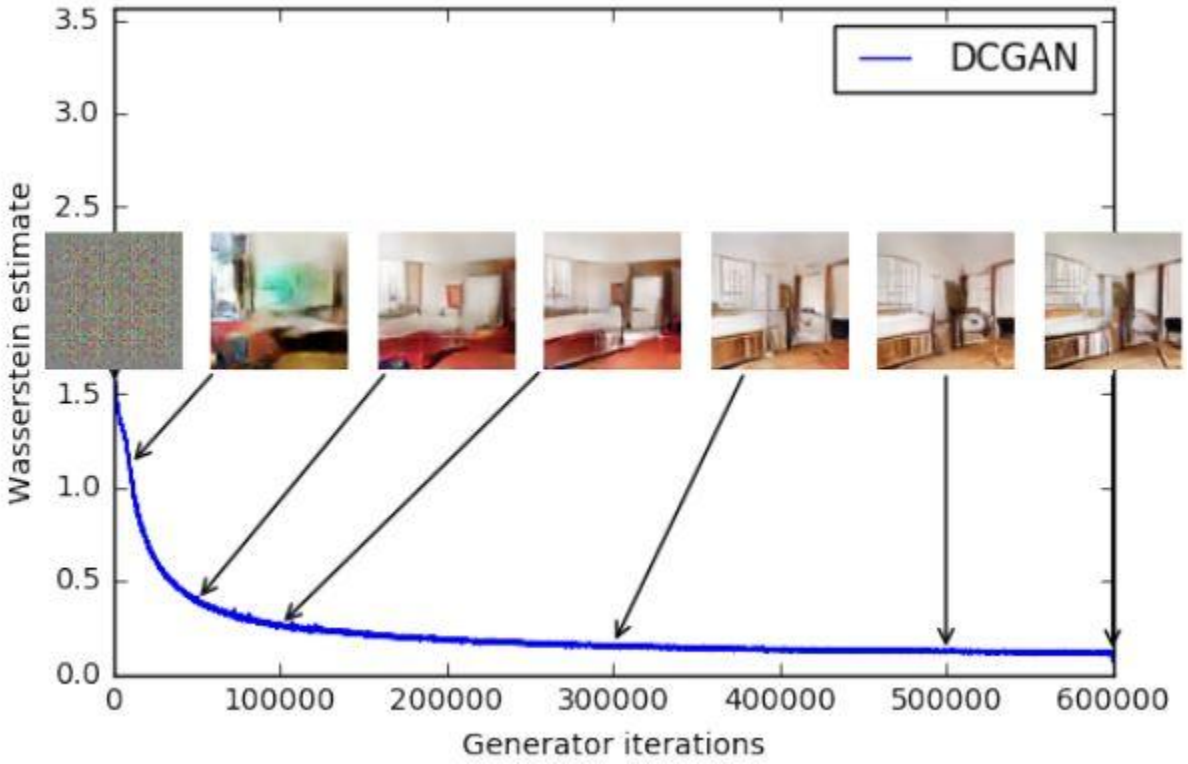
Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**
  for $t = 0, \ldots, n_{\text{critic}}$ **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

WGAN: before and after



Do you know when to stop training just by looking at this figure? Me neither



This is the plot of the WassGAN loss function. The lower the loss, the higher the image quality. Neat!

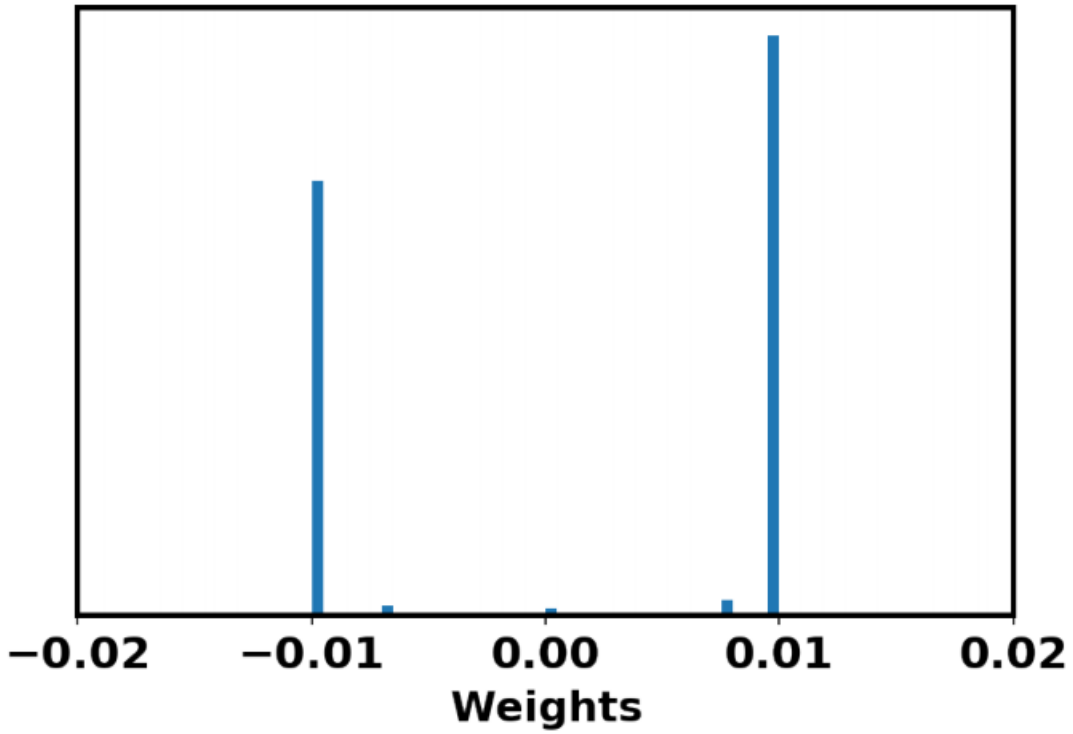
WGAN-GP

- WGAN is not well generalized for a deeper model
  - most WGAN parameters are localized at -0.01 and 0.01 because of parameter clipping. This will dramatically reduce the modeling capacity of D
- WGAN-GP
  - + gradient penalty for restricting $||f||_L \leq K$ for the discriminator

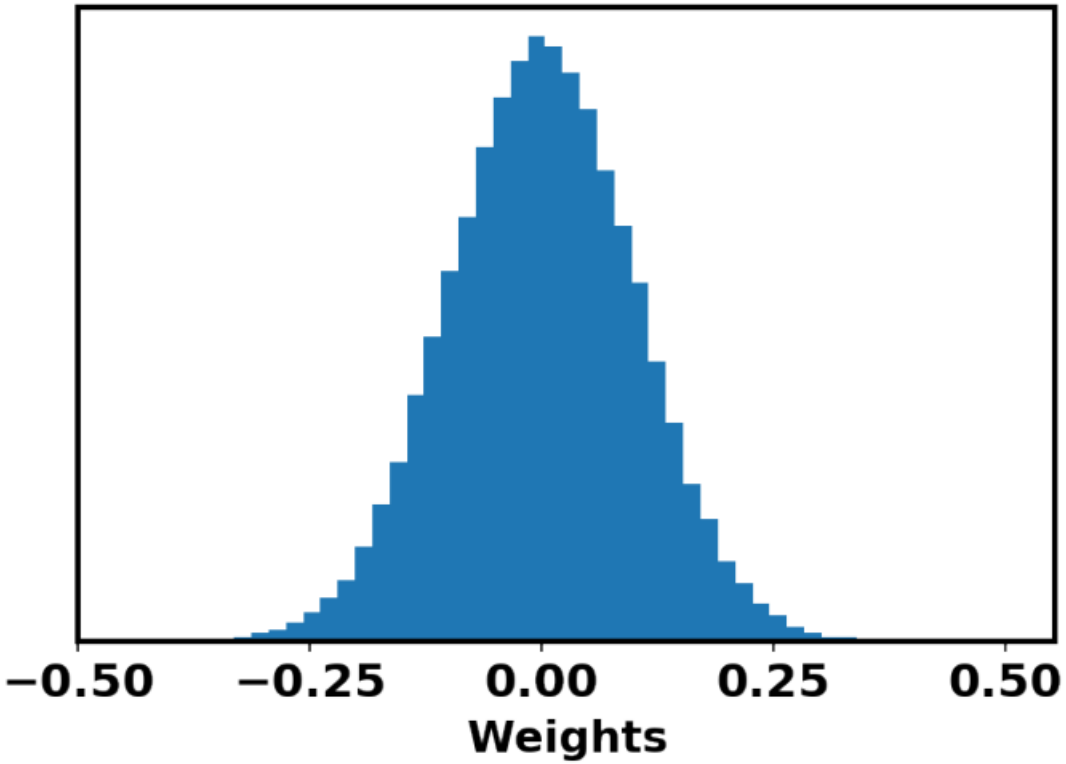
$$\mathcal{L}_D = \mathbb{E}_{\mathbf{x}_g \sim p_g}[D(\mathbf{x}_g)] - \mathbb{E}_{\mathbf{x}_r \sim p_r}[D(\mathbf{x}_r)] + \lambda \mathbb{E}_{\hat{\mathbf{x}} \sim p_{\hat{x}}}[(\|\nabla_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}})\|_2 - 1)^2]$$

WGAN penalty

  - Training is more stable
    - Adam allowed



(a) Weights of WGAN



(b) Weights of WGAN-GP

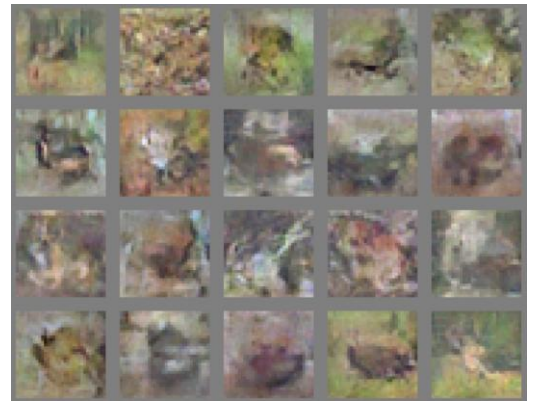
WGAN, WGAN-GP, Spectral Normalization

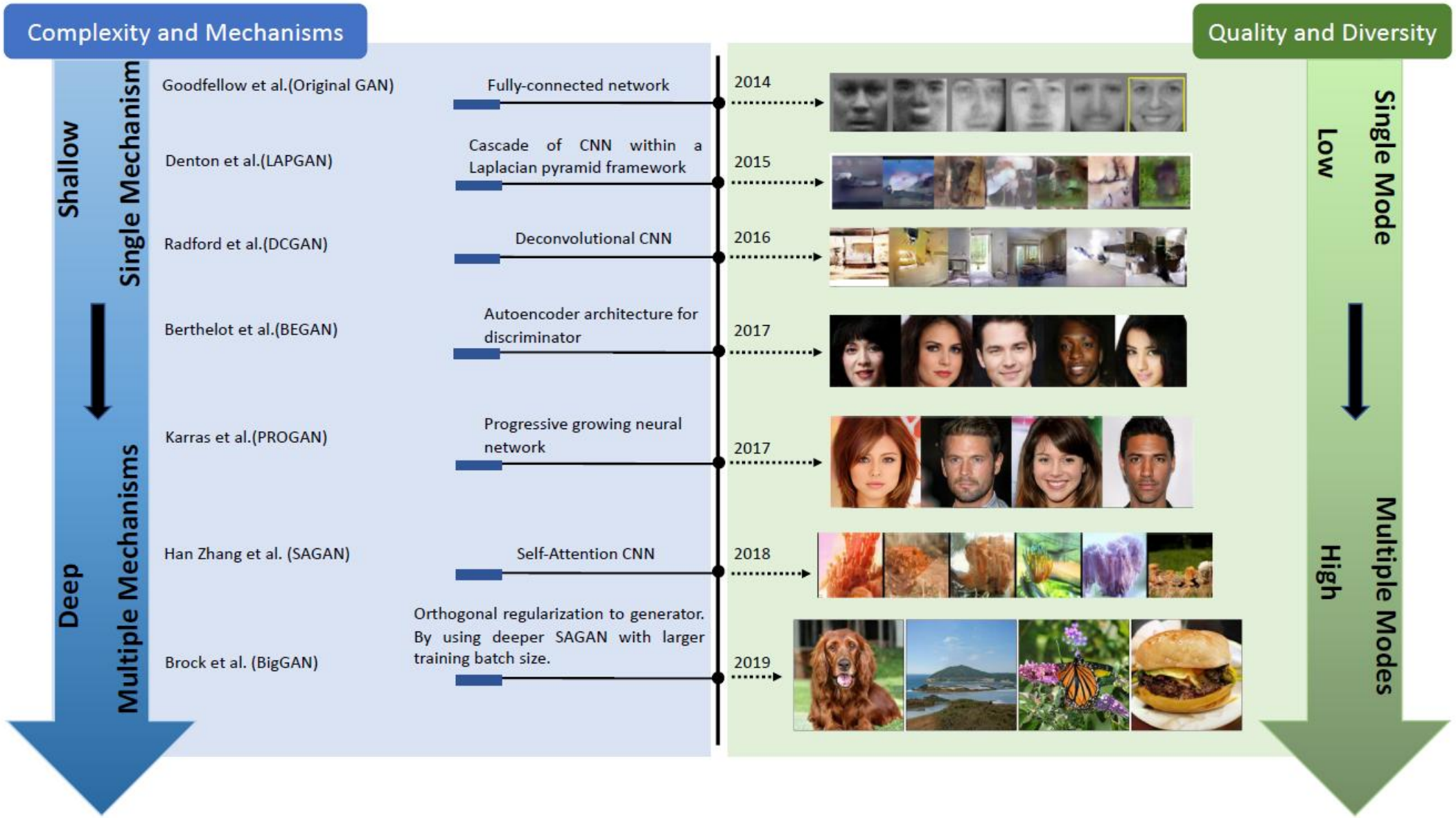
- To use Wasserstein distance
  - discriminator must be Lipschitz continuous
- WGAN
  - parameter clipping to force discriminator satisfy the Lipschitz continuity
  - parameters in the discriminator locates to the edges of clipping range
    - Low capacity
- WGAN-GP
  - use of gradient penalty to make discriminator is Lipschitz continuous
- Spectral normalization
  - constrain the discriminator under the Lipschitz continuous requirement
  - Recommended for all GAN applications

