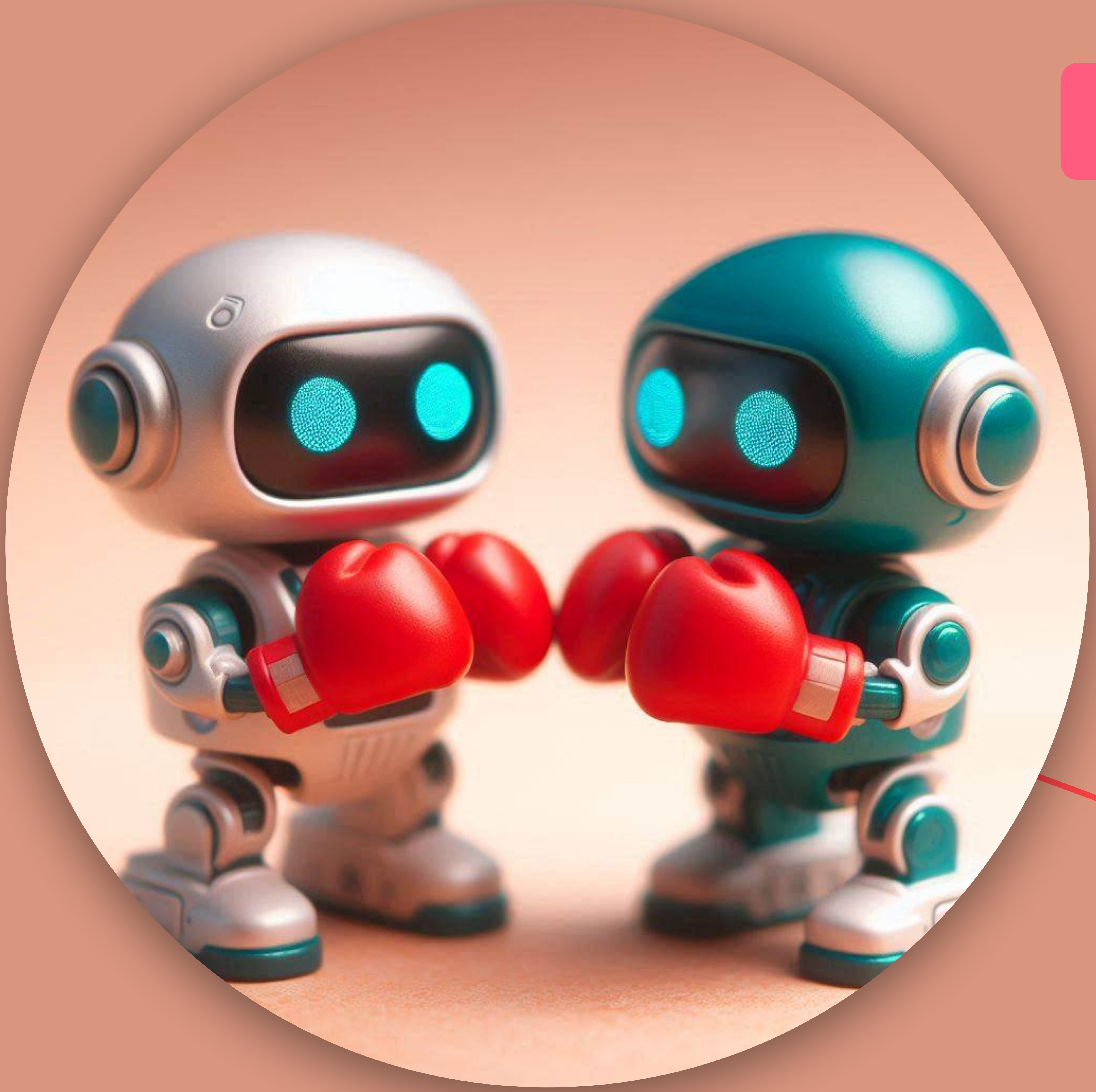


生成式 AI：文字與圖像生成的原理與實務

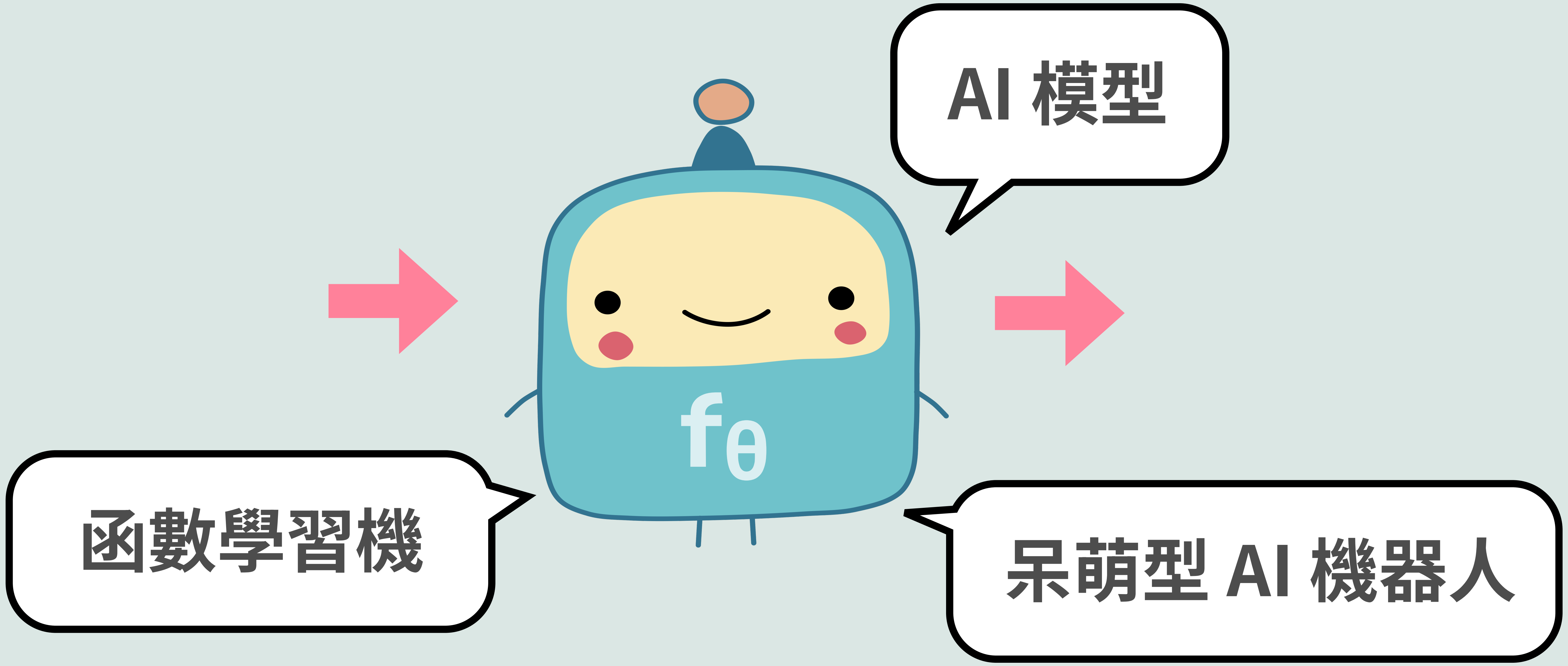
03. GAN 生成對抗網路

蔡炎龍
政治大學應用數學系

01.

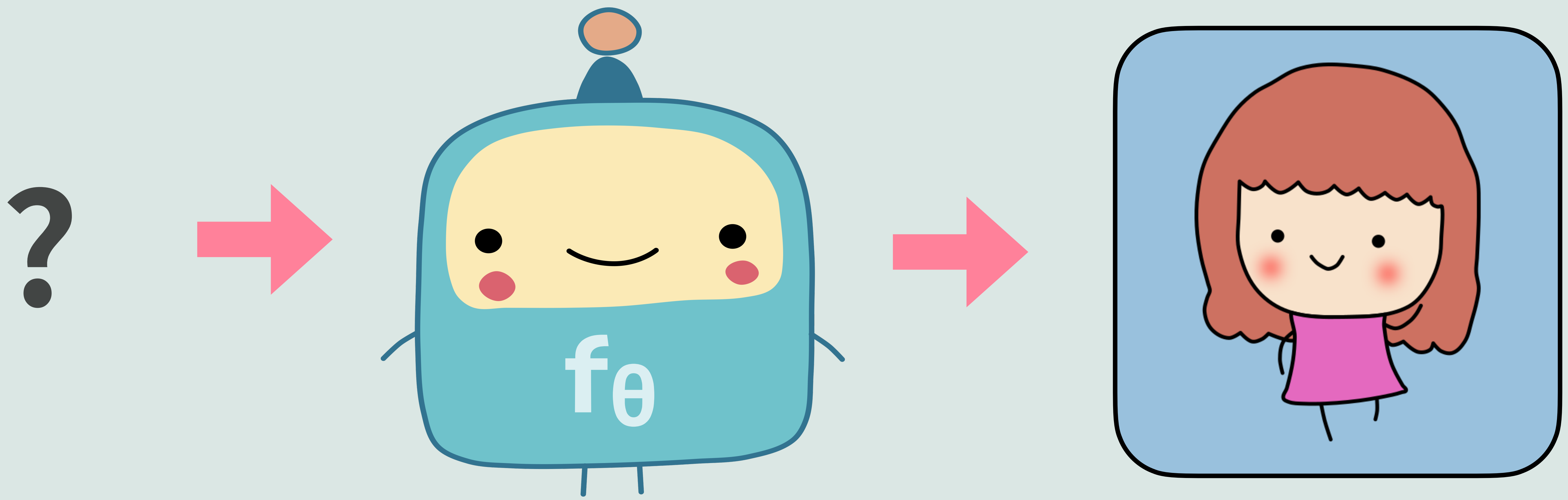
創作型的 AI 很難做嗎？

記得現代的 AI 是要設計呆萌型 AI 機器人



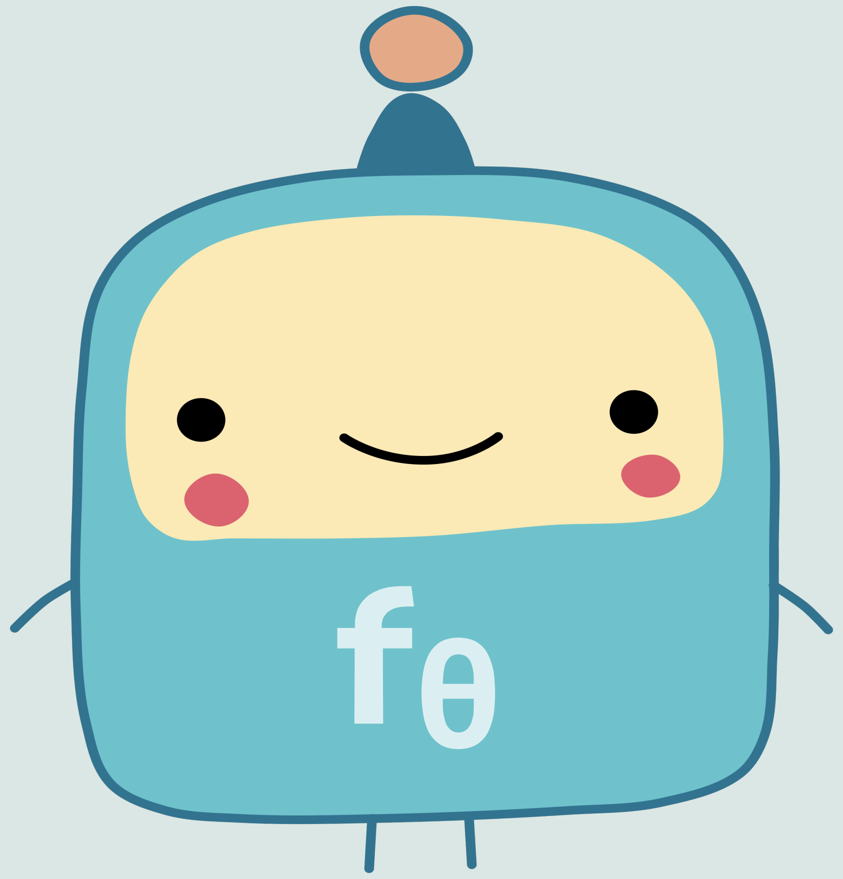
就是知道輸入是什麼、輸出是長什麼樣子

創作型 AI 要輸入什麼、輸出什麼呢?



難道是...

“一個可愛的女孩"

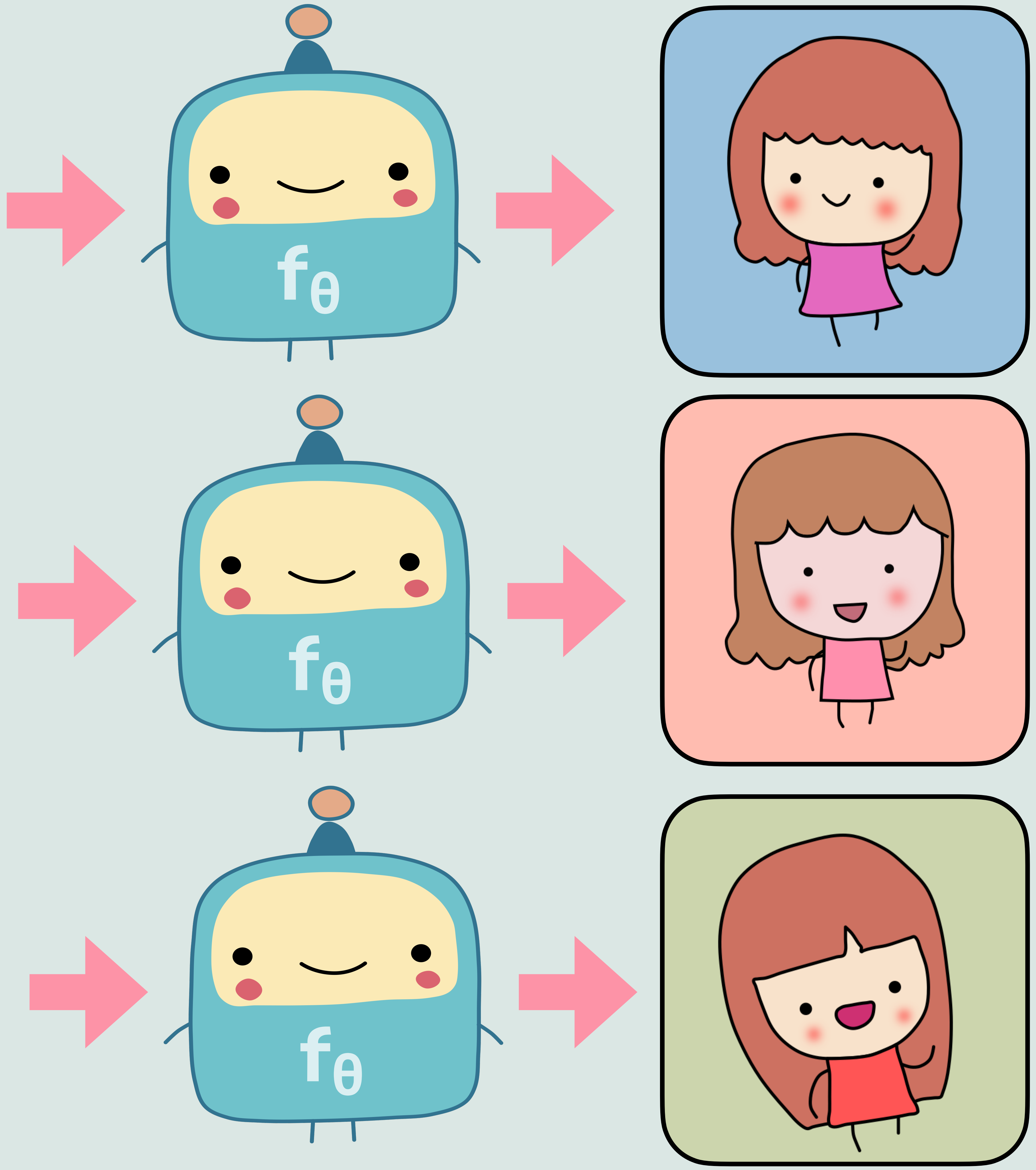


同樣的輸入有很多不同輸出的可能

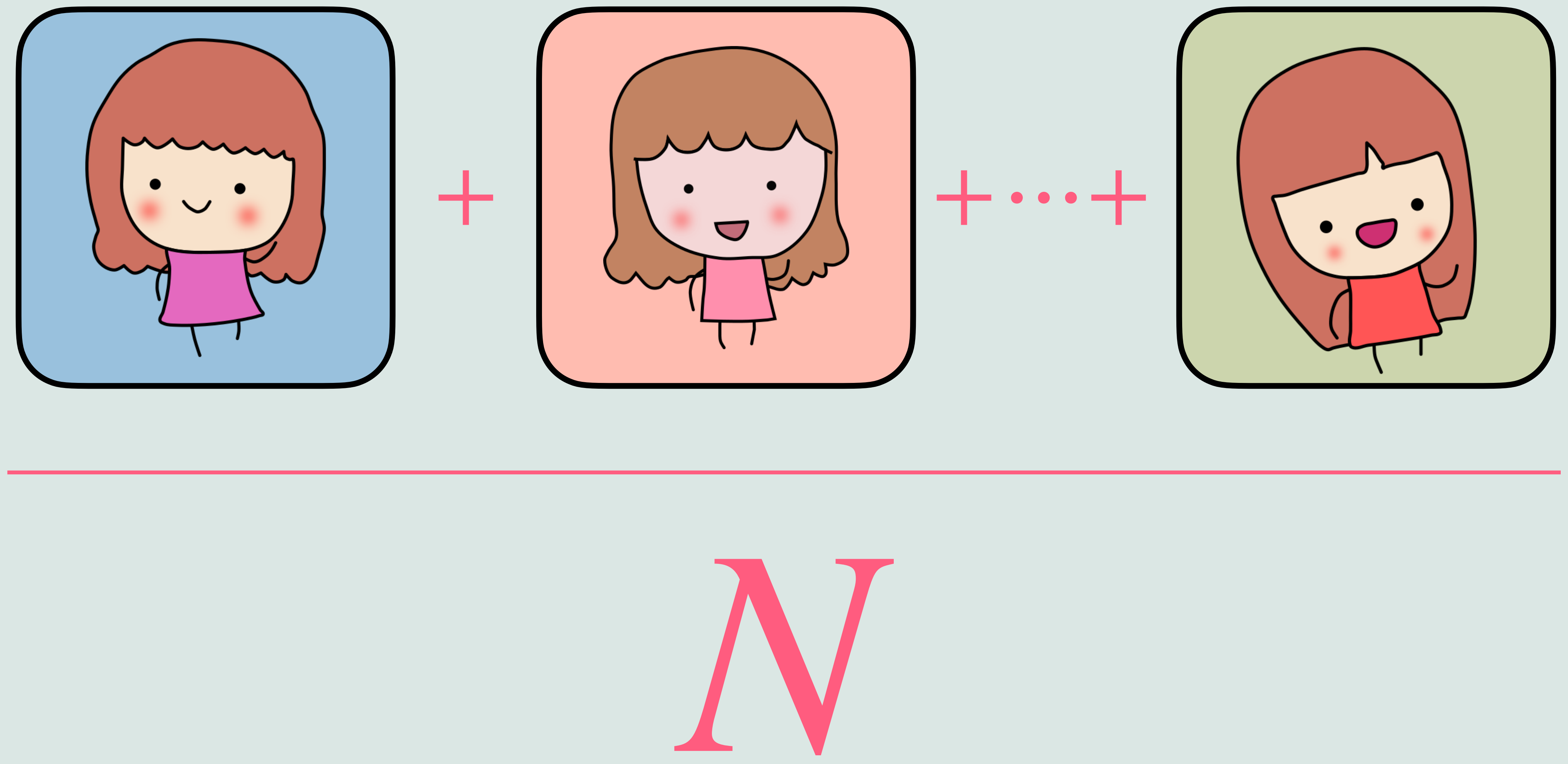
"一個可愛的女孩"

"一個可愛的女孩"

"一個可愛的女孩"



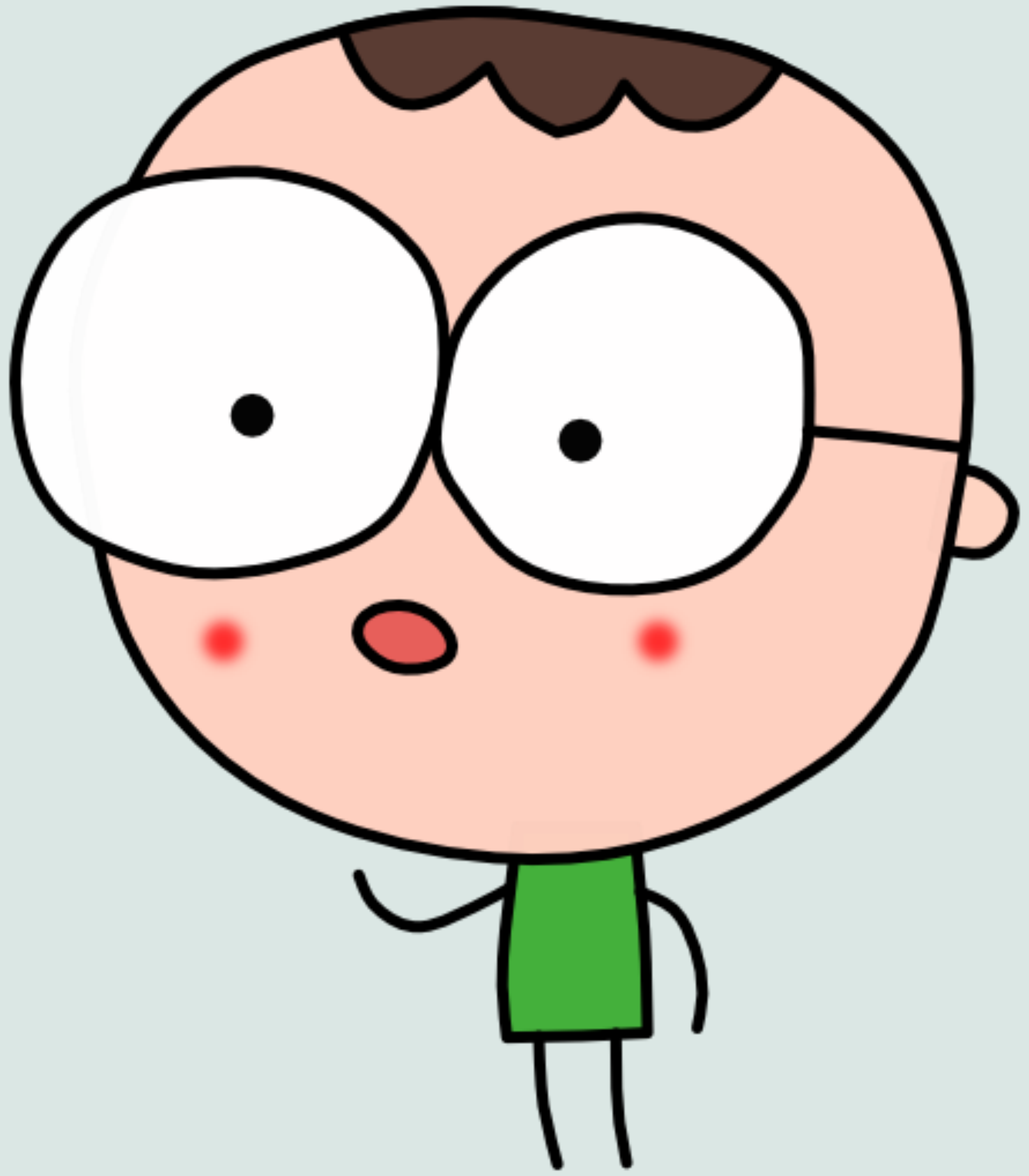
注意這不是函數 (一對多)



硬是要訓練, 同樣的輸入, 會得到輸出的平均!

這不是我們想要的!

