生成式 AI：文字與圖像生成的原理與實務

10.
從 VAE 開始的奇幻旅程

蔡炎龍
政治大學應用數學系

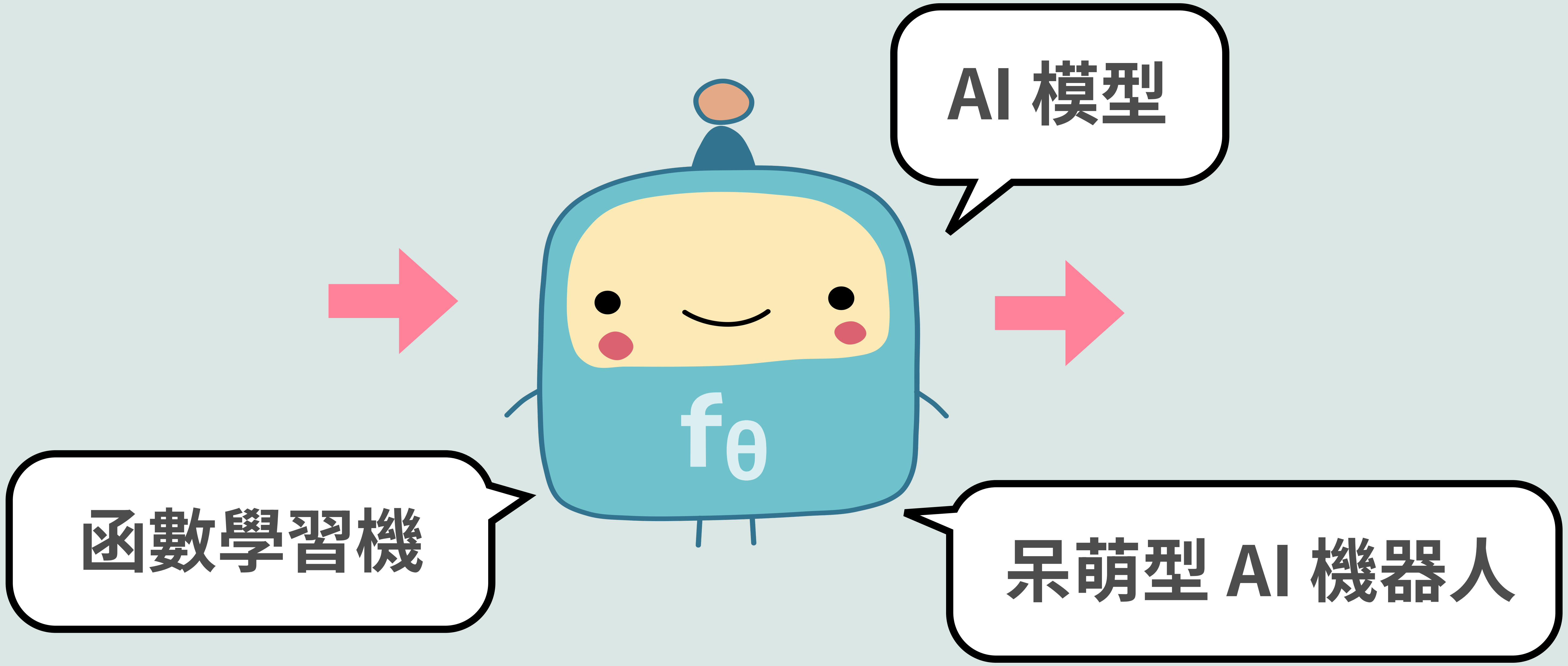
生成式 AI 的可能形式

1. 生成對抗網路 GAN
2. 預測下一個 token 的 LLM
3. (本來) 生成圖像的擴散模型

01.

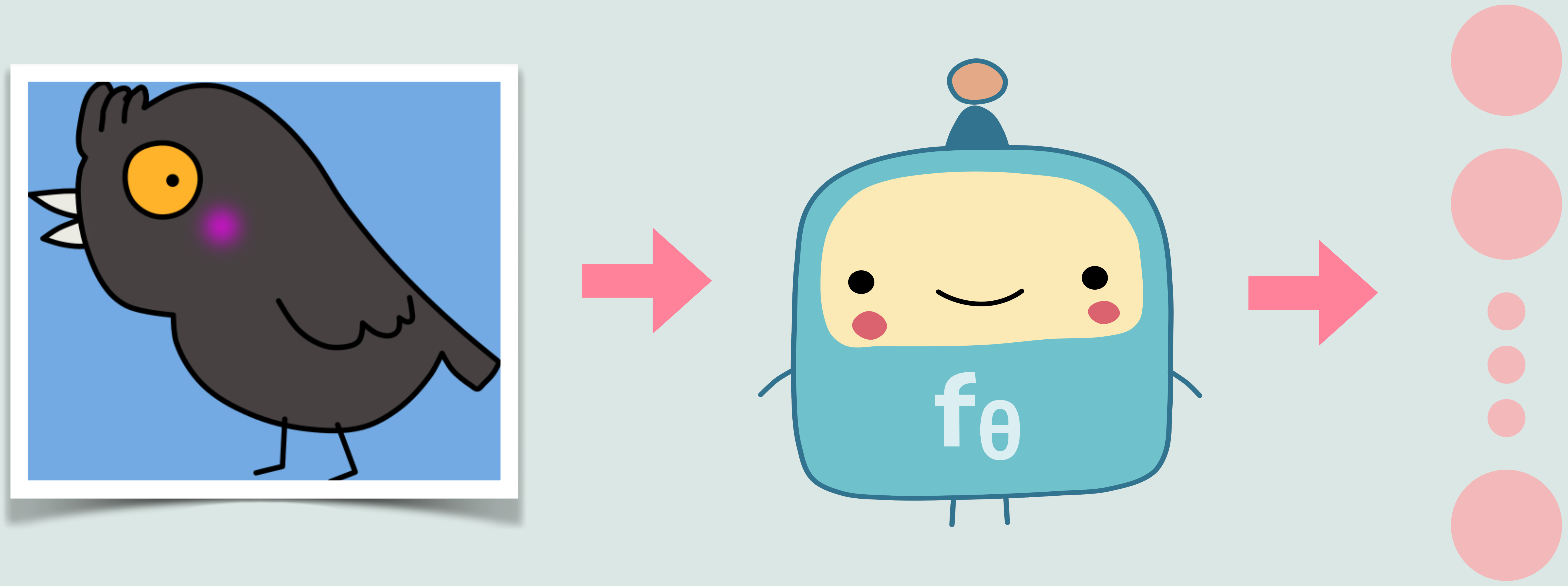
Embeddings

當前所有的 AI 都只是一個呆萌型 AI 機器人



就是知道輸入是什麼、輸出是長什麼樣子

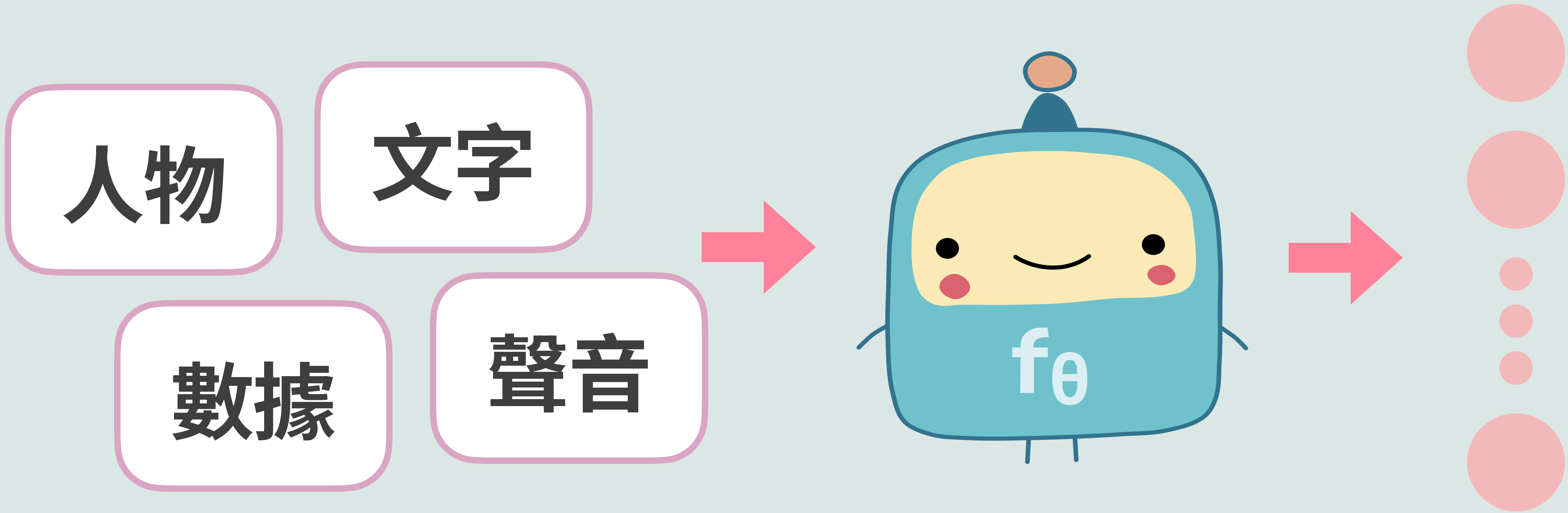
有一種任務我們很喜歡, 但有點困擾...



特徵代表向量,
embedding,
latent vector

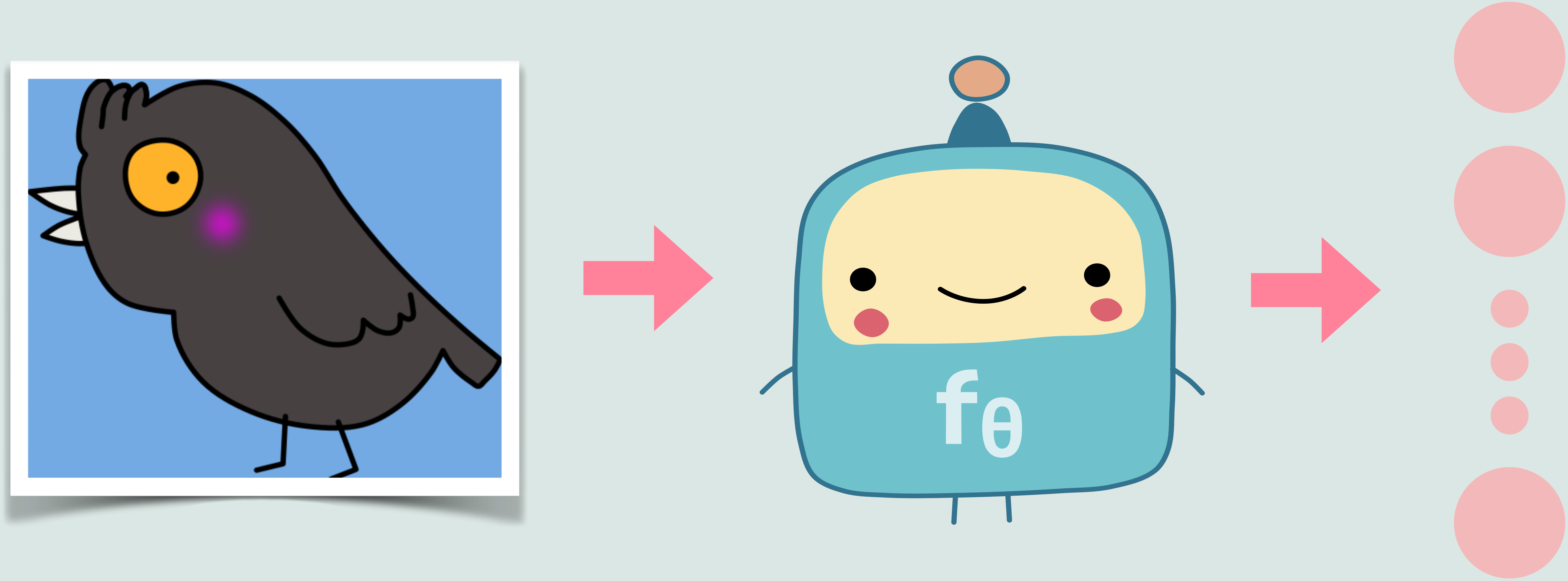
我們想找輸入的特徵代表向量

當然不只是圖像



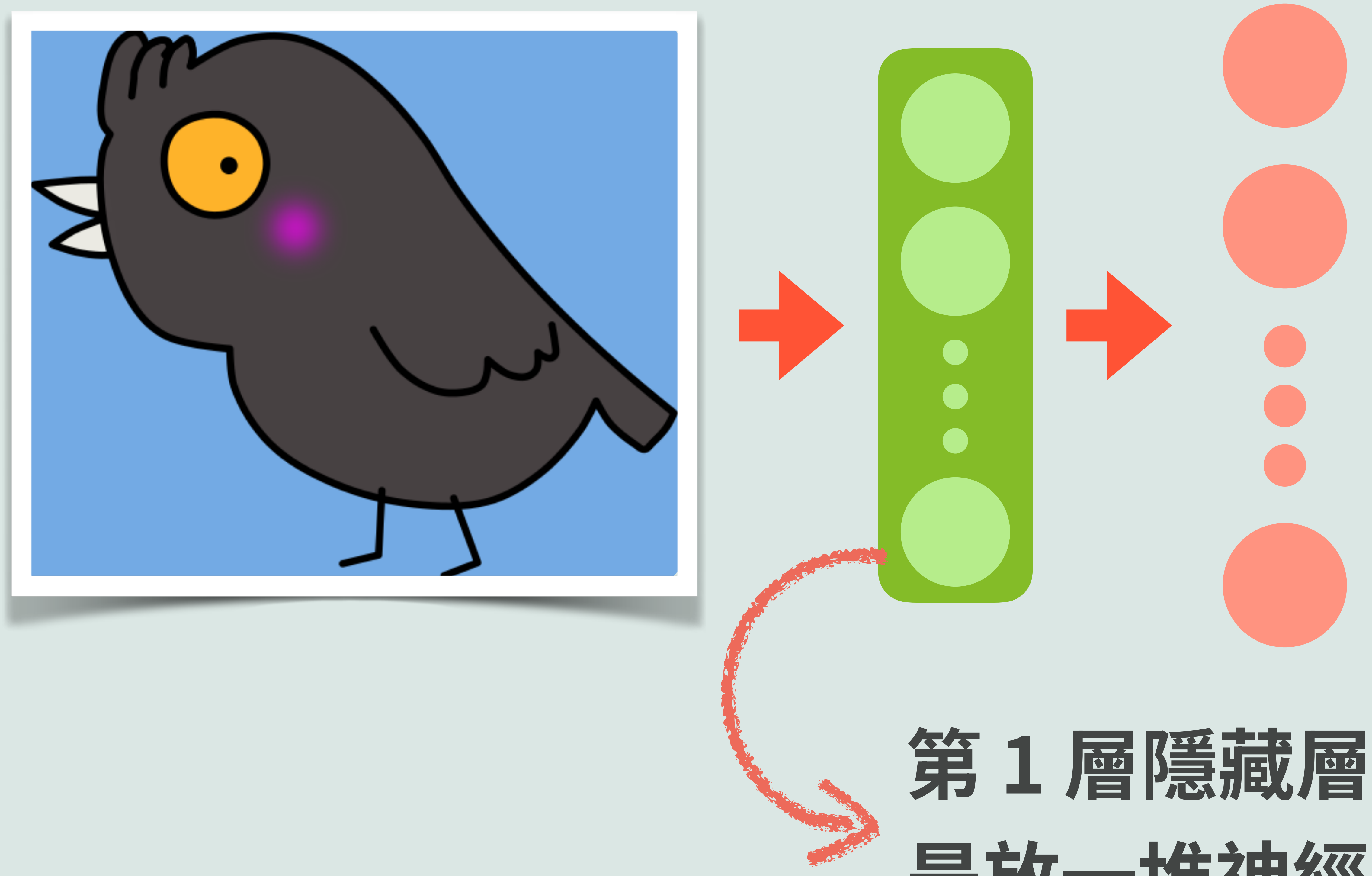
總之所有可能的輸入, 都有可能要找特徵代表向量

問題是訓練資料要怎麼找呢?