ARCHITECTURE-VARIANT GANS

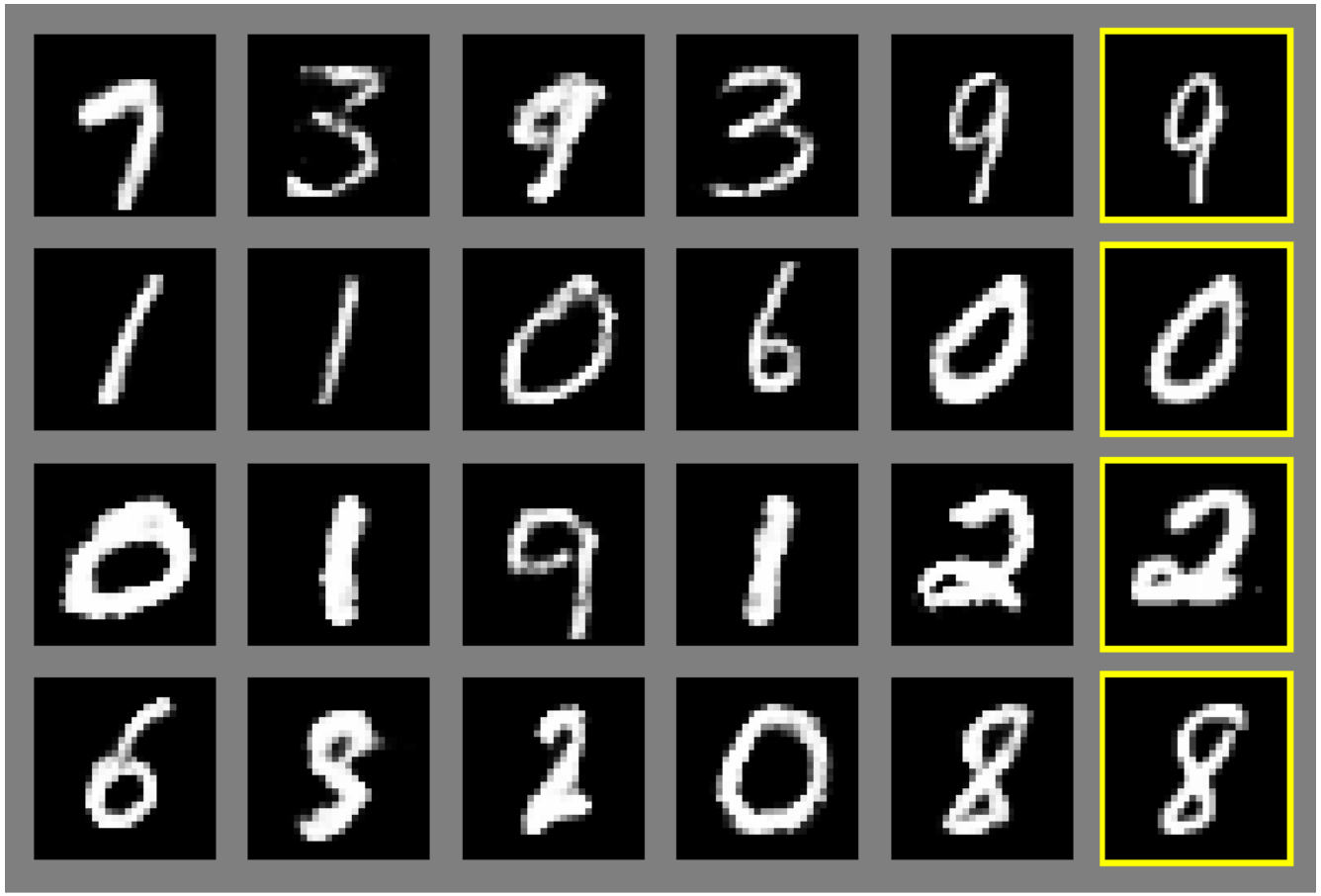
GANs基於Network 的改進

- Original GAN
  - Fully connected network
  - Bad for more complex image types
- + CNN
  - DCGAN
- +self attention
  - SAGAN (self attention GAN)
- BIGGAN
  - Based on SAGAN




Complexity and Mechanisms
Quality and Diversity
Shallow
Single Mechanism
Deep
Multiple Mechanisms
Low
Single Mode
High
Multiple Modes
Goodfellow et al.(Original GAN)
Fully-connected network
2014
Denton et al.(LAPGAN)
Cascade of CNN within a Laplacian pyramid framework
2015
Radford et al.(DCGAN)
Deconvolutional CNN
2016
Berthelot et al.(BEGAN)
Autoencoder architecture for discriminator
2017
Karras et al.(PROGAN)
Progressive growing neural network
2017
Han Zhang et al. (SAGAN)
Self-Attention CNN
2018
Brock et al. (BigGAN)
Orthogonal regularization to generator. By using deeper SAGAN with larger training batch size.
2019

No Convolution Needed to Solve Simple Tasks



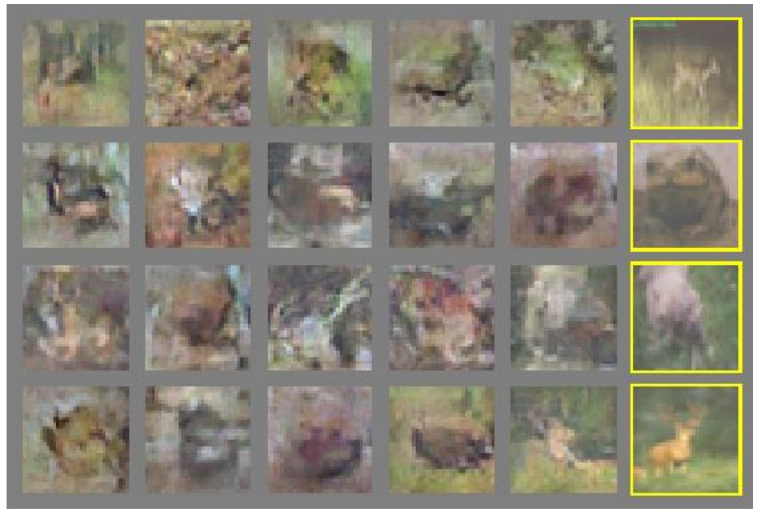
Original GAN, 2014

(Goodfellow 2018)

Depth and Convolution for Harder Tasks

Original GAN (CIFAR-10)

No convolution



One convolutional layer

DCGAN (ImageNet)

Many convolutional layers
(Radford et al, 2015)

(Goodfellow 2018)

DCGAN: deep convolutional GAN

- Architecture guidelines for stable deep convolutional GANs
  - No pooling layers (all convolutional layers)
    - Use strided convolution (discriminator) and fractional stride covolution (generator)
    - 讓網路學自己的upsampling/downsampling
  - No fully connected layers
    - Generator 的convolutional layer輸出直接連discriminator 輸入
  - 採用batch normalization，結果比較銳利
    - Batch Normalization證明了生成模型初始化的重要性，避免生成模型崩潰
    - 直接將batchnorm用在所有層會導致採樣振盪（sample oscillation）和模型不穩定
    - 生成器的輸出端不採用batchnorm，判決器的輸入端不採用batchnorm，則可以避免上述問題
  - Activation 選擇
    - Geneator 除輸出層用tanh，其餘用ReLU
    - Discriminator 用LeakyReLU，避免sparse gradient，影響訓練

https://arxiv.org/abs/1511.06434

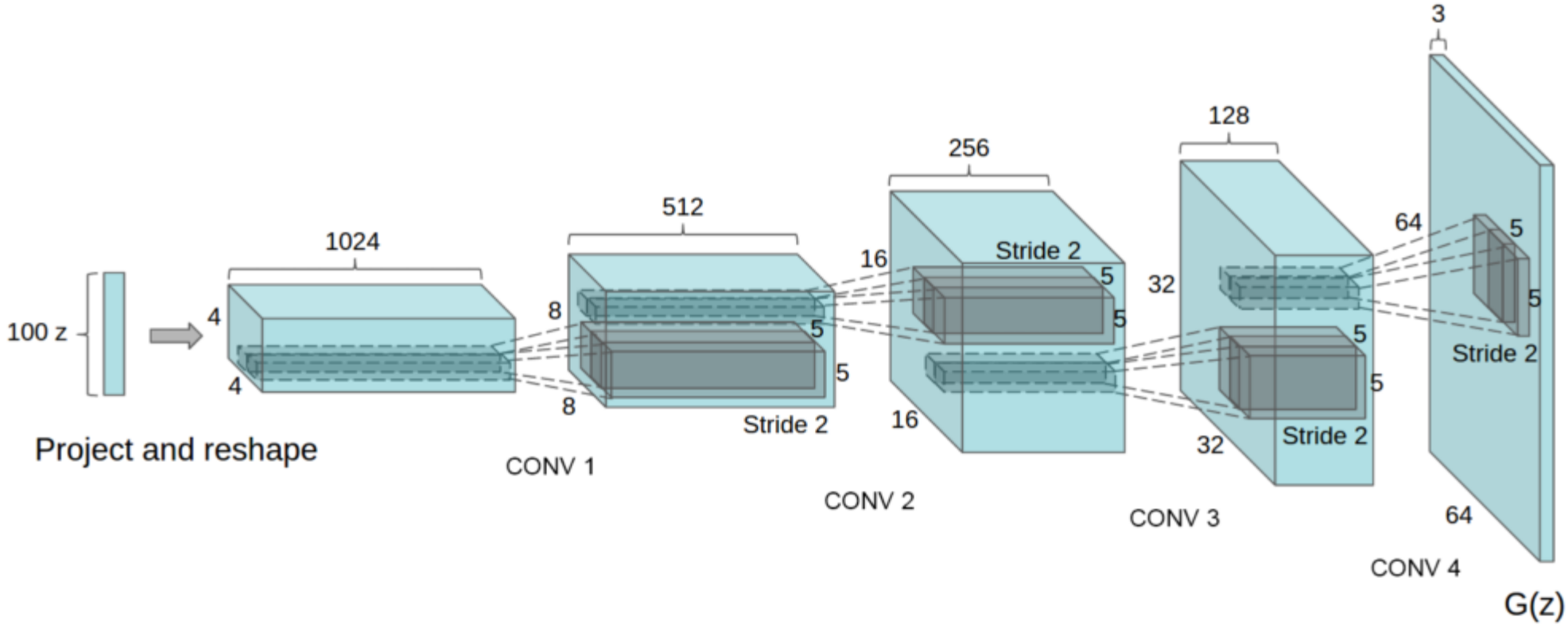


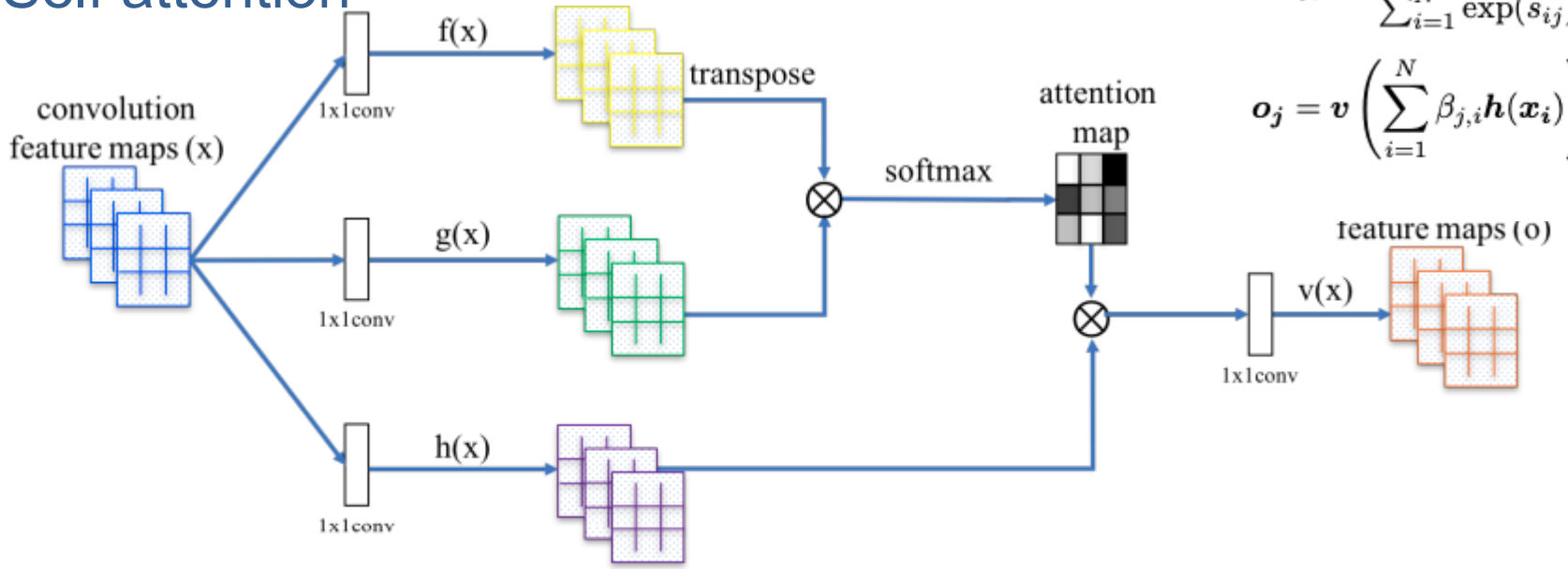
Figure 1: DCGAN generator used for LSUN scene modeling. A 100 dimensional uniform distribution Z is projected to a small spatial extent convolutional representation with many feature maps. A series of four fractionally-strided convolutions (in some recent papers, these are wrongly called deconvolutions) then convert this high level representation into a 64×64 pixel image. Notably, no fully connected or pooling layers are used.

DCGAN: Bedroom images

Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv:1511.06434 (2015).

Self Attention GAN: SAGAN

- Target problem: generate high resolution image
  - Need large receptive field
    - Deeper CNN, or fully connected: high FLOPS or large parameters
  - Self attention



$$\beta_{j,i} = \frac{\exp(s_{ij})}{\sum_{i=1}^{N} \exp(s_{ij})}, \text{ where } s_{ij} = \boldsymbol{f}(\boldsymbol{x_i})^T \boldsymbol{g}(\boldsymbol{x_j}), \quad (1)$$

$$\boldsymbol{o_j} = \boldsymbol{v}\left(\sum_{i=1}^{N} \beta_{j,i} \boldsymbol{h}(\boldsymbol{x_i})\right), \ \boldsymbol{h}(\boldsymbol{x_i}) = \boldsymbol{W_h} \boldsymbol{x_i}, \ \boldsymbol{v}(\boldsymbol{x_i}) = \boldsymbol{W_v} \boldsymbol{x_i}. \quad (2)$$

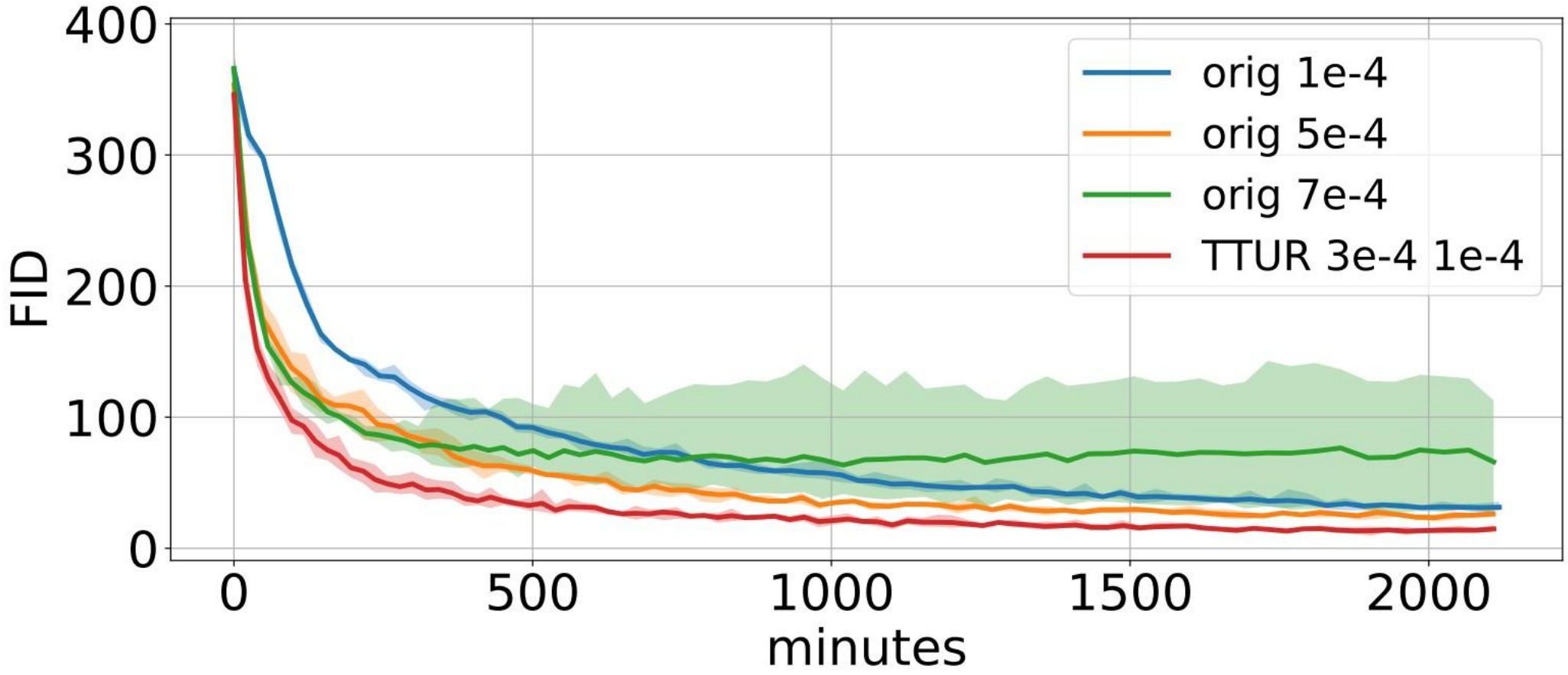
$$\boldsymbol{y_i} = \gamma \boldsymbol{o_i} + \boldsymbol{x_i},$$

Figure 2. The proposed self-attention module for the SAGAN. The ⊗ denotes matrix multiplication. The softmax operation is performed on each row.