- 平均之後很可能四不像
- 就算運氣好可以接受, 也不是我們要的 (創作)

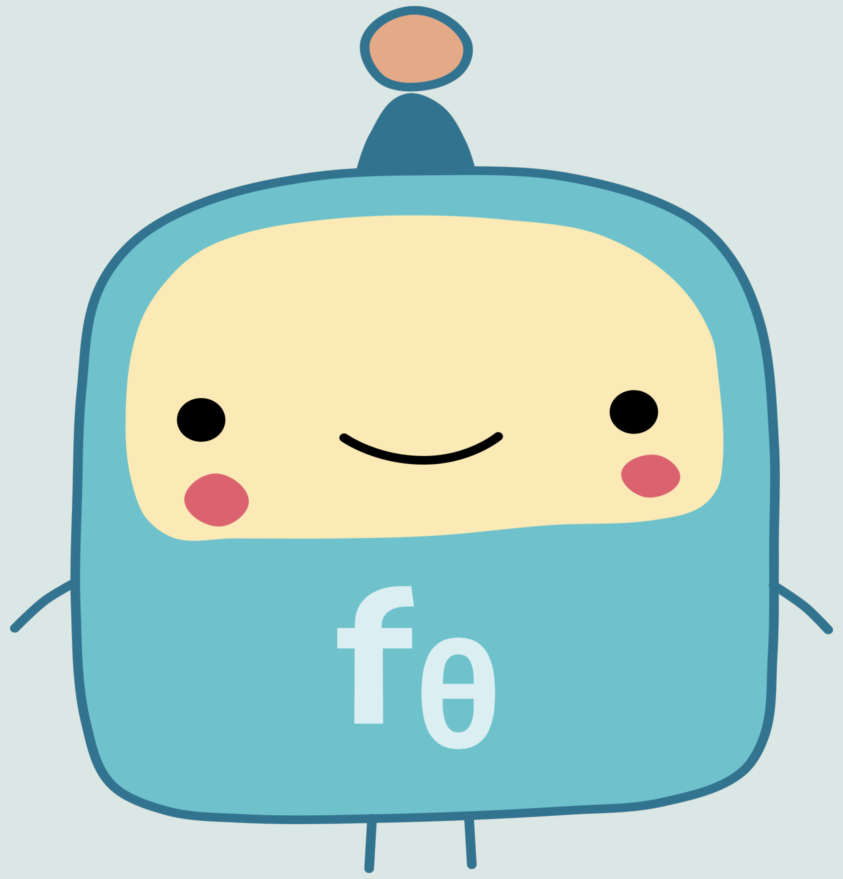


創作者機器人應該是怎麼樣的呢?

天馬行空的「想法」。

$\mathbf{z}$

latent tensor
(潛張量)



創作作品

Latent Tensor 就是某個數據的特徵表現 Tensor

02.

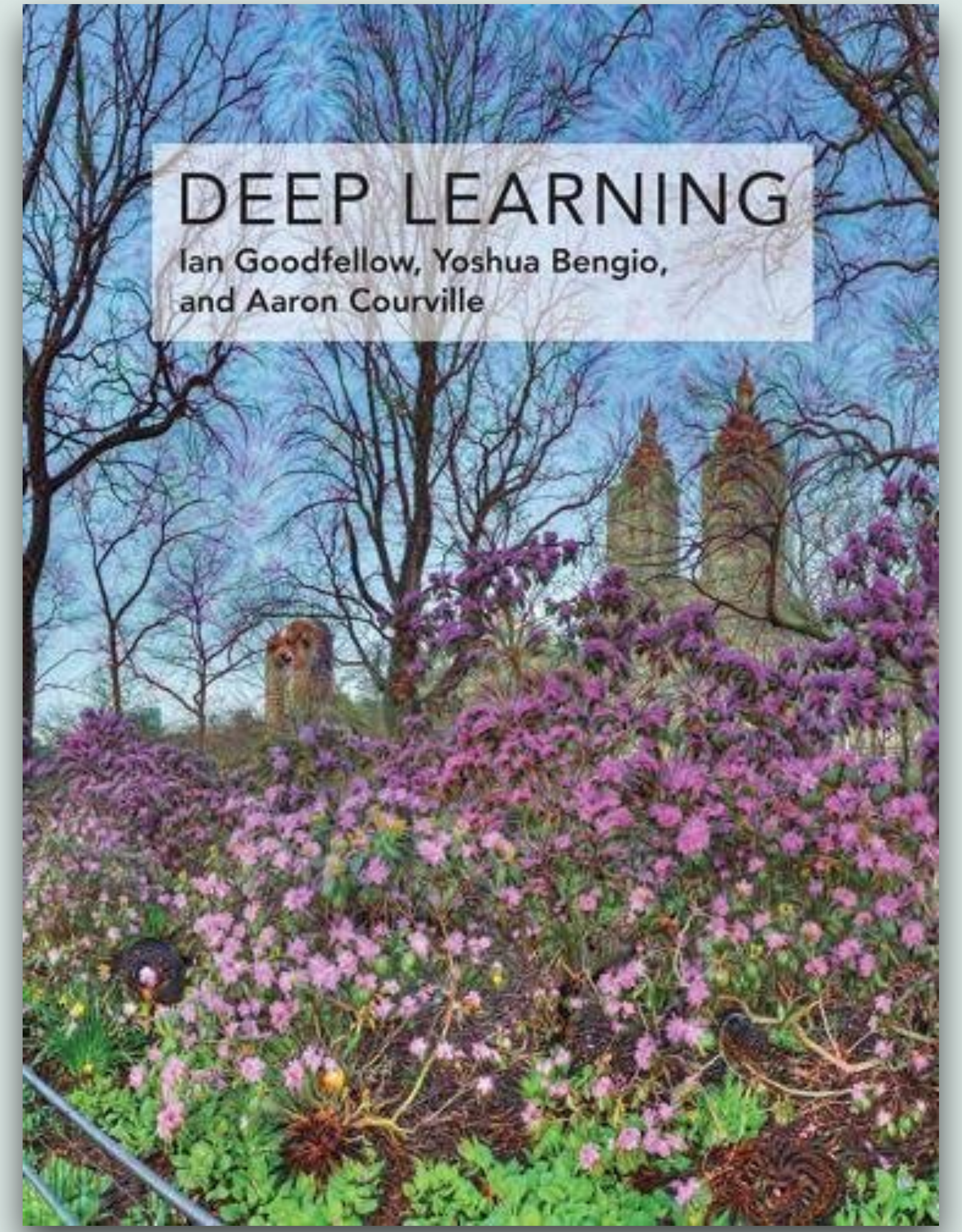
GAN 的出現

LeCun 認為 GAN 是深度學習最有潛力的 model

> There are many interesting recent development in deep learning…
>
> The most important one, in my opinion, is adversarial training (also called GAN for Generative Adversarial Networks).

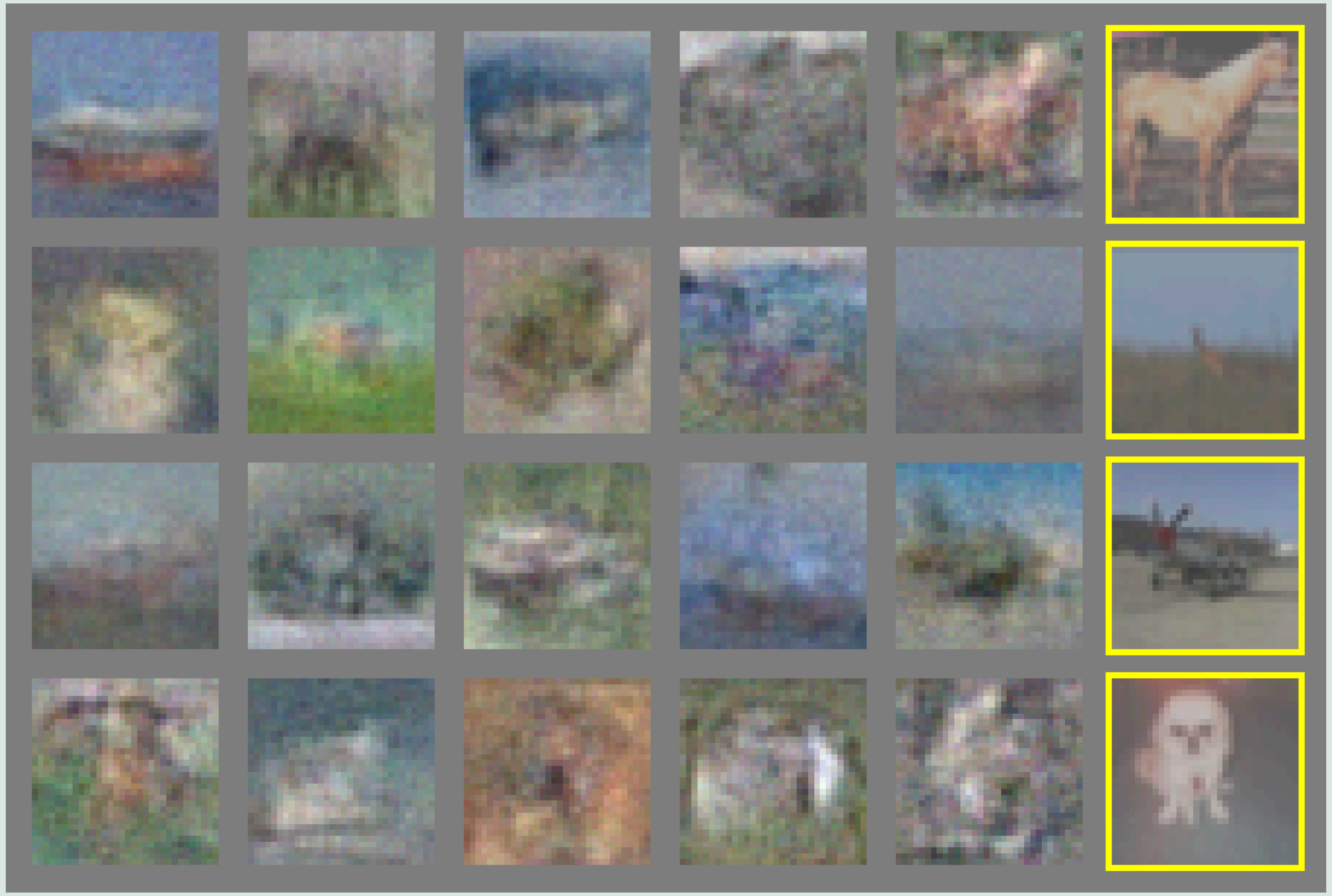
—**Yan LeCun (楊立昆), 2016**

作者 Ian Goodfellow



原創者 Ian Goodfellow, 之前最著名的是和他博士時期兩位老師 Yoshua Bengio 和 Aaron Courville 寫了號稱 Deep Learning 聖經的書。

就是這篇文章



Generative Adversarial Networks

Ian Goodfellow 等人 (NIPS 2014)

https://arxiv.org/abs/1406.2661

原創者親授

Ian Goodfellow 的 GAN 教學（NIPS 2016）

https://youtu.be/AJVyzd0rqdc

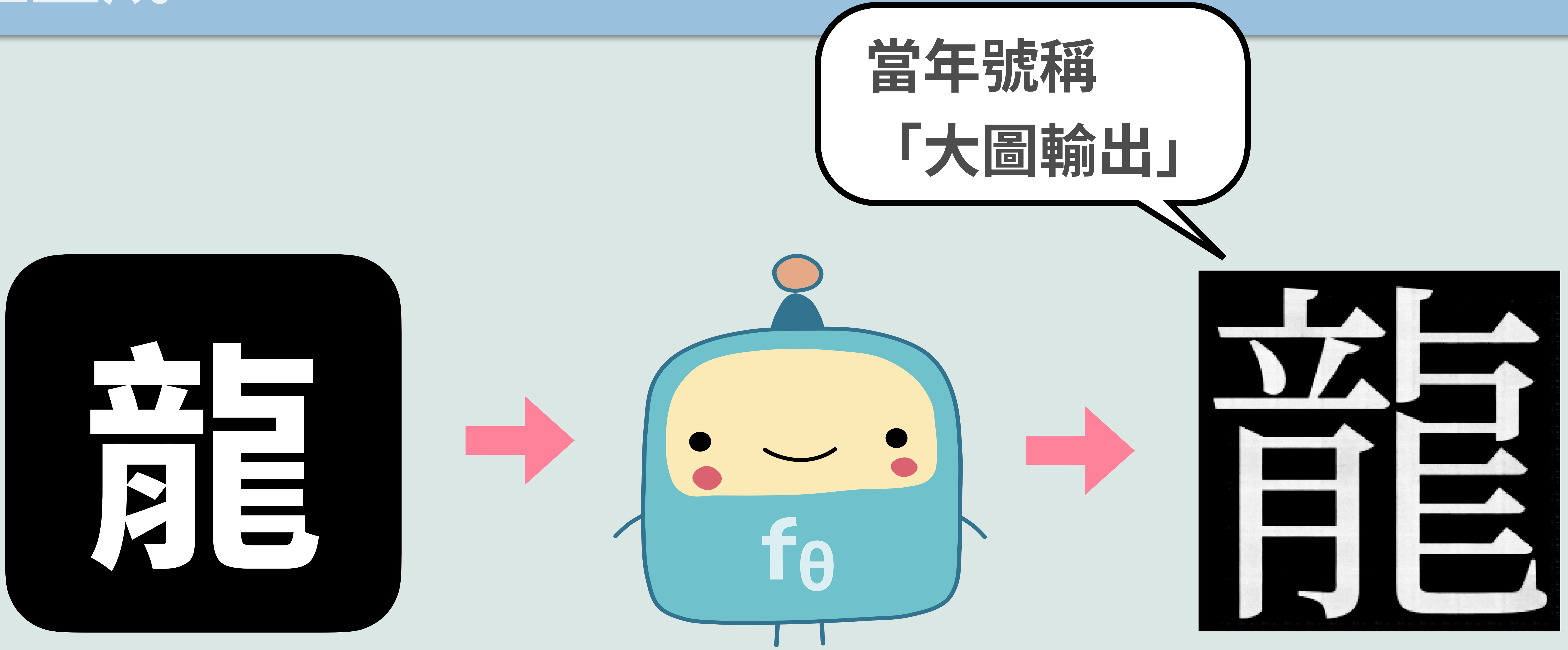
The GAN Zoo

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

政大學長姐也做了一個、參加比賽得佳作

聽說 Soumith Chintala (WGAN 作者) 來, 參加 GAN 比賽的同學

字型生成 GAN



512 × 512

Variational Grid Setting Network

Yu-Neng Chuang, Zi-Yu Huang, and Yen-Lung Tsai

08 創作機器人

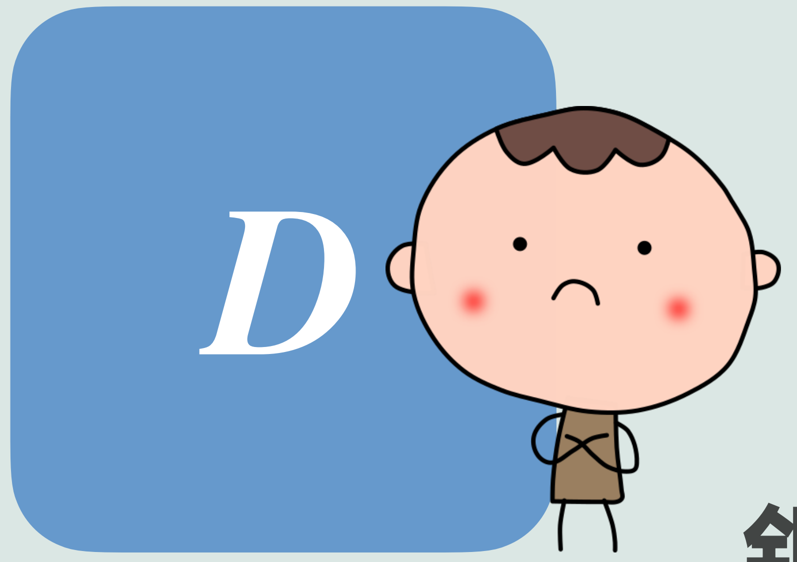
結果是訓練兩個神經網路!

靈感 →

→ 一小段曲子

生成器

一小段曲子 →

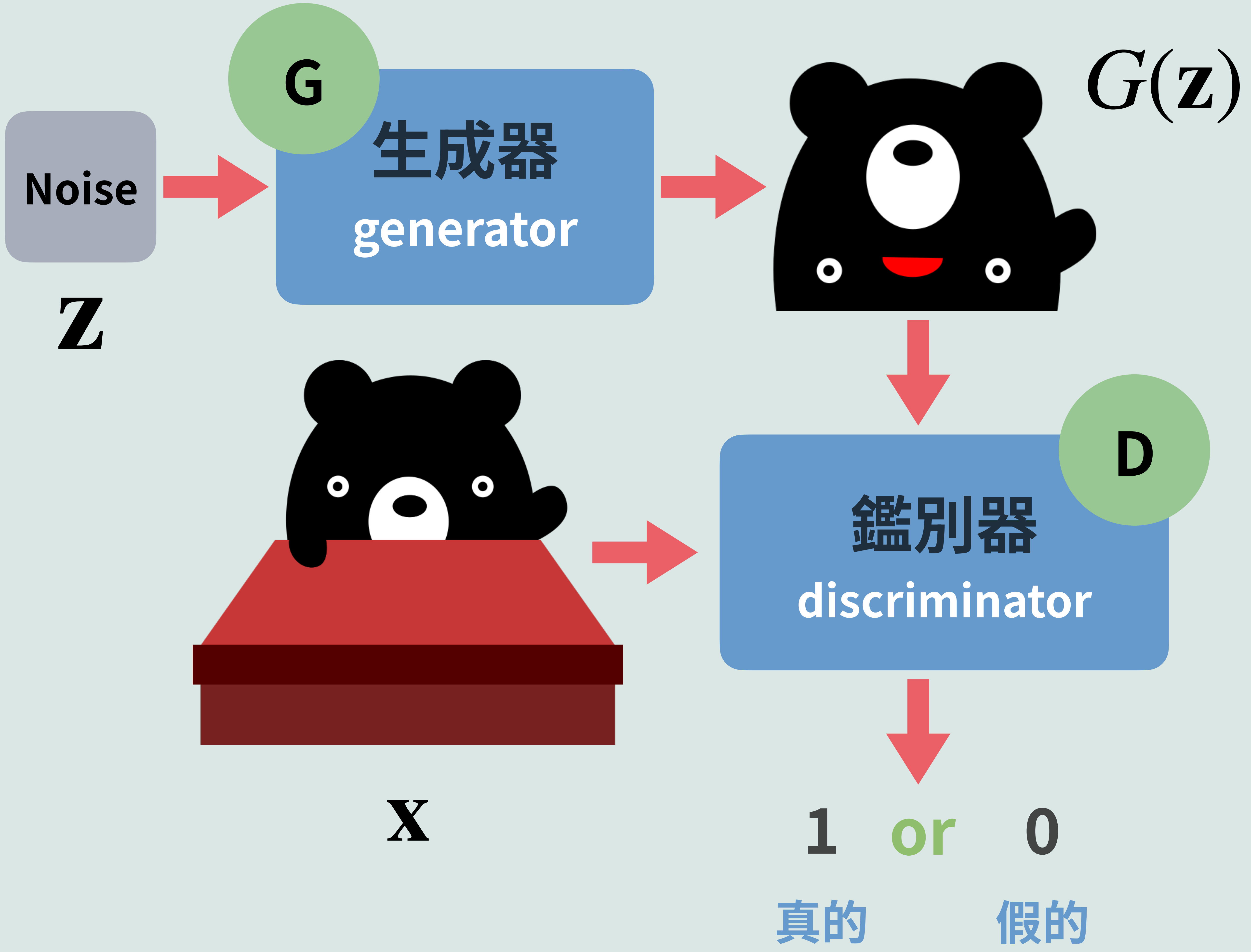


→ 像 or 不像
一小段曲子

鑑別器

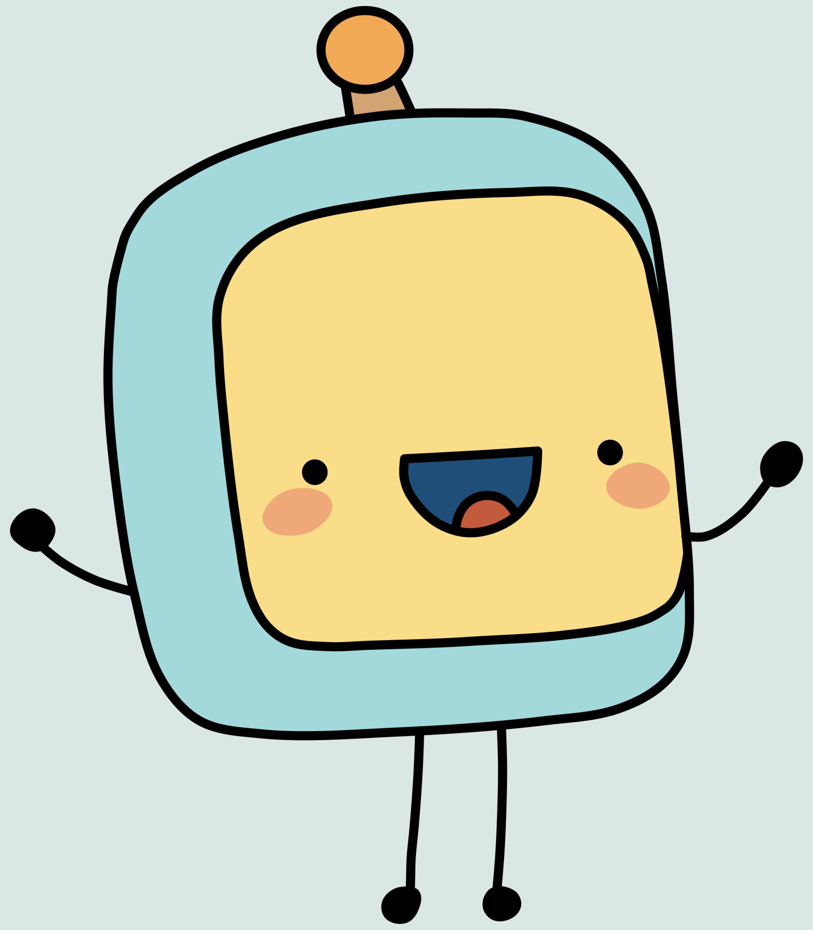
這就是所謂的**生成對抗網路 (GAN)**!

GAN 的架構



GAN 是兩個神經網路,
一個叫**生成器**、一個叫
鑑別器, 相互對抗!

GAN 就是生成器、鑑別器大對抗!