不知什麼是「**適當的**」特徵代表向量!

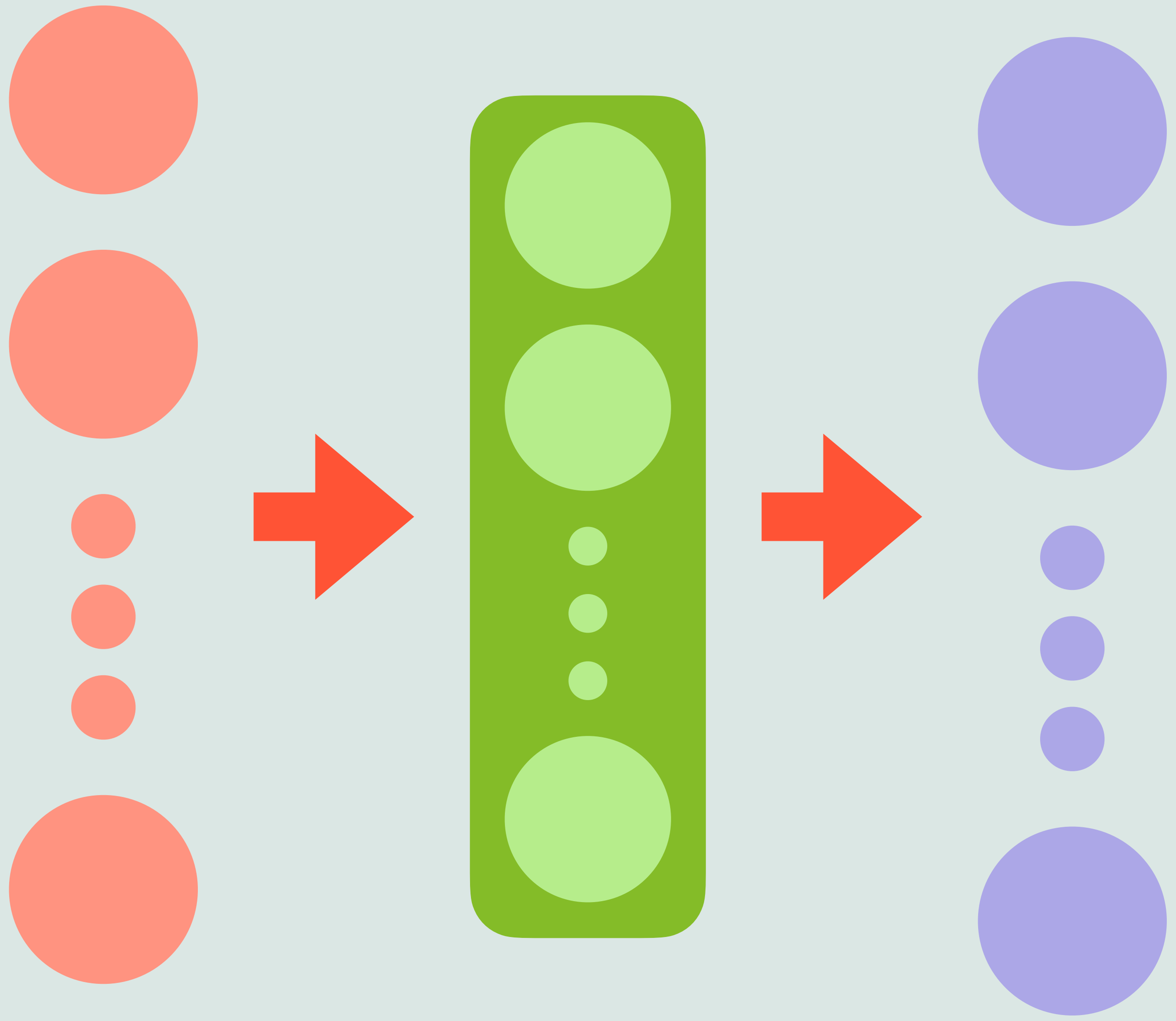
每一層神經網路把輸入轉為另一個 tensor



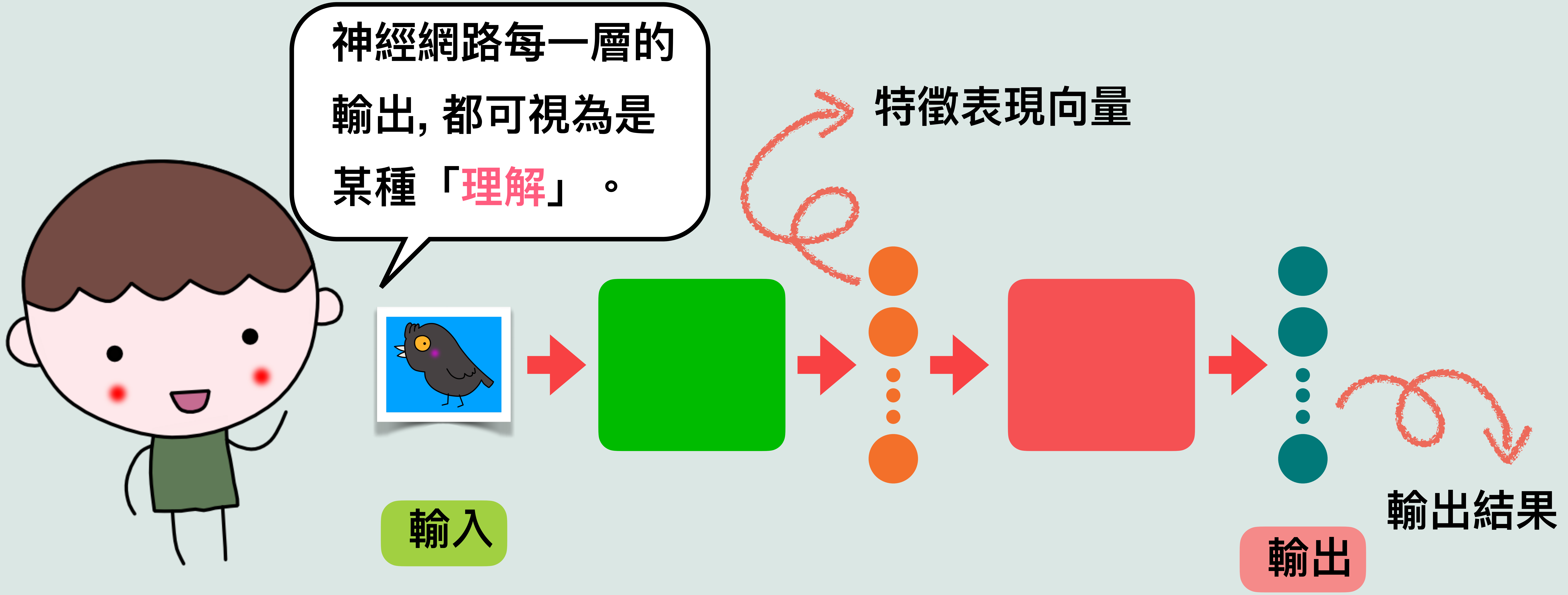
第 1 層隱藏層, 基本上就
是放一堆神經元在裡面

每一層神經網路把輸入轉為另一個 tensor

上一層的輸出就
是下一層的輸入,
如此不斷下去...

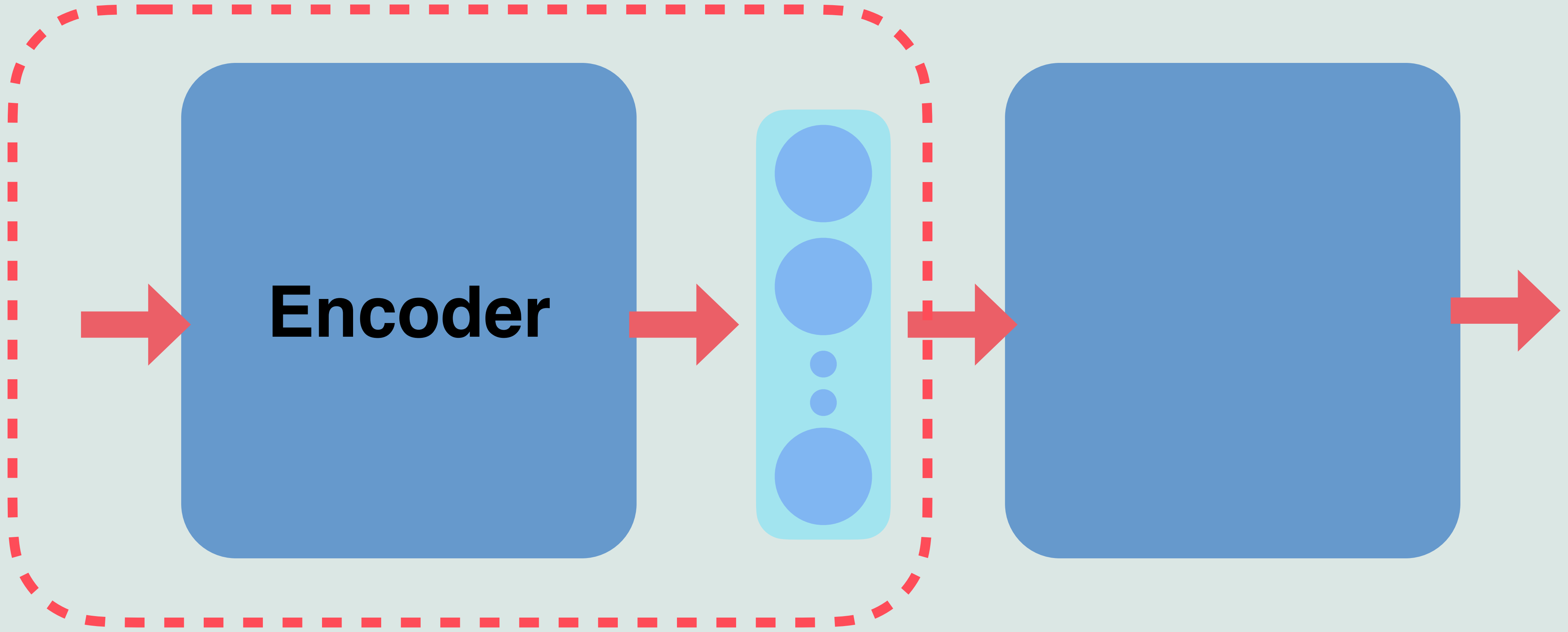


電腦的「理解」



設計代理任務 (Pretext Task)

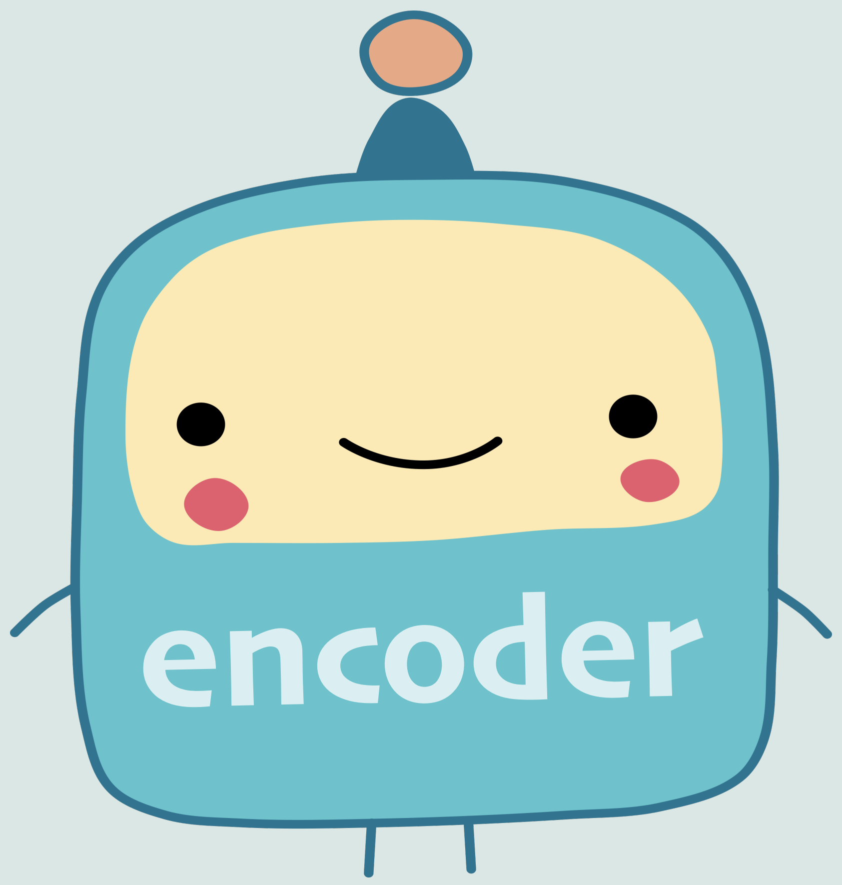
我們可以讓電腦去做一些小任務, 這個任務是我們覺得「電腦要懂文字的意」才能完成的任務。這種不是我們真正最後的目標, 通常是為了訓練好的表示向量的叫 pretext task。

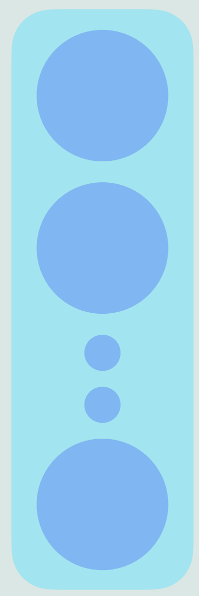


我們看要 embed 到幾維向量, 比如說 V=128 維, 那就在神經網路中間的隱藏層, 放 128 個神經元!