- Two approaches to stabilize the training of GAN
- Spectral normalization
  - SAGAN為D和G加入了譜範數歸一化的方式，
  - 讓D滿足了1-lipschitz限制，同時也避免了G的參數過多導致梯度異常，使得整套訓練較為平穩和高效
- 生成器和判別器使用單獨的學習率（TTUR）
  - 判別器的正則化通常會減慢GAN學習過程。實際上，使用正則化判別器的方法通常在訓練期間每個生成器需要多個更新步驟（例如，5個）
  - 使用TTUR來補償正則化判別器中慢學習的問題，使得對於每個判別器步驟使用更少的生成器步驟成為可能

Two-Timescale Update Rule

SA-GAN 對生成器和判別器同時做spectral normalization；
在生成器和判別器中使用不同的學習率（TTUR）。
這樣做可以解決判別器正則化過程中學習速度過慢的問題



Spectral Normalization

- 對於多次神經網絡*f* 來說，*f* 一般可以看作是一個複合函數，即許多函數的嵌套操作，此時，對於*f* 的Lipschitz常數滿足

$$\|f\|_{\text{Lip}} \leq \|(\boldsymbol{h}_L \mapsto W^{L+1}\boldsymbol{h}_L)\|_{\text{Lip}} \cdot \|a_L\|_{\text{Lip}} \cdot \|(\boldsymbol{h}_{L-1} \mapsto W^{L}\boldsymbol{h}_{L-1})\|_{\text{Lip}}$$
$$\cdots \|a_1\|_{\text{Lip}} \cdot \|(\boldsymbol{h}_0 \mapsto W^{1}\boldsymbol{h}_0)\|_{\text{Lip}} = \prod_{l=1}^{L+1} \|(\boldsymbol{h}_{l-1} \mapsto W^{l}\boldsymbol{h}_{l-1})\|_{\text{Lip}} = \prod_{l=1}^{L+1} \sigma(W^l). \quad (7)$$

- 為了使discriminator方程*f* 的Lipschitz常數不超過1，所以我們只需要保證卷積操作時1-Lipschitz的。因此對於卷積層的權重W，我們有

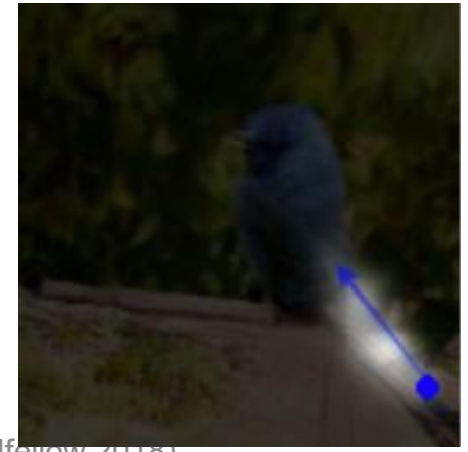
$$\bar{W}_{SN}(W) := \frac{W}{\sigma(W)}$$

其中σ(W)表示W的spectral norm，即最大的奇異值。直接使用SVD來計算，計算量太大，使用迭代法可以加速計算

Self-Attention GAN (SA-GAN)




Use layers from Wang et al 2018

(Goodfellow 2018)


goldfish
(**44.4**, 58.1)
indigo
bunting
(**53.0**, 66.8)
redshank
(**48.9**, 60.1)
saint
bernard
(**35.7**, 55.3)

BigGAN: Large Scale GAN Training for High Fidelity Natural Image Synthesis

BIGGAN 產生更逼真的圖片

- Need large scale GAN
- Large batch, wide channels

IS（反應圖像的生成質量），FID（注重生成的多樣性）

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

Table 1: Fréchet Inception Distance (FID, lower is better) and Inception Score (IS, higher is better) for ablations of our proposed modifications. *Batch* is batch size, *Param* is total number of parameters, *Ch.* is the channel multiplier representing the number of units in each layer, *Shared* is using shared embeddings, *Skip-z* is using skip connections from the latent to multiple layers, *Ortho.* is Orthogonal Regularization, and *Itr* indicates if the setting is stable to 10^6 iterations, or it collapses at the given iteration. Other than rows 1-4, results are computed across 8 random initializations.

BigGAN

- Base
  - SA-GAN + Hinge Loss
  - G + class conditional BN,
  - D + projection
- Large scale training
  - Large batch size (8X over prior art)
  - Large model (2x~4X over prior art)
  - 512x512 image generation
- Model
  - ResNet + skip-z + shared class embedding

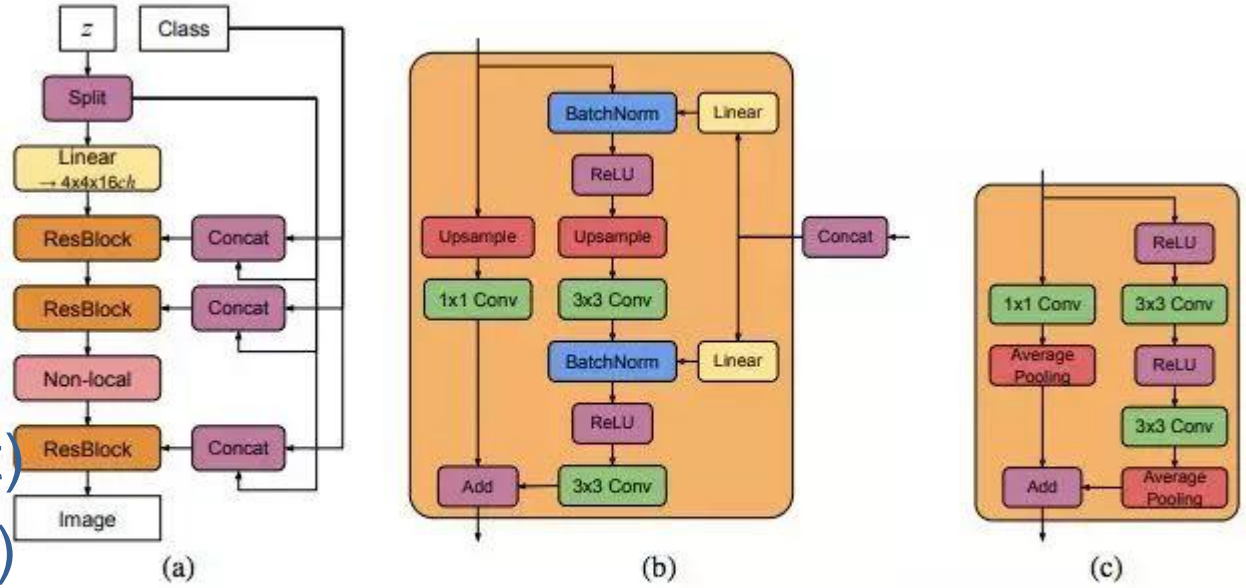


Figure 15: (a) A typical architectural layout for BigGAN's **G**; details are in the following tables. (b) A Residual Block (*ResBlock up*) in BigGAN's **G**. (c) A Residual Block (*ResBlock down*) in BigGAN's **D**.

https://github.com/ajbrock/BigGAN-PyTorch

- Single shared class embedding in G
  - to reduce parameter size and speedup training
- direct skip connections (skip-z)
  - from the noise vector z to multiple layers of G rather than just the initial layer to directly influence features at different resolutions and levels of hierarchy
  - Hierarchical Latent Space (e.g. Z = 12 34 56, => 12 (input), 34 + c, 56+c)
- Ortho.: Orthogonal Regularization
  - Brock's orthogonal regularization:
    - 希望 W 與輸入的 Matrix 盡量可以 Orthogonal，避免訊號消失或是訊號爆炸的問題
    - 要讓所有的 Singular value 都趨近於 1
  - SNGAN: 要讓最大的 Singular value 去限縮為 1, conflict with orthogonal regu
  - Modified orthogonal regularization:
    - 只要不要更動到對角線的 Singular value 就好
    - 讓 Truncation trick 在 60% 的模型上可用

Gradient Normalization for Generative Adversarial Networks

- Imposing a Lipschitz constraint on the discriminator
  - Model- or modulewise constraint
  - Sampling-based or non-sampling based constraint
  - Hard or soft constraint
- Gradient normalization fits above constraints
  - enforce the gradient norm bounded by 1 for the discriminator model through dividing the outputs by the gradient norm of the discriminator.

$$\hat{f}(x) = f(x) / \left(\|\nabla_x f(x)\| + |f(x)| \right)$$

$$\|\nabla_x \hat{f}(x)\| = \left| \frac{\|\nabla f\|}{\|\nabla f\| + |f|} \right|^2 \leq 1. \qquad (12)$$

Table 4: Inception Score and FID with conditional image generation on CIFAR-10.

Method	Inception Score↑	FID(test)↓
BigGAN	9.22	14.73
BigGAN-CR [35]	-	11.48
(our) GN-BigGAN	9.22±.13	**10.05±.23**

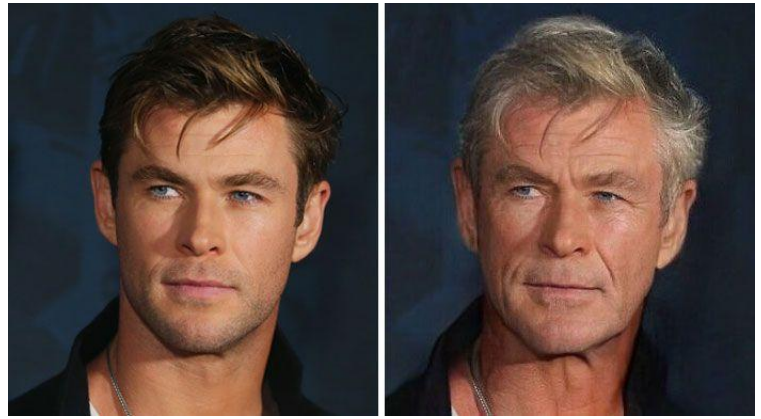
Figure 1: Generated samples on CelebA-HQ 256×256. FID=7.67.

Figure 2: Generated smaples on LSUN Church Outdoor 256×256. FID=5.405.

LATENT SPACE IN GANS

Latent Space in GANs

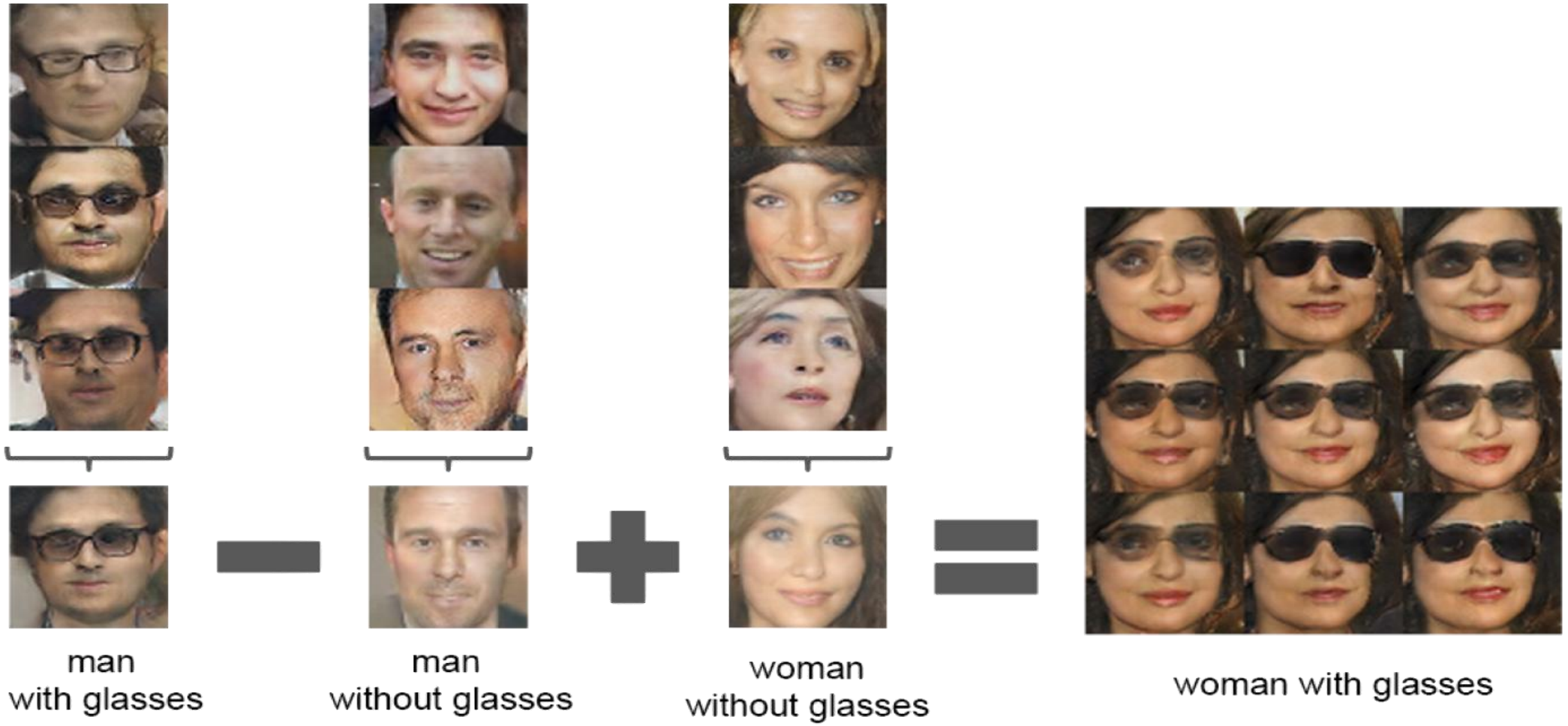
- + condition: GAN probability changed to conditional probability
  - cGAN: conditional GAN
- Application
  - Picture as condition: pix2pix
  - Unpaired data: CycleGAN
- AC-GAN
- InfoGAN



CycleGAN for Age Conversion

https://www.boredpanda.com/old-age-filter-photos-faceapp/

Latent vectors capture interesting patterns…



Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv:1511.06434 (2015).