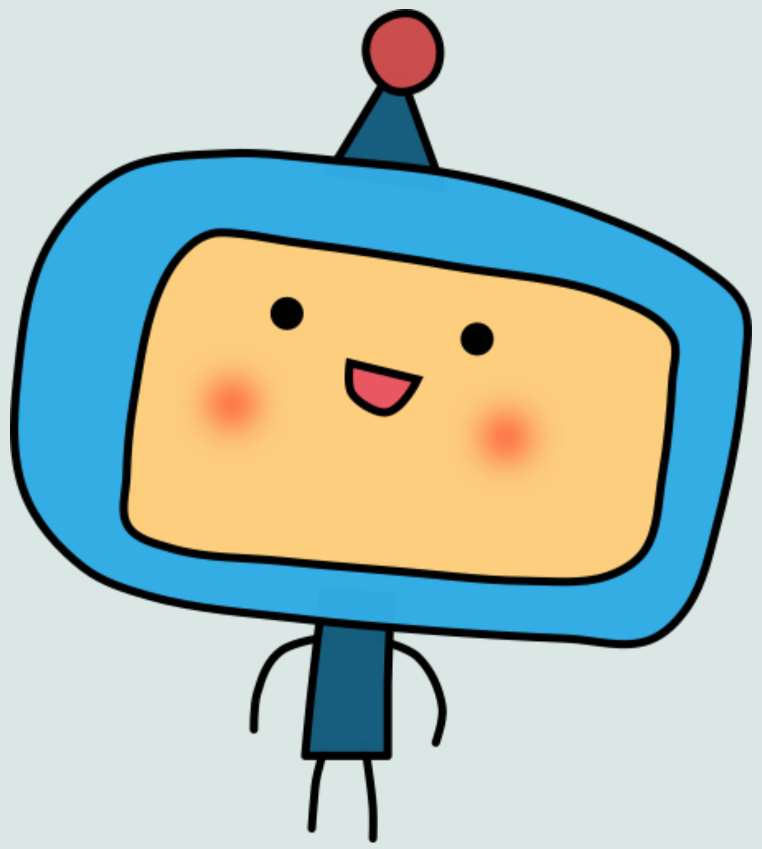


生成器 G

希望

$$D(G(\mathbf{z}))$$

接近 1



鑑別器 D

希望

$$D(G(\mathbf{z}))$$

接近 0

$$D(\mathbf{x})$$

接近 1

又來到 paper 嚇人時間

$p_{\mathbf{z}}$

latent vector 的分布

$p_{\mathbf{x}} = p_{\text{data}}$

真實世界的分布

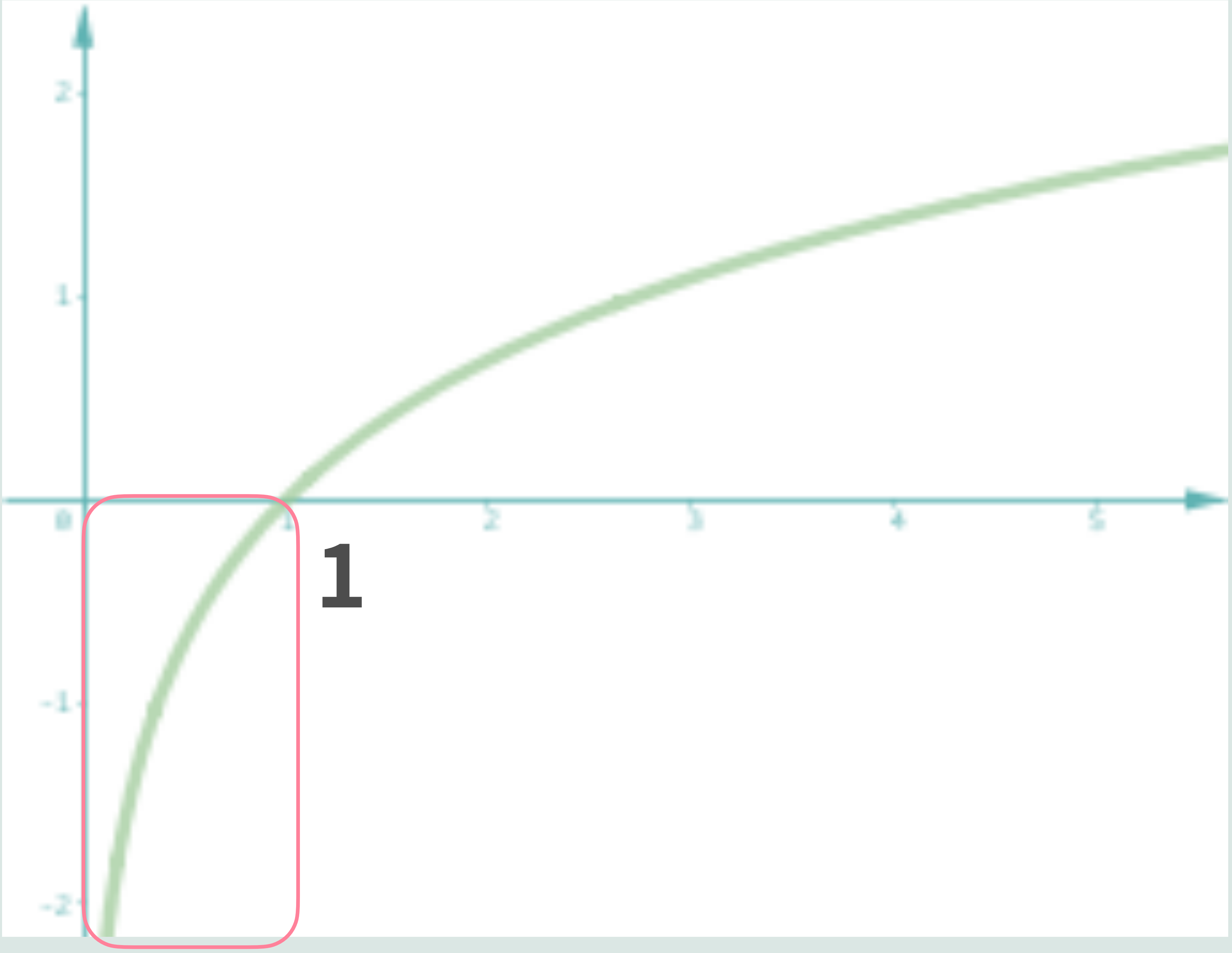
$p_{\mathbf{g}} = p_{\text{model}}$

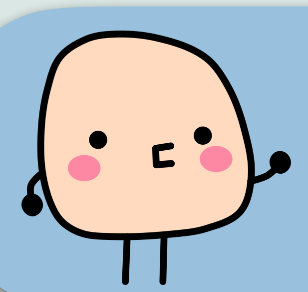
我們呆萌 AI 生成器的分布

再來也沒有什麼的...

$$\mathbf{x} \sim p_{\text{data}}$$

插播: 可愛的 log





插播: 可愛的 log

$$\log(a \cdot b) = \log(a) + \log(b)$$

關於 GAN 的 loss

1. $\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log(D(\mathbf{x}))]$

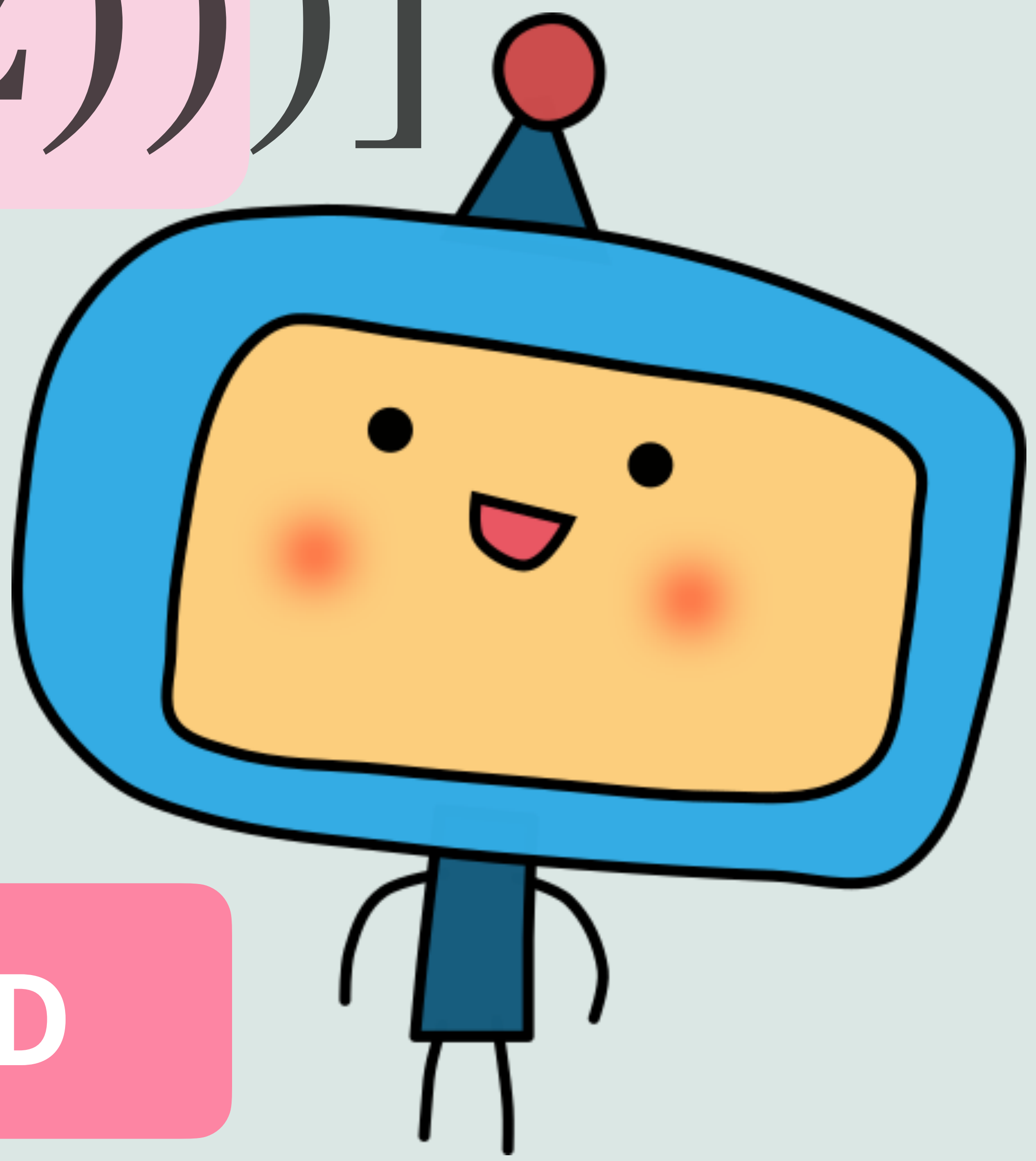
越接近 1 越好。

2. $\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[\log(1 - D(G(\mathbf{z})))]$

越接近 0 越好。

兩者都越大越好

鑑別器 D

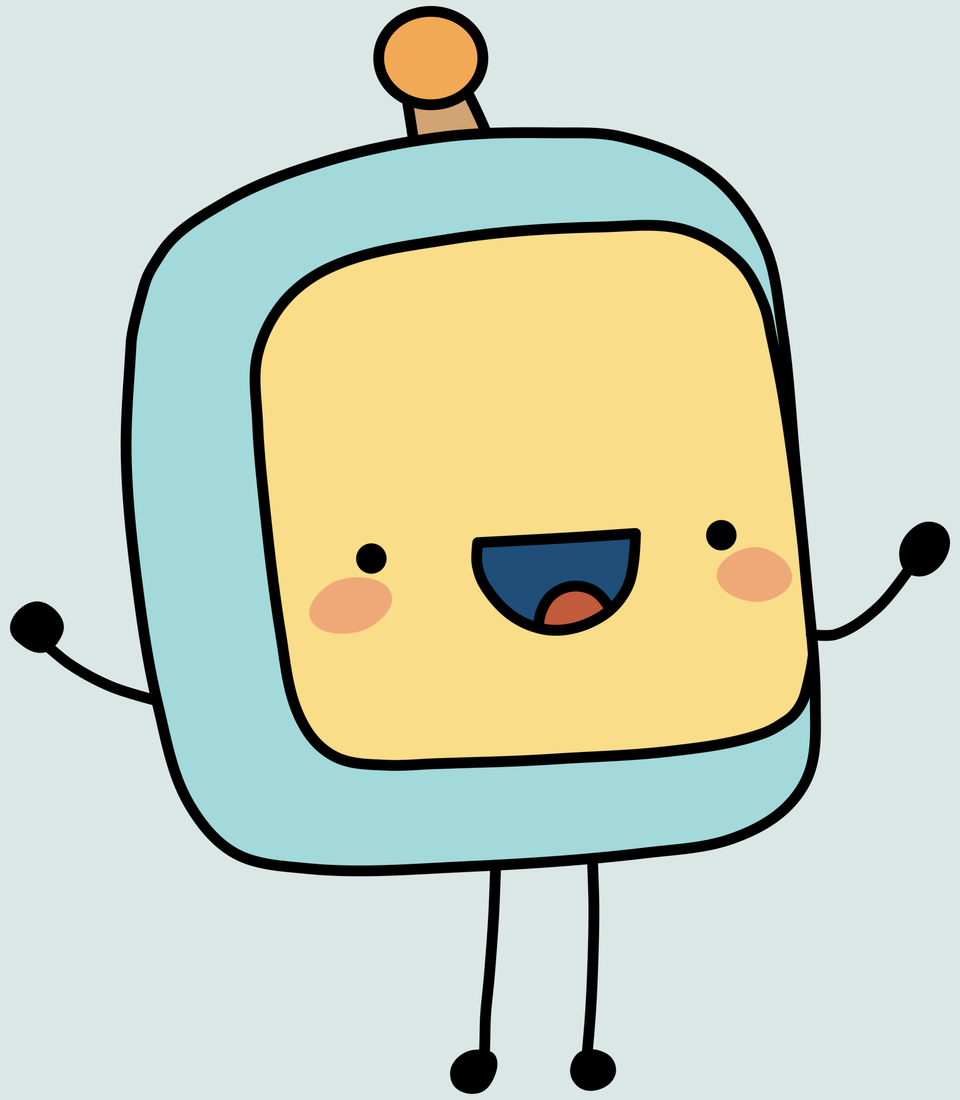


關於 GAN 的 loss

越接近 1 越好。

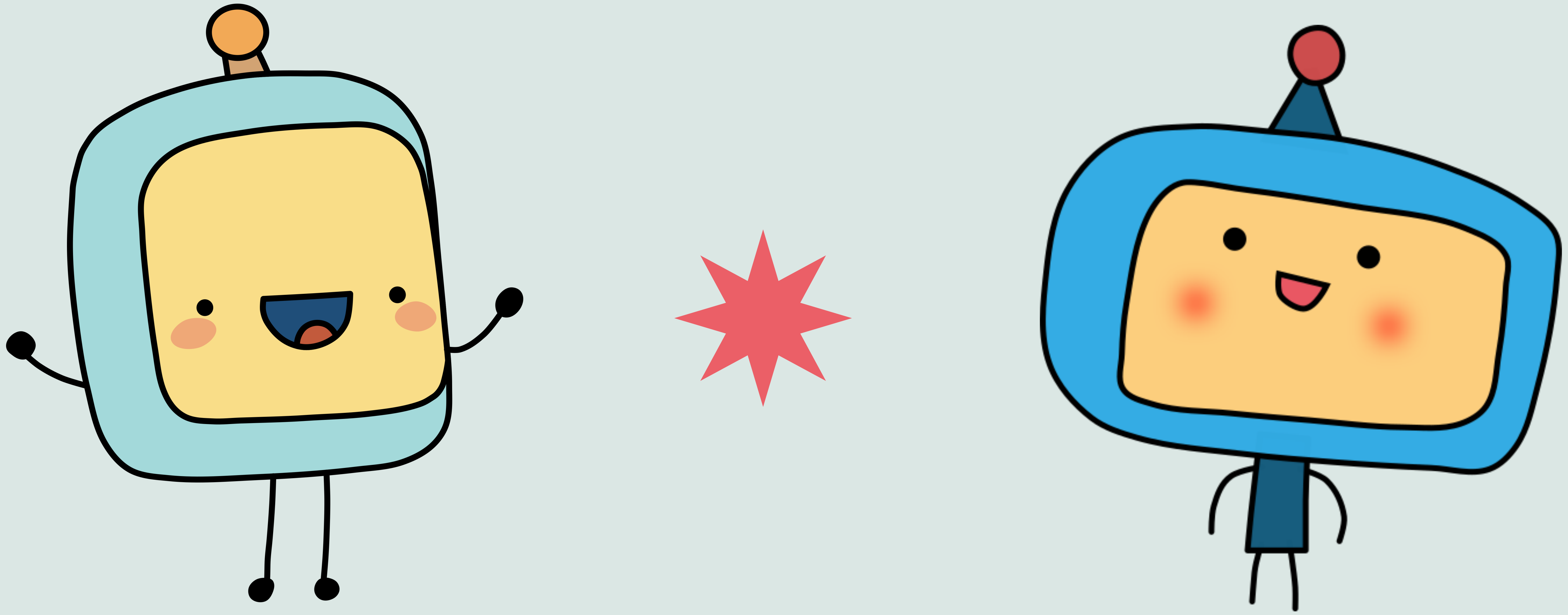
2 $\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[\log(1-D(G(\mathbf{z})))]$

這個越小越好。



生成器 G

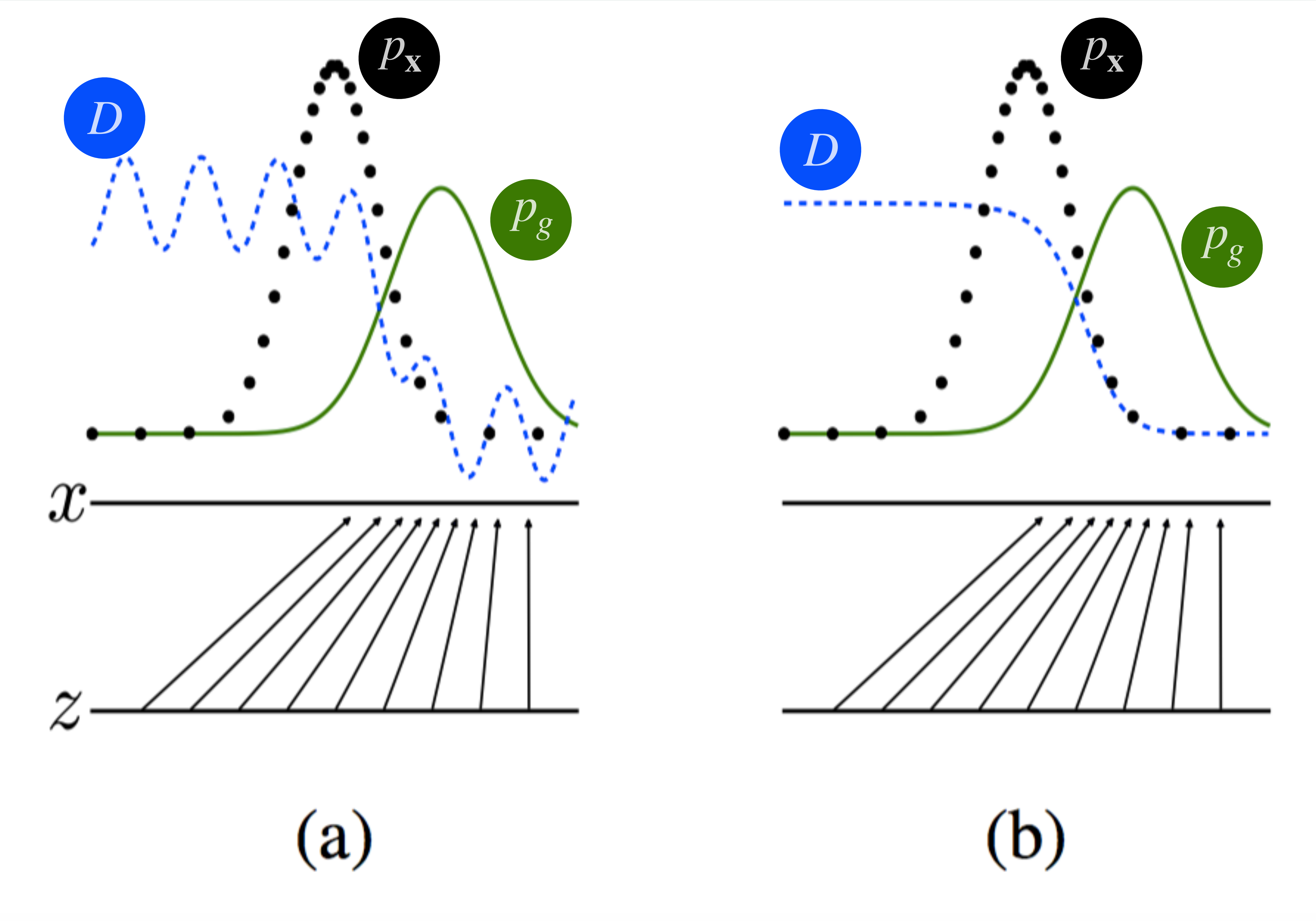
關於 GAN 的 loss



生成對抗

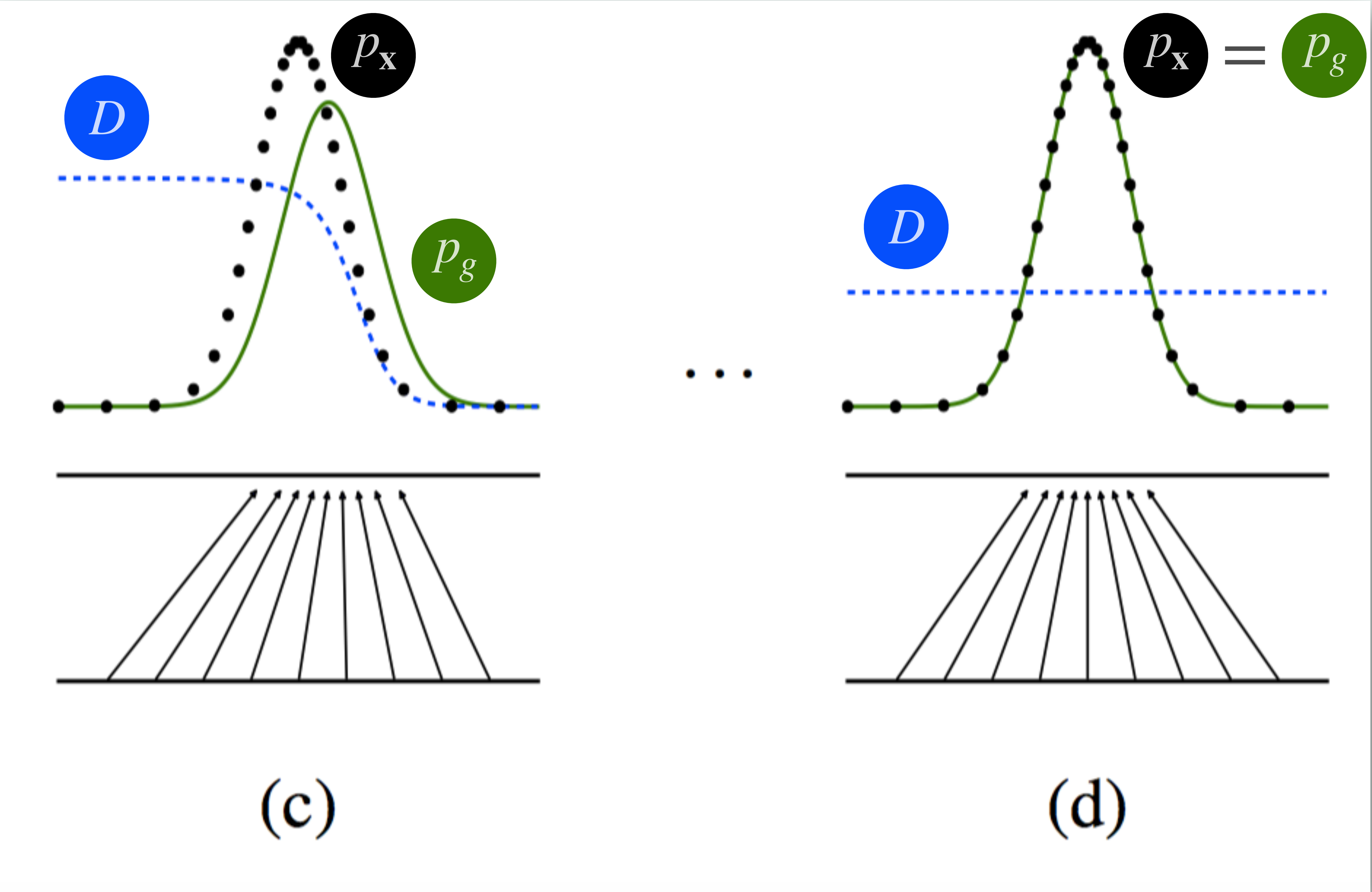
$$\min_{G}\max_{D} V(D, G) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log(D(\mathbf{x}))] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[\log(1 - D(G(\mathbf{z})))]$$

學習過程

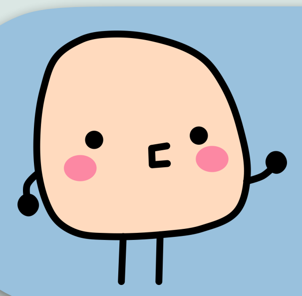


* 來自 Ian Goodfellow 等人 GAN 的原始論文 (2014)