訓練好代理任務, 前面那段模型叫 Encoder

輸入 (圖像、文字
或其他任何數據)



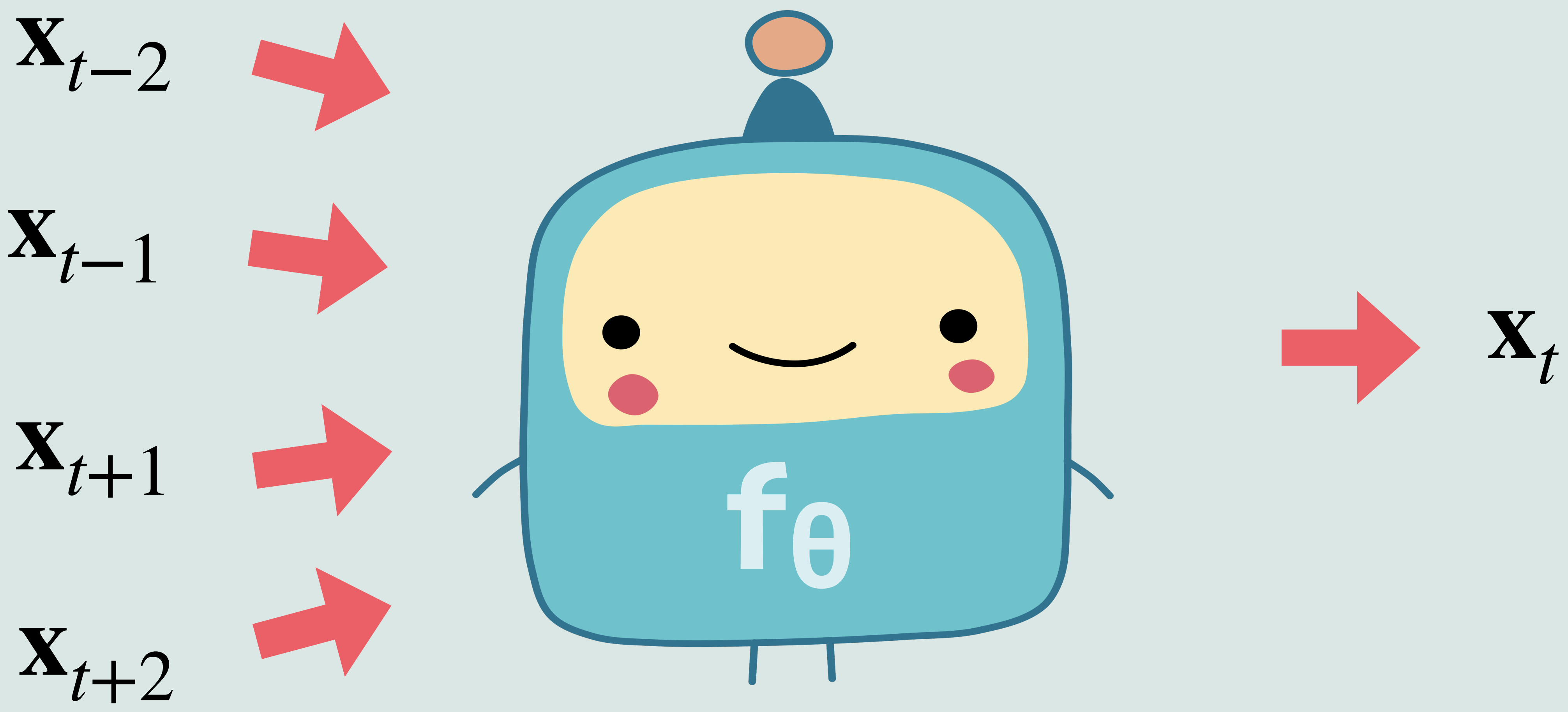


輸入的特徵
代表向量

latent vector,
embedding

比如說 Word2Vec 的兩個小任務

Word2Vec 就設計兩種任務。

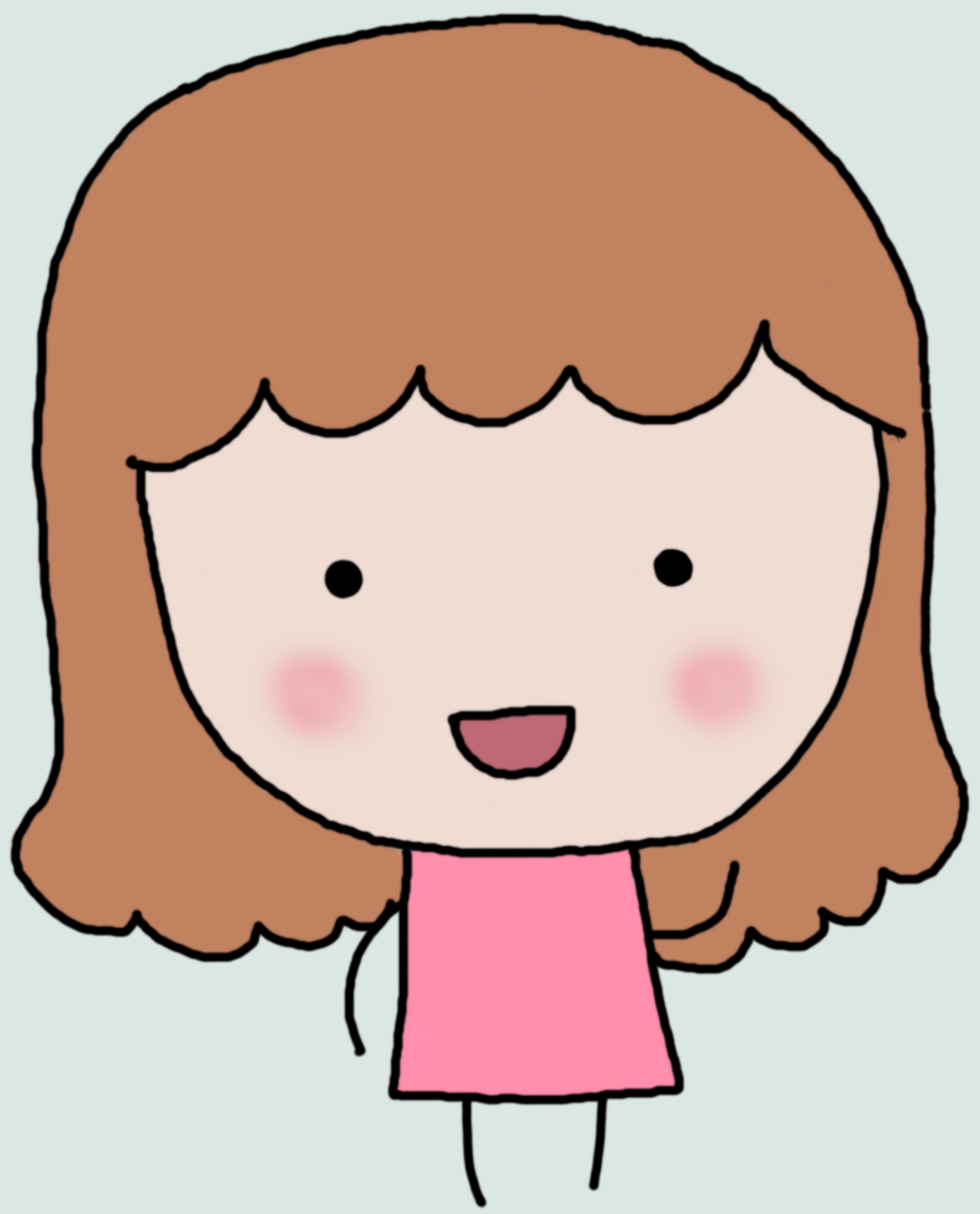


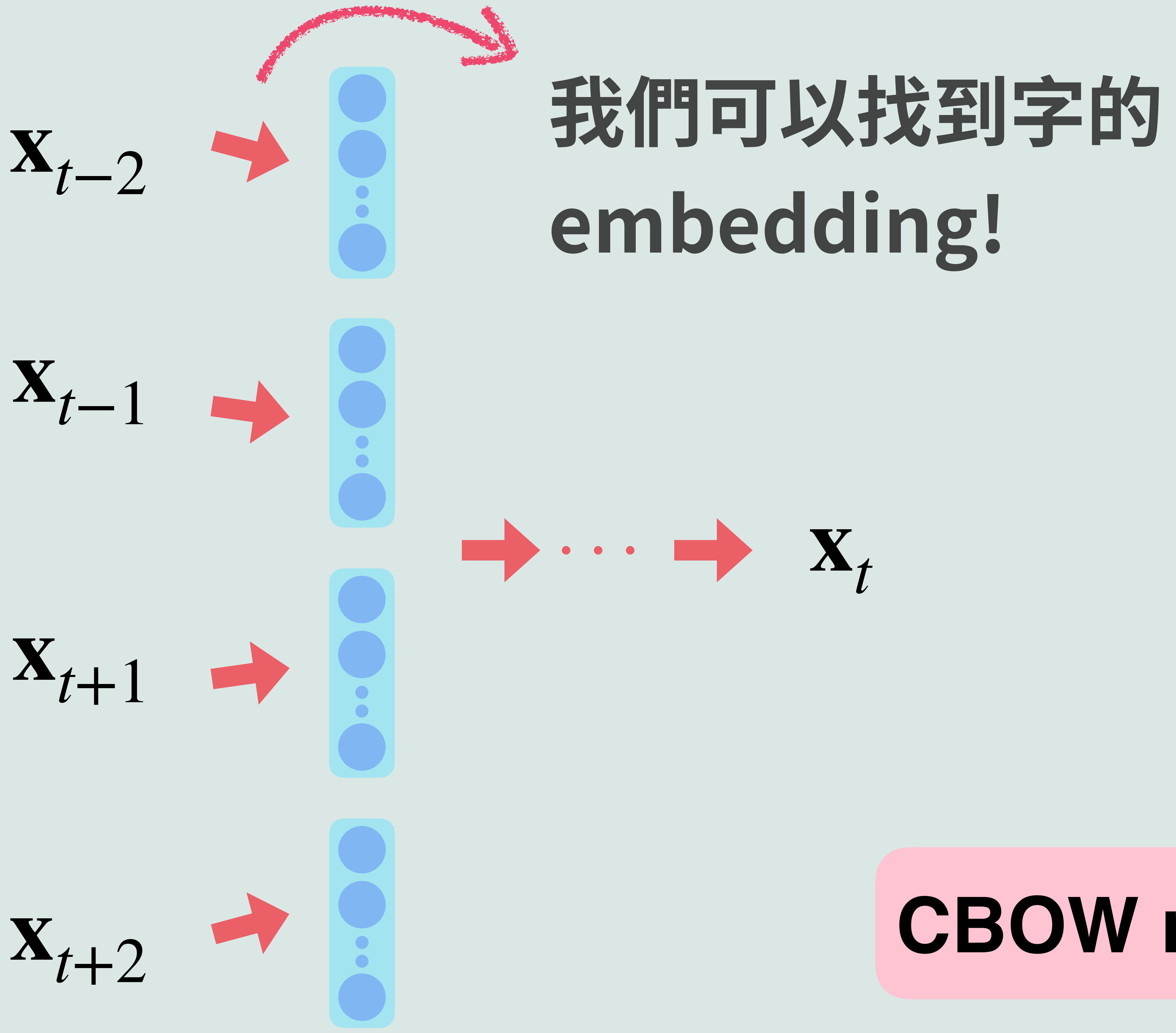
CBOW model

用周圍的字預測中間的字。

$\mathbf{x}_{t-2}$	$\mathbf{x}_{t-1}$	?	$\mathbf{x}_{t+1}$	$\mathbf{x}_{t+2}$

某個隱藏層輸出就是 embedding

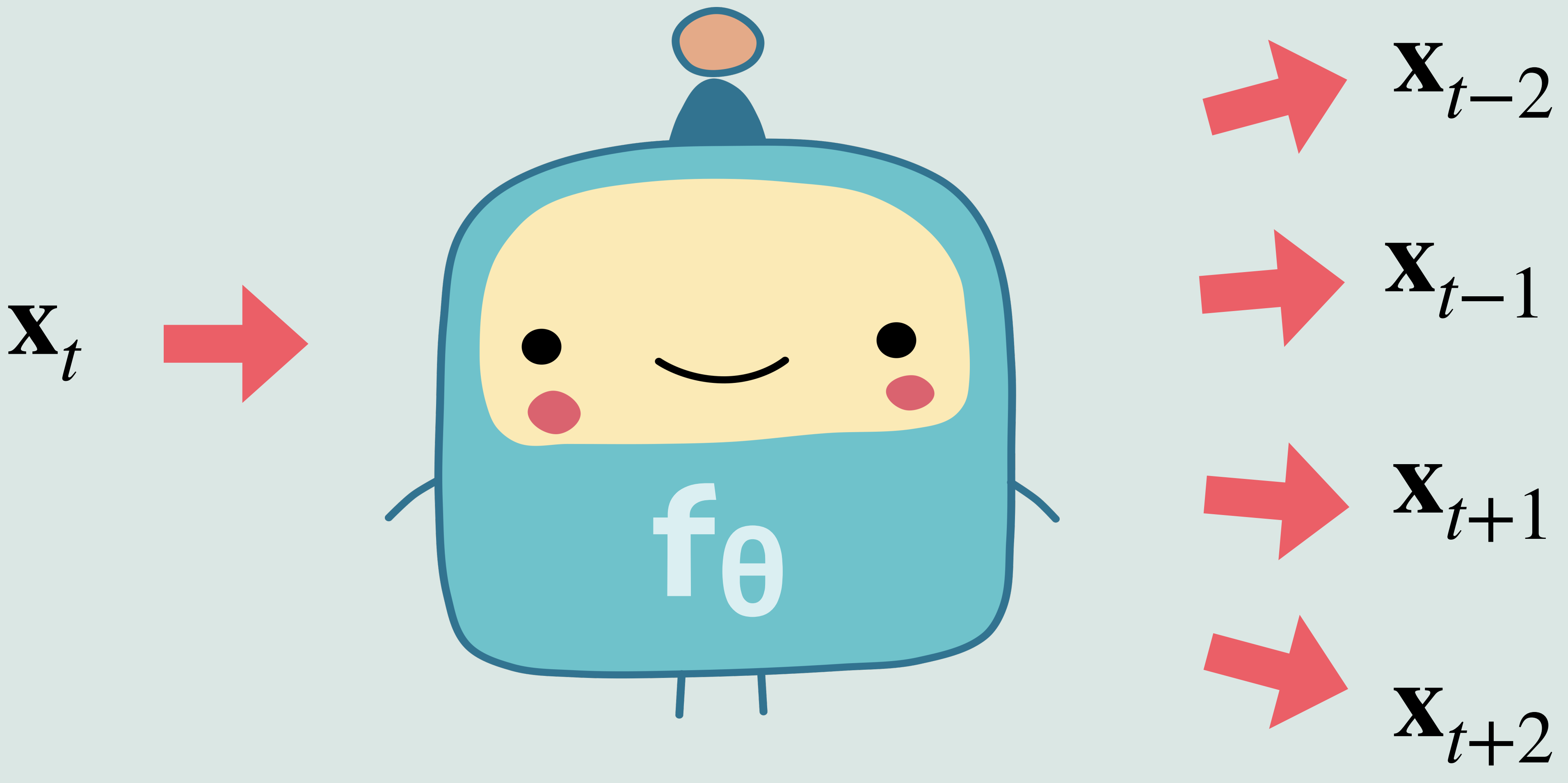




CBOW model

Word2Vec 的兩個小任務

或是更炫的去訓練這樣的函數!