Conditional GAN

指定label，產生對應的圖

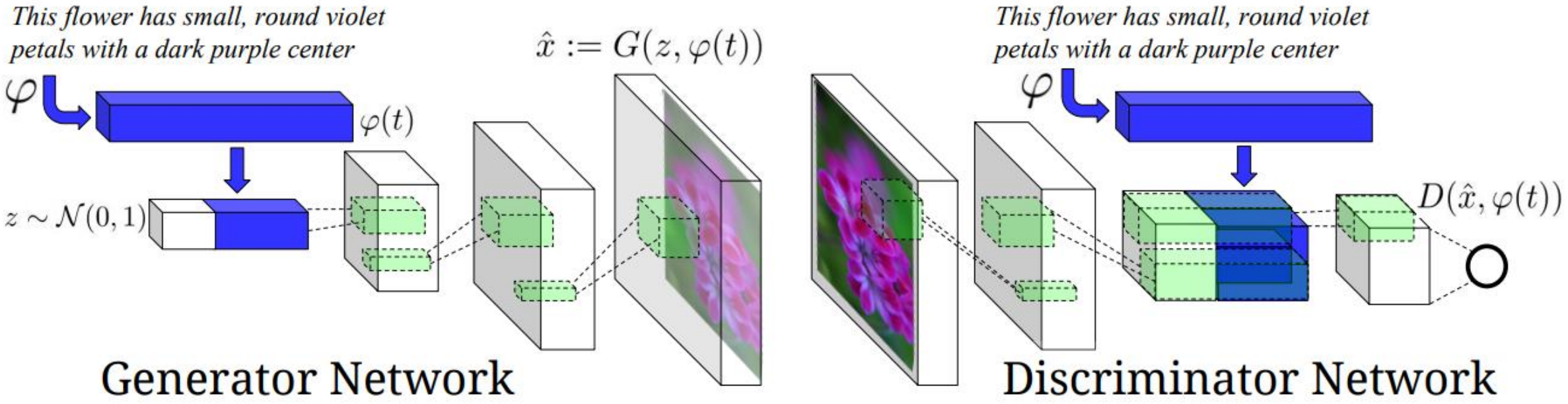
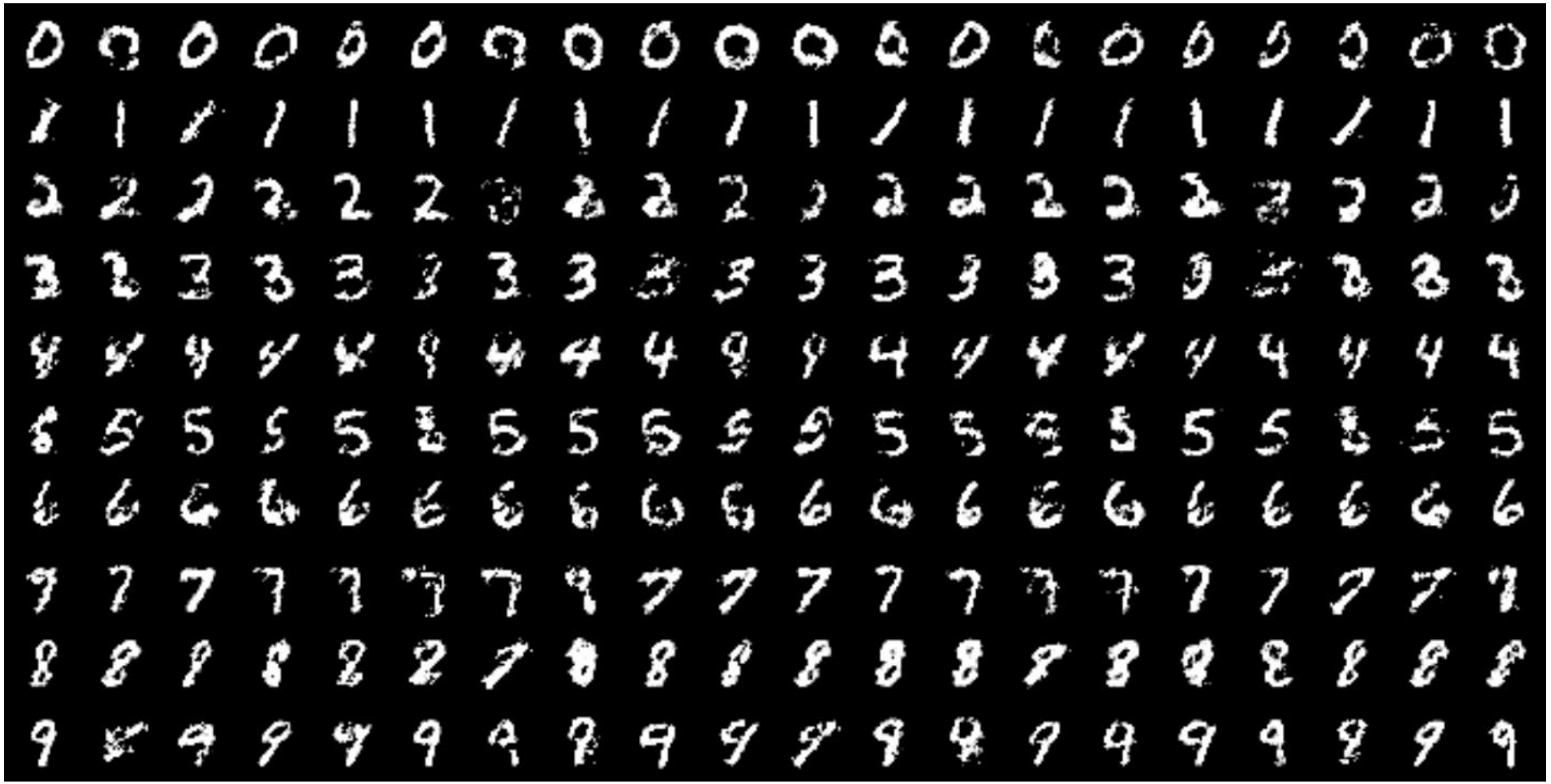


Figure 2. Our text-conditional convolutional GAN architecture. Text encoding $\varphi(t)$ is used by both generator and discriminator. It is projected to a lower-dimensions and depth concatenated with image feature maps for further stages of convolutional processing.

Class-Conditional GANs



(Mirza and Osindero, 2014)

(Goodfellow 2018)

Conditional GANs

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x}|\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}|\boldsymbol{y})))]. \quad (2)$$

```
disc_condition_input = Input(shape=(10,))
gen_condition_input = Input(shape=(10,))

def get_discriminator(input_layer, condition_layer):
    hid = LeakyReLU(alpha=0.1)(hid)
    hid = Flatten()(hid)
    merged_layer = Concatenate()([hid, condition_layer])
    hid = Dense(512, activation='relu')(merged_layer)
    out = Dense(1, activation='sigmoid')(hid)

def get_generator(input_layer, condition_layer):
    merged_input = Concatenate()([input_layer, condition_layer])
    hid = Dense(128 * 8 * 8, activation='relu')(me
    hid = BatchNormalization(momentum=0.9)(hid)
```

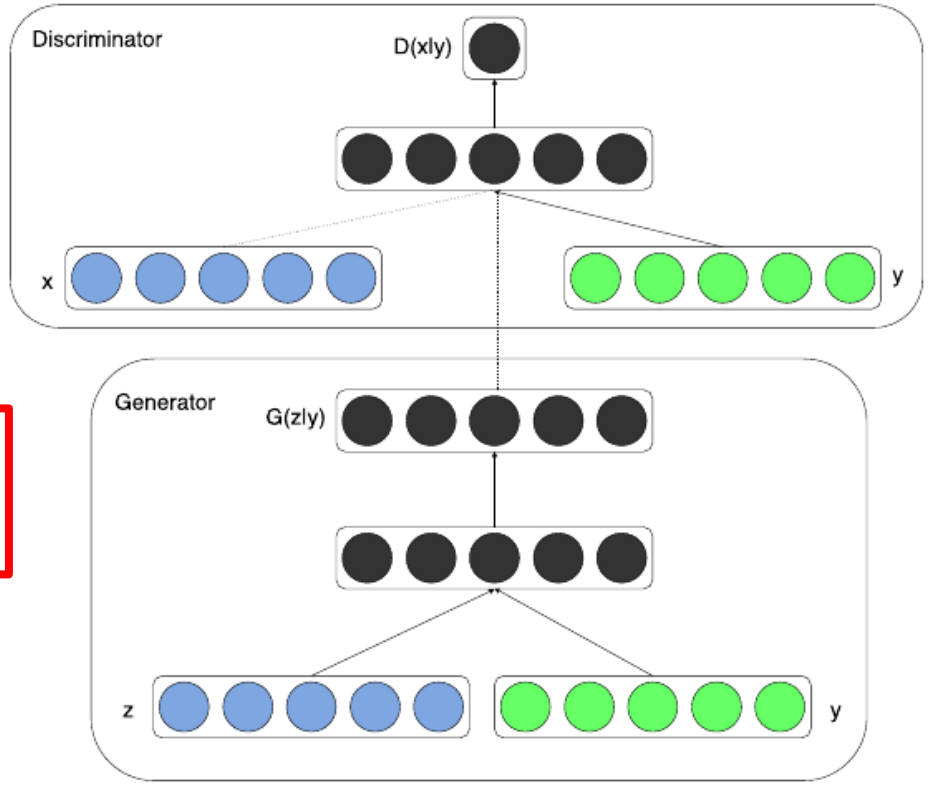
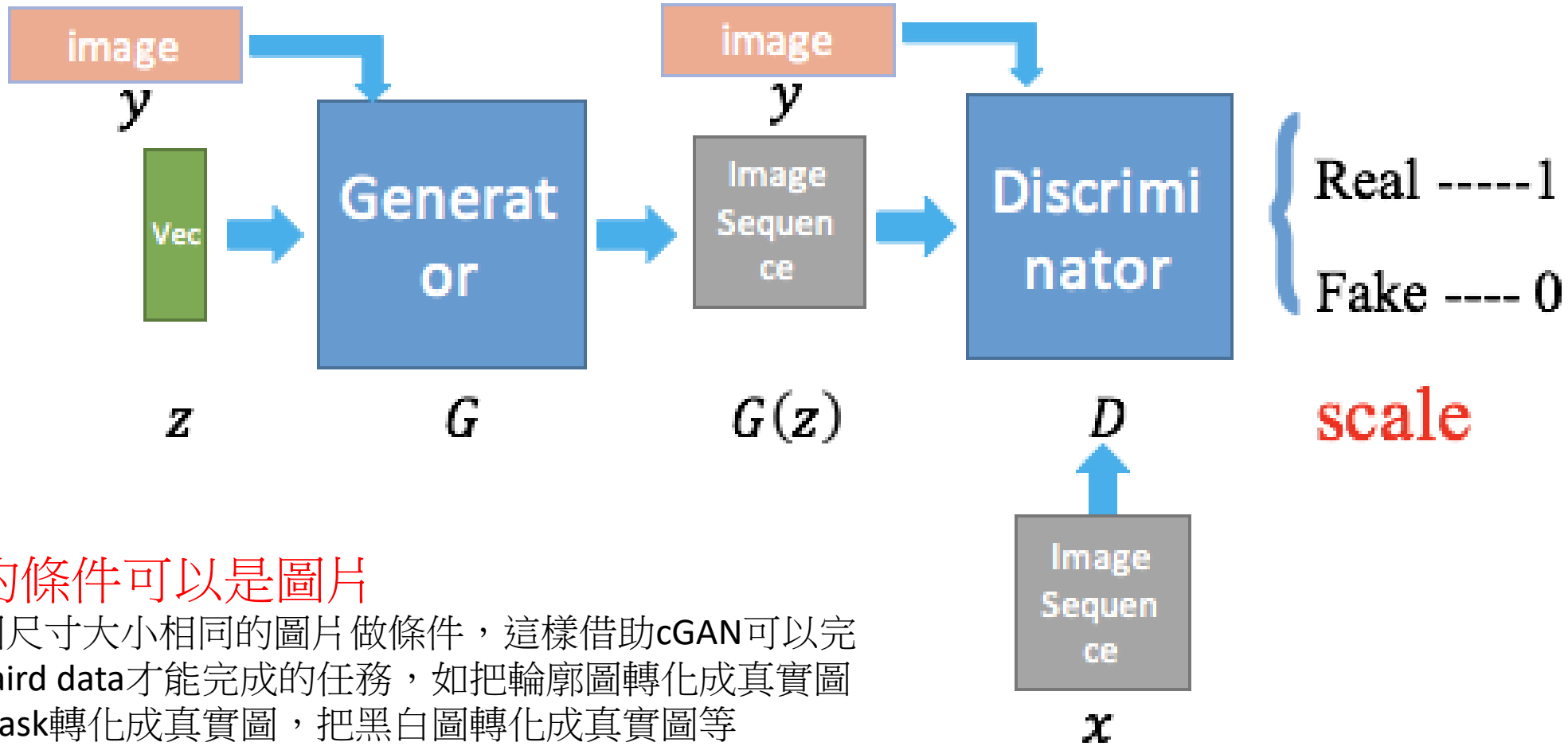