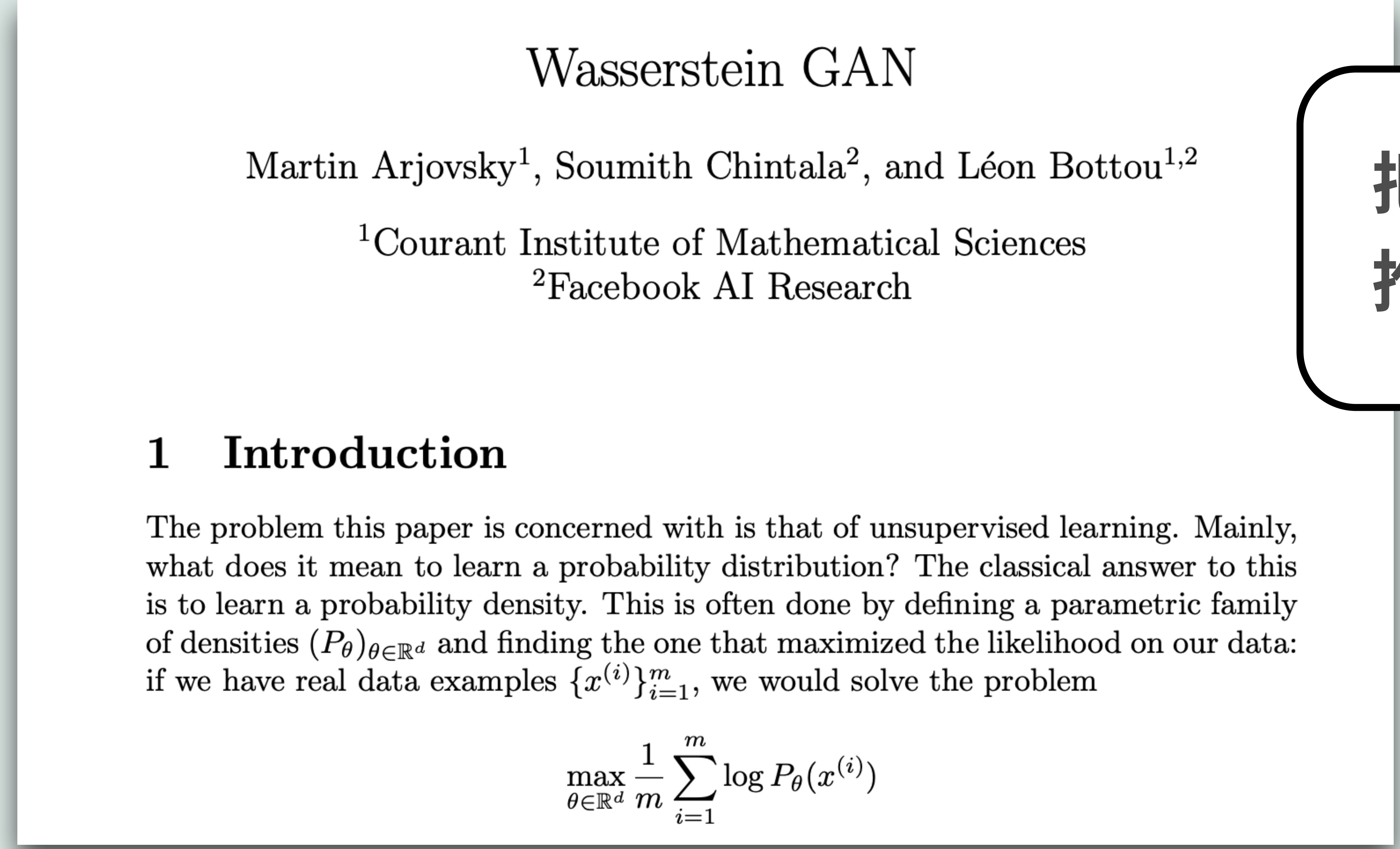
學習過程



* 來自 Ian Goodfellow 等人 GAN 的原始論文 (2014)



令人拍案叫絕的 WGAN



Wasserstein GAN

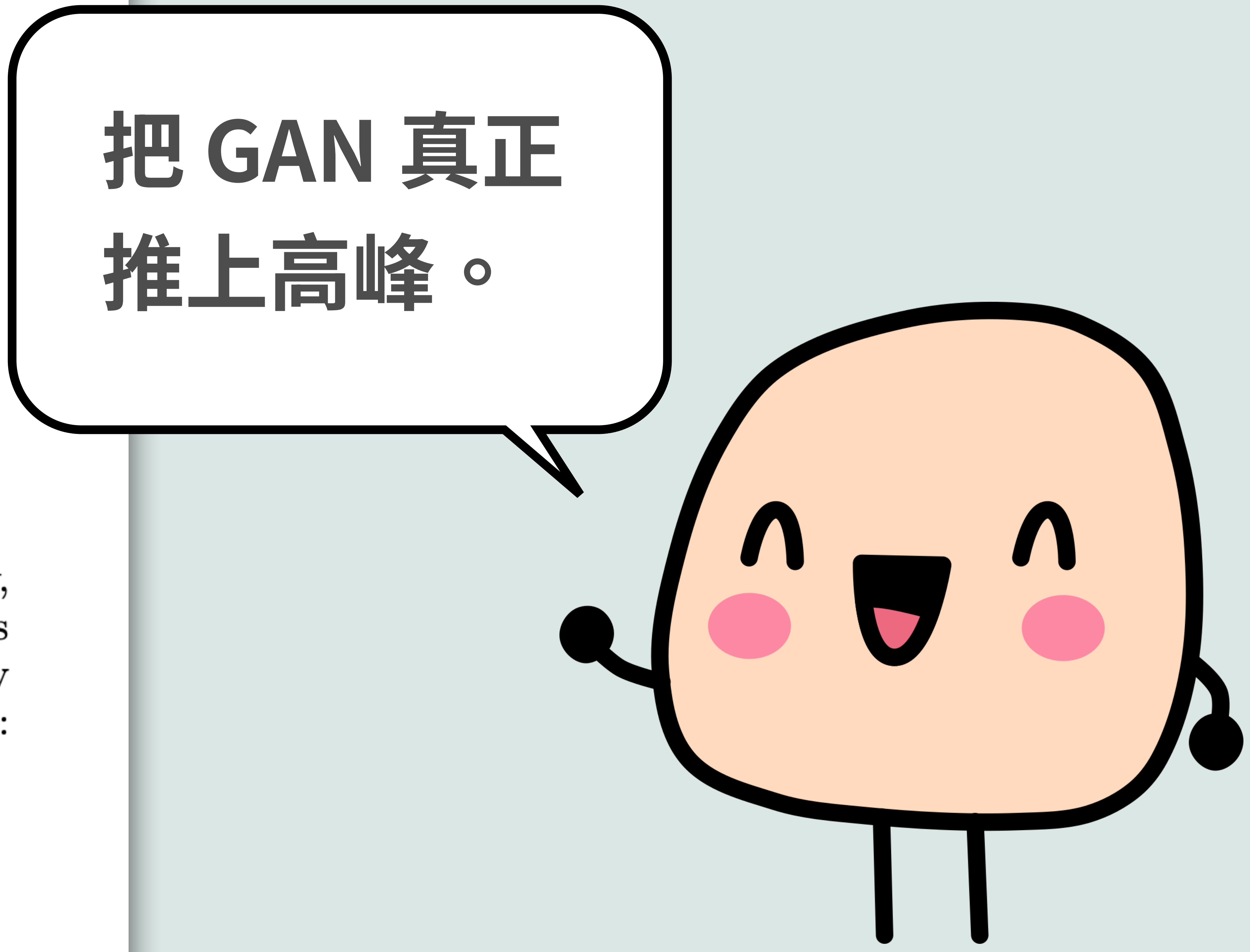
Martin Arjovsky[1], Soumith Chintala[2], and Léon Bottou[1,2]

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

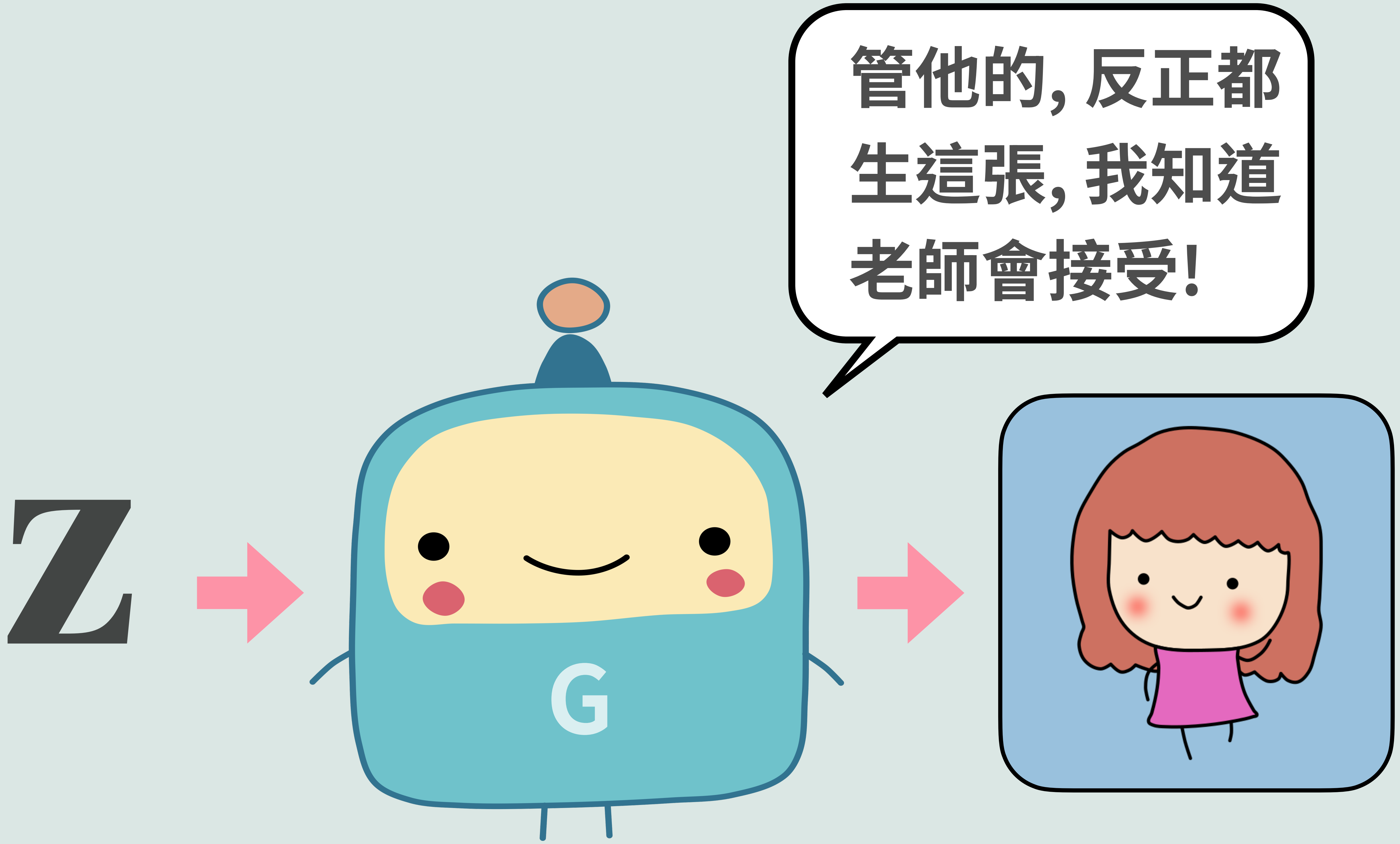
1 Introduction

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

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



WGAN 解決問題之一: Collapse 崩壞



03.

GAN 的高光時刻

GAN 一度成為電腦創作的王牌

StyleGAN 前傳: Progressive GAN 生成明星照片

Karras-Aila-Laine-Lehtinen

Progressive Growing of GANs for Improved Quality, Stability, and Variation

Progressive GAN

Karras 等 NVIDIA 團隊 (ICLR 2018)

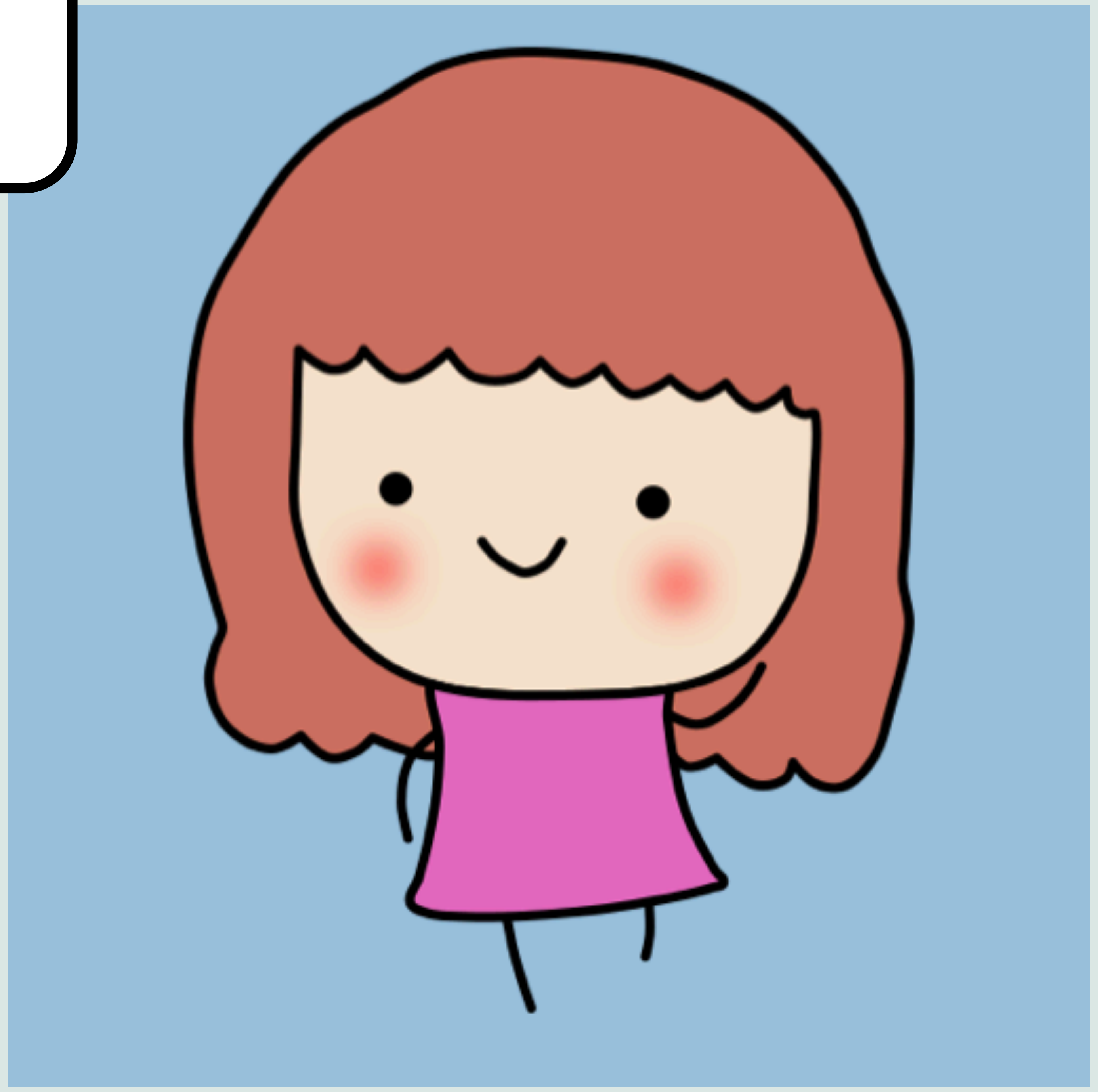
https://arxiv.org/abs/1710.10196

"Progressive Growing of GANs for Improved Quality, Stability, and Variation"

- NVIDIA 團隊很有名的文章
- 當初用現在基本上沒有在用的 Theano, Python 2, 而且只有單 GPU

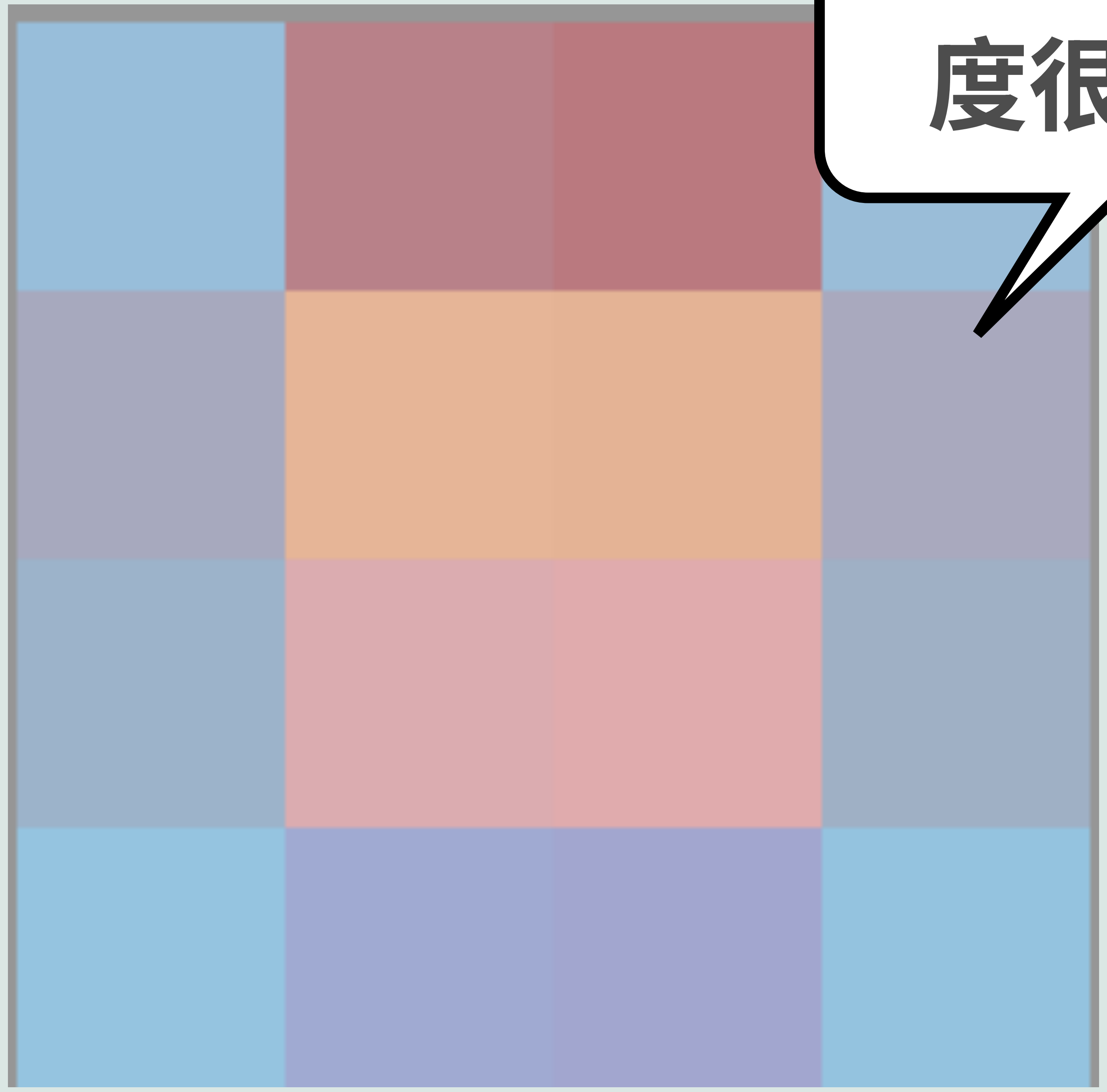
Progressive GAN 原理

直接生高解析度很難。



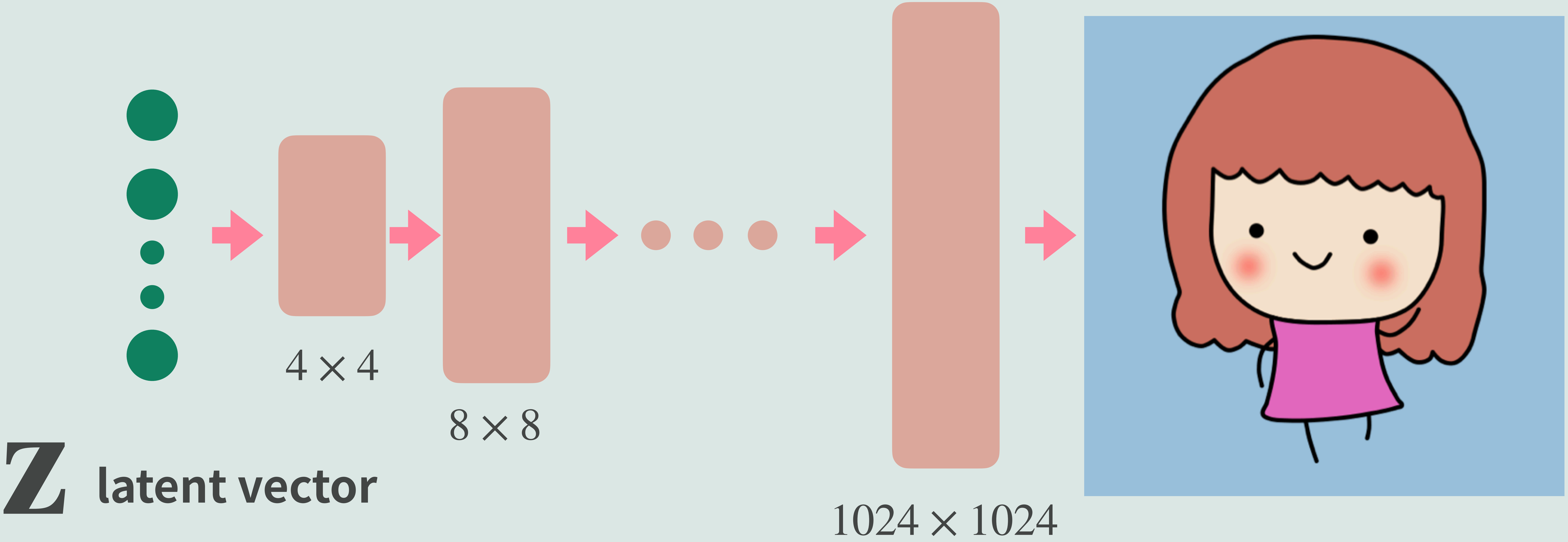
1024 × 1024

直接生低解析度很容易。



4 × 4

生成器就從簡單學到精細



Progressive GAN 生成明星照片

這攏係假ㄟ啦 (1024x1024 明星照)

This X Does Not Exist

This Word Does Not Exist

Using generative adversarial networks (GAN), we can learn how to create realistic-looking fake versions of almost anything, as shown by this collection of sites that have sprung up in the past month. Learn how it works.

This Person Does Not Exist

The site that started it all, with the name that says it all. Created using a style-based generative adversarial network (StyleGAN), this website had the tech community buzzing with excitement and intrigue and inspired many more sites.

This Cat Does Not Exist

These purr-fect GAN-made cats will freshen your feeline-gs and make you wish you could reach through your screen and cuddle them. Once in a while the cats have visual deformities due to imperfections in the model – beware, they can cause nightmares.

This Rental Does Not Exist

Why bother trying to look for the perfect home when you can create one instead? Just find a listing you like, buy some land, build it, and then enjoy the rest of your life.

我是不是可以有點「控制」生成的東西?