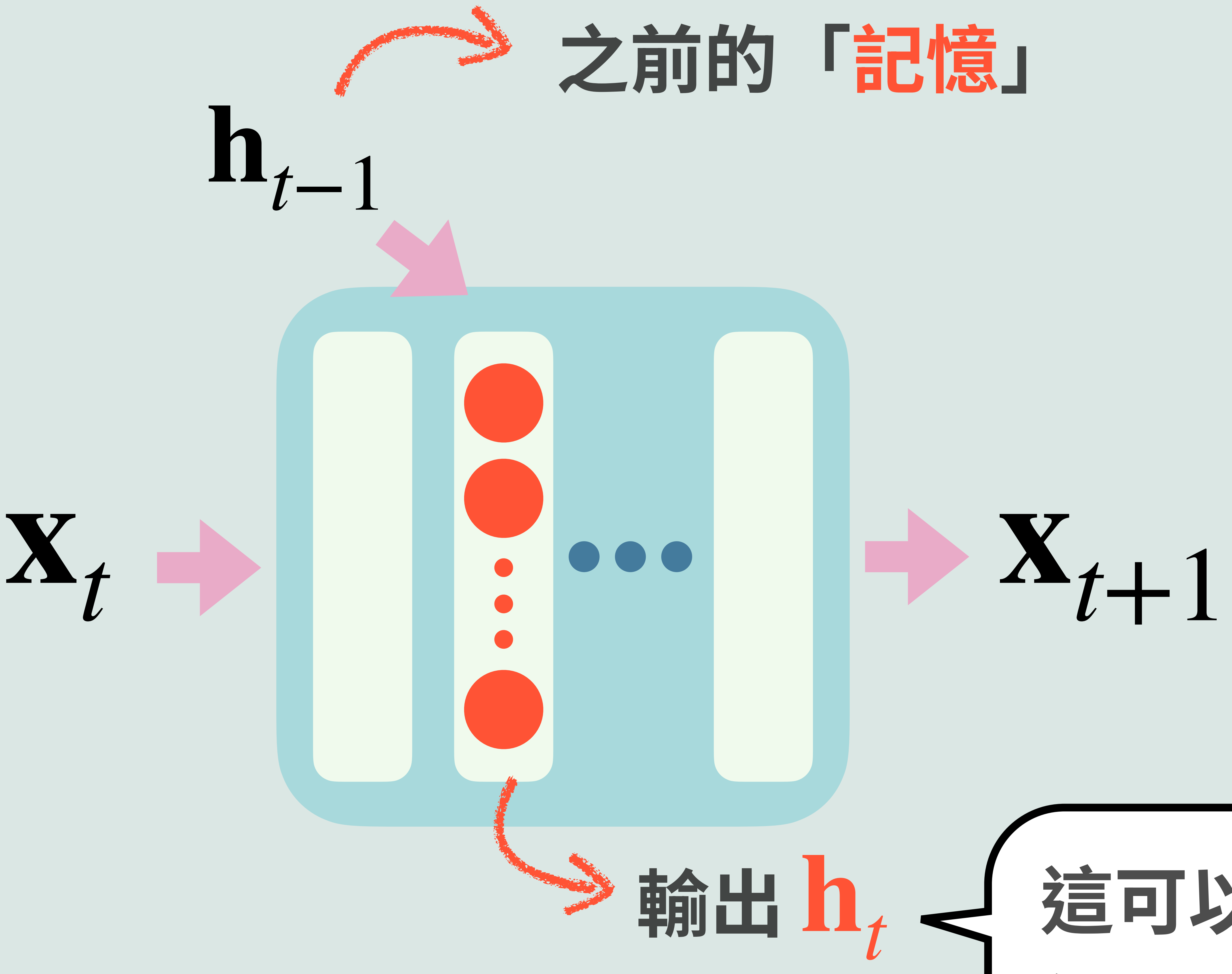


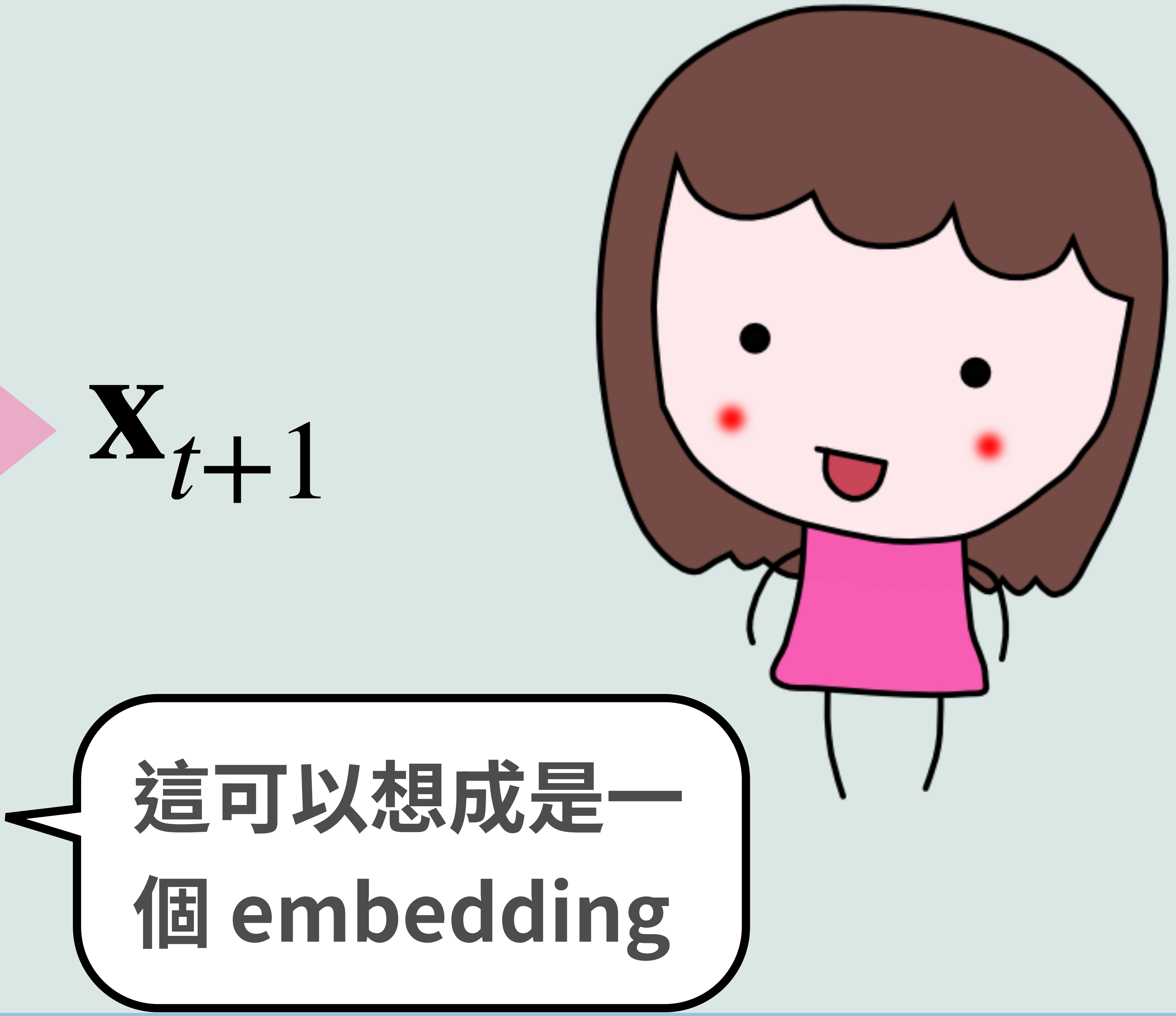
Skip-Gram model

中間的字預測週圍的字

?	?	$\mathbf{x}_t$	?	?

事實上前一個字預測下一個字也是!