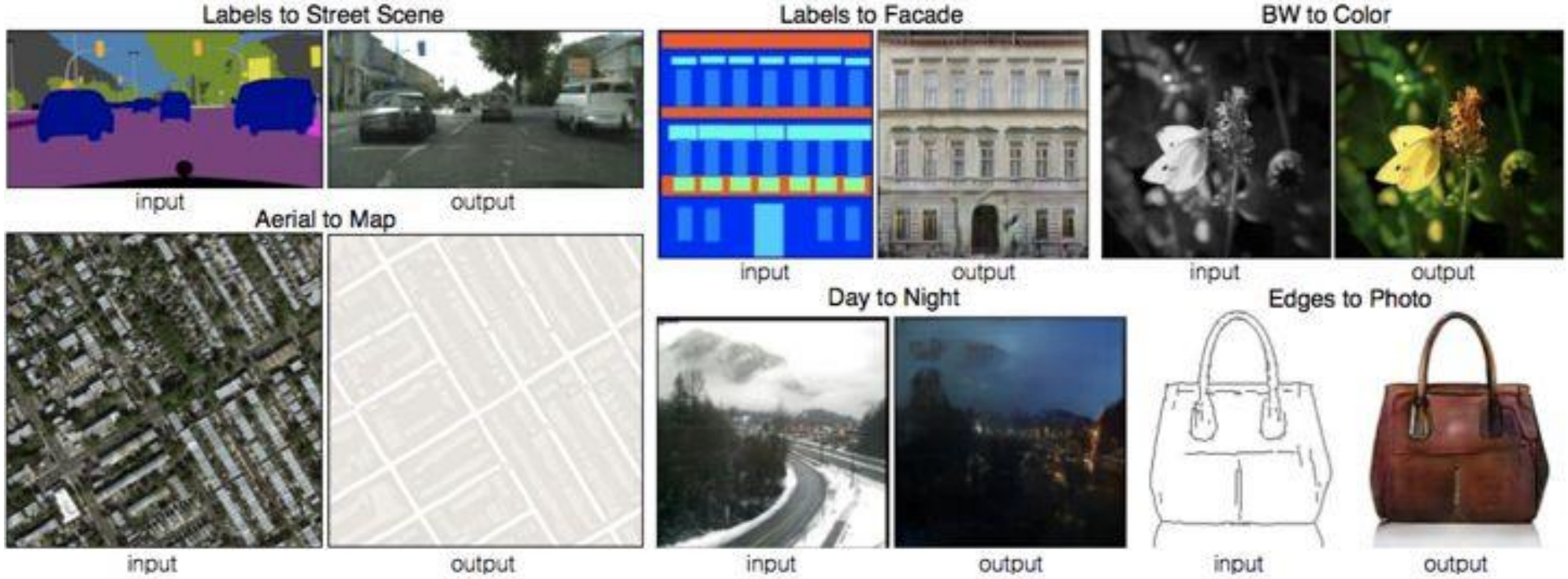
Figure 1: Conditional adversarial net

Pix2Pix



cGAN的條件可以是圖片

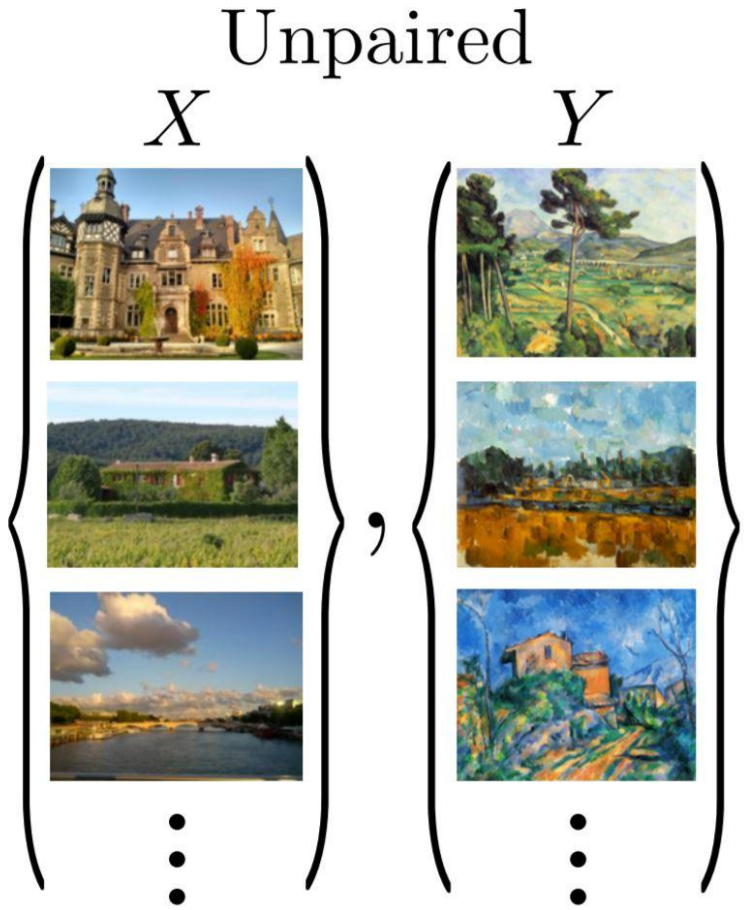
用跟原圖尺寸大小相同的圖片 做條件，這樣借助cGAN可以完成一些paird data才能完成的任務，如把輪廓圖轉化成真實圖片，把mask轉化成真實圖，把黑白圖轉化成真實圖等


Labels to Street Scene
input
output
Labels to Facade
input
output
BW to Color
input
output
Aerial to Map
input
output
Day to Night
input
output
Edges to Photo
input
output


Input
Ground truth
Output
Input
Ground truth
Output

CycleGAN: Unpaired Image-to-Image Translation

- Paired image to image translation needs a lot of training data
- What if we don't have such data



Cycle GAN

直接用conditional GAN

Domain X

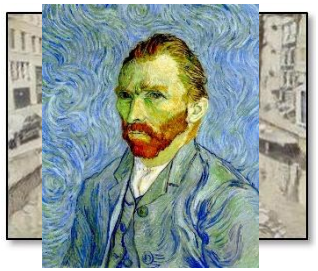
Domain Y

Domain X

ignore input

$G_{X \to Y}$

Become similar to domain Y



Not what we want

D_Y

scalar

Input image belongs to domain Y or not

Domain Y

https://arxiv.org/abs/1703.10593

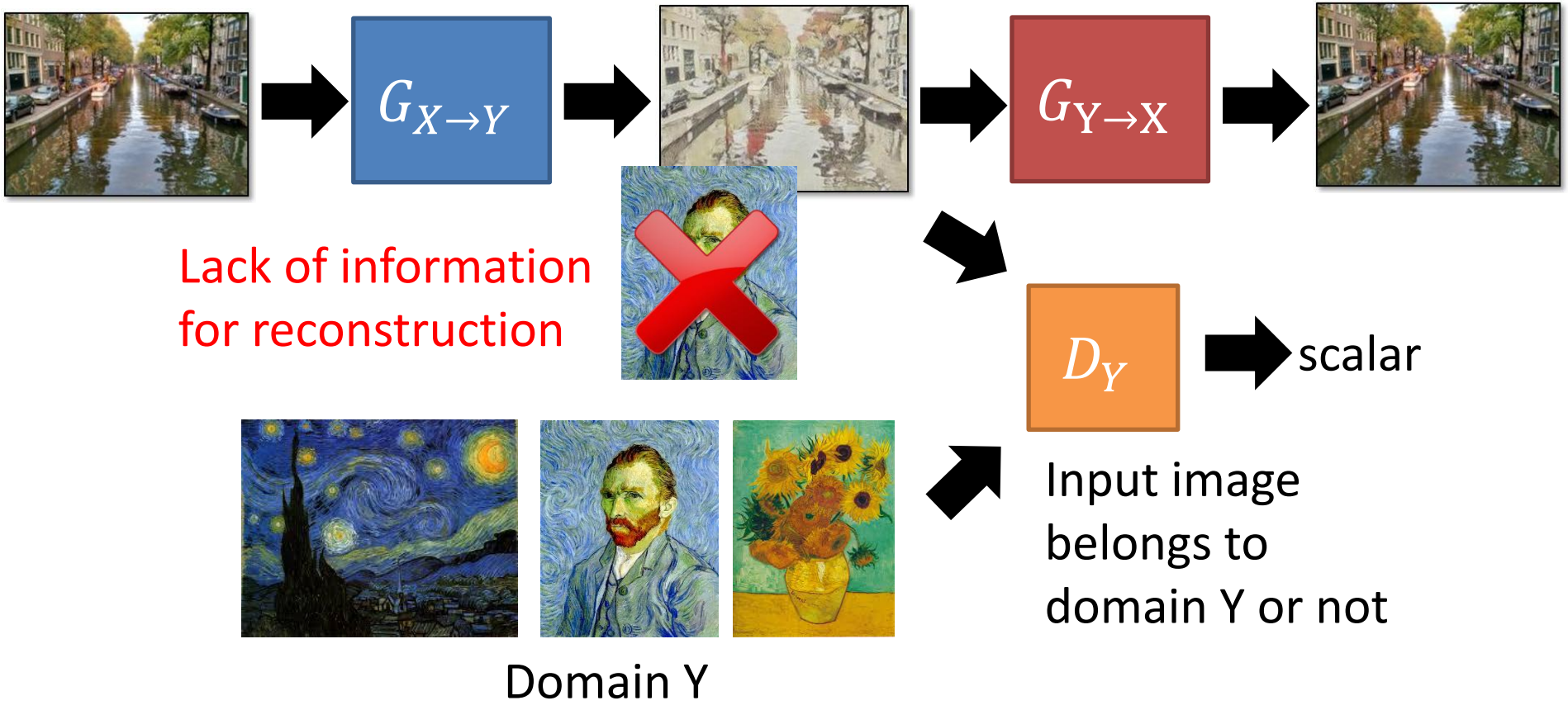
https://junyanz.github.io/CycleGAN/

Source: Hung-Yi Lee, 2017

Cycle GAN

- **Cycle consistency**

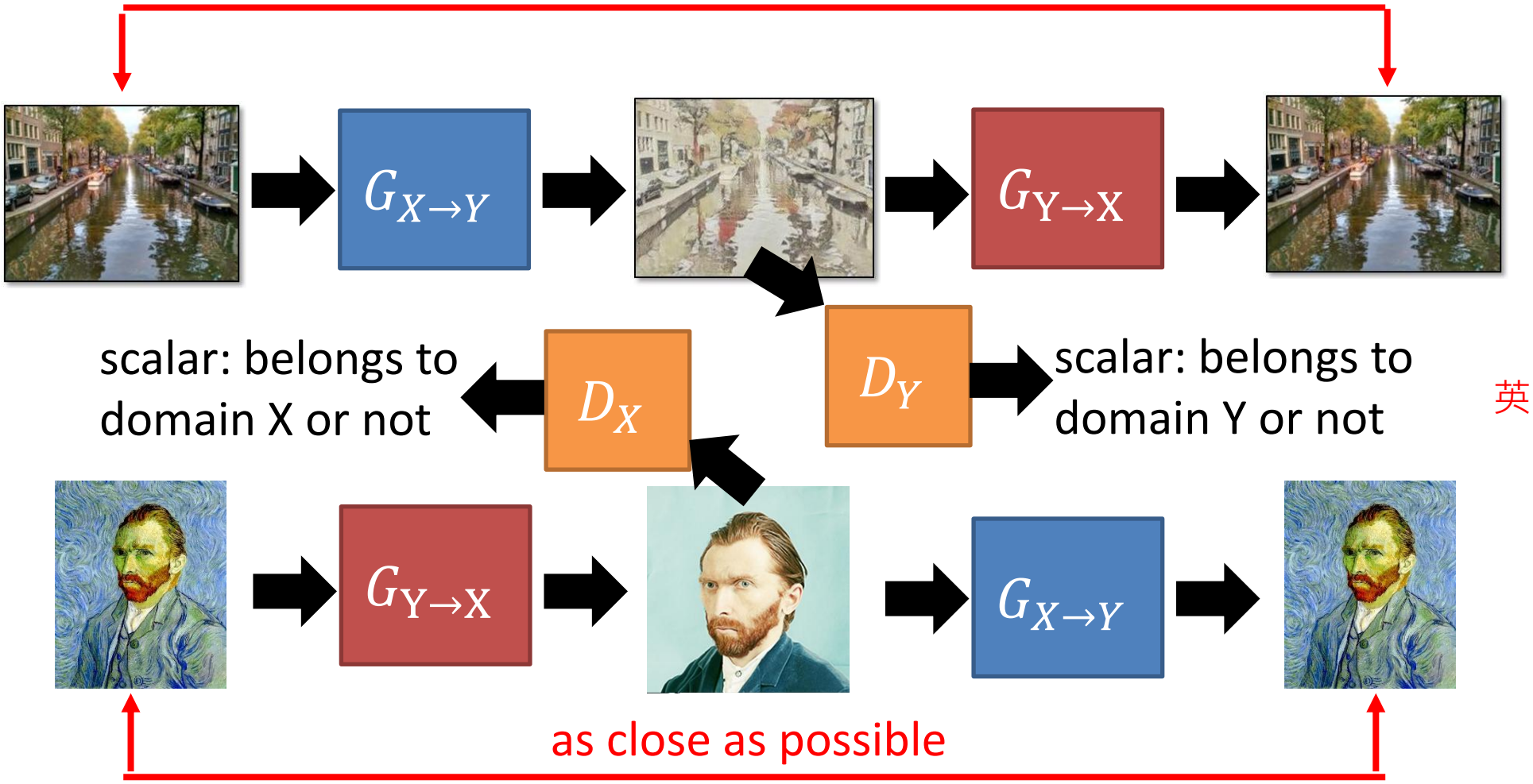
as close as possible



Source: Hung-Yi Lee, 2017

Cycle GAN

- Cycle consistency



英翻中 中翻英

Source: Hung-Yi Lee, 2017

CycleGAN: Cycle Consistency

- 轉換失敗例子

CycleGAN, a Master of Steganography

- AI 「偷吃步」
  - 透過將特徵巧妙編碼到另一張圖片，CycleGAN 讓衛星地圖的細節變成在街道地圖只有 AI 可見的「小抄」

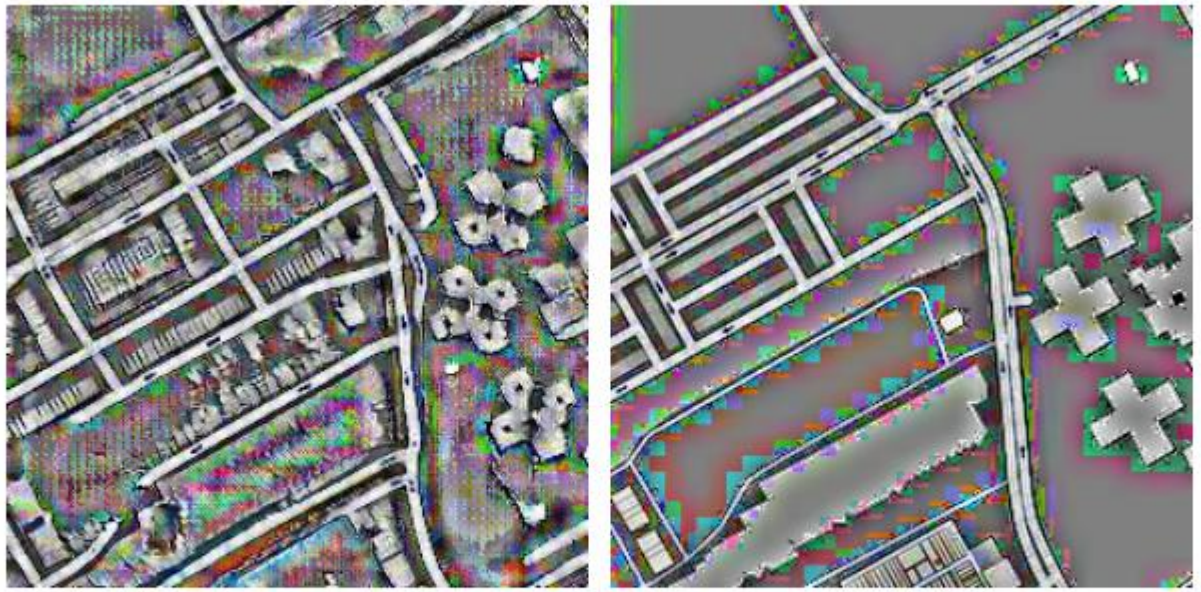
原圖　　轉換圖　　再轉換圖

(a) Aerial photograph: x.　(b) Generated map: Fx.　(c) Aerial reconstruction: GFx.