新的 latent vector

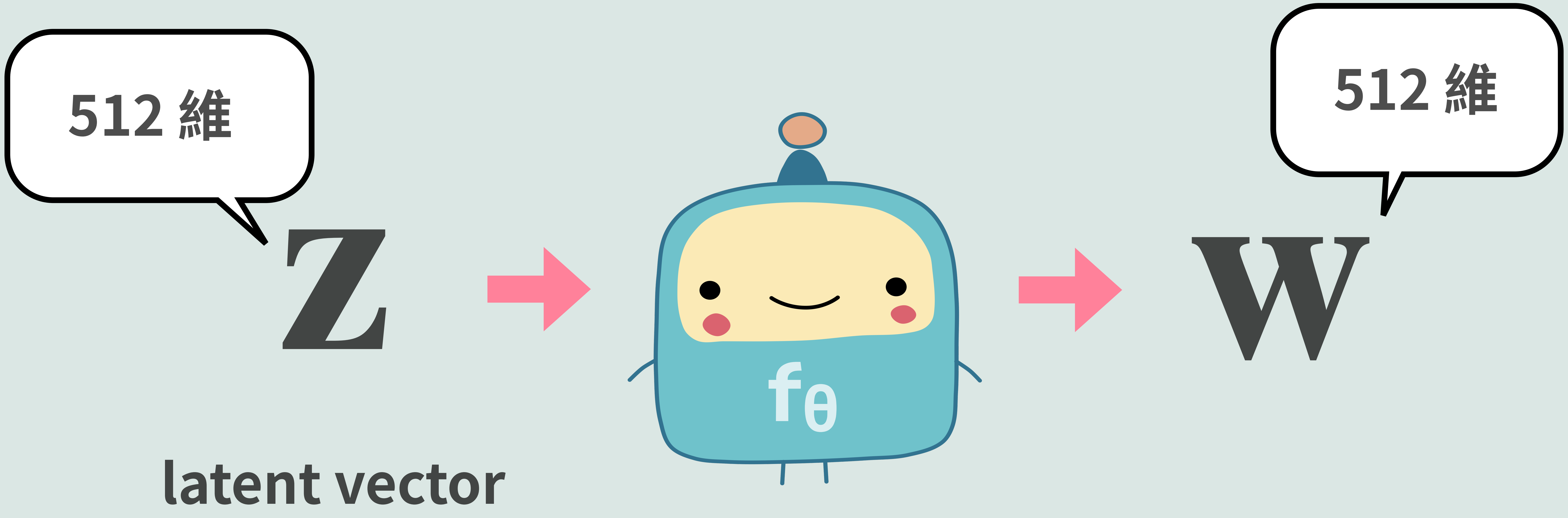
$$\mathbf{z}' = \mathbf{x} + \mathbf{w}$$

content vector

style vector

我們可以想像, 在一個 latent vector 中是內容向量 (或天馬行空的創意), 「加上」 (不一定真的加, 反正就合起來) 風格向量。

StyleGAN 原版用看來比較有學問的加法



因為種種的理由, 我們先把輸入的 latent vector 轉成 w

然後做一次 affine transformation

學來的

學來的

$$A\mathbf{w} + \mathbf{b}$$

$\mathbf{y}_s$ $\mathbf{y}_b$

$$\mathbf{y} = (\mathbf{y}_s, \mathbf{y}_b)$$

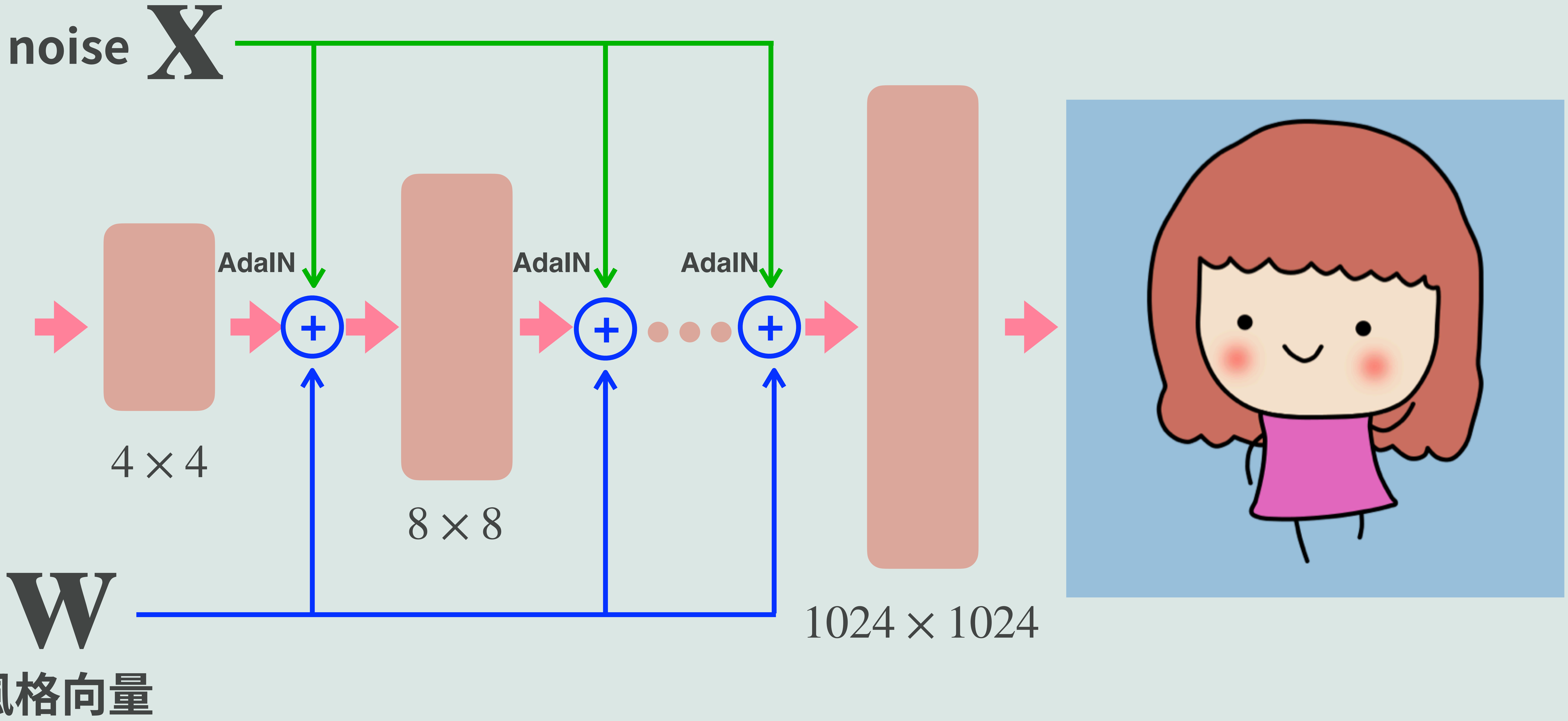
真正是 AdaIN 這樣加入!

對第 i 個 channel 做 normalization

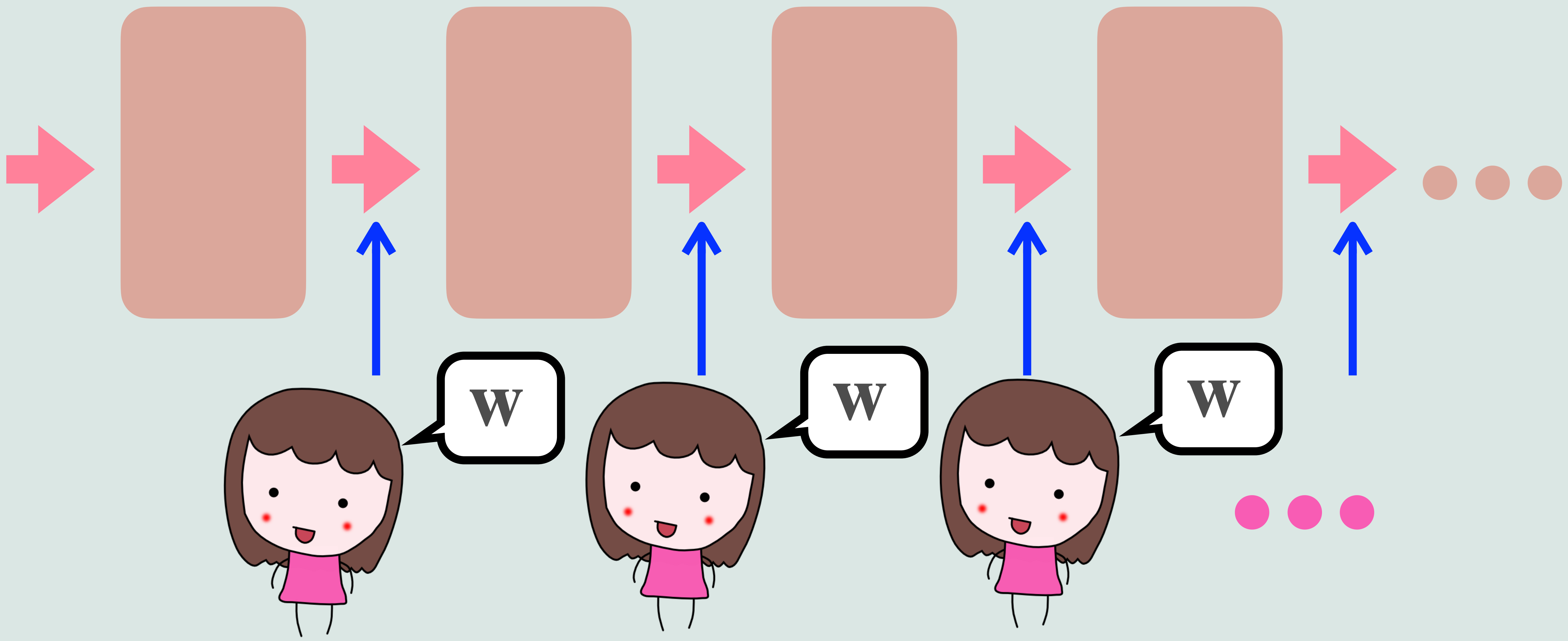
$$\text{AdaIN}(\mathbf{x}_i, \mathbf{y}) = \mathbf{y}_{s,i} \frac{\mathbf{x}_i - \mu(\mathbf{x}_i)}{\sigma(\mathbf{x}_i)} + \mathbf{y}_{b,i}$$

第 i 個 channel

生成器就從簡單學到精細

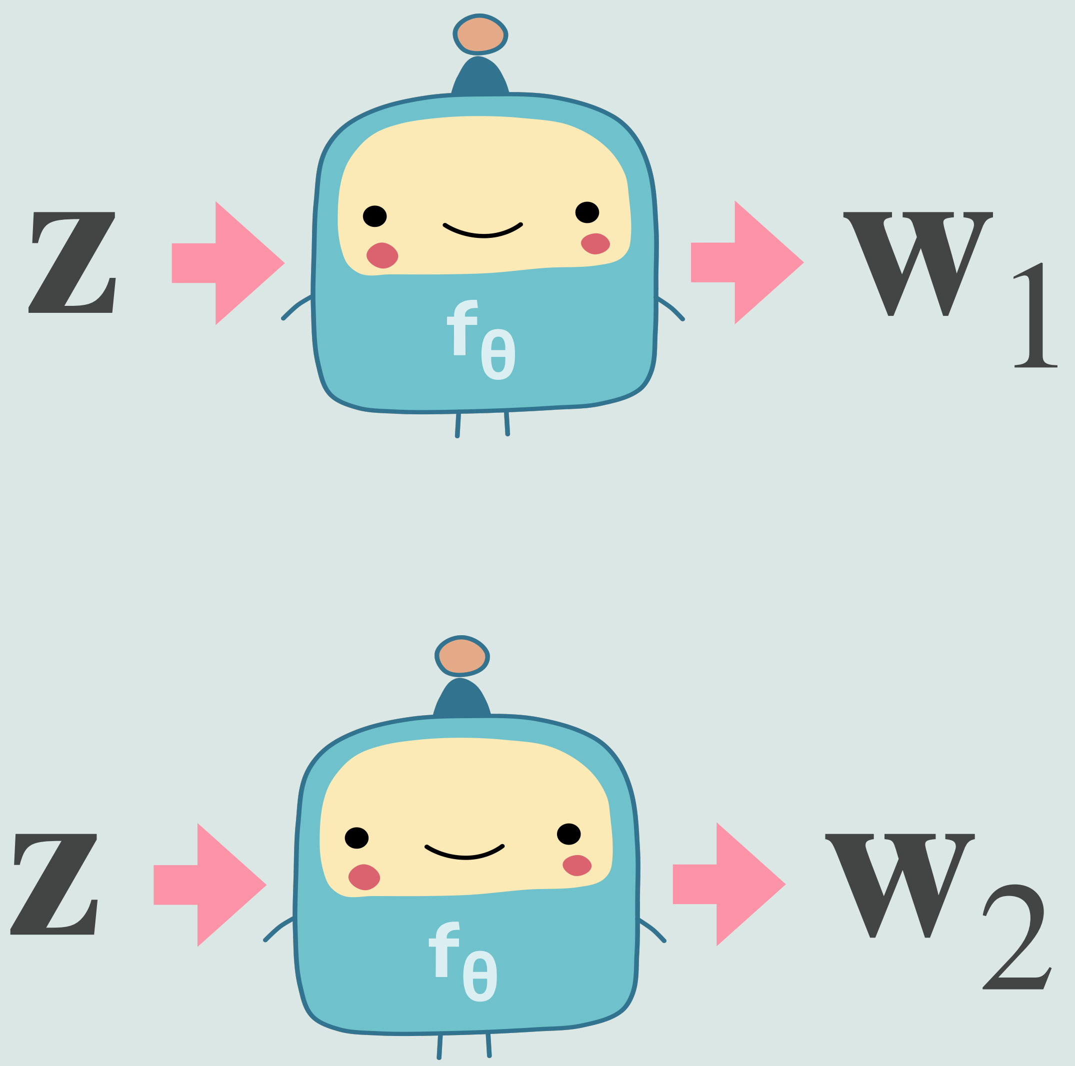


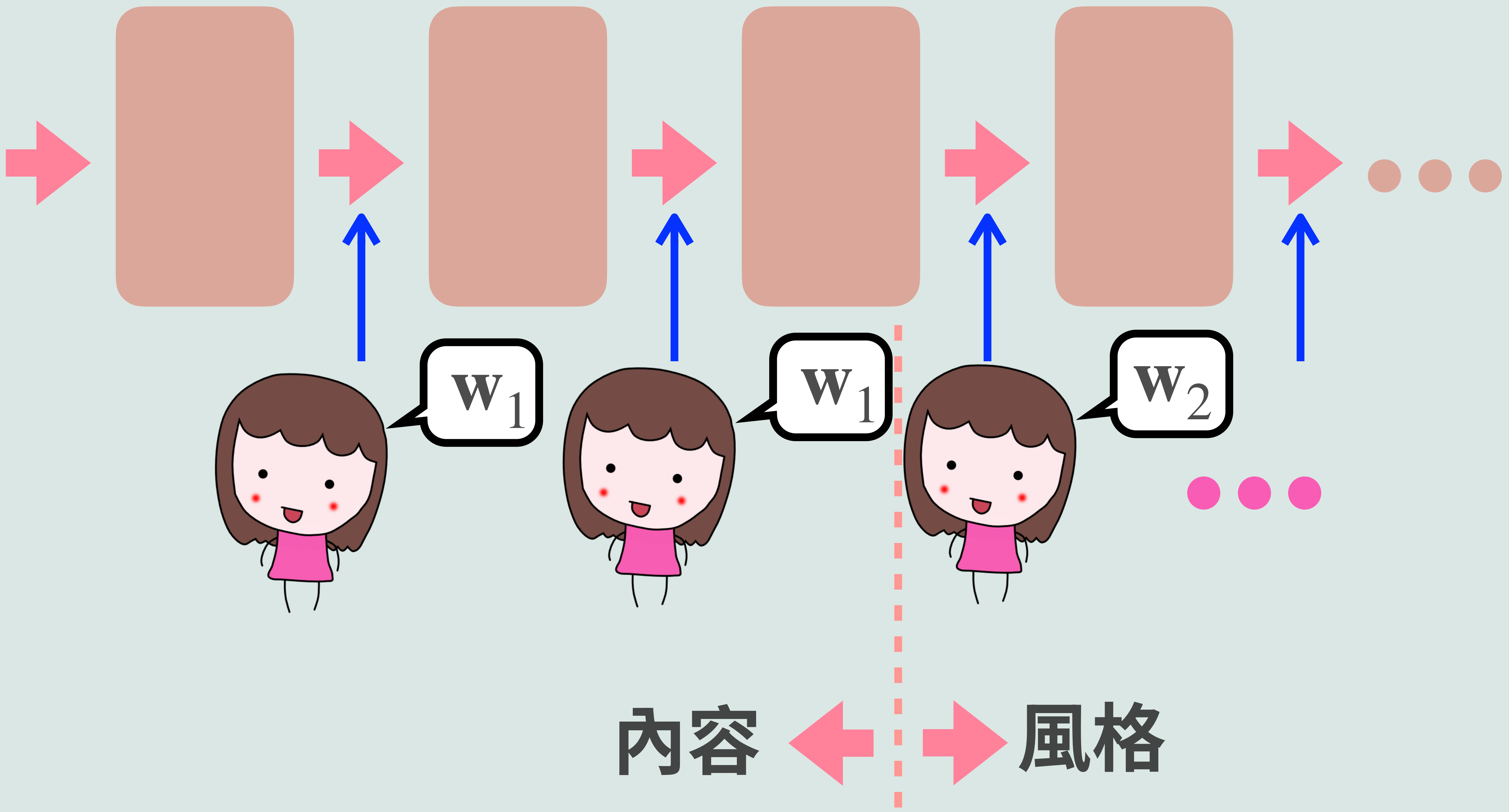
W 就像是 prompt



不斷提醒 AI 該怎麼畫

我們可以變換提示



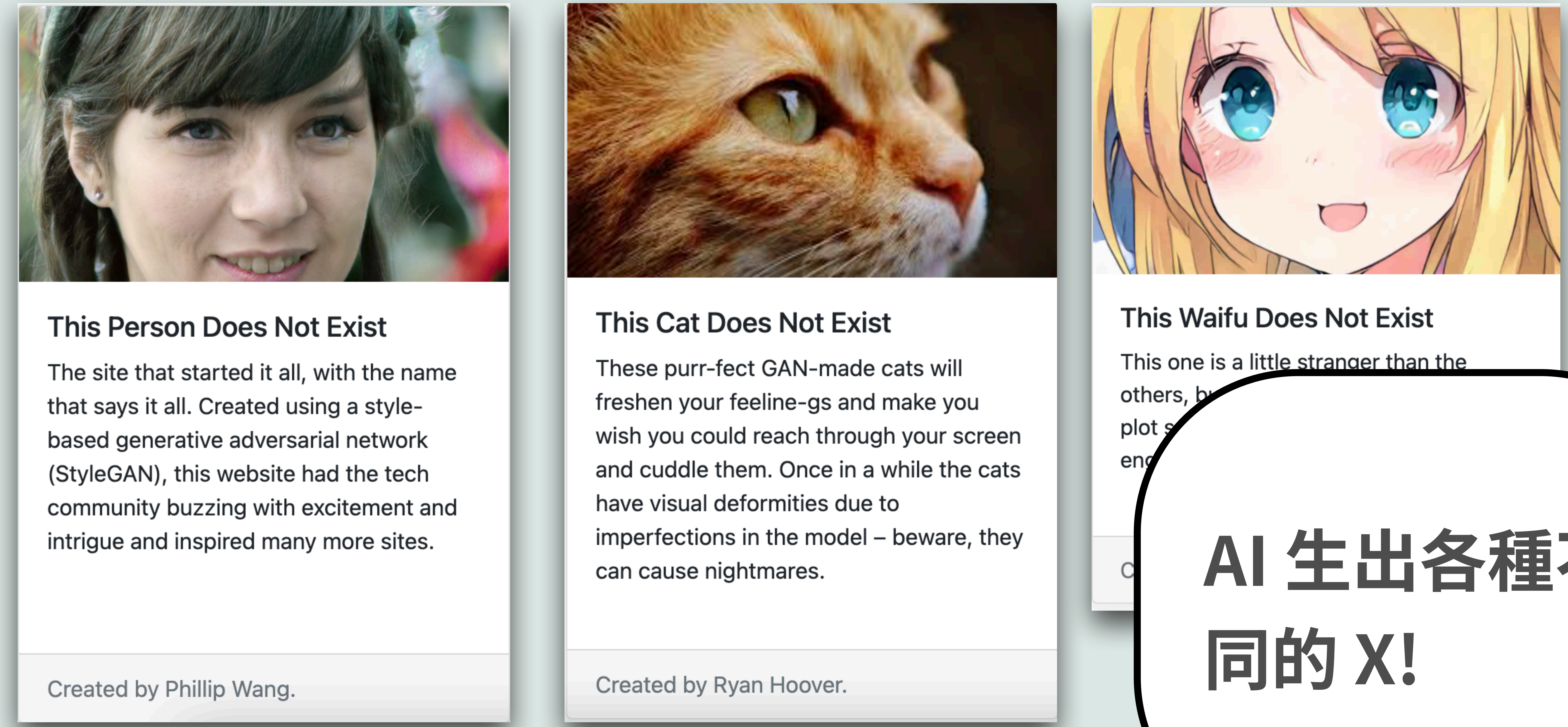


生成人又比第一代強!



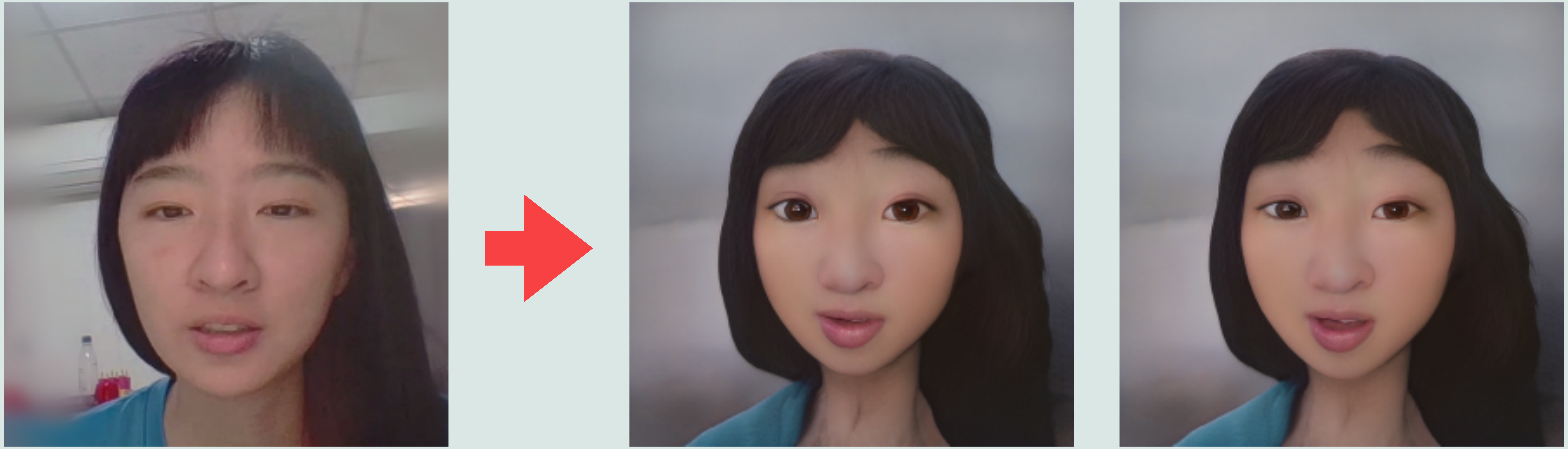
https://thispersondoesnotexist.com/

This X Does Not Exist



https://thisxdoesnotexist.com/

人臉變迪士尼風格!