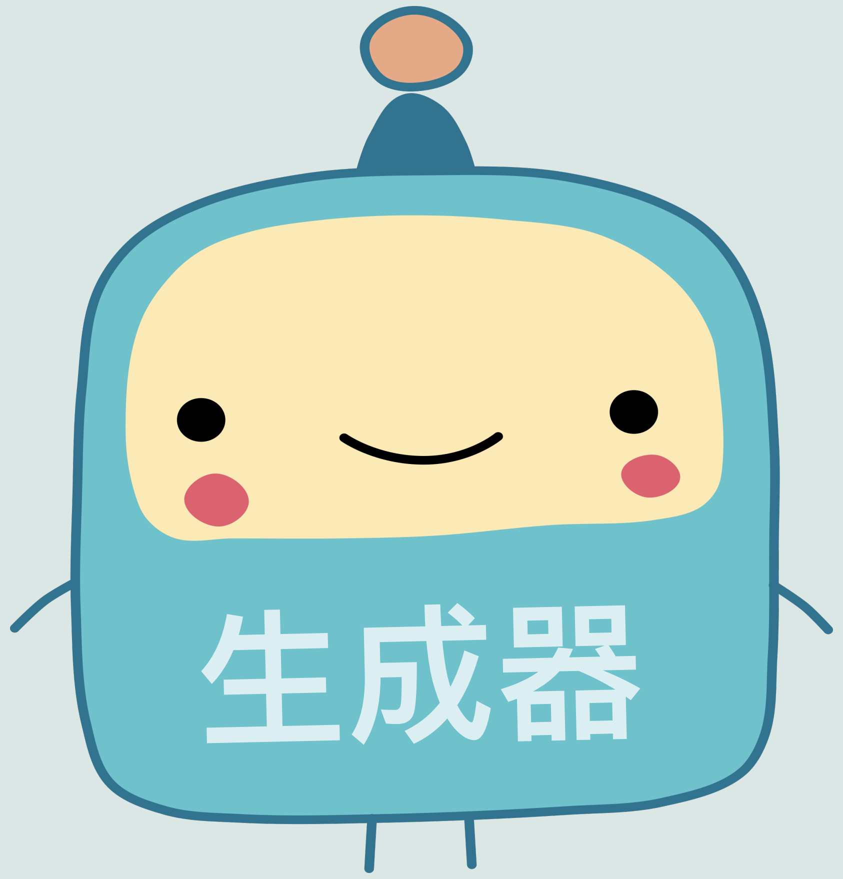


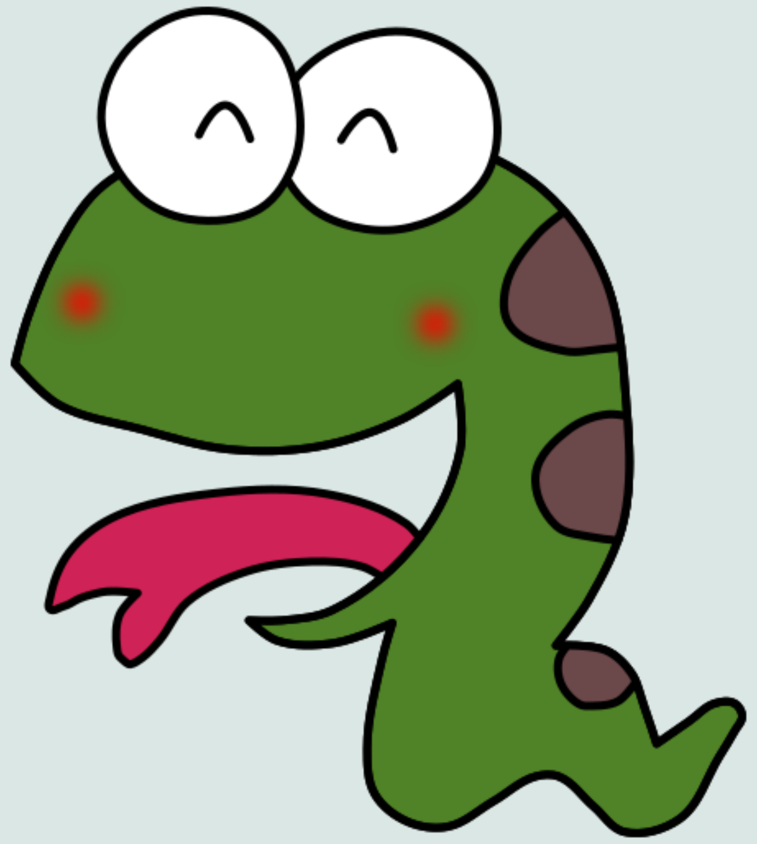


生成模式

z

latent vector
特徵代表向量

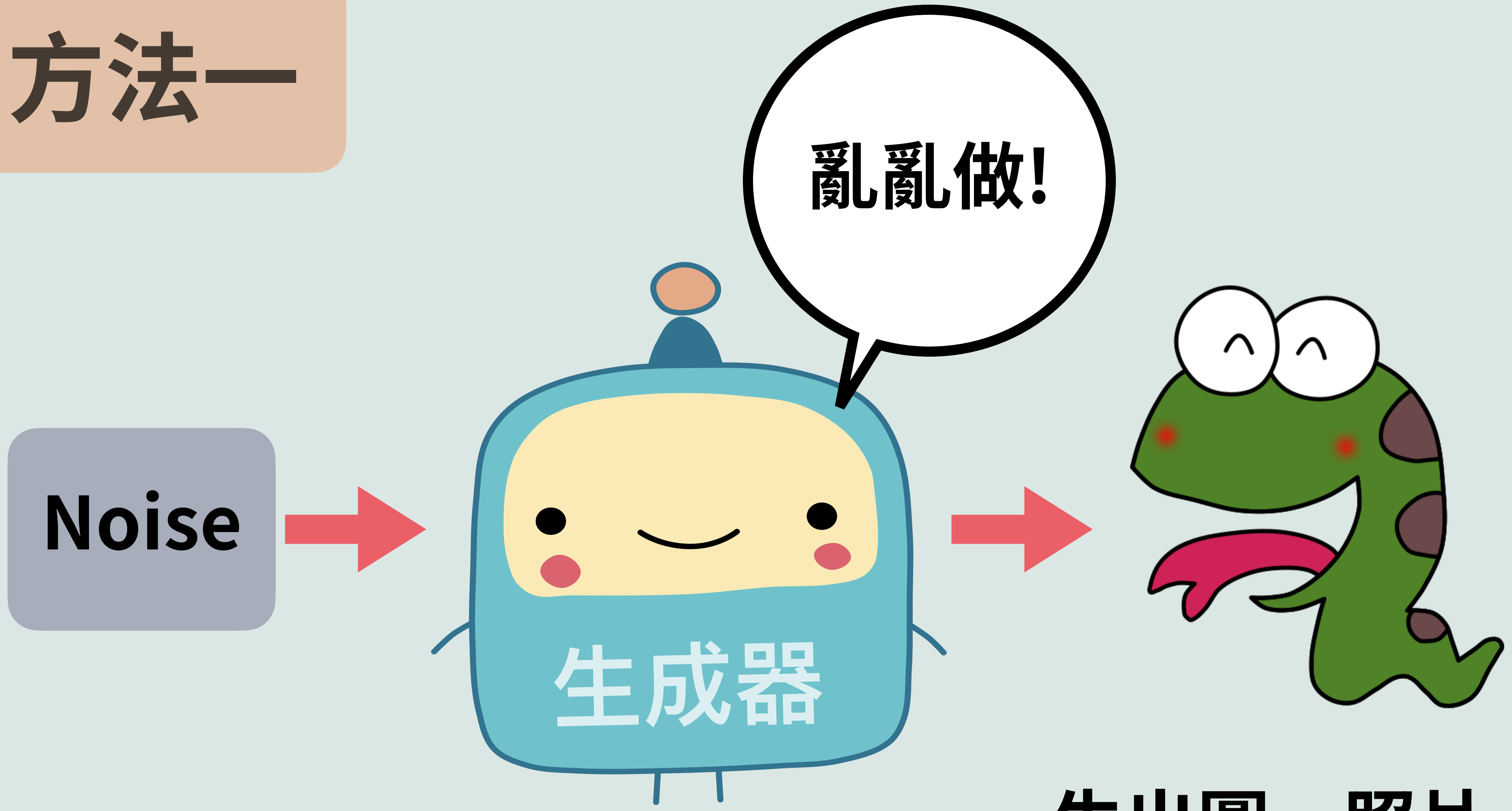




生出圖、照片、文字

輸入的特徵向量基本上有兩種作法

方法一



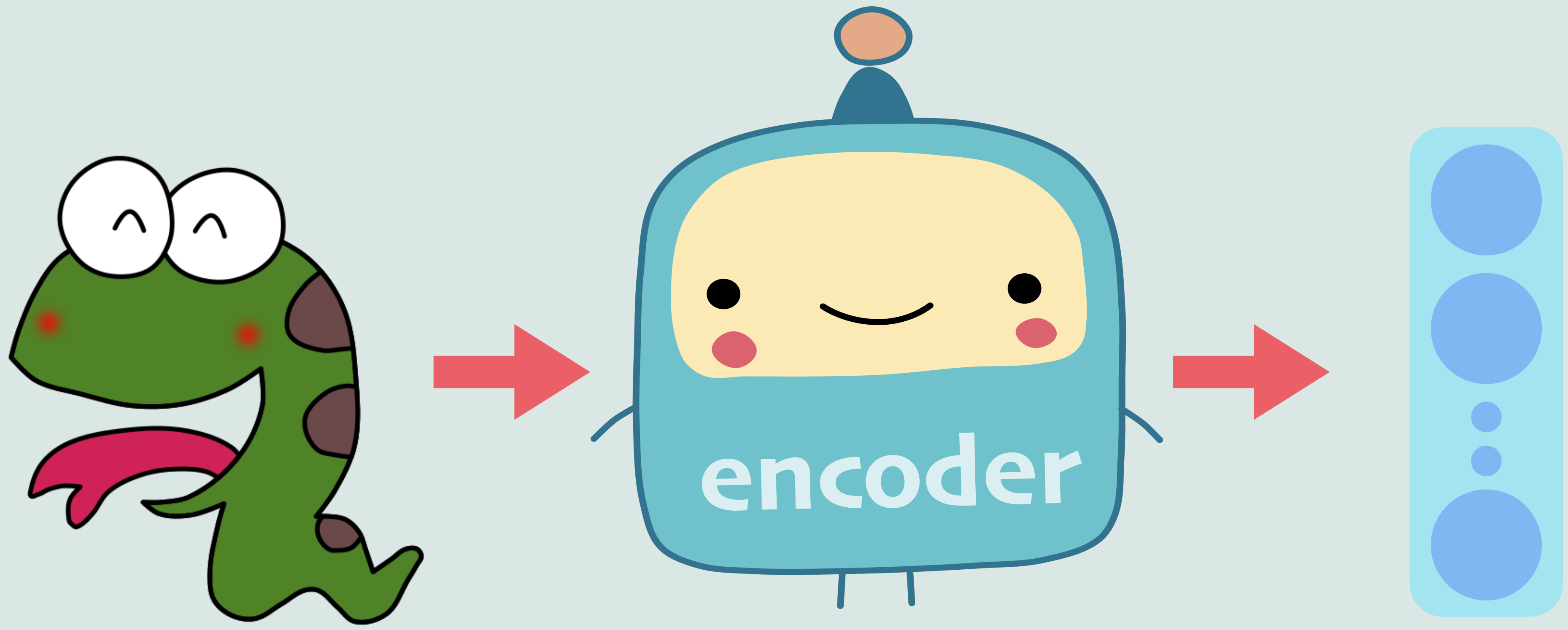
生出圖、照片、文字

(保證生出「正確格式」的東西就好)

在 GAN 裡, 這是最常用的方法!

輸入的特徵向量基本上有兩種作法

方法二



特徵代表向量,

latent vector

02.

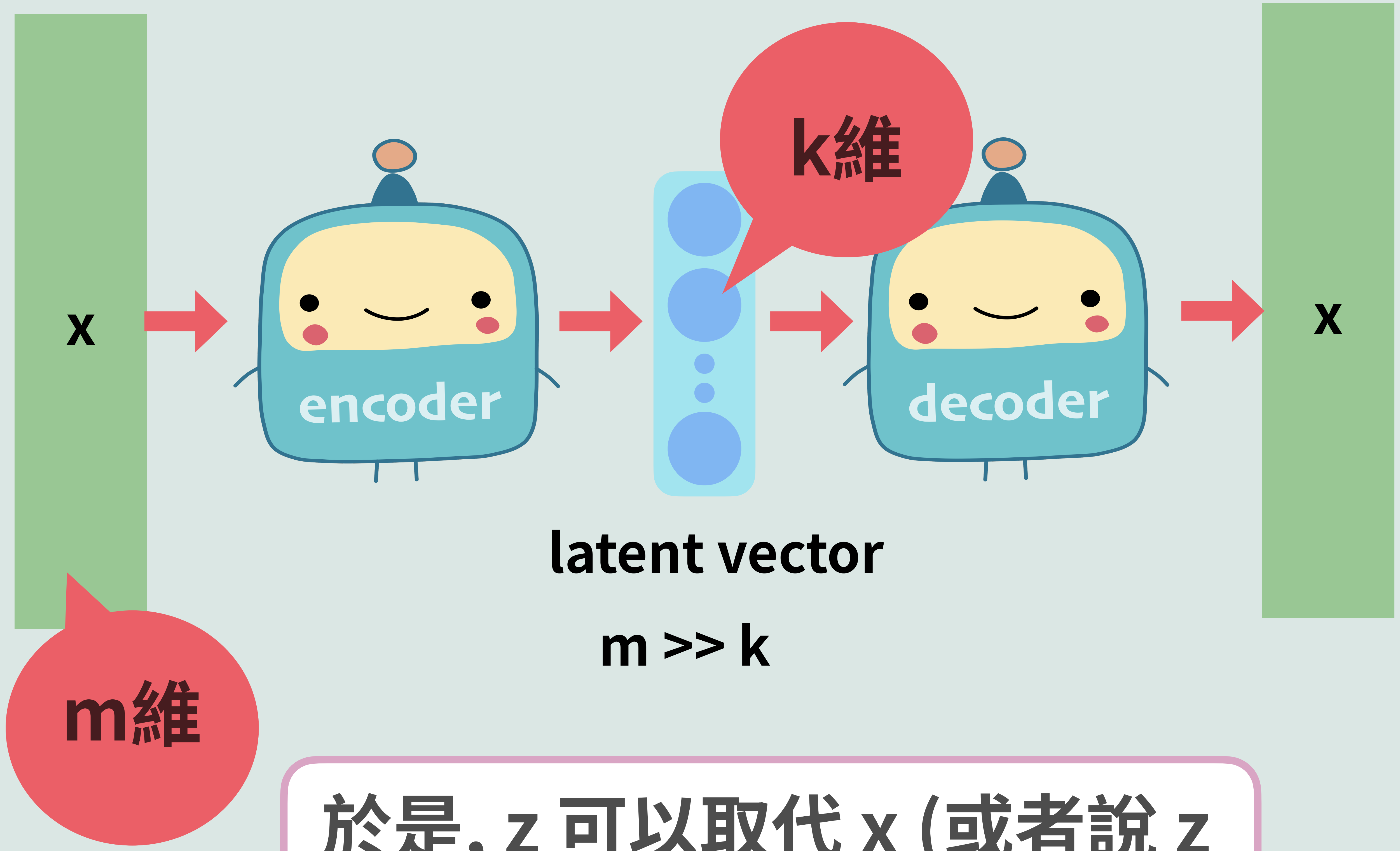
自編碼器 Autoencoder

Autoencoder

方法二看來有點神奇, 怎麼能訓練一個截取特徵向量的函數的? 這裡介紹一個常用手法叫**自編碼器 (Autoencoder)**。

Autoencoder

Autoencoder



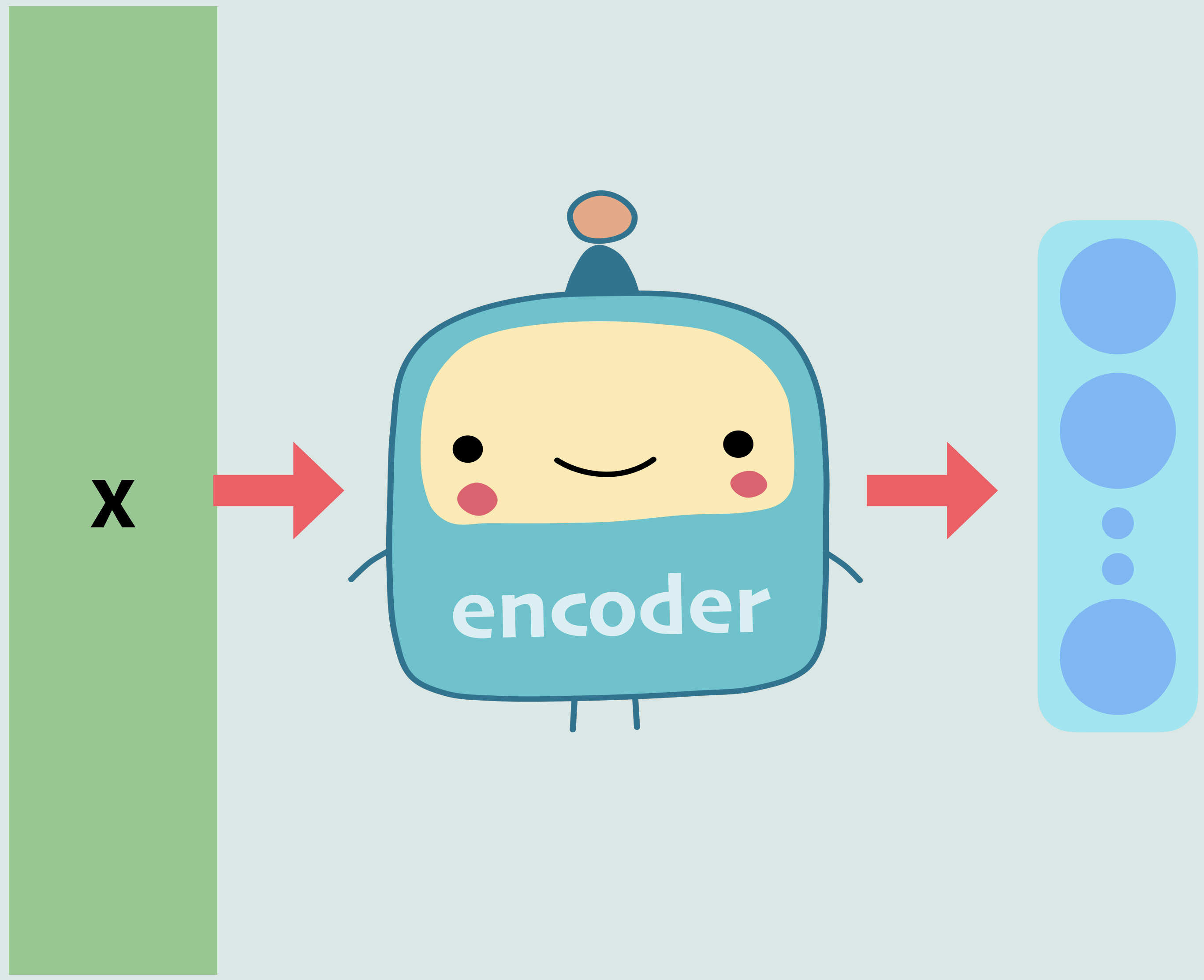
於是, z 可以取代 x (或者說 z 是 x 的一個 presentation)