Figure 1: Details in x are reconstructed in GFx, despite not appearing in the intermediate map Fx.

https://arxiv.org/pdf/1712.02950.pdf

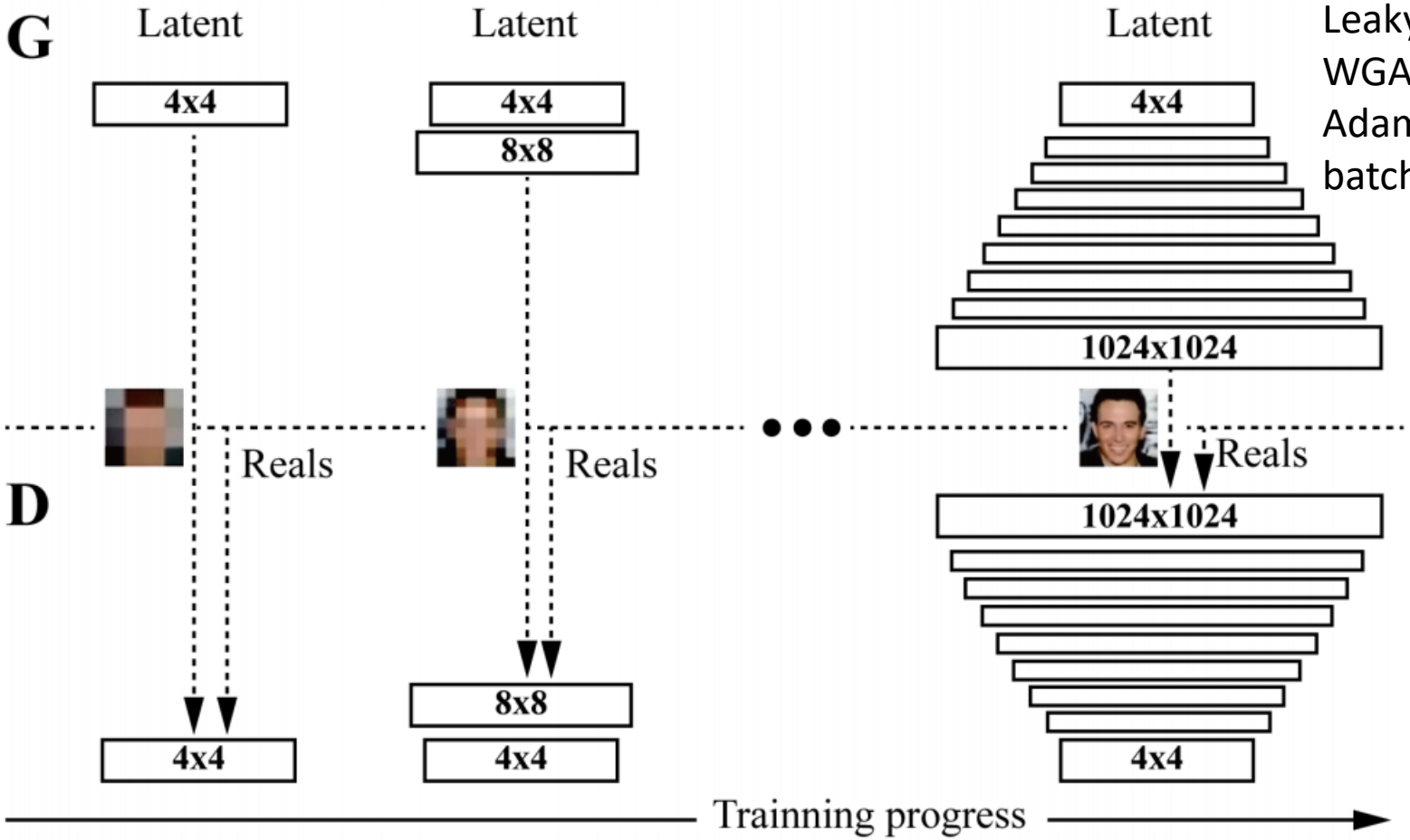
- 用adaptive histogram equalization 放大細節
  - 產生的影像有額外的高頻訊號



(a) Generated map. (b) Training map, for comparison.

Figure 2: Maps with details amplified by adaptive histogram equalization. Information is present in the generated map even in regions that appear empty to the naked eye.

Progressive GAN

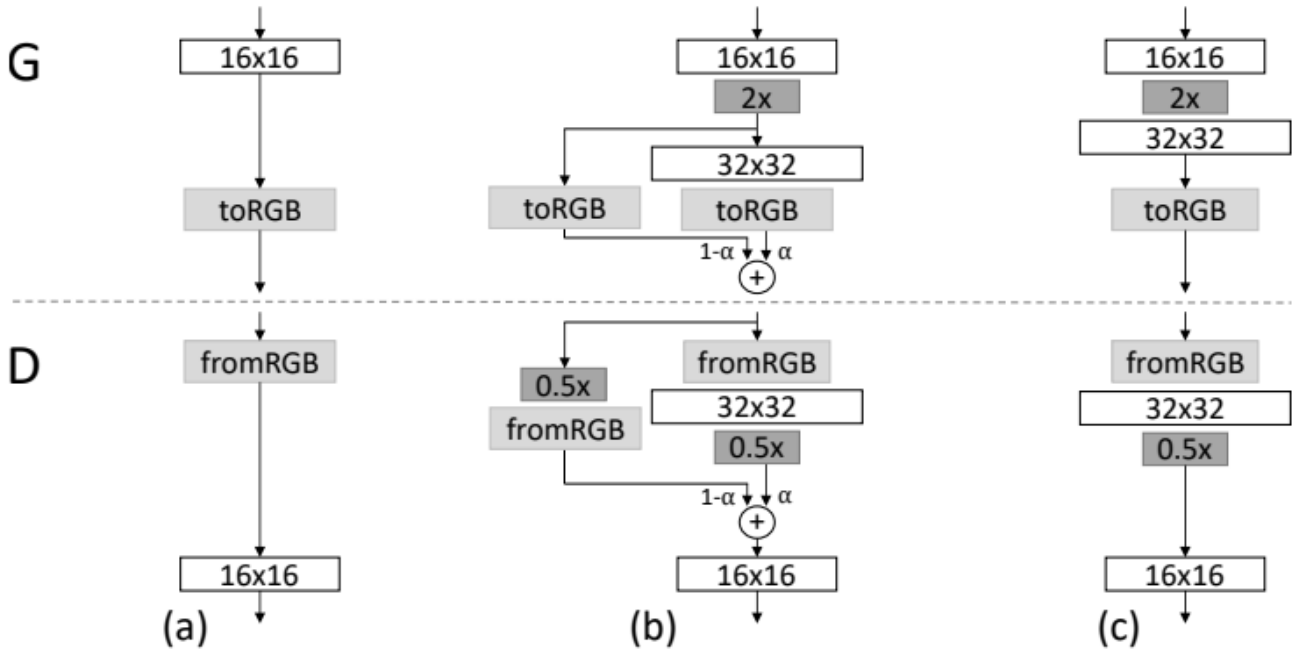


pixelwise normalization after 3x3 conv
Leaky ReLU
WGAN-GP
Adam
batch size decreasing

PGGAN 隨著訓練進行網絡會不斷加深，網絡結構是在不斷改變的。這樣做最大的好處就是，PGGAN 大部分的迭代都在較低分辨率下完成，訓練速度比傳統GANs 提升了2-6 倍。

Fig. 12. Progressive growing step for PROGAN during the training process. Training starts with 4×4 pixels image resolution. With the training step growing, layers are incrementally added to G and D which increases the resolution for the generated images. All existing layers are trainable throughout the training stage. Figure regenerated from [78].

Smooth Fading



從4 × 4 的輸出變為8 × 8 的輸出的過程中，網絡層數的突變會造成GANs 的急劇不穩定，使得GANs 需要花費額外的時間從動盪狀態收斂回平穩狀態，這會影響模型訓練的效率。為了解決這一問題，PGGAN 提出了平滑過渡技術。

Figure 2: When doubling the resolution of the generator (G) and discriminator (D) we fade in the new layers smoothly. This example illustrates the transition from 16 × 16 images (a) to 32 × 32 images (c). During the transition (b) we treat the layers that operate on the higher resolution like a residual block, whose weight α increases linearly from 0 to 1. Here $2\times$ and $0.5\times$ refer to doubling and halving the image resolution using nearest neighbor filtering and average pooling, respectively. The toRGB represents a layer that projects feature vectors to RGB colors and fromRGB does the reverse; both use 1×1 convolutions. When training the discriminator, we feed in real images that are downscaled to match the current resolution of the network. During a resolution transition, we interpolate between two resolutions of the real images, similarly to how the generator output combines two resolutions.

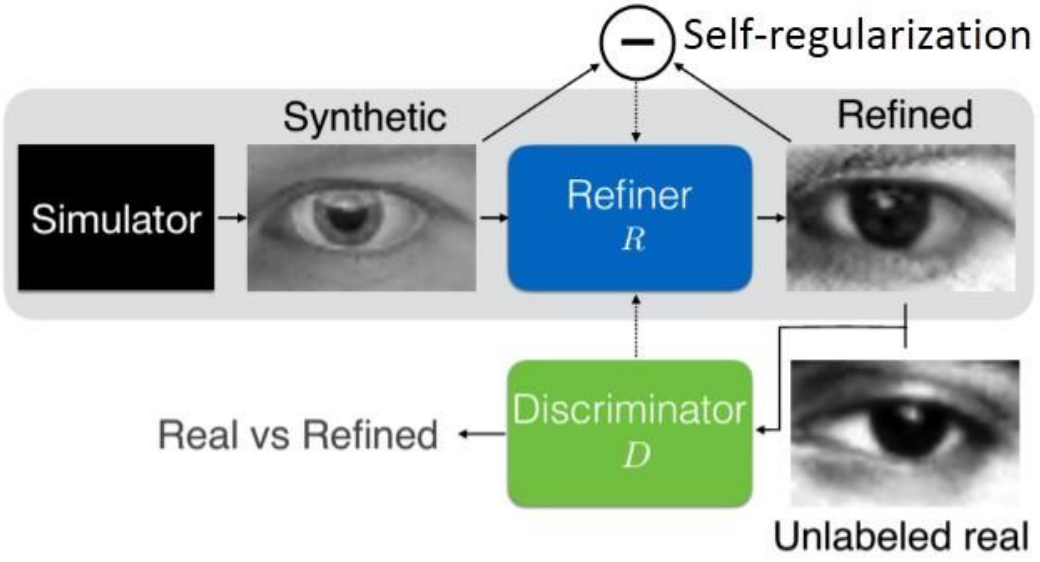
GAN APPLICATIONS

DeOldfy

- combine GANs and feature losses
  - created a black and white ImageNet, and then trained a model to re-colorize it

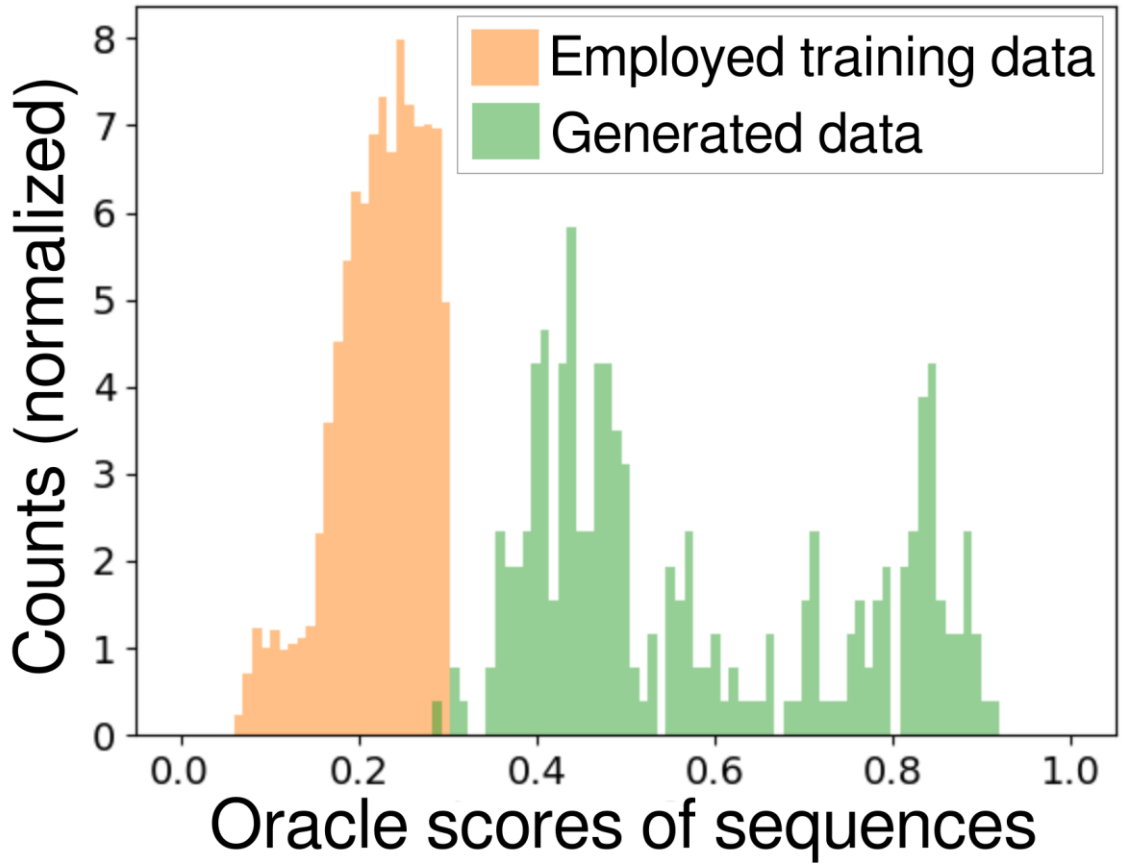
GAN的應用

- 產生更多測試資料，幫助supervised learning
- 人眼檢測
  - 利用無標籤的真實圖像，來精細化合成圖像使其看起來更加真實



Gou C, Wu Y, Wang K, Wang K F, Wang F Y, Ji Q. **A joint cascaded framework for simultaneous eye detection and eye state estimation**. Pattern Recognition, 2017, 67: 23-31

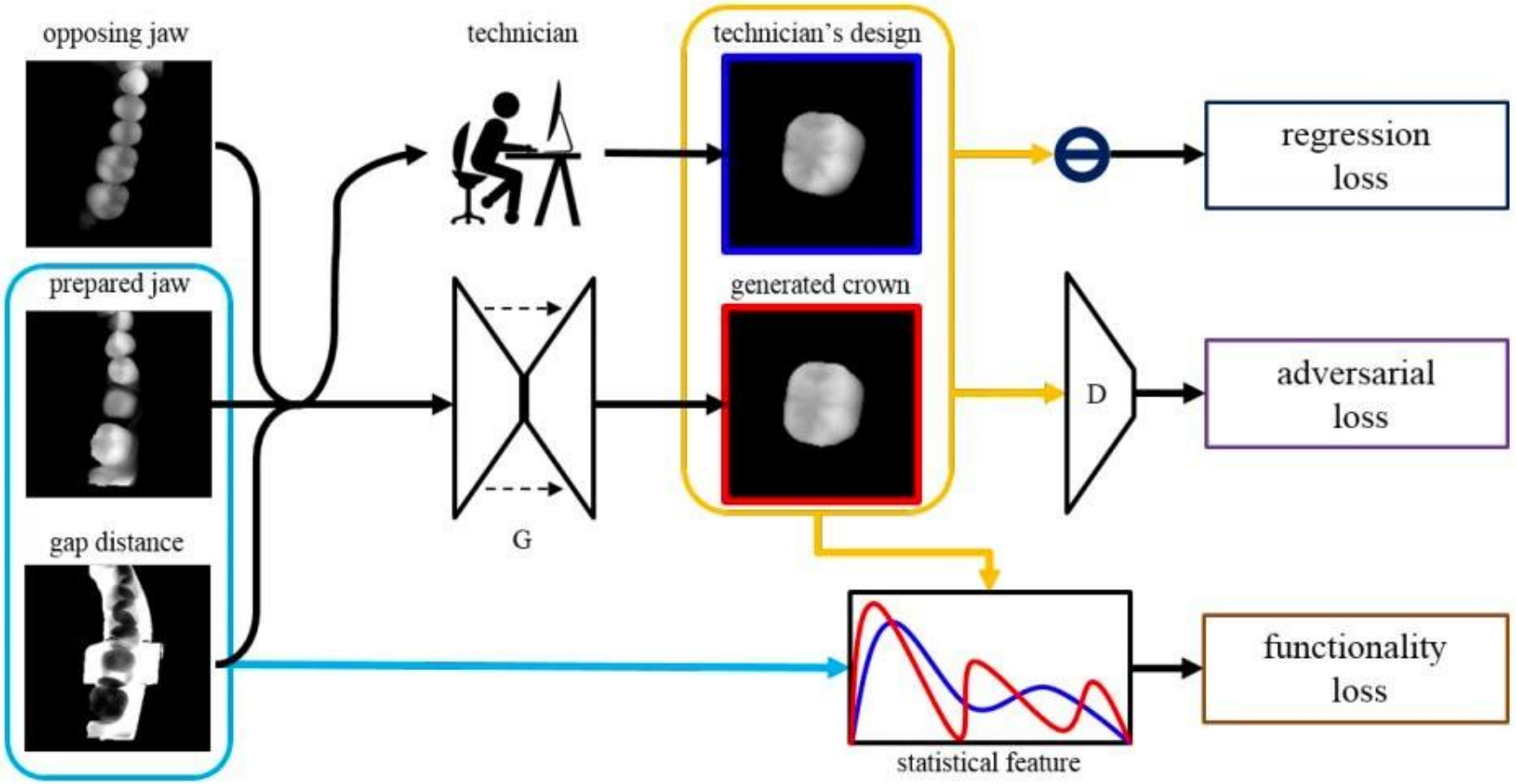
Designing DNA to optimize protein binding