魏澤人老師整理成 Colab Notebook:

https://bit.ly/colab_toonify

Pix2pix: 簡單畫畫就產生很真實的相片

Pix2pix 把衛星圖變地圖。

* 來自 Isola, 朱俊彥等人的原始論文 (2017)

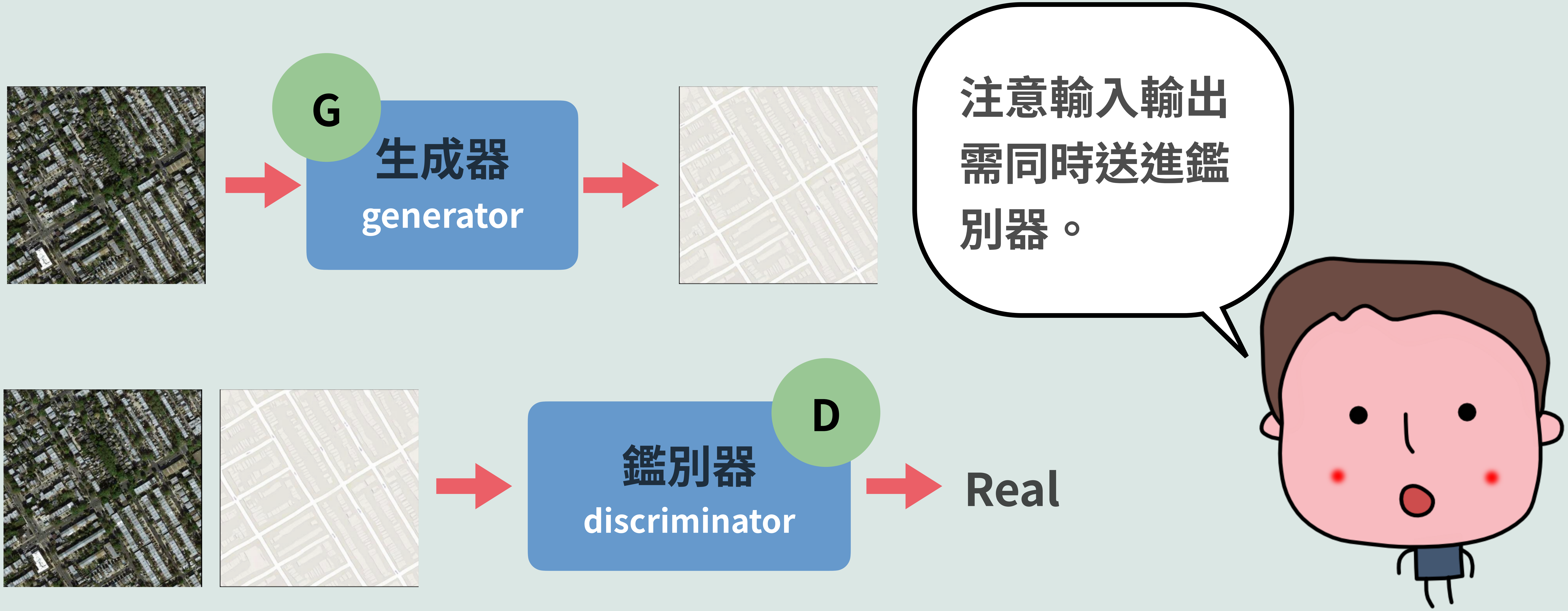
Pix2Pix

Isola, 朱俊彥等人 (CVPR 2017)

https://arxiv.org/abs/1611.07004

"Image-to-Image Translation with Conditional Adversarial Networks"

Pix2pix: 簡單畫畫就產生很真實的相片

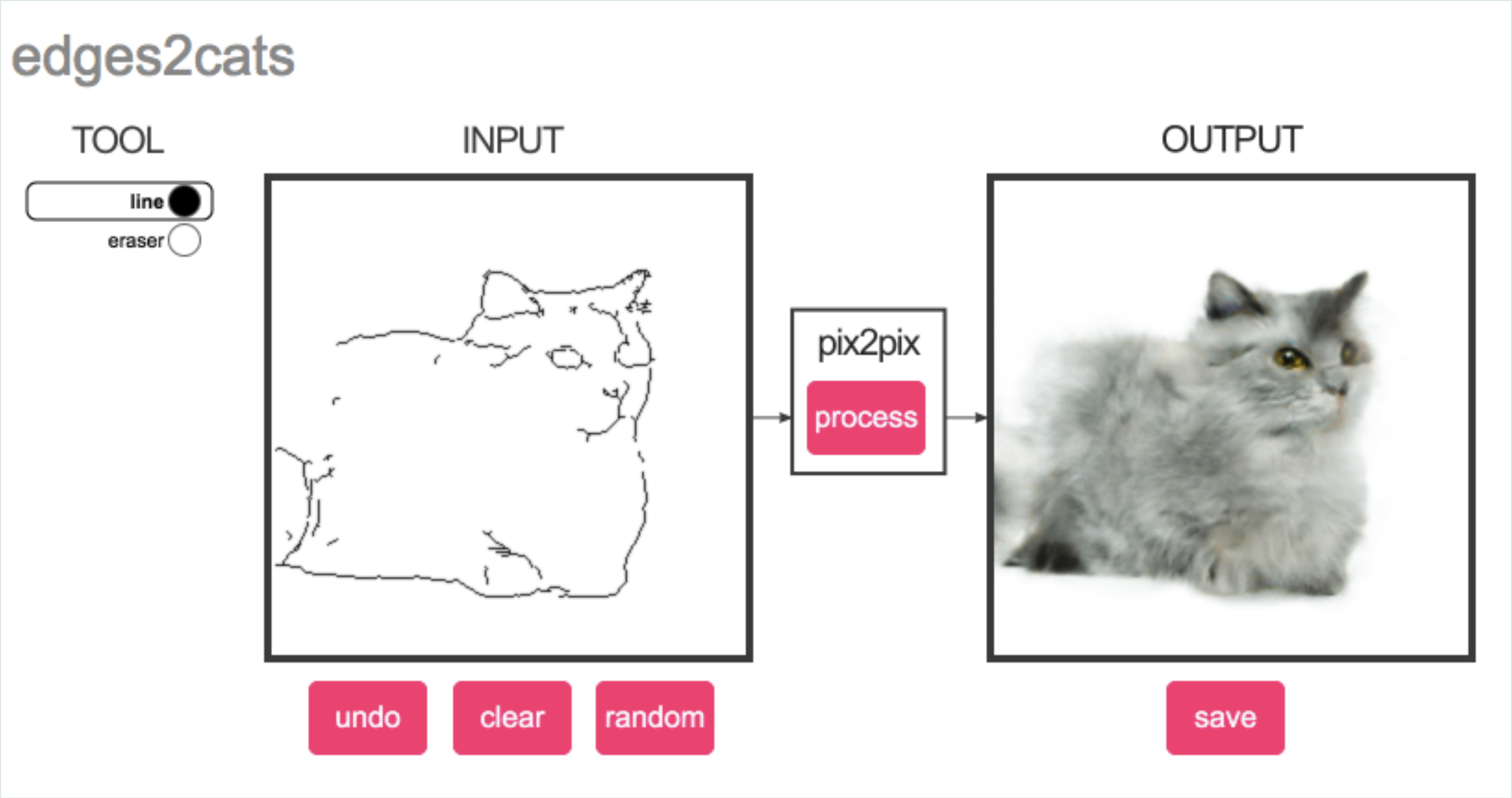


Pix2pix: 簡單畫畫就產生很真實的相片

Pix2pix 隨手畫畫變街景。

* 來自 Isola, 朱俊彥等人的原始論文 (2017)

Pix2pix: 簡單畫畫就產生很真實的相片



Pix2pix 線上版

https://affinelayer.com/pixsrv/

* Christopher Hesse 依原論文做出

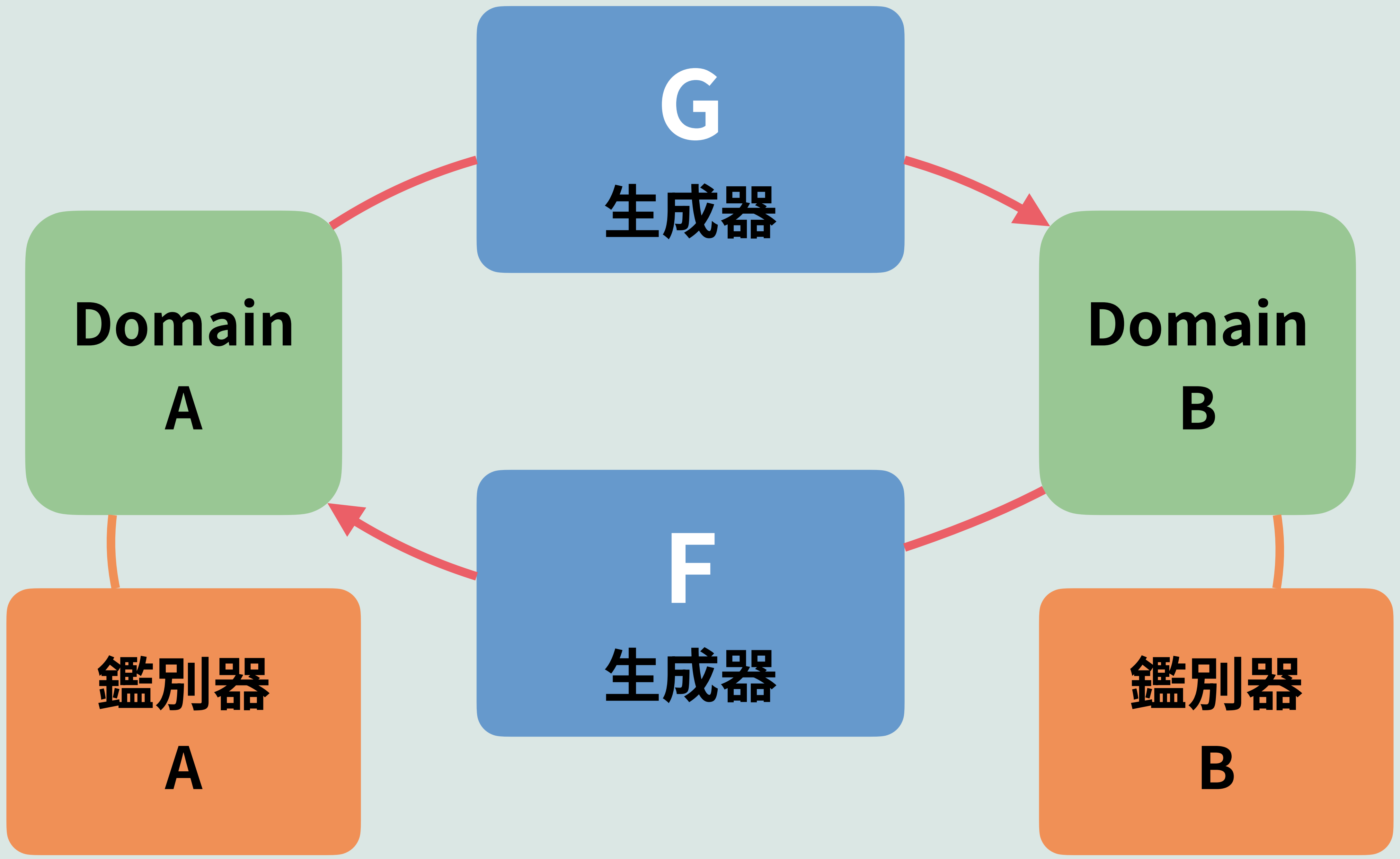
CycleGAN

朱俊彥等人 (ICCV 2017)
https://arxiv.org/abs/1703.10593

"Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks"

CycleGAN: 讓人驚呆的魔法

CycleGAN



資料「不需要」配對!

於是有無限可能...

CycleGAN: 讓人驚呆的魔法

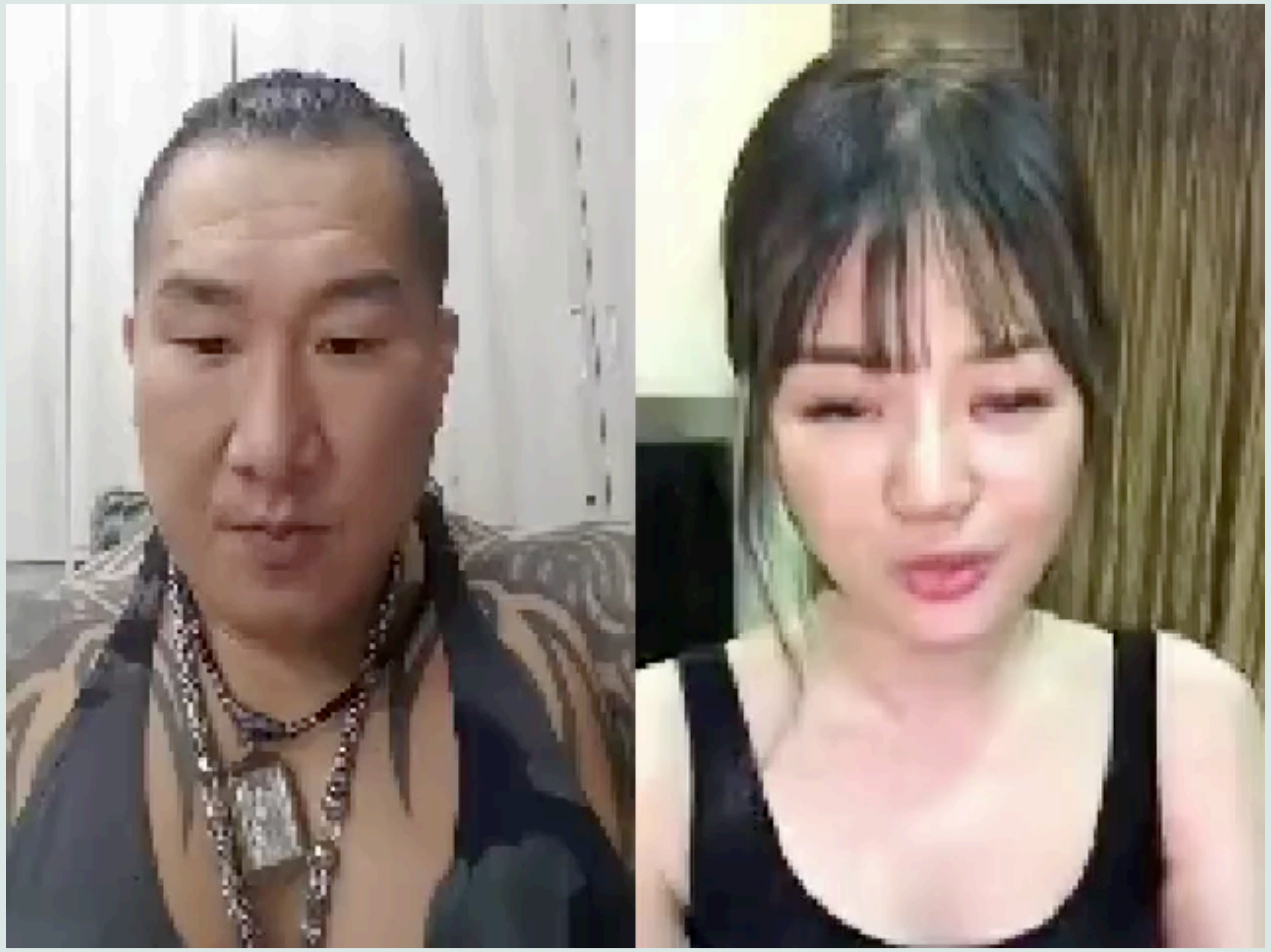
全世界的人都驚呆了的馬變斑馬。

https://youtu.be/9reHvktowLY

CycleGAN: 讓人驚呆的魔法

CycleGAN 作者群幽自己一默的失敗例子。

CycleGAN: 讓人驚呆的魔法

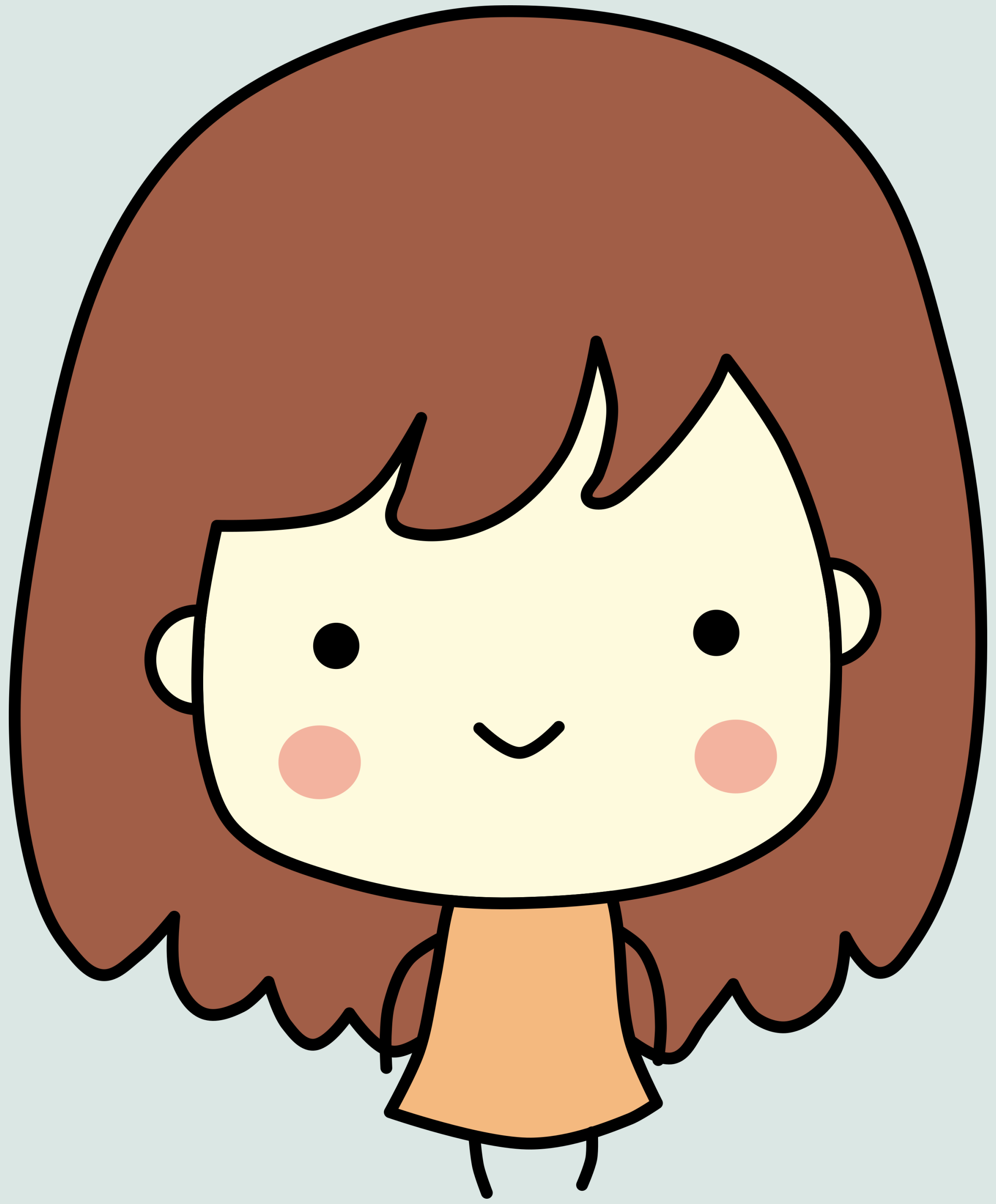


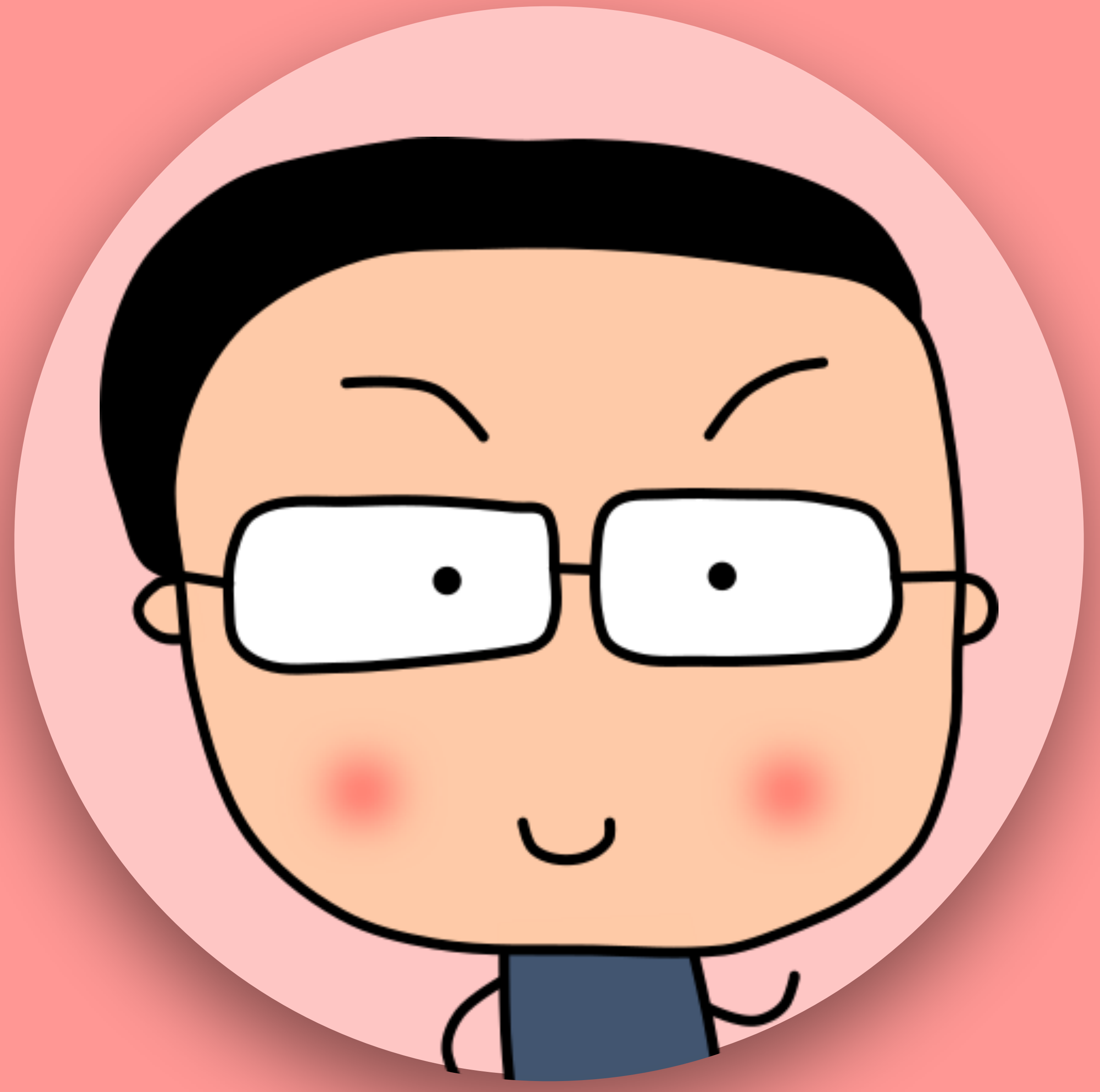
Goodfellow 也關注的館長變陳沂 (魏澤人老師)。

https://youtu.be/Fea4kZq0oFQ

大家一度認為, GAN 是電腦創作的巔峰

直到出現
diffusion models。

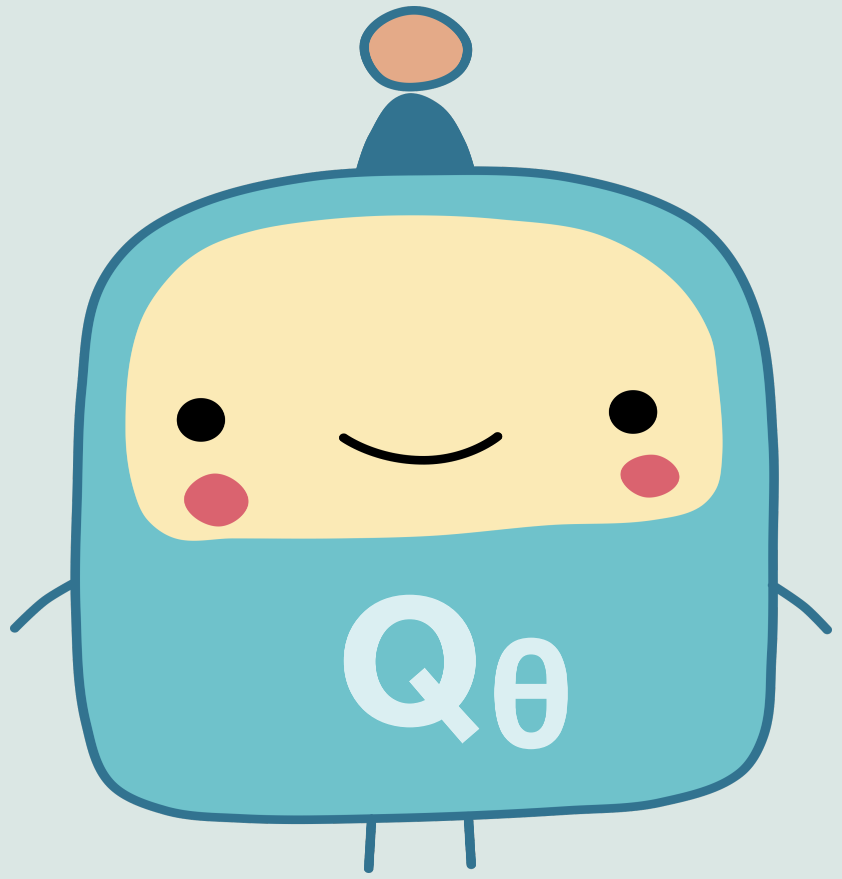




附錄.