Autoencoder 是輸入什麼, 就輸出什麼的函數。

感覺很奇怪, 原來是中間我們會用一層比較小的 k 維神經元。

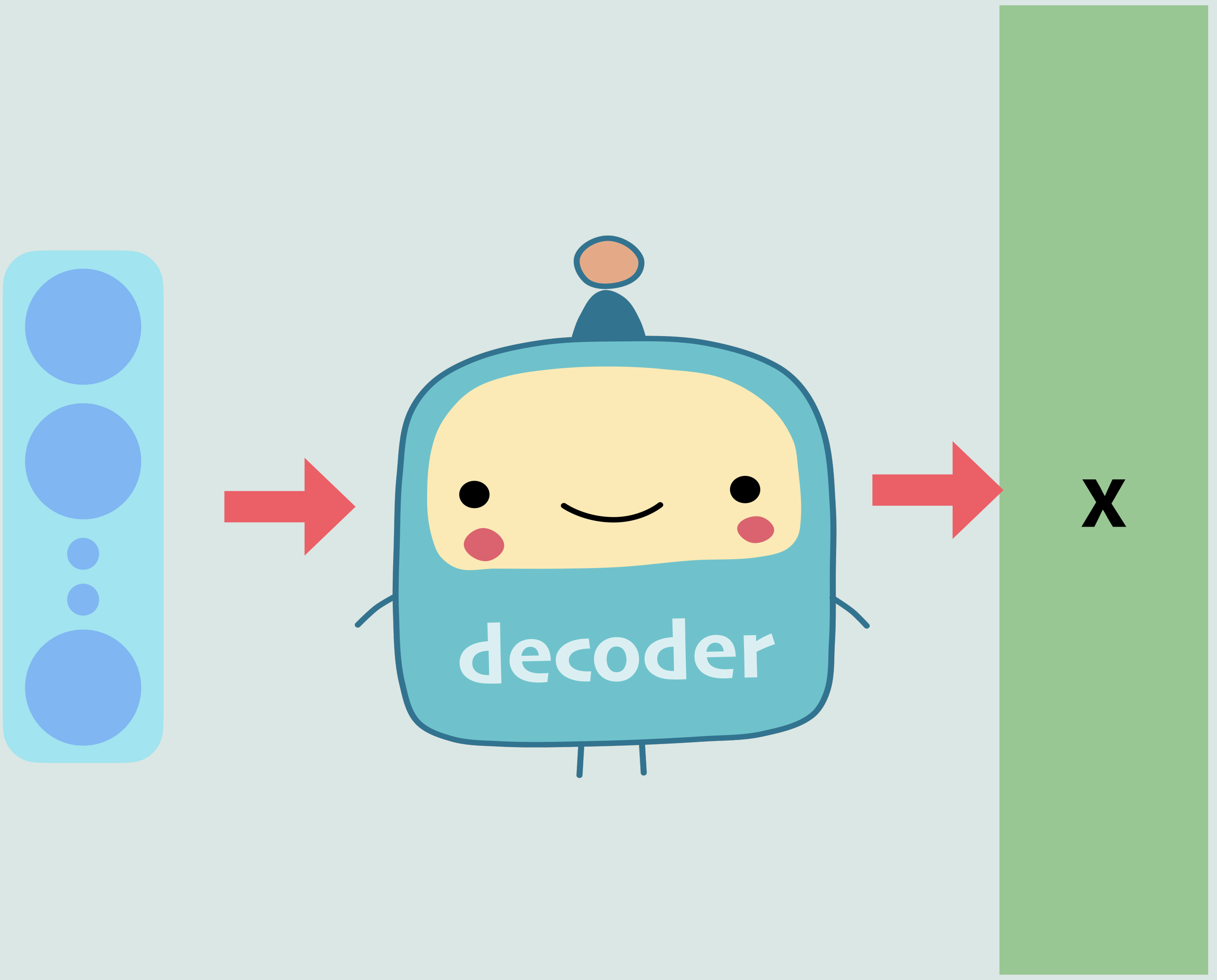
這一層的輸出就是我們準備當特徵向量的。

Autoencoder



以後我們基本上可以
用比較小的 z 來取代
比較大的 x。

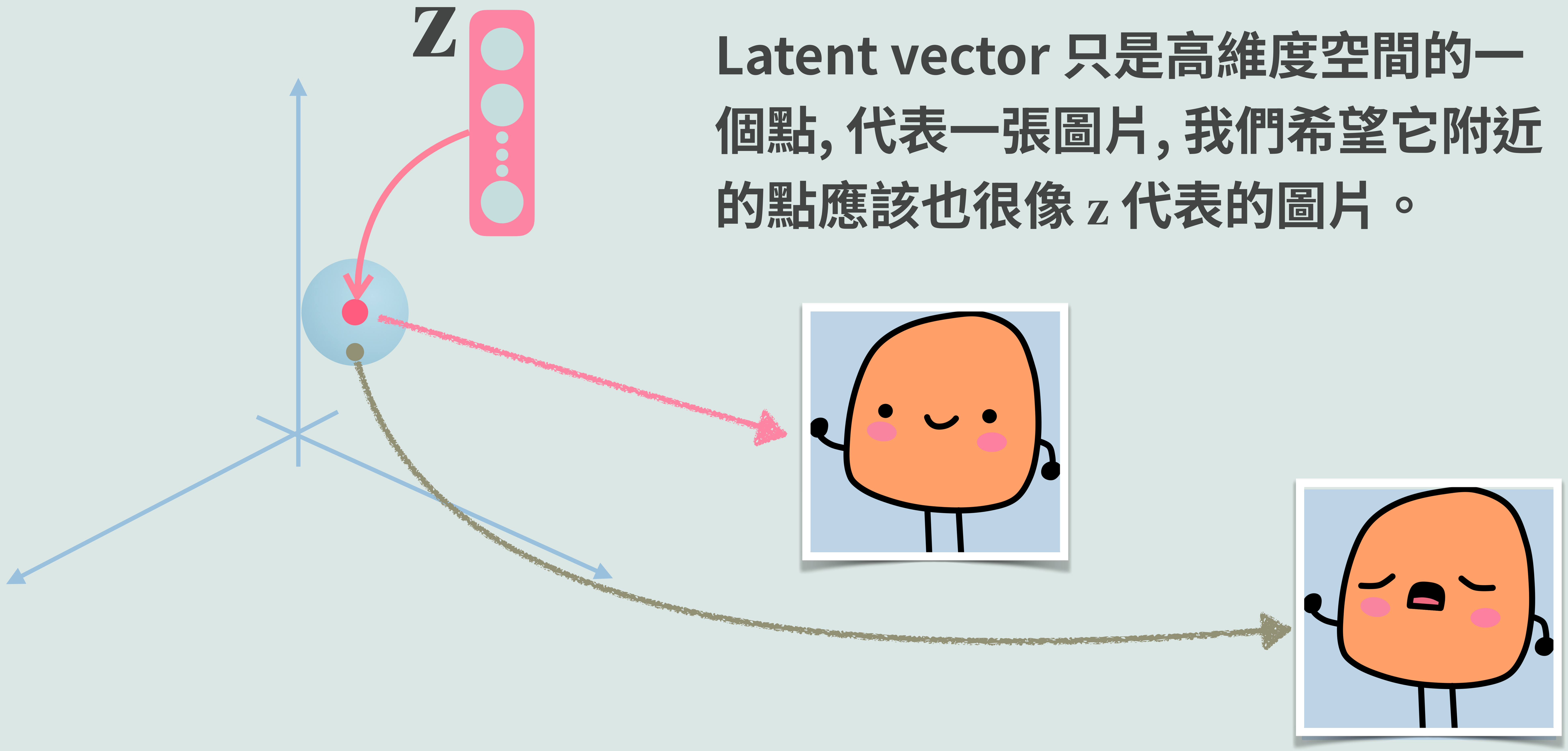
Autoencoder



假設我們有隻兔子的特徵向量, 改一點點會不會生出一隻很像的兔子?

另一邊就可以當生成器, 輸入一個特徵向量, 就生出一個我們要的圖或任何東西。

也就是說我們希望...



Latent vector 只是高維度空間的一個點, 代表一張圖片, 我們希望它附近的點應該也很像 z 代表的圖片。

Autoencoder

答案是不太行

隨便生兩個 latent vectors, 數學上距離很近, 但生出來的東西不一定有什麼關係。

白話文是 z 差不多就是亂數, 我們無以掌控。

03.

變分自編碼器 VAE

改善自編碼器問題的 VAE

Variational **A**uto**E**ncoder

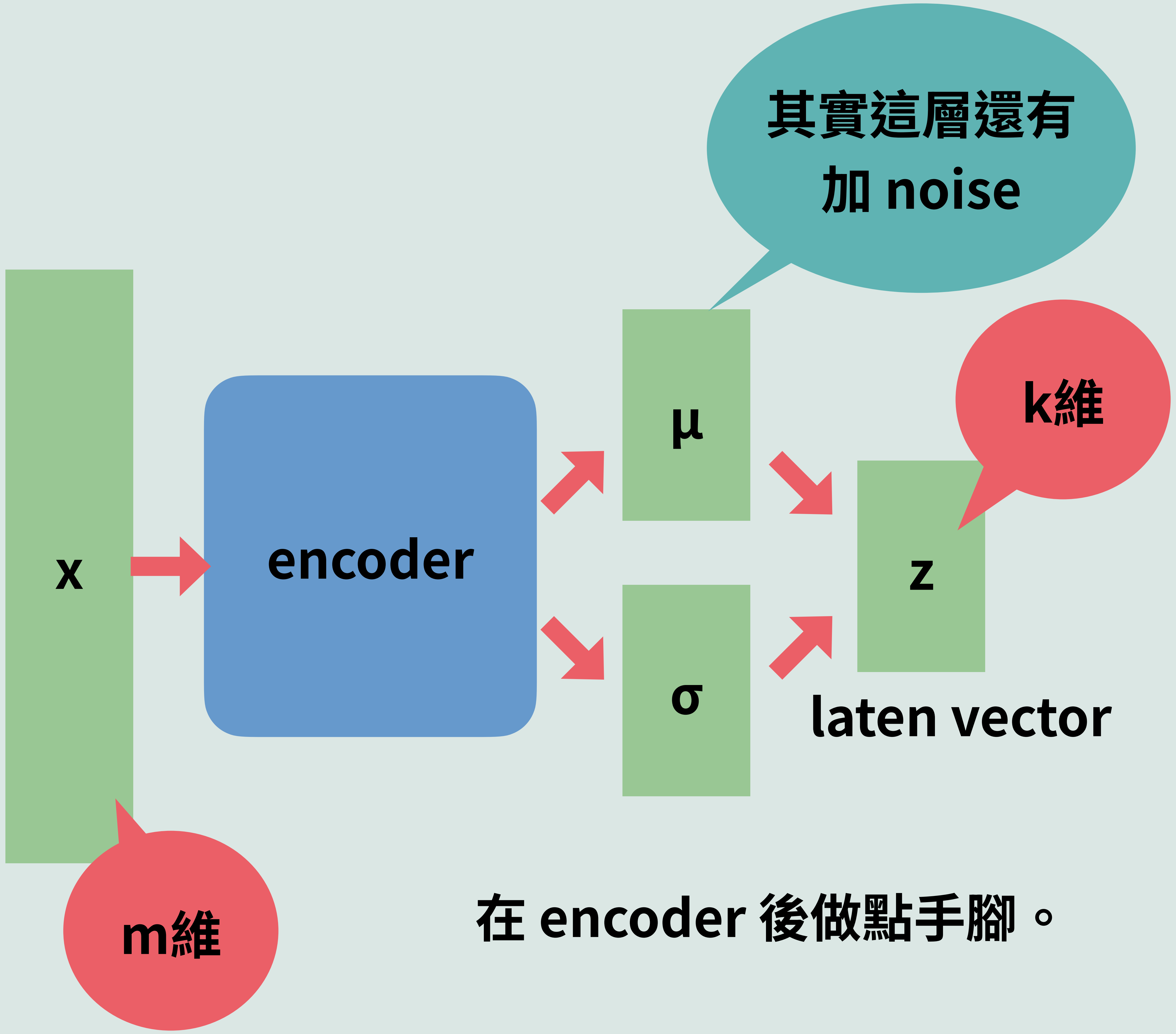
希望 latent vector 每個元素符合常態分佈

$$z = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_k \end{bmatrix} \qquad z_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$$

我們想辦法找每個數的平均值和變異數 (or 標準差)

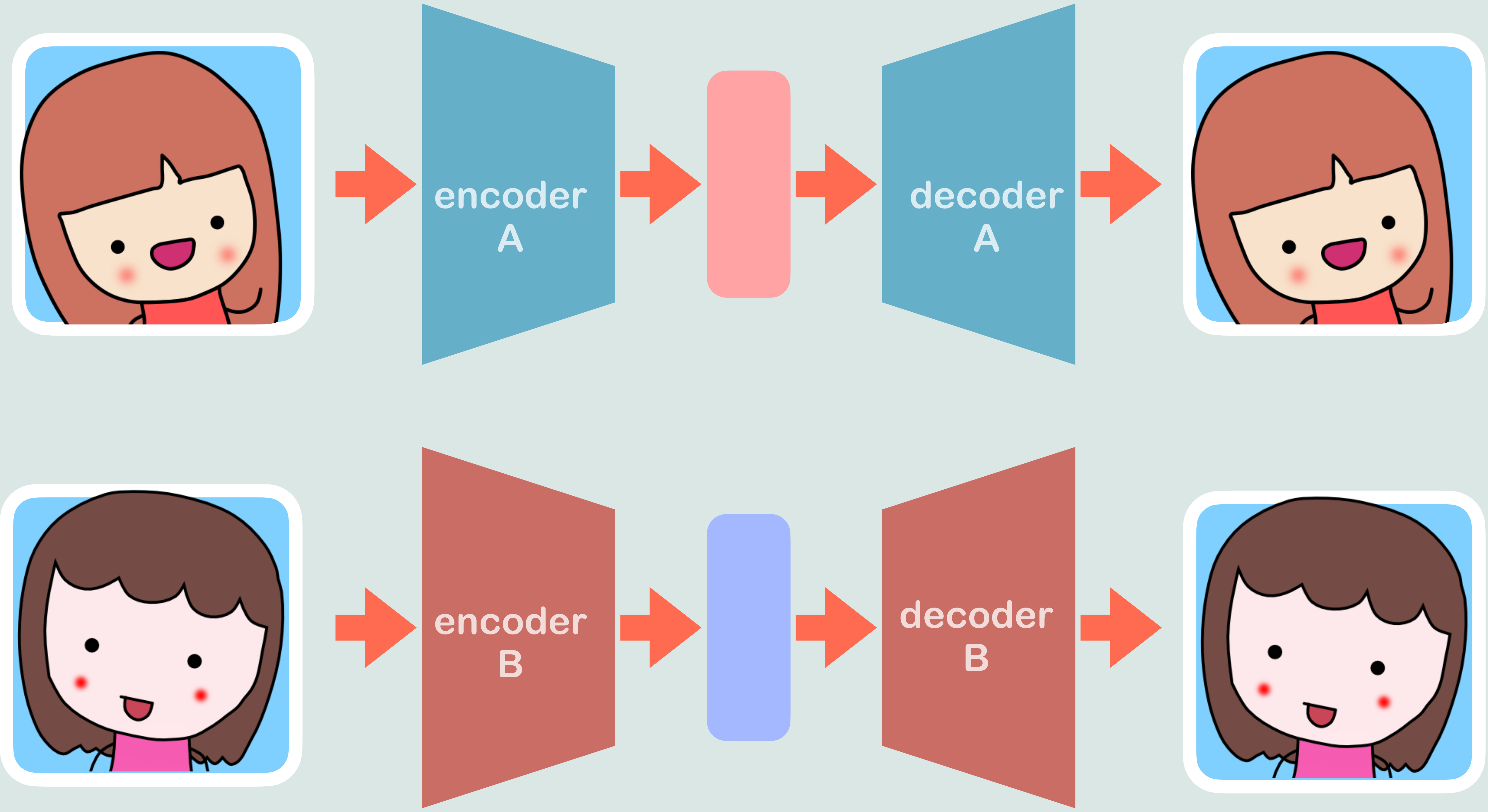
神秘編碼 laten vector 每個數字是符合某常態分布的, 這樣我們容易掌控!