(Killoran et al, 2017)

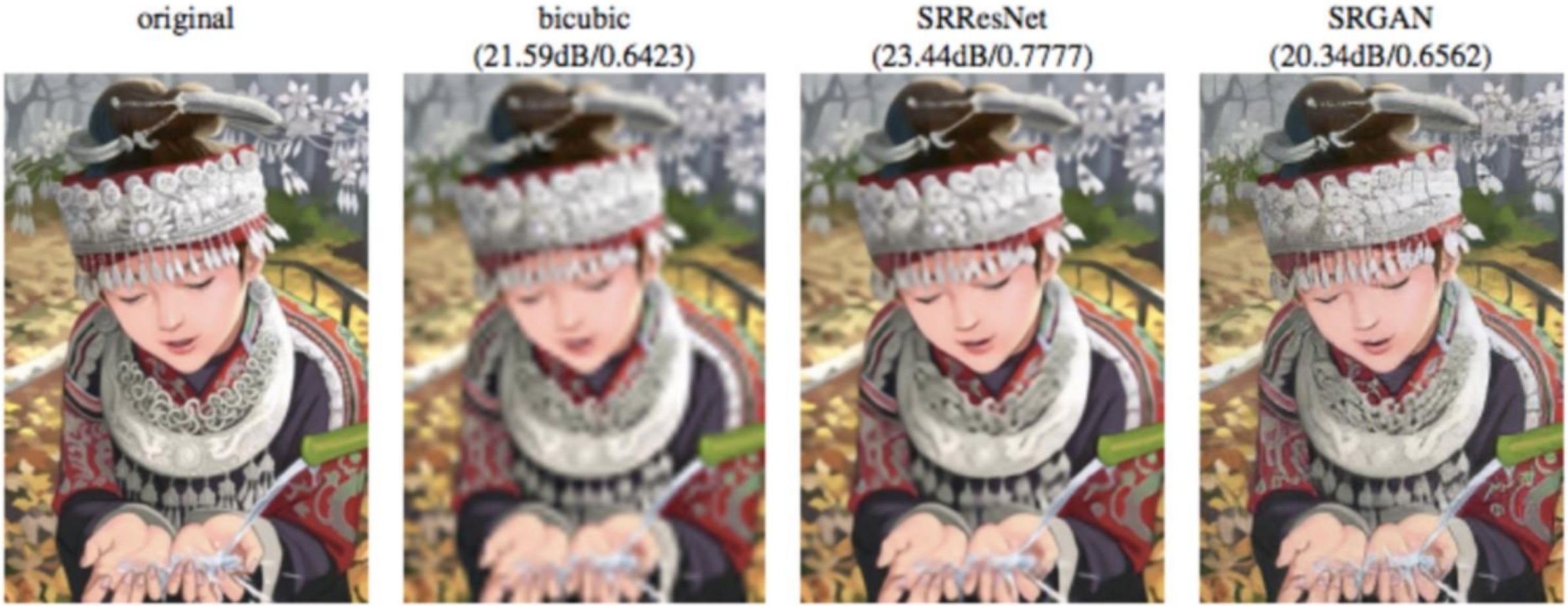
(Goodfellow 2018)

Personalized GANufacturing



(Hwang et al 2018)

Super-resolution



Reference

- 探索及應用生成對抗網路
- 一日搞懂生成式對抗網路
- GAN papers categorized by application
  - https://github.com/zhangqianhui/AdversarialNetsPapers
- 100+ different GAN variations: GAN zoo
  - https://github.com/hindupuravinash/the-gan-zoo

OPTION

Self-Supervised GAN Compression

Table 1. GAN compression comparison (network pruning)

	Generator(s)		Discriminator		Loss Terms				Results	
Technique	Compressed	Init Scheme	Init Scheme	Fixed	L-Gc	L-Dc	L-Go	L-Do	Qualitative	FID Score
(a) No Compression	Dense	Random	Dense,Random	No	-	-	Yes	Yes	Good	6.113
(b) Self-Supervised **(ours)**	Dense,Sparse	From Dense	Dense,Pretrained	No	Yes	Yes	Yes	Yes	Good	6.929
(c) Small & Dense Network	Dense	Random	Dense,Random	No	-	-	Yes	Yes	Mode collapse	72.821
(d) One-shot Pruning & Fine-Tuning	Sparse	From Dense	Dense,Pretrained	No	Yes	Yes	-	-	Facial artifacts	24.404
(e) Gradual Pruning & Fine-Tuning	Sparse	From Dense	Dense,Random	No	Yes	Yes	-	-	Facial artifacts	35.677
(f) Gradual Pruning during Training	Sparse	Random	Dense,Random	No	Yes	Yes	-	-	No faces	84.941
(g) One-shot Pruning & Distillation	Dense,Sparse	From Dense	-	-	Yes	-	Yes	-	Mode collapse	45.461
(h) (d) & Distillation	Dense,Sparse	From Dense	Dense,Pretrained	No	Yes	Yes	Yes	-	Color artifacts	38.985
(i) (g) & Fix Original Loss	Dense,Sparse	From Dense	Dense,Pretrained	Yes	Yes	Yes	-	-	Facial artifacts	15.182
(j) Adversarial Learning	Dense,Sparse	Random	Dense,Random	No	Yes	Yes	Yes	Yes	Mode collapse	92.721
(k) Knowledge Distillation	Dense,Sparse	From Dense	Dense,Random	No	Yes	-	Yes	Yes	Mode collapse	103.094
(l) Distill Intermediate (LIT)	Dense,Sparse	From Dense	Dense,Pretrained	Yes	-	-	-	-	Mode collapse	61.150
(m) E-M Pruning	Dense,Sparse	From Dense	Sparse,Pretrained	No	Yes	Yes	Yes	-	Color artifacts	159.767
(n) G & D Both Pruning	Dense,Sparse	From Dense	Sparse,Pretrained	No	Yes	Yes	Yes	-	Mode collapse	46.453

Why Regular Pruning Fails for GAN?

- No explicit evaluation like in classification (softmax)
  - GAN: evaluated subjectively
- Unstable GAN training
  - G and D must be well matched
  - Pruning can disrupt this well balance
- Energy of input and output is constant in GAN
  - Classification has output with much less entropy than the input
    - More tolerance in the reduced information space

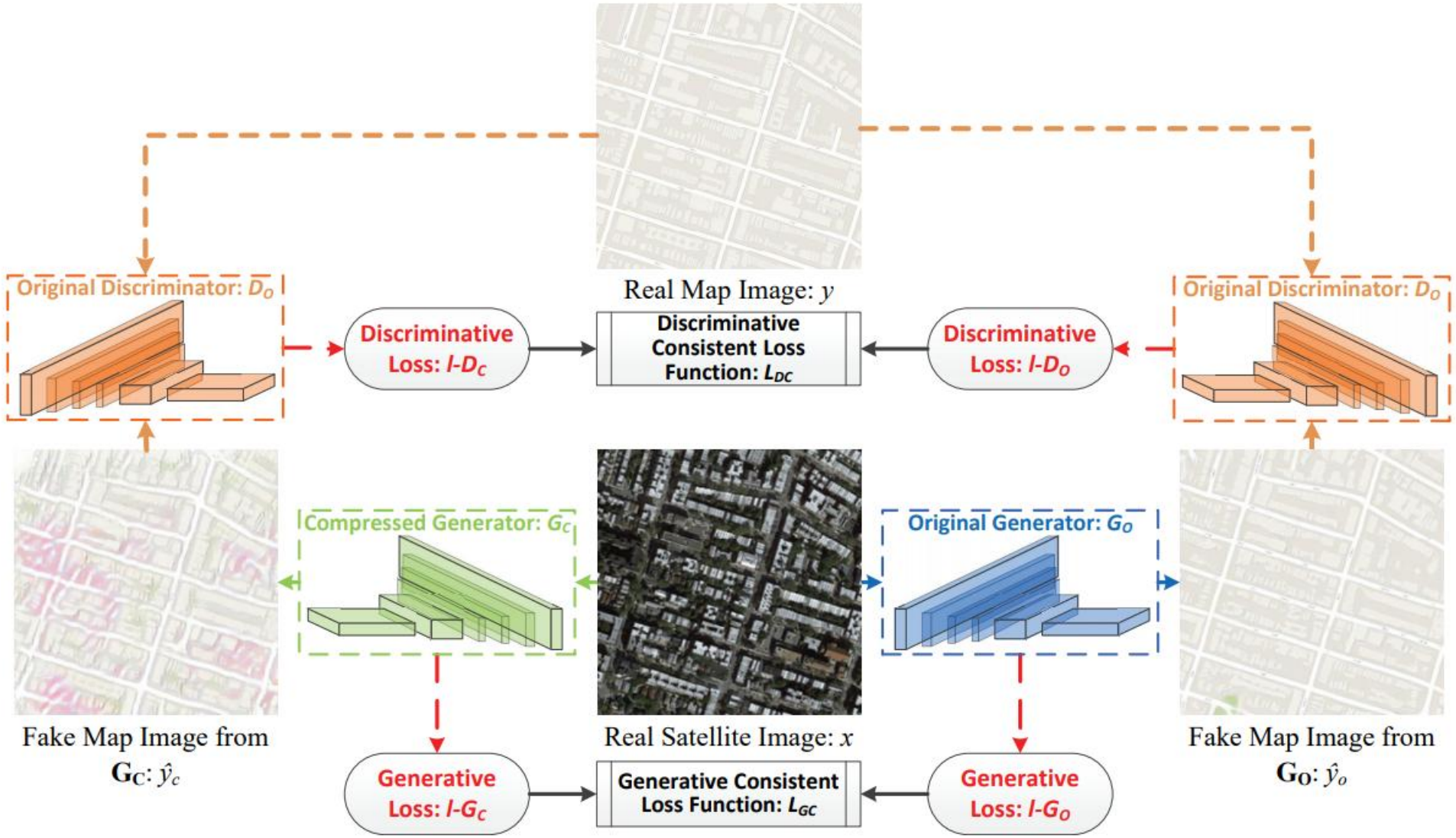


Figure 2. Workflow chart of GAN compression process.

Table 2. Tasks and networks overview

Task	Network	Dataset	Resolution	FID Scores when Pruned to				
				0% (dense)	25%	50%	75%	90%
Image Synthesis	DCGAN	MNIST	64x64	50.391	50.128	50.634	50.805	51.356
Domain Translation	Pix2Pix	Sat $\rightarrow$ Map	256x256	17.636	17.897	17.990	20.235	24.892
Domain Translation	Pix2Pix	Sat $\leftarrow$ Map	256x256	30.826	30.628	30.720	34.051	38.936
Style Transfer	CycleGAN	Monet $\rightarrow$ Photo	256x256	63.152	63.410	63.662	66.394	70.933
Style Transfer	CycleGAN	Monet $\leftarrow$ Photo	256x256	31.987	32.102	32.346	33.913	41.409
Image-Image Translation	CycleGAN	Zebra $\rightarrow$ Horse	256x256	60.930	61.005	61.102	65.898	68.450
Image-Image Translation	CycleGAN	Zebra $\leftarrow$ Horse	256x256	52.862	52.631	52.688	58.356	63.274
Image-Image Translation	StarGAN	CelebA	128x128	6.113	6.307	6.929	6.714	7.144
Super Resolution	SRGAN	DIV2K	$\geq$ 512x512	14.653	15.236	16.609	17.548	18.376

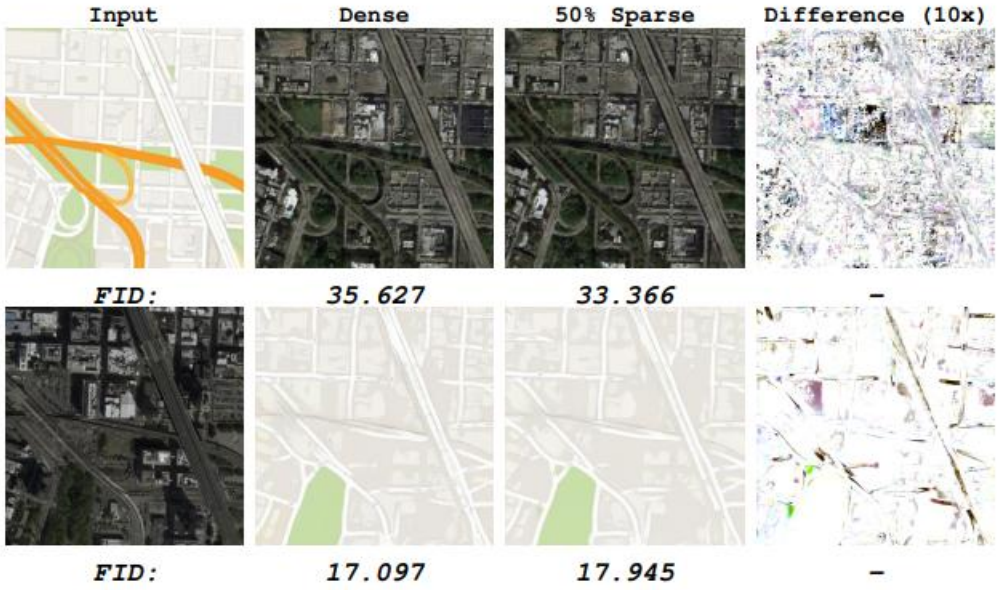


Figure 4. Representative results for domain translation: pix2pix. Row 1: map to satellite task, Row 2: satellite to map task.

Figure 6. Representative image-to-image translation results: CycleGAN. Row 1: zebra to horse, Row 2: horse to zebra.