學 AI 你一定要弄懂的 Cross Entropy

前情摘要



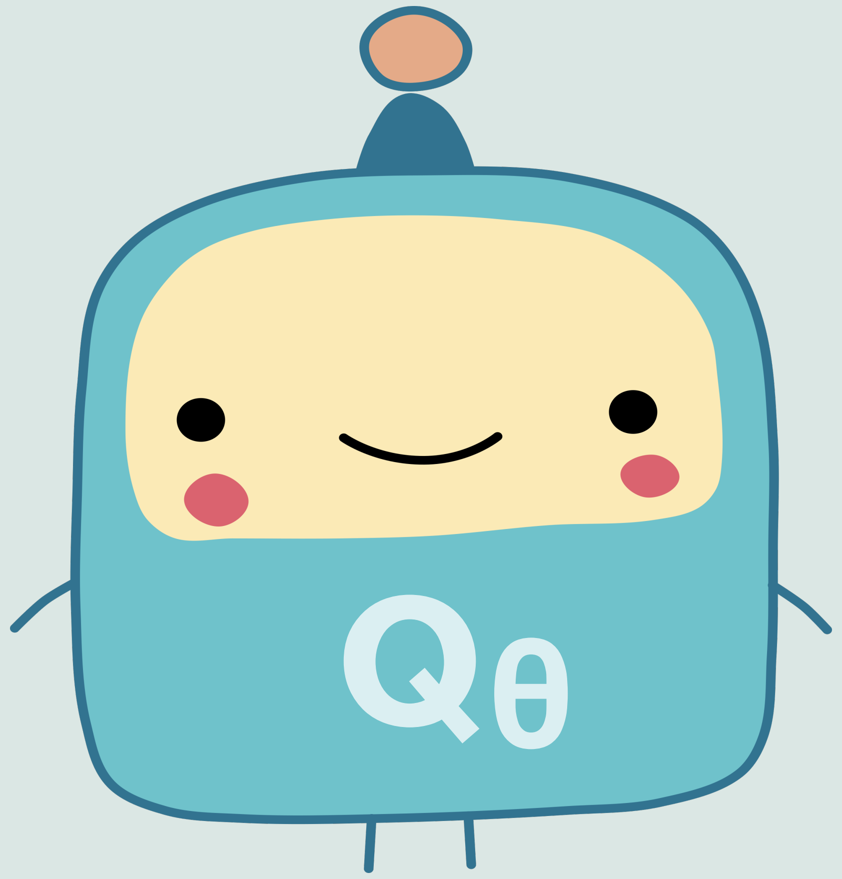
$\hat{y}_1$ 土八哥

$\hat{y}_2$ 白尾八哥

$\hat{y}_3$ 家八哥

最後經 softmax, 八哥辨識就可以看成每次輸出一個機率分布

比如說



$\hat{y}_1$ 0.6 土八哥

$\hat{y}_2$ 0.3 白尾八哥

$\hat{y}_3$ 0.1 家八哥

正確答案是 [1, 0, 0], 所以的確是「正確」的

計算 loss

我們的答案 [0.6, 0.3, 0.1] 和正確答案 [1, 0, 0] 差多少呢?

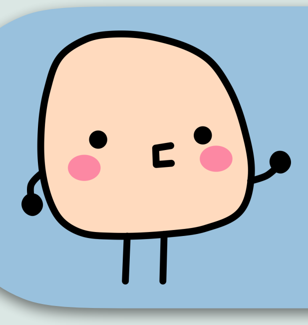
$$(1-0.6)^2+(0-0.3)^2+(0-0.1)^2=0.26$$

如果是 [0.6, 0.2, 0.2] 和正確答案 [1, 0, 0] 差多少呢?

$$(1-0.6)^2+(0-0.2)^2+(0-0.2)^2=0.24$$

但是

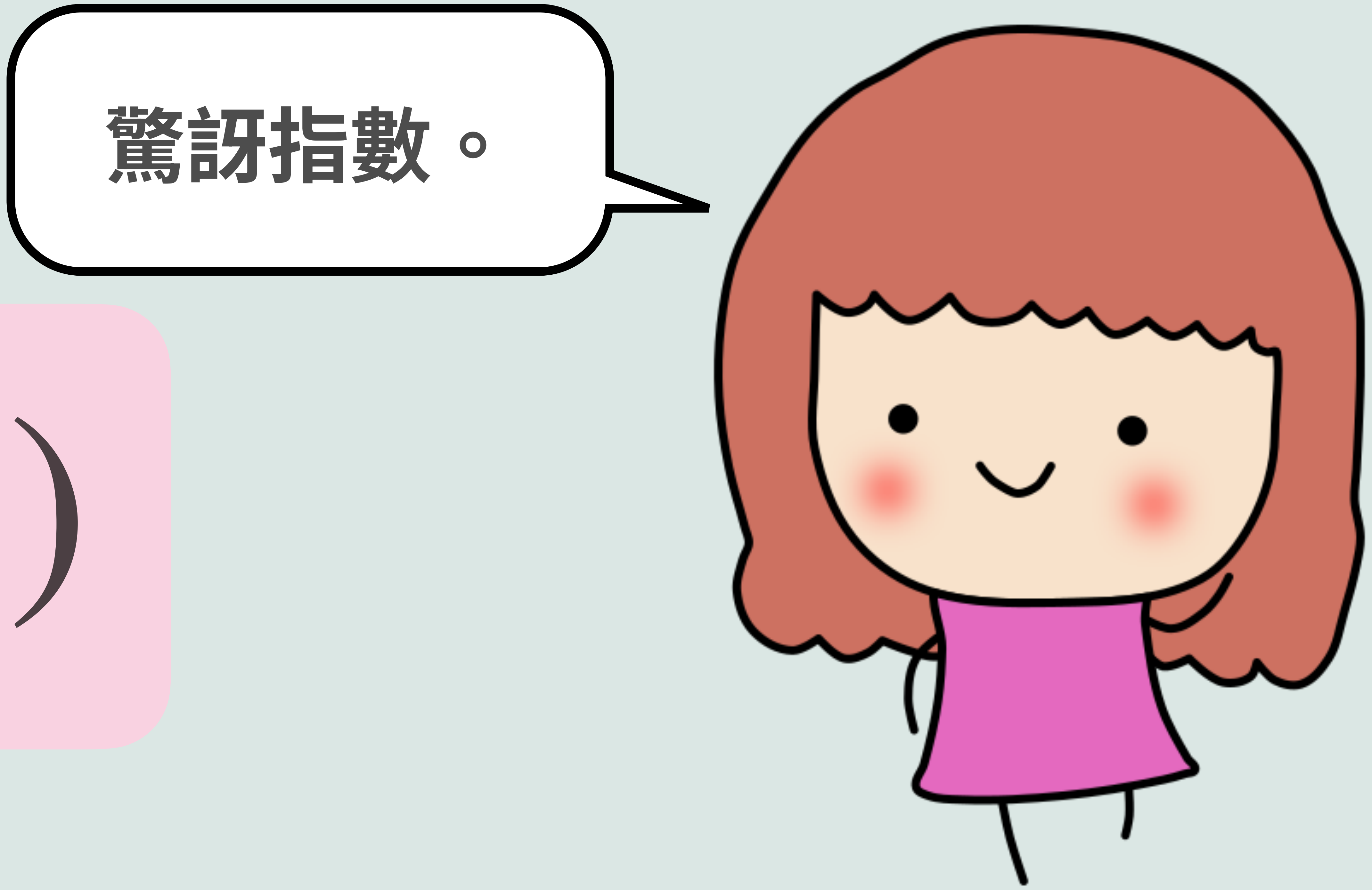
這兩個答案, 都算正確, 很多時候也是「一樣好」 (因為正確都是 60%), 有沒有可能有算法讓 loss 一樣呢?



資訊量 (驚訝指數)

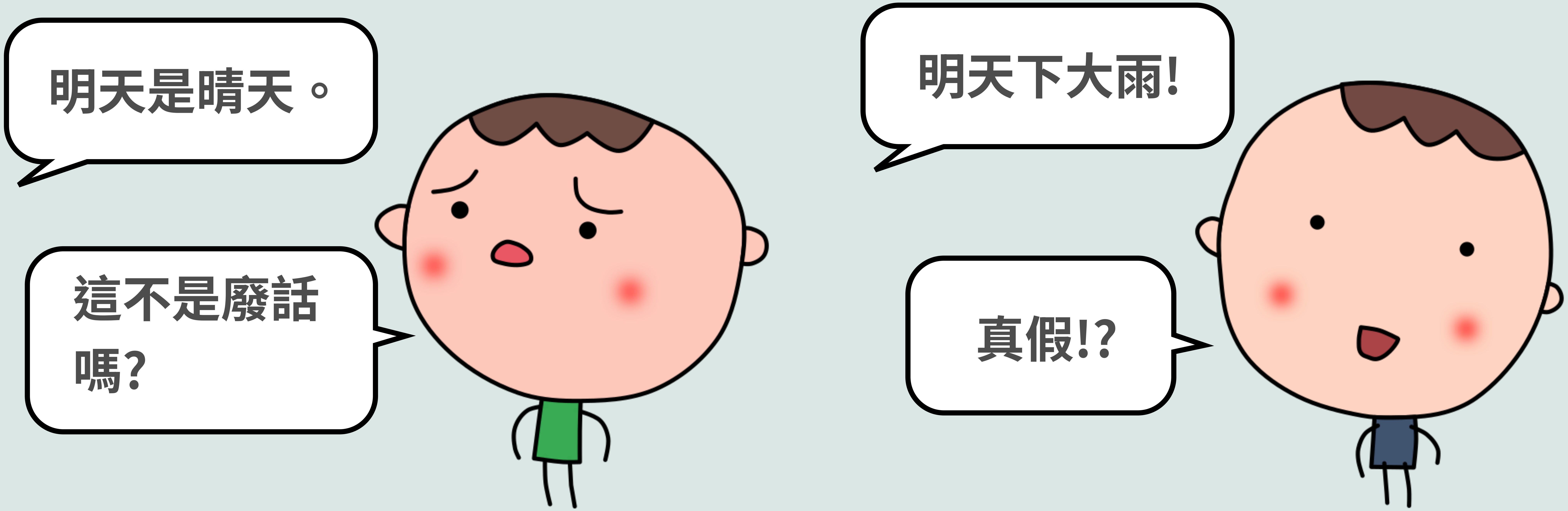
發生一件事情 x, 如果發生機率 p 很低的話, 我們會很驚訝; 反之, 機率 p 很高, 我們就一點也不驚訝。這樣要怎麼表示呢, 你認真想想, 就會發現應該是...

$$-\log(p)$$



機率越低的事, 訊息資訊量越大

比如在幾乎不下雨的加州...



資訊量小

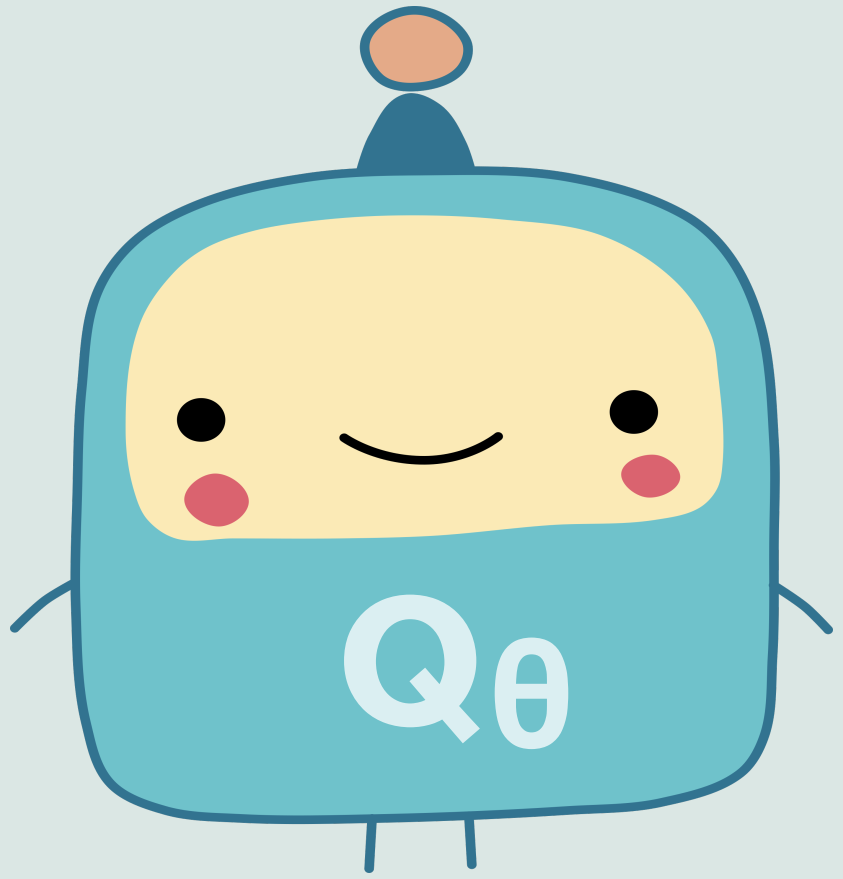
資訊量大

資訊量 (驚訝指數)

正式的名稱是 x 事件的「資訊量」。

$$I(x) = -\log P(x)$$

回到我們的八哥



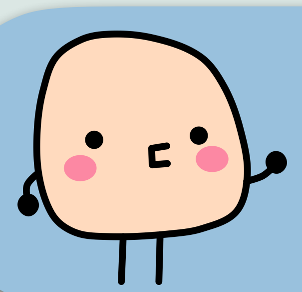
$\hat{y}_1$	0.6	土八哥
$\hat{y}_2$	0.3	白尾八哥
$\hat{y}_3$	0.1	家八哥

我們只在意正確答案 (土八哥) 那的得分, 如果土八哥得分越高, 扣分越少; 反之土八哥得分越低, 我們就越「驚訝」...

Cross Entropy

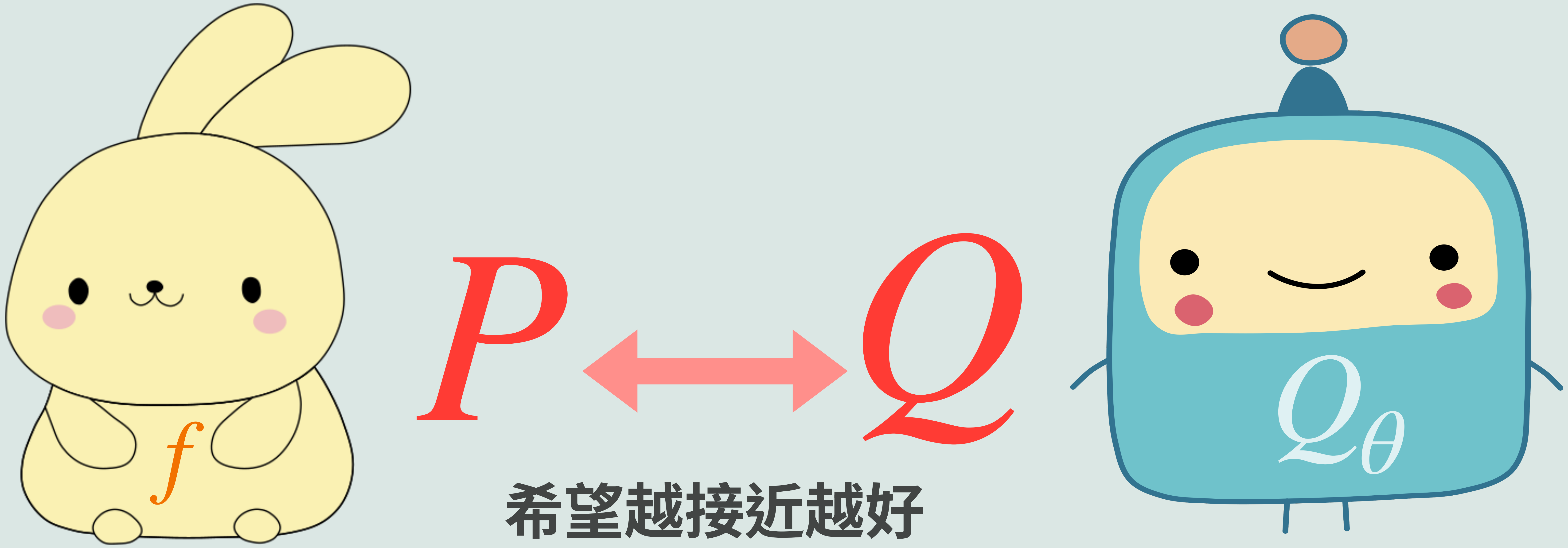
假設八哥分類正確答案 [1, 0, 0], 我們的答案是 [0.6, 0.3, 0.1], 其 Cross Entropy 為:

$$-\log(\hat{y}_1)$$



總言之, 就是比較真實世界和神經網路學到的機率分佈

雖然我們說過, 神經網路的機率分佈是我們用 softmax 硬是弄成的。

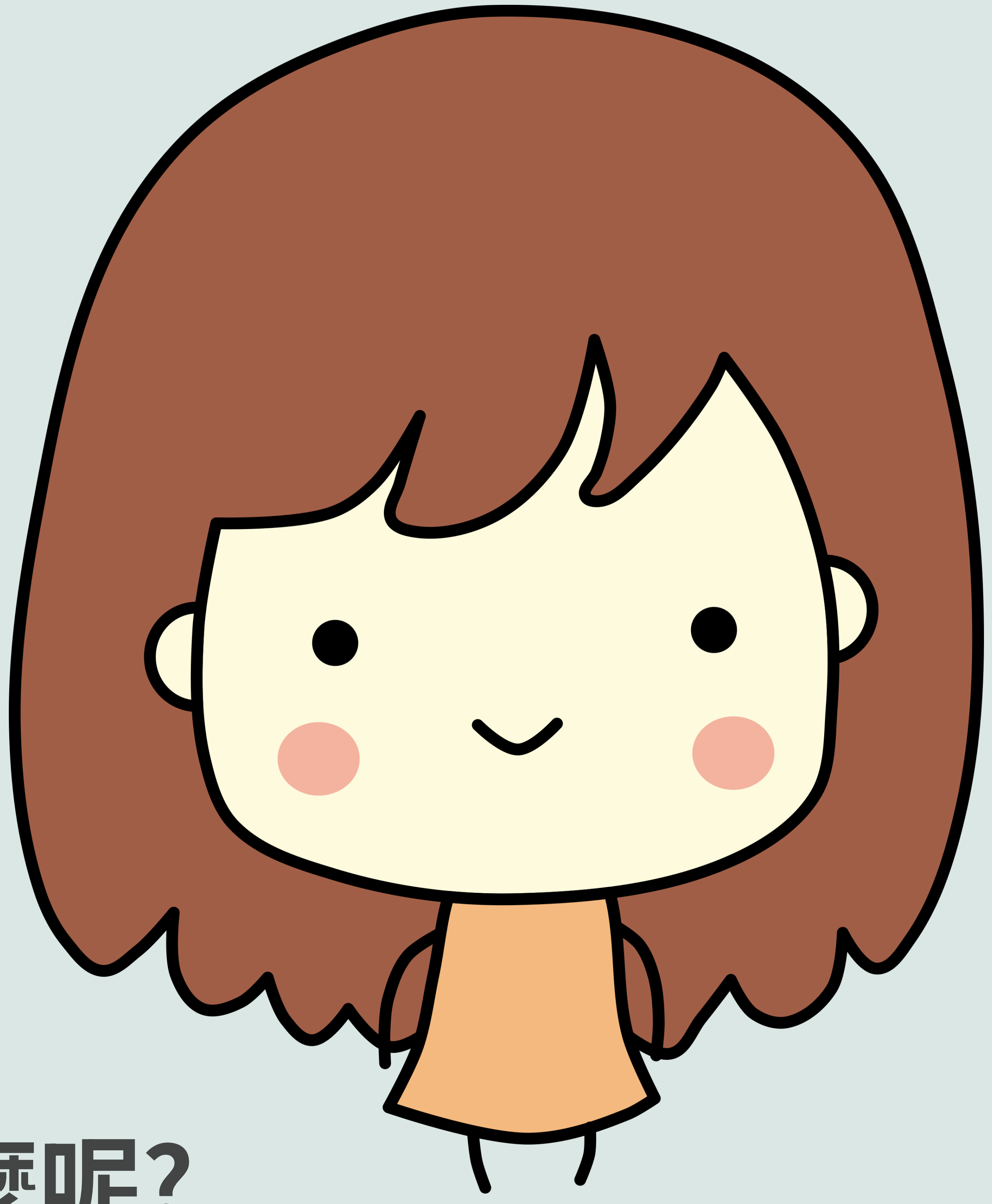


真實世界

神經網路

Cross Entropy

Cross Entropy 也可以當我們的 loss function (事實上分類問題這個還更適合)。



但是, 我們到底 cross 在哪? Entropy 又是什麼呢?

假設一個機率分佈有 n 個事件

機率分別是 $p_1, p_2, \cdots, p_n$。

$$P = [p_1, p_2, \cdots, p_n]$$

Entropy (熵, 能趨疲) 是訊息量的期望值 (平均)

Entropy 意思是平均訊息量。越大表示平均訊息量越大, 也就是可能的情況越複雜; 當只有一個事件, 發生機率是 1, 所以 entropy 會是 0。也因此大家常說 entropy 是「亂度」。

$$H(P) = -\sum_{i=1}^{n} p_i \log p_i$$

寫成期望值會這樣表示

設 P 為某機率分布, 其 entropy 定義為:

$$H(X) = -\mathbb{E}_{x\sim P}[\log P(x)]$$
$$= -\sum_{x\sim P} P(x)\log P(x)$$

(Shannon) Entropy 熵

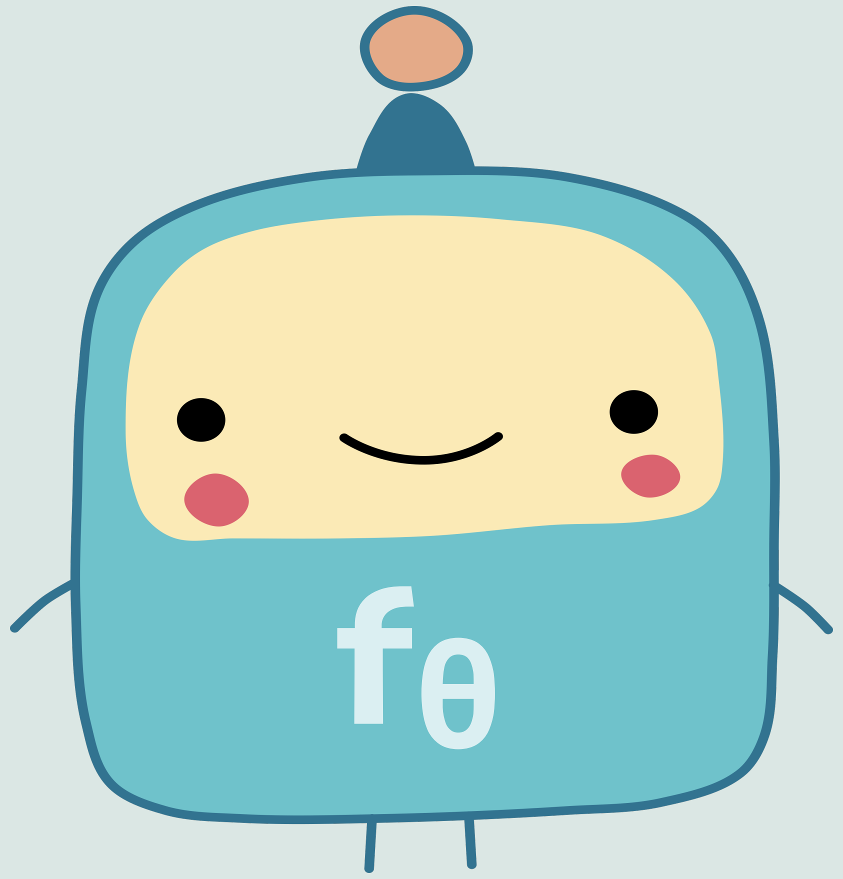
有一個機率分布 P 有三個可能的事件, 機率分別是 $P = (p_1, p_2, p_3)$

$$H(X) = -\sum_{i=1}^{3} p_i \log(p_i)$$

$$= -[p_1 \log(p_1) + p_2 \log(p_2) + p_3 \log(p_3)]$$

以八哥的例子來說

$\mathbf{x}$



$\hat{\mathbf{y}}$

- $\hat{y}_1$ 土八哥
- $\hat{y}_2$ 白尾八哥
- $\hat{y}_3$ 家八哥

$$Q \quad \hat{\mathbf{y}} = [\hat{y}_1, \hat{y}_2, \hat{y}_3]$$

神經網路的答案

用個簡單的例子看 Cross Entropy

假設有機率分布 P, Q 有共同的三種事件 (比如三種八哥), 機率分別是 $P = (y_1, y_2, y_3)$, $Q = (\hat{y}_1, \hat{y}_2, \hat{y}_3)$, 則 P, Q 的 cross entropy $H(P, Q)$ 為:

$$H(P, Q) = -\sum_{i=1}^{3} y_i \log(\hat{y}_i)$$

和正確答案的比較

P $\mathbf{y} = [y_1, y_2, y_3]$

正確答案

$$H(P,Q) = -(y_1 \log \hat{y}_1 + y_2 \log \hat{y}_2 + y_3 \log \hat{y}_3)$$

0 0

注意正確答案是如 [1,0,0] 這樣的 one-hot encoding。

分類問題的 cross entropy 其實很簡單

假設第 i 類是正確答案, 即 $y_i = 1$, 而我們的神經網路說的答案是 $\hat{y}_i$。

$$H(P, Q) = -y_i \log \hat{y}_i = -\log \hat{y}_i$$

分類問題的 cross entropy 其實很簡單

假設第 i 類是正確答案, 即 $y_i = 1$, 而我們的神經網路說的答案是 $\hat{y}_i$。

$$H(P, Q) = -y_i \log \hat{y}_i = -\log \hat{y}_i$$

再一次…

注意 $\hat{y}_i$ 很小, 這個值很大, 意思是...

$$-\log \hat{y}_i$$

複習: 均方差 MSE

就是高維度空間兩個點距離的平方。

$$L(\theta) = \frac{1}{2k} \sum_{i=1}^{k} \|\mathbf{y_i} - \hat{\mathbf{y}}_\mathbf{i}\|^2$$

為什麼不用 MSE 呢?