VAE (Variational AutoEncoder)



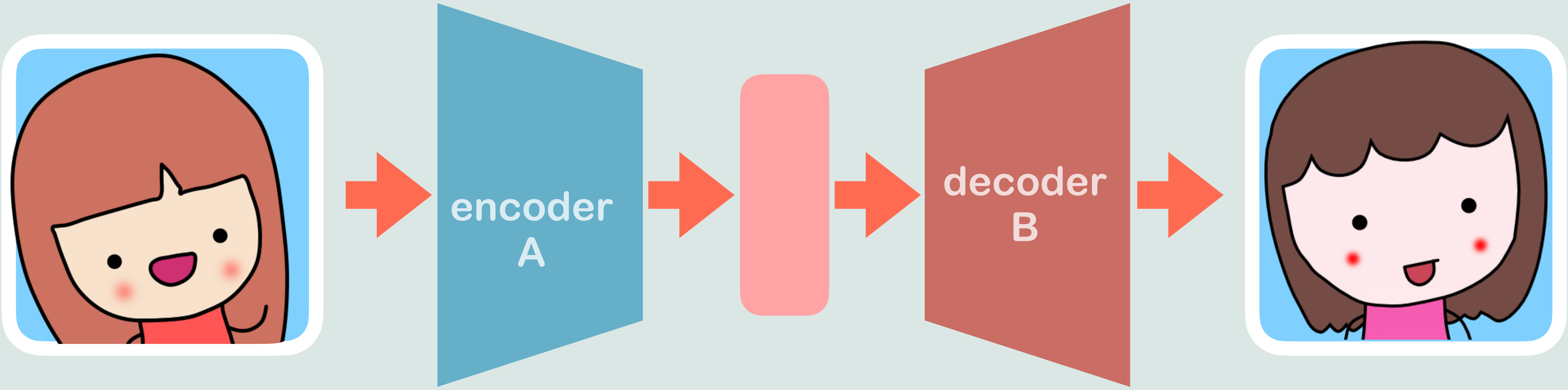
在 encoder 後做點手腳。

Deepfake



用 autoencoder 就可以做到 deepfake

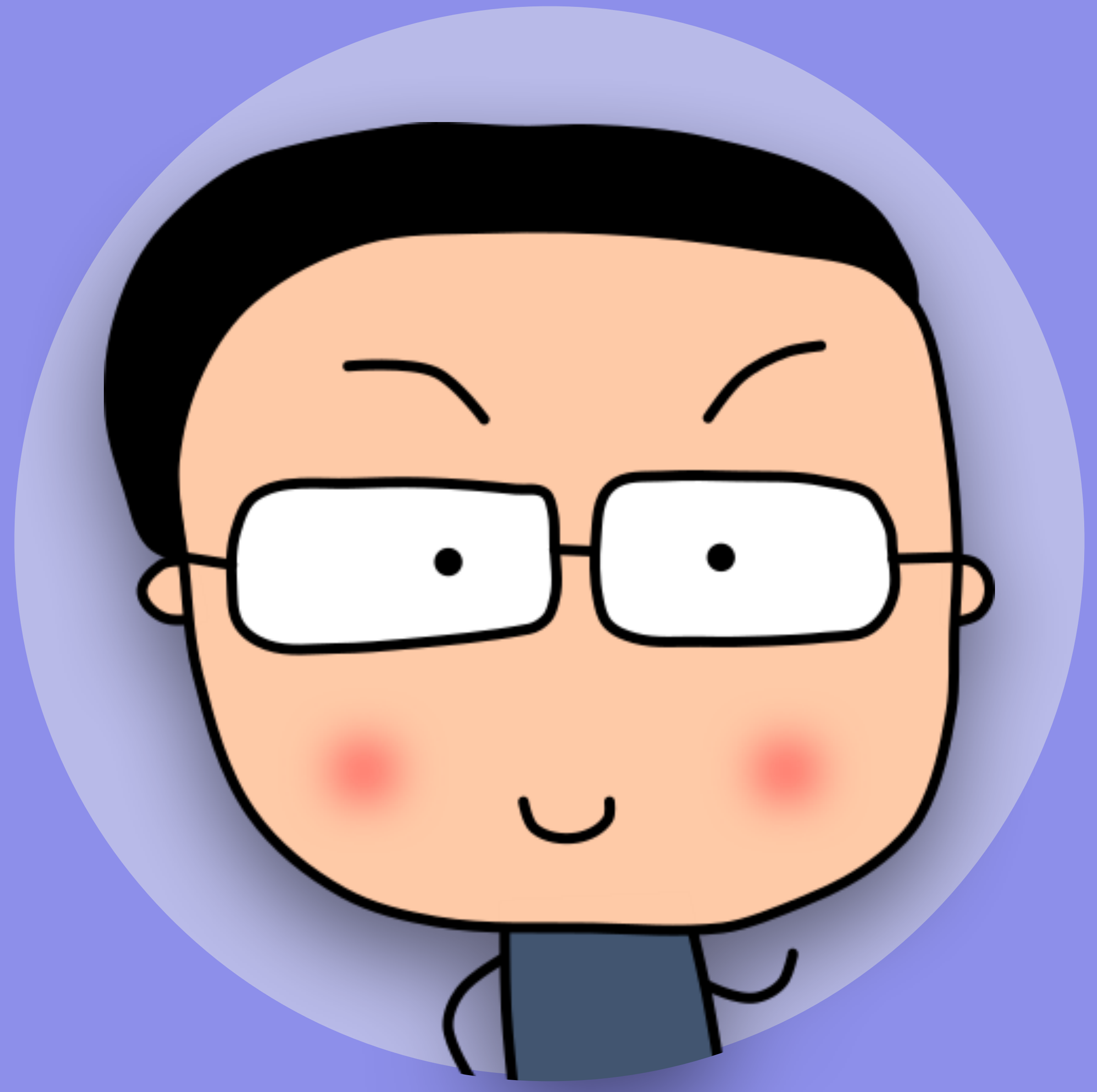
Deepfake



用 encoder A 做出的 latent vector, 送進 decoder B 之中。

後話說在前面

大家覺得 autoencoder 好像變化有點少, 品質也不是太好, 生成模型一度 GAN 獨大, 不過後來世界又變了...



04.

横空出世的 Diffusion Models

2022 年起忽然人人都在電腦創作!

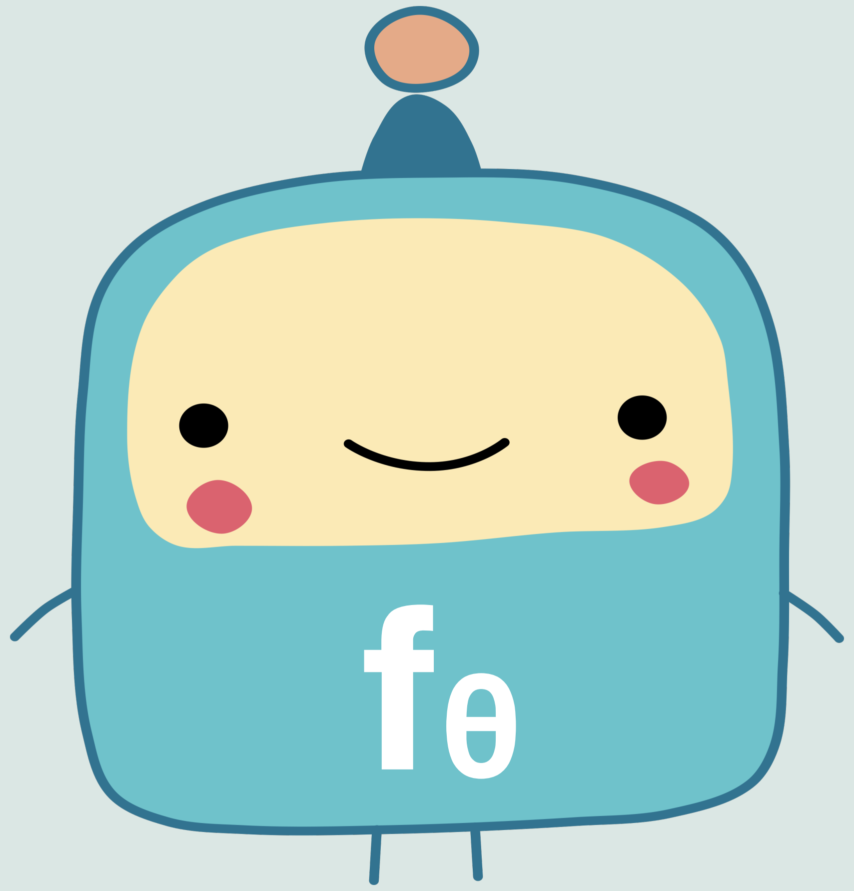
DALL·E 2

Stable Diffusion

Midjourney

Text to Image Models

"a rabbit wearing
a rabbit ear hat"



2025 年

Bing Create

SDXL

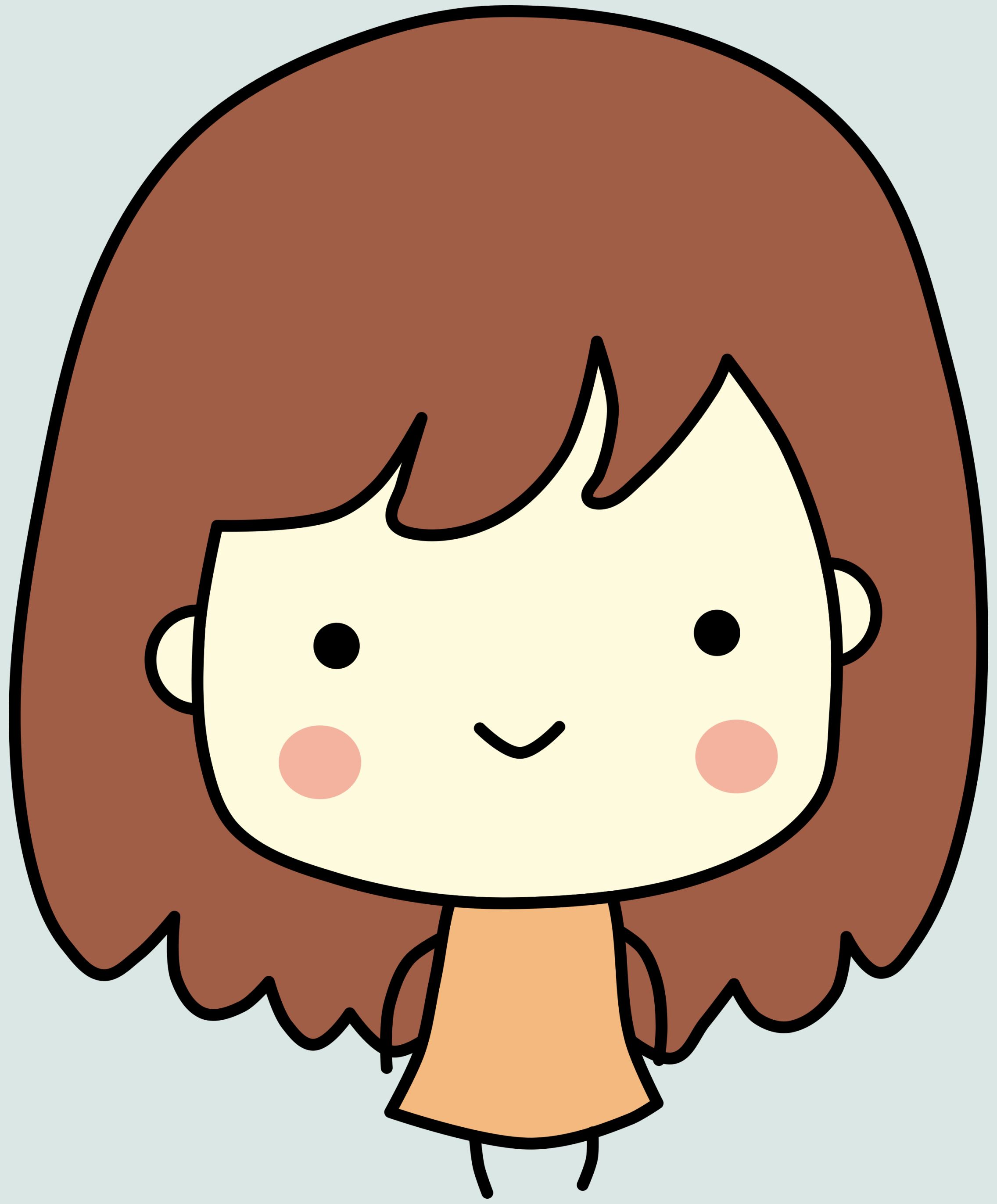
Midjourney

Diffusion Models

「收費型」文字生圖 AI

Midjourney

Leonardo.Ai



Microsoft Bing Create

DALL-E 3

https://www.bing.com/images/create

免費雲端圖像生成 AI

Bing Create

https://www.bing.com/images/create

5位台灣的大學生, 在咖啡店裡, 用一台筆電在討論東西的照片。

Whimsical Watercolor Illustration

whimsical watercolor illustration, 一個在施展魔法的可愛小女巫

Claymation

claymation, 一隻可愛的熊貓, 載著眼鏡, 在沙發上用著他的 MacBook 筆電。

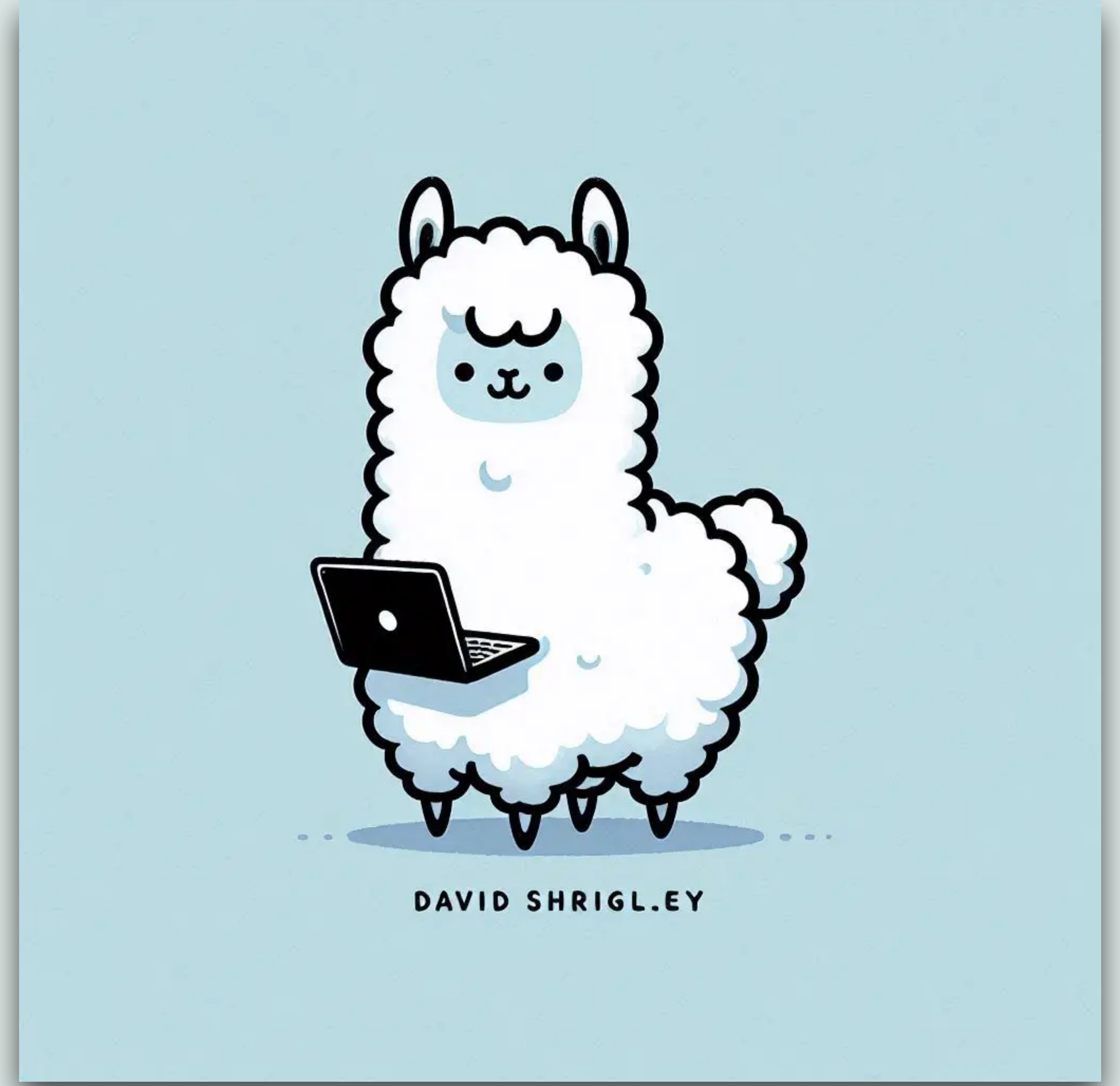
3D Pixar 卡通風格

3D animation, Pixar 卡通風格, 超可愛的機器人, 拿著水彩筆和調色盤, 在畫一幅水彩畫。

Simple 2D Vector Art

simple 2D vector art, minimalist, very few details, pastel colors, 一個可愛的女孩在咖啡店中用她的筆電

一點也不像的 David Shrigley 風格



A white llama holding a laptop, with its hands visible from behind the laptop, is depicted in a simple Japanese animation style on a light blue background. It is drawn as a David Shrigley style illustration, using only black lines on a clean sky blue background. This minimalist design conveys its cute yet simple look, while maintaining a clear contrast between the shadows and highlights.