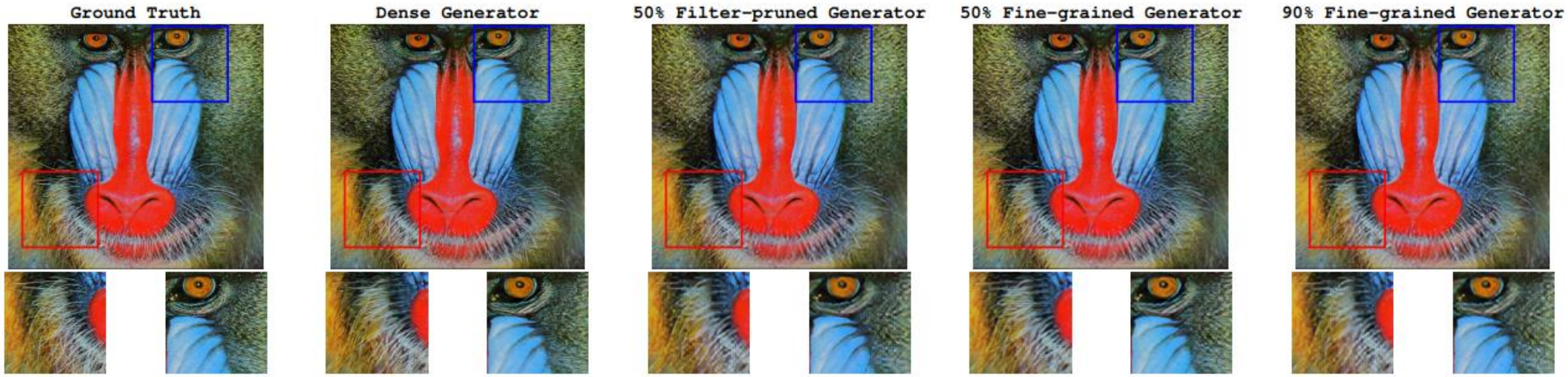


Figure 7. Representative super resolution results: SRGAN (with enlargements of boxed areas).

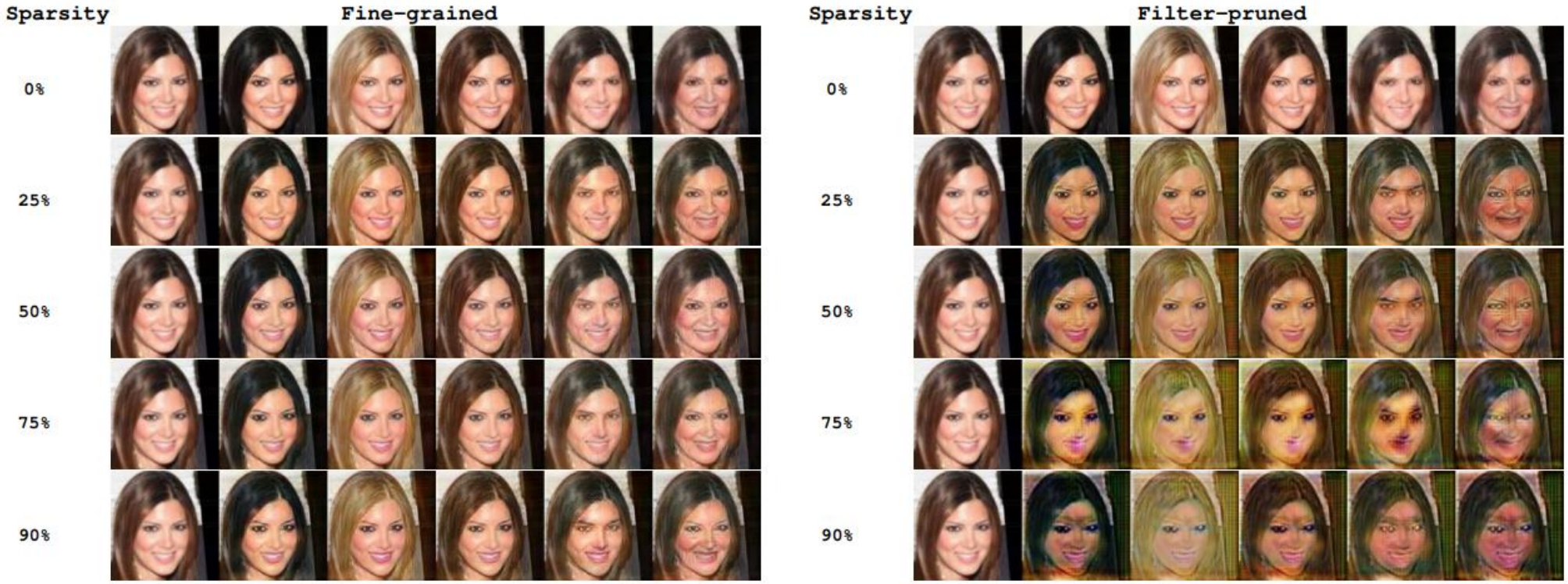
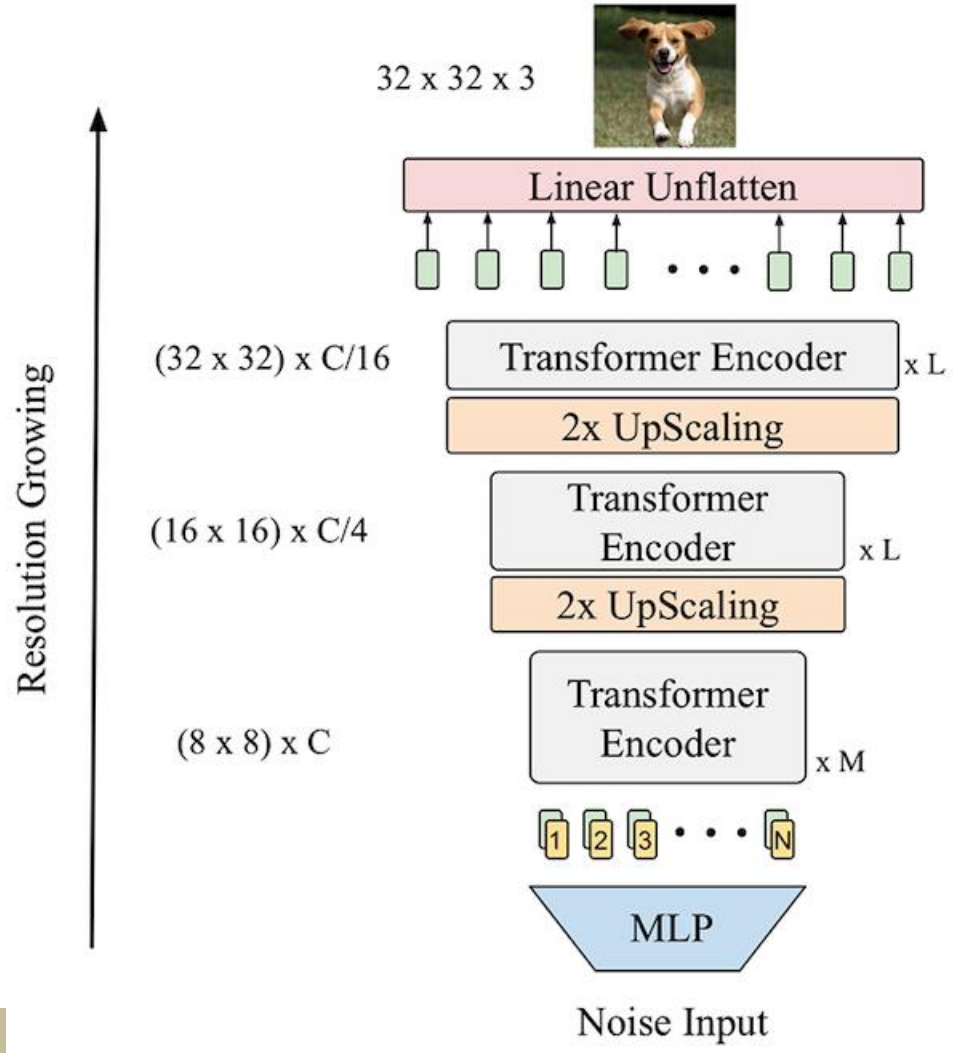


Figure 9. Representative results for pruning rate and granularity study of image-to-image translation.

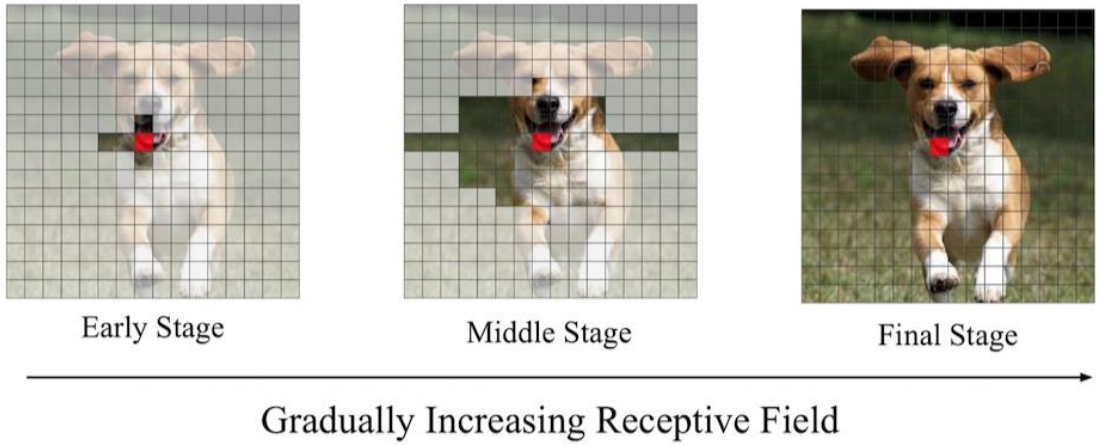
Dom.	Subject with applied model names
Image	Handwritten font generation: DenseNet-CycleGAN [36], LS-CGAN [37], GlyphGAN [38]
	Anime characters generation: DRA-GAN [45], PS-GAN [46]
	Image blending: GP-GAN [47], GCC-GAN [50]
	Image in-painting: Ex-GAN [51], PG-GAN [54]
	Face aging: Age-CGAN [56], CAAE [57], IP-CGAN [58], Wavelet -GANs [59 - 60]
	Text synthesis: AttnGAN [61], StackGAN [62], ACGAN [63], TACGAN [64], SISGAN
	Human pose synthesis:PG^2[65], Deformable-GAN [66]
	Stenographic applications: S-GAN [67], SS-GAN [68], Stegano-GAN [69]
	Image manipulation: IGAN [70], TAGAN [71], IAN [72], AttGAN [73], DGAN [234]
	Visual saliency prediction: Sal-GAN [74], SalCapsule-CGAN [75], DSAL-GAN [76]
	Object detection: SeGAN [77], PGAN [78], MTGAN [79], GANDO [80]
	3D image synthesis: 3DGAN [81], PrGAN [82]
	Medical applications: SeGAN [86], MedGAN [87]
	Facial makeup transfer: BGAN [94], PairedCycleGAN [95], DMTGAN [96]
	Face landmark detection: SAN [97], Expose-GAN [98]
	Image super-resolution: SR-GAN [99], ESR-GAN [100], SR-DGAN [101], T-GAN [102]
	Texture synthesis: MGAN [103], SGAN [105], PSGAN [106]
	Sketch synthesis: TGAN [107], Sketchy-GAN [110], CA-GAN [111]
	Image-to-image translation: Pix2pix [34], PAN [112], CycleGAN [113], Disco-GAN [114], Dual-GAN [115], StarGAN [116], UNIT [117], MUNIT [118], DRIT [119], DRIT++ [120]
	Face frontal view generation: DRGAN [121], TPGAN [122], FFGAN [123], FTGAN [125]
Audio	Speech and audio synthesis: Rank-GAN [150], VAW-GAN [151]
	Music generation: C-RNNGAN [152], Seq-GAN [154], ORGAN[155]
Video	Video applications:VGAN [168], MoCoGAN [169], DRNET [170], DVD-GAN [171]

TransGAN: **Two Transformers can make One Strong GAN**

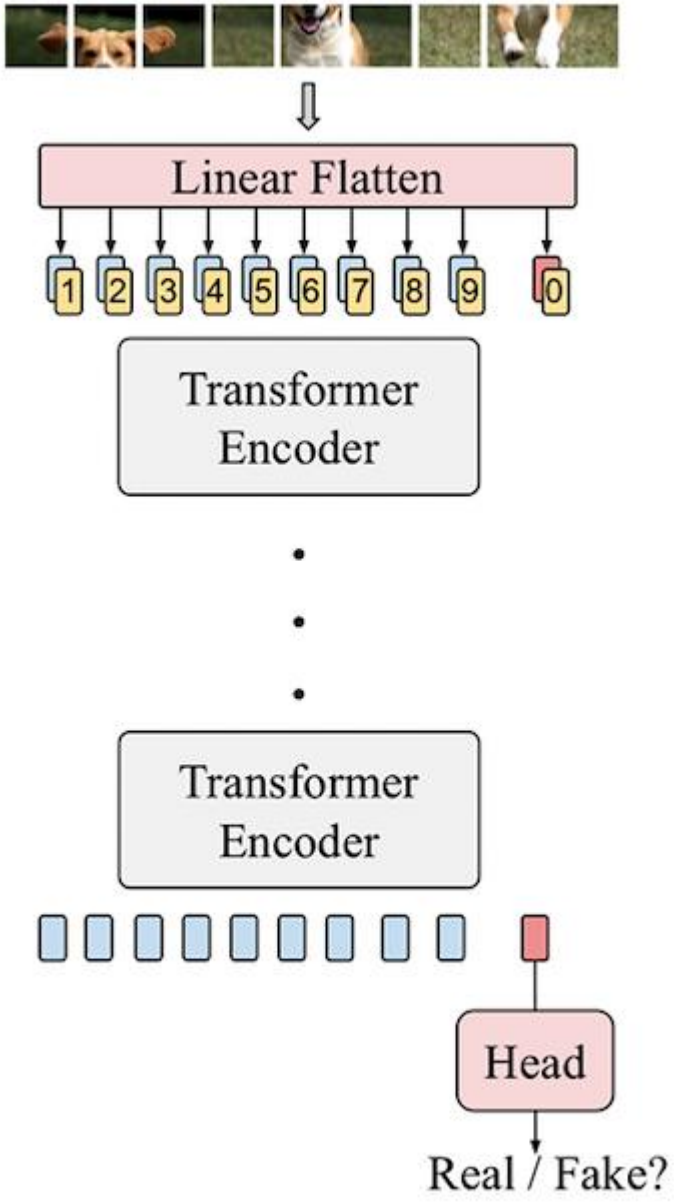
- A CNN free GAN network



Data Augmentation to increase efficiency
Co-Training with Self-Supervised auxiliary task
Locality — Aware Localization for Self-Attention



localized self-attention is most helpful at early training stages, but can hurt the later training stage and the final achievable performance.



Discriminator

METHODS	IS	FID
WGAN-GP (GULRAJANI ET AL., 2017)	6.49 ± 0.09	39.68
LRGAN (YANG ET AL., 2017)	7.17 ± 0.17	-
DFM (WARDE-FARLEY & BENGIO, 2016)	7.72 ± 0.13	-
SPLITTING GAN (GRINBLAT ET AL., 2017)	7.90 ± 0.09	-
IMPROVING MMD-GAN (WANG ET AL., 2018A)	8.29	16.21
MGAN (HOANG ET AL., 2018)	8.33 ± 0.10	26.7
SN-GAN (MIYATO ET AL., 2018)	8.22 ± 0.05	21.7
PROGRESSIVE-GAN (KARRAS ET AL., 2017)	8.80 ± 0.05	15.52
AUTOGAN (GONG ET AL., 2019)	8.55 ± 0.10	12.42
STYLEGAN V2 (ZHAO ET AL., 2020B)	**9.18**	**11.07**
TRANSGAN-XL	8.63 ± 0.16	11.89