假設正確答案: $\mathbf{y} = [1,0,0]$

兩次的答案: $\hat{\mathbf{y}}_a = [0.7,0.19,0.11], \hat{\mathbf{y}}_b = [0.7,0.3,0]$

模型輸出	MSE	Cross Entropy
[0.7,0.19,0.11]	0.14	0.36
[0.7,0.3,0]	0.18	0.36

單純在意正確答案。

Cross Entropy 最小的時候

從 cross entropy 的公式可以看出, 當 $Q = P$ 時, $H(P, Q)$ 有最小值, 即 P 的 cross entropy $H(P)$。

$$H(P, Q) = -\sum_{i=1}^{n} p_i \log q_i$$

$Q = P$

$$= -\sum_{i=1}^{n} p_i \log p_i$$

$$= H(P)$$

注意最小值不一定是 0

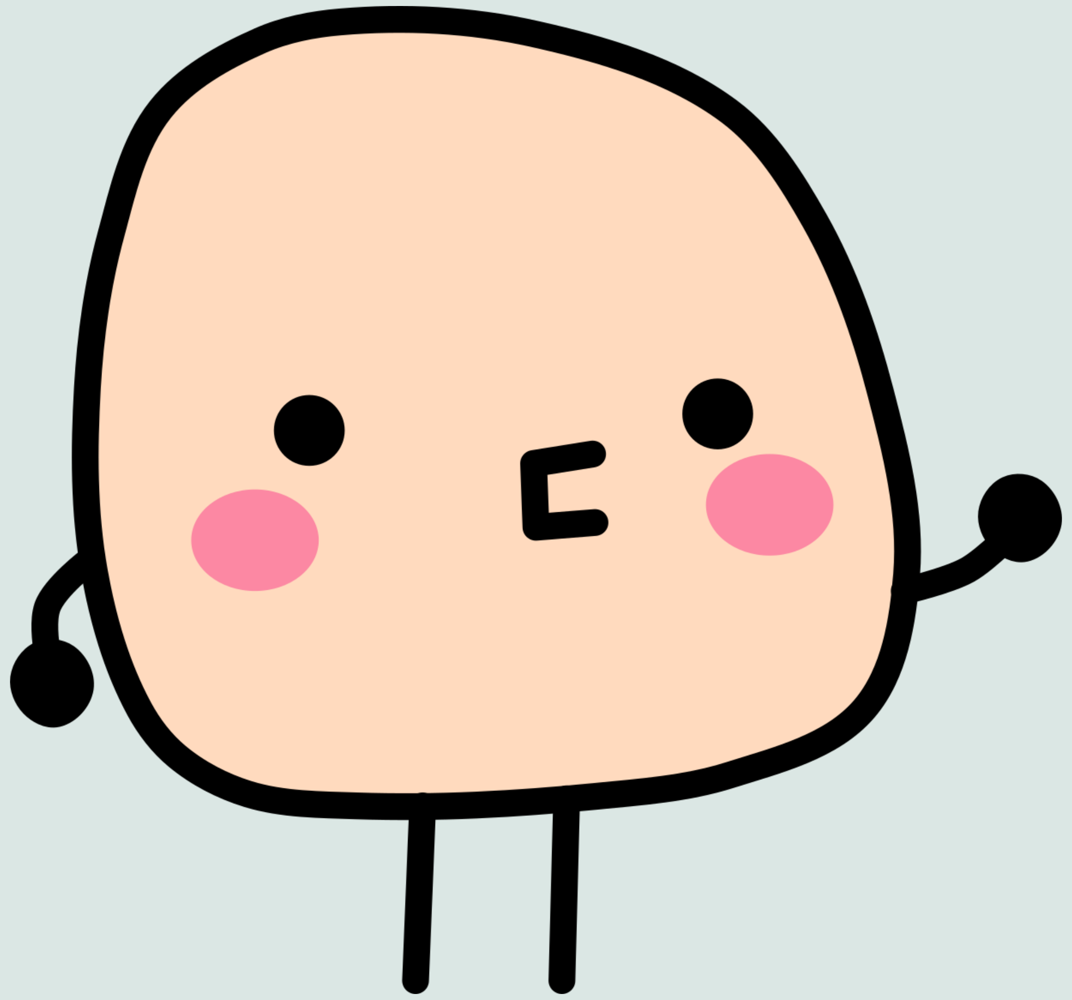
Kullback-Leibler Divergence (KL Divergence)

當 $Q = P$ 時, KL 散度為 0!

$$D_{KL}(P \| Q) = H(P, Q) - H(P)$$

我們可能會覺得這在做什麼!

$$D_{KL}(P\|Q) = H(P,Q) - H(P)$$



意思是 cross entropy 越小, KL 散度也越小。

這是真實世界的 entropy, 也就是固定的數值。

注意差異一樣, 但是差距的量級不同

假設正確答案: $\mathbf{y} = [0.7, 0.2, 0.1]$

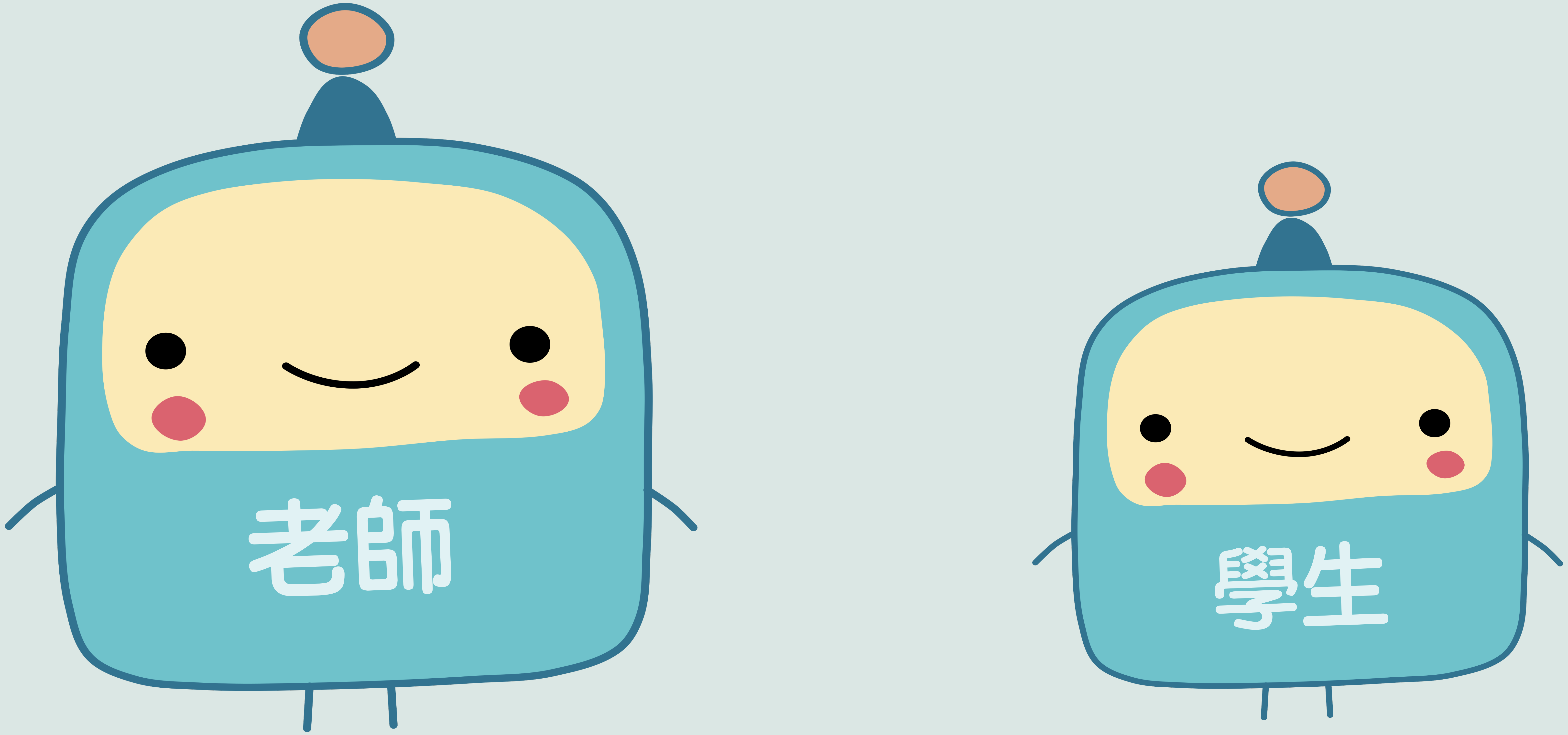
兩次的答案: $\hat{\mathbf{y}}_a = [0.75, 0.15, 0.1], \hat{\mathbf{y}}_b = [0.6, 0.3, 0.1]$

模型輸出	Cross Entropy	KL Divergence
$[0.75, 0.15, 0.1]$	0.81	0.01
$[0.6, 0.3, 0.1]$	0.83	0.03

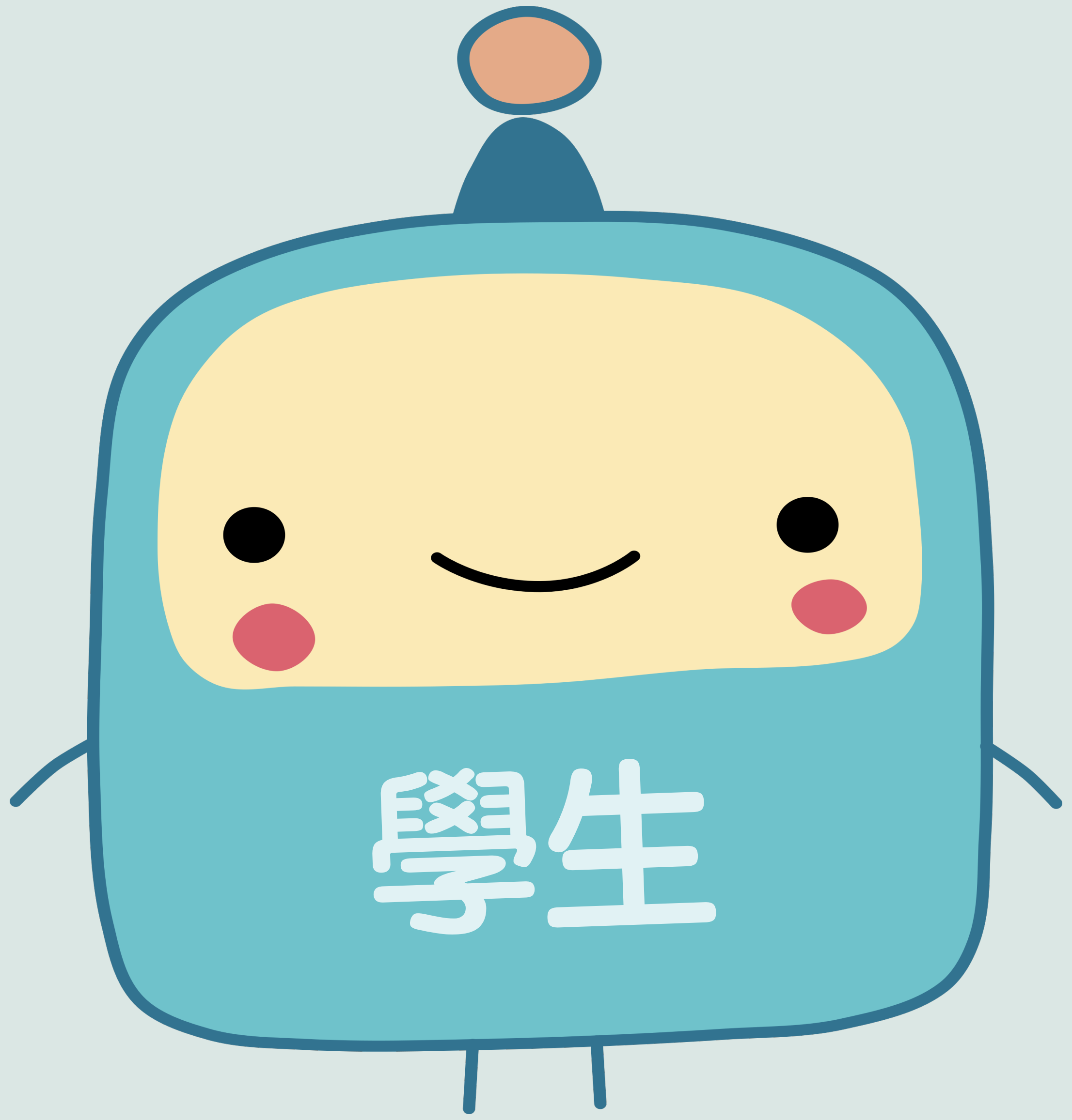
3 倍!

需要學 soft target 的時候，就該考慮 KL 散度

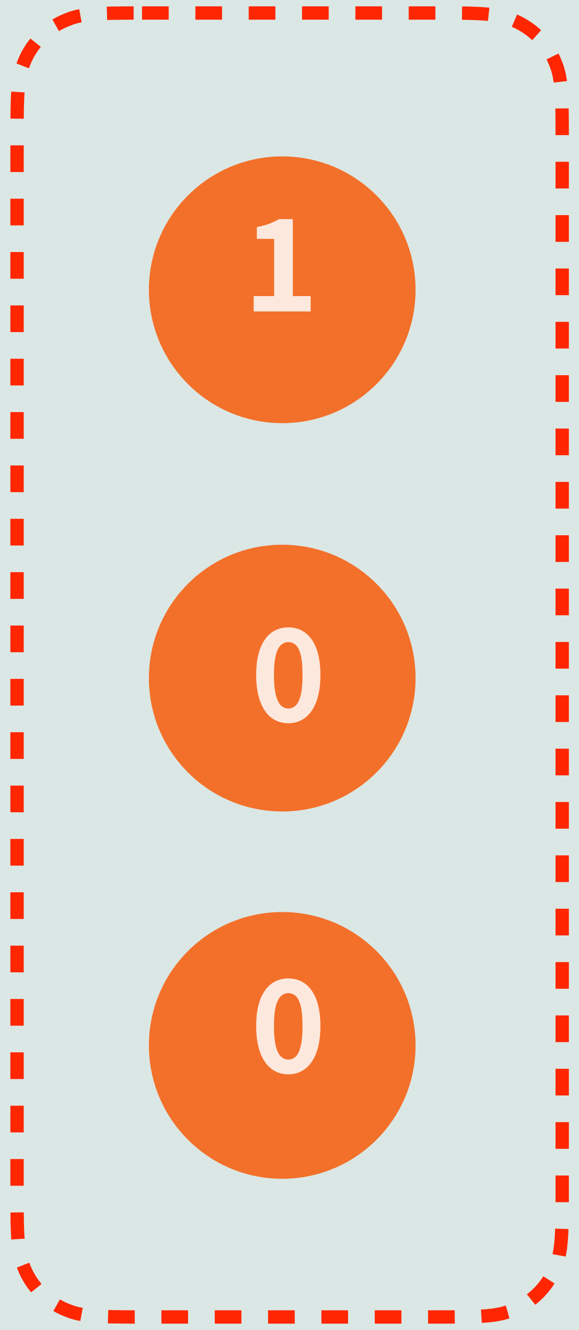
比如說**蒸餾**



假設我們要學八哥辨識，一般這樣學...

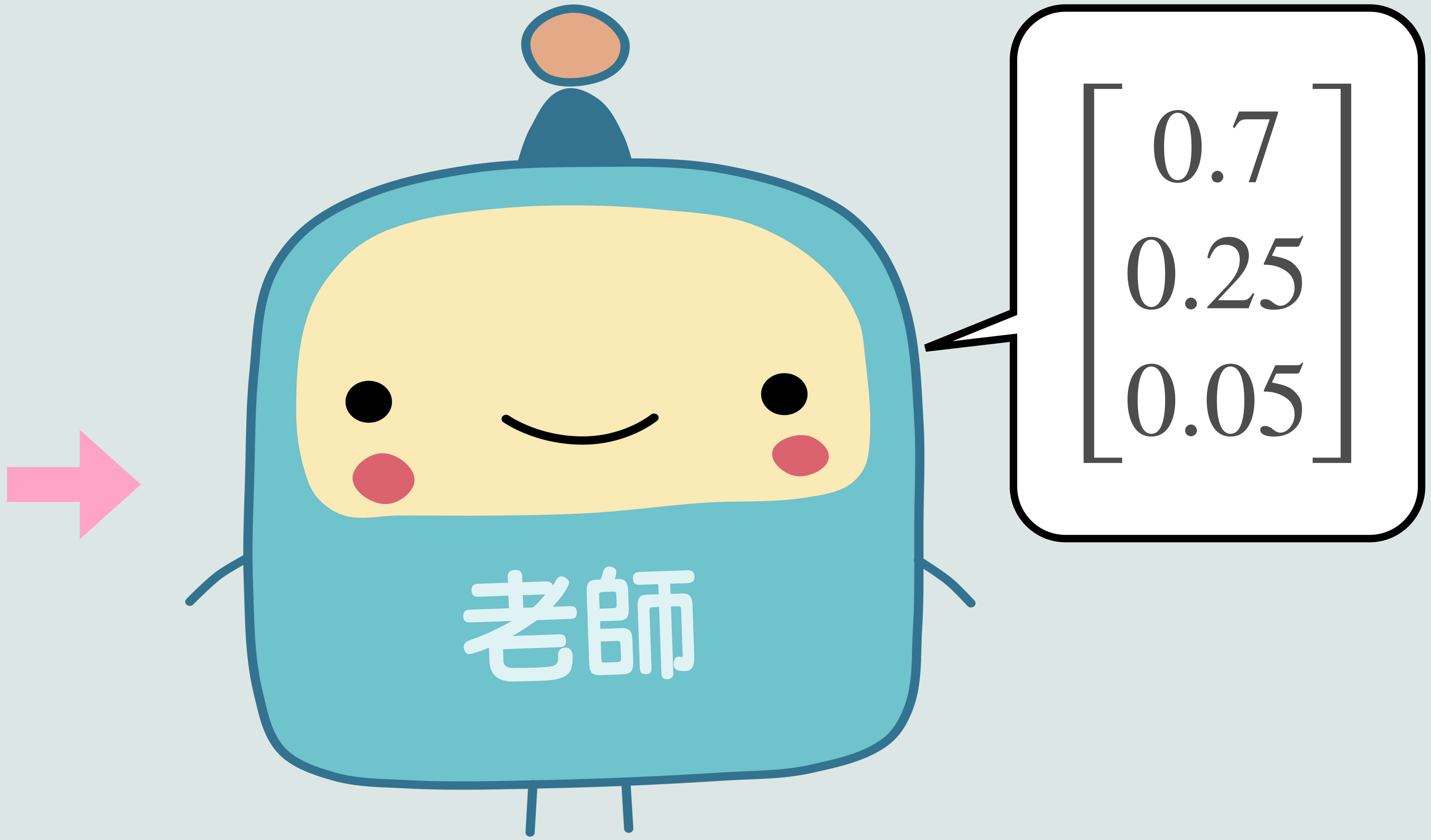


目標

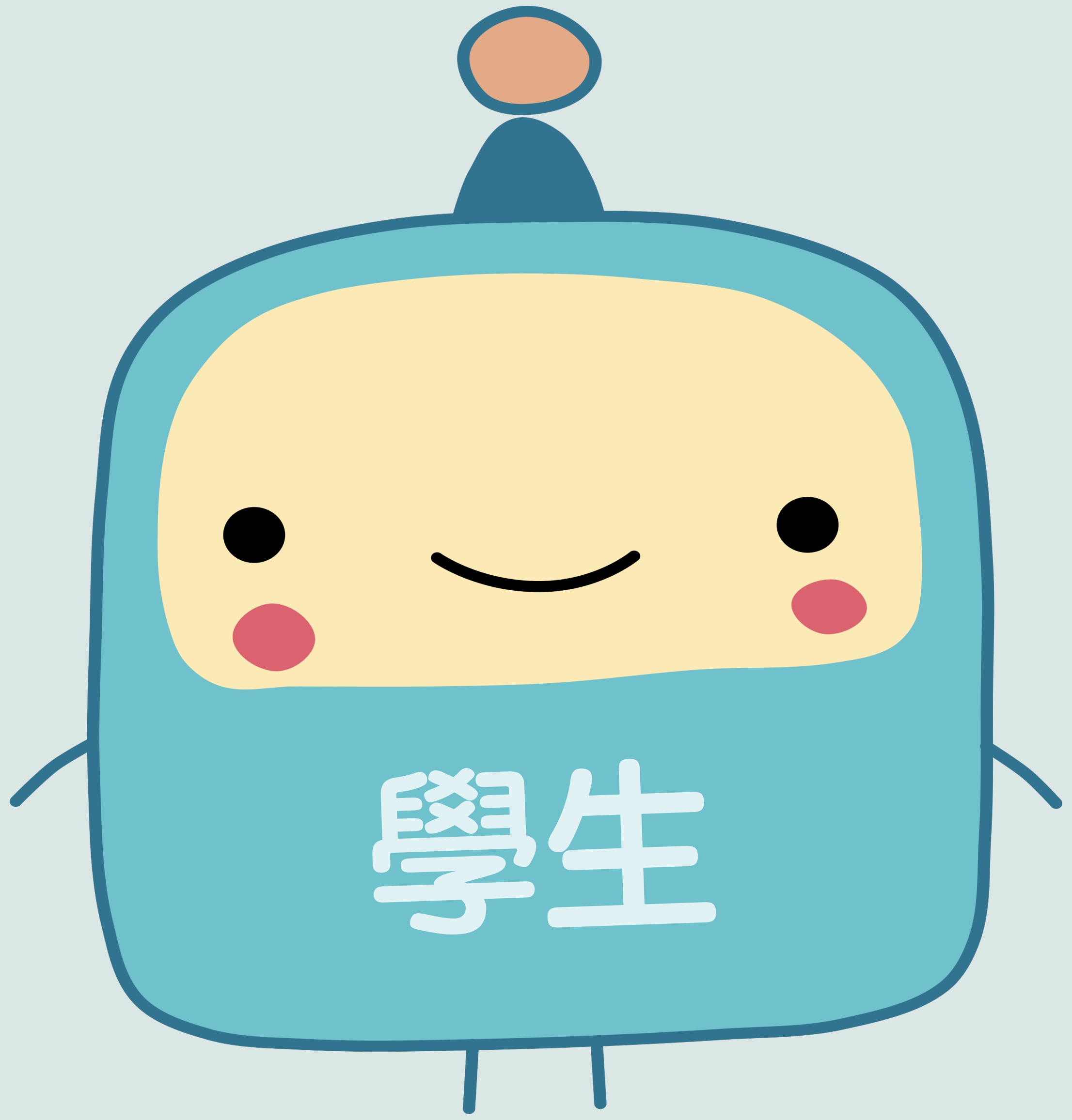


Hard Target

假設大 (老師) 模型說這樣



學生模型試著學老師的「思路」



目標

0.7

0.25

0.05

Soft Target

Q & A