作業

* 使用 Microsoft Bing Create (最好是選 DALL-E 3), 找到一個你喜歡的風格。
* 最好是一開始先決定你想要的風格, 試著接近。說明你怎麼改變、最後不太像但你滿意也沒有問題。
* 試著用這樣的風格畫幾張不同主題的圖出來。

開源的龍頭 Stable Diffusion

Hugging Face

diffusers 套件

automatic1111

我們會介紹更簡潔方便的 Fooocus

像 Midjourney 一樣容易的
Stable Diffusion Web UI!

Fooocus

05.

Diffusion Models 原理

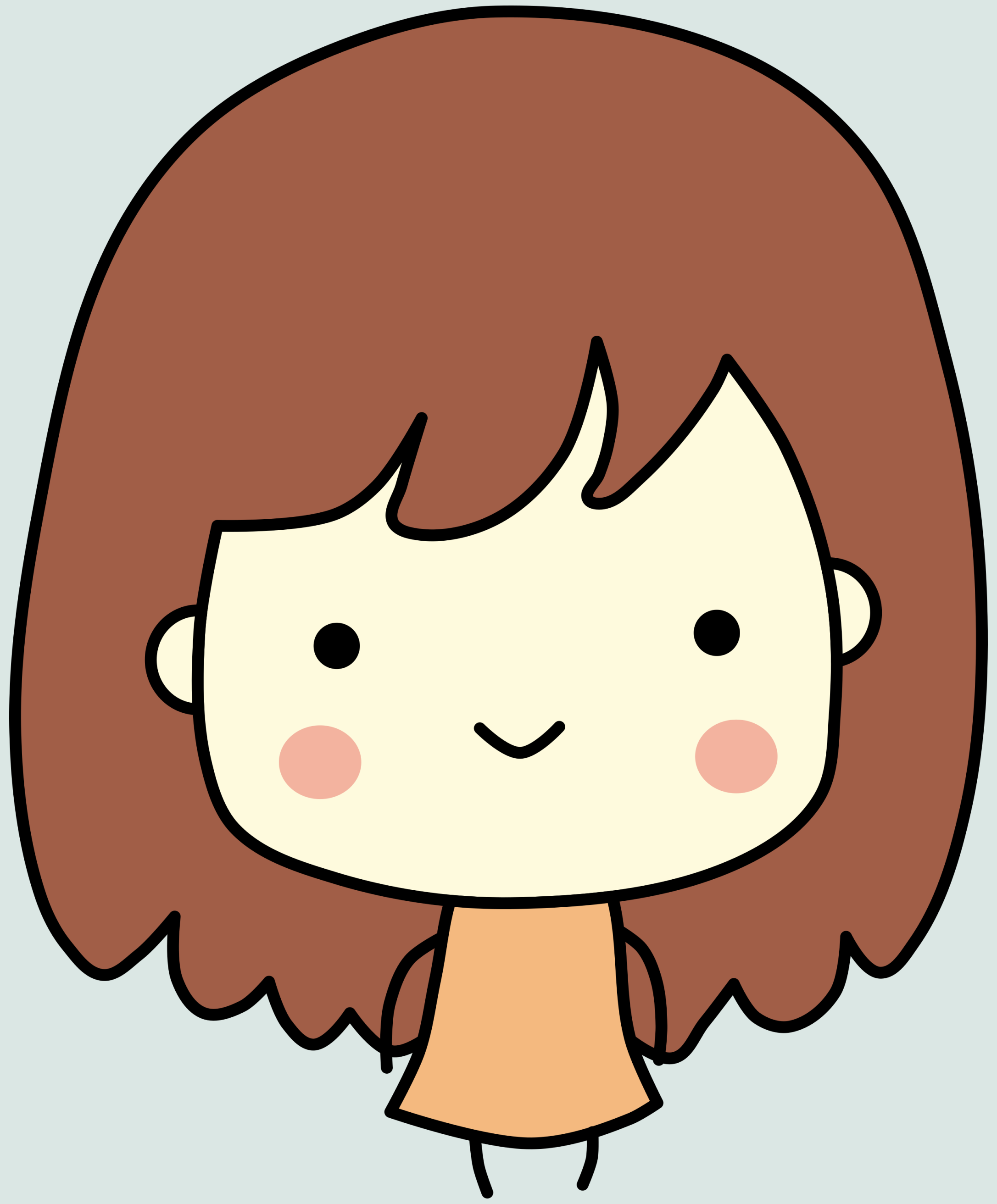
當然說「橫空出世」也不太對...

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli "Deep Unsupervised Learning using Nonequilibrium Thermodynamics," 2015.

https://arxiv.org/abs/1503.03585

但關鍵是這篇

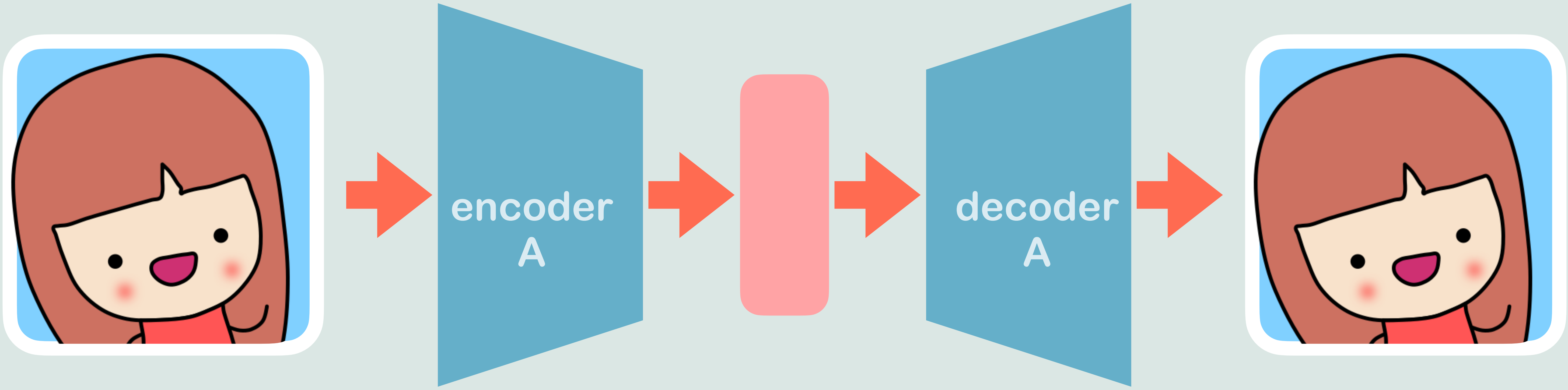
這是 OpenAI 的。



Prafulla Dhariwal and Alex Nichol "Diffusion Models Beat GANs on Image Synthesis," 2021.

https://arxiv.org/abs/2105.05233

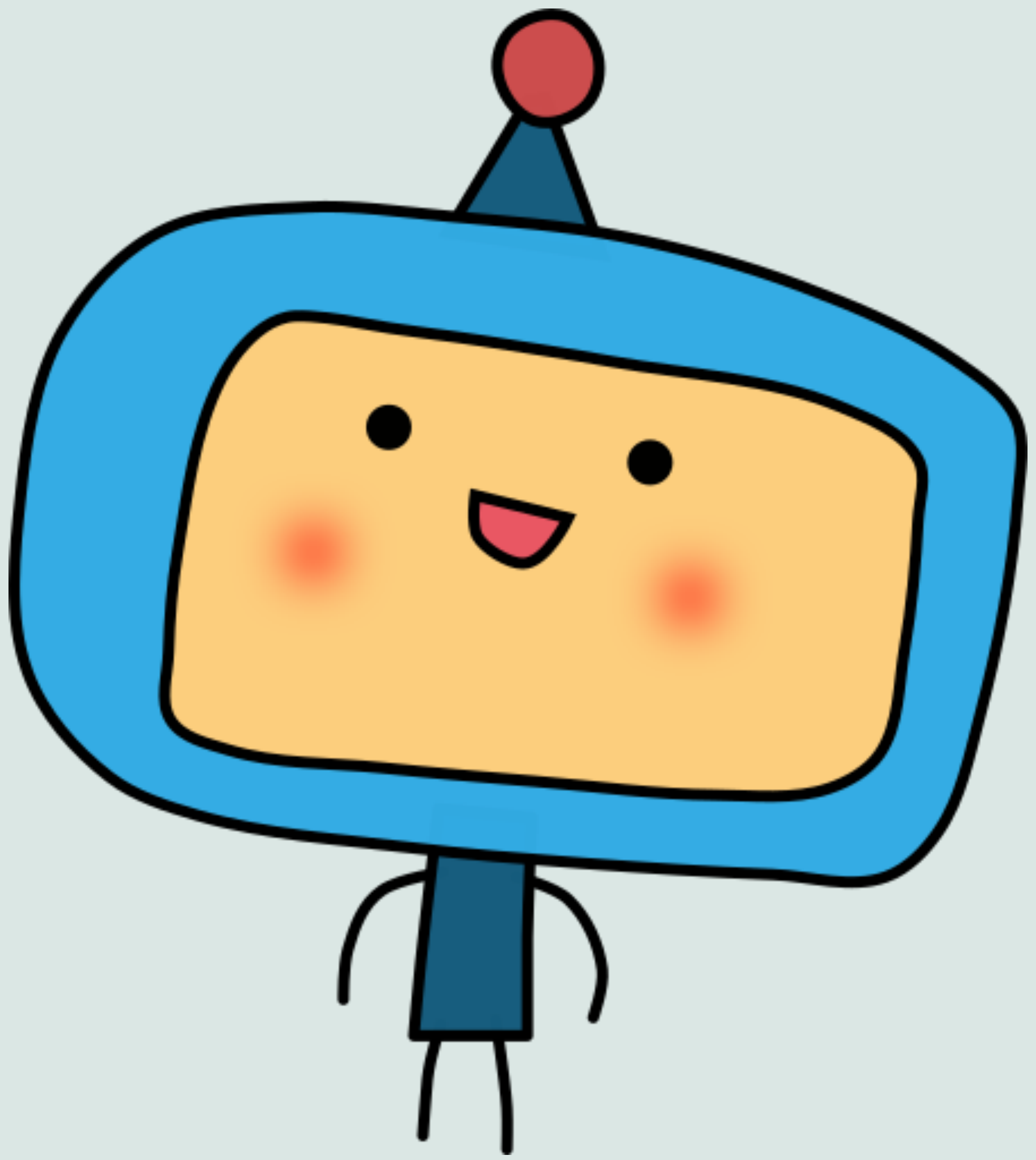
大家還記得 Autoencoder 嗎?



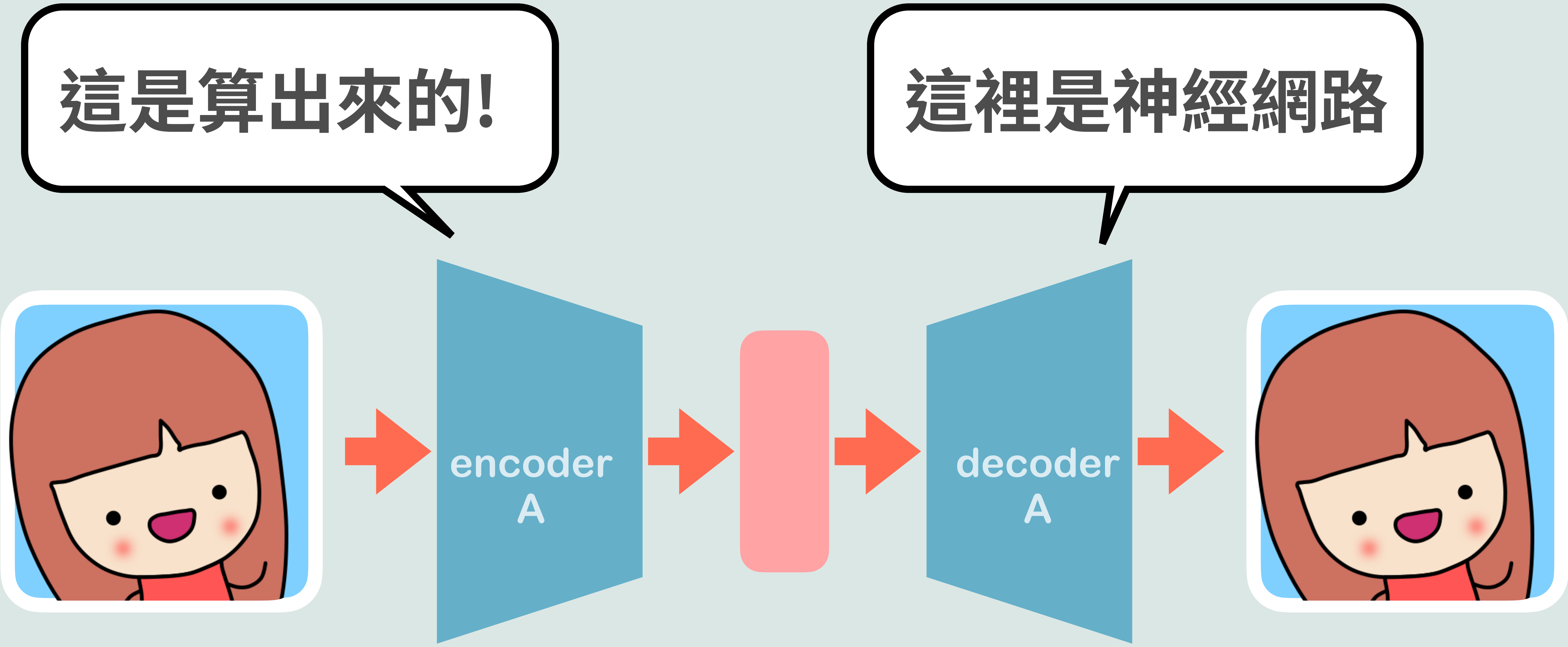
那個輸入和輸出都一樣的超呆機器人...

GPT 唬爛王給我們的啟發

也許看得夠多, 唬爛, 我是說「創作」的能力就會強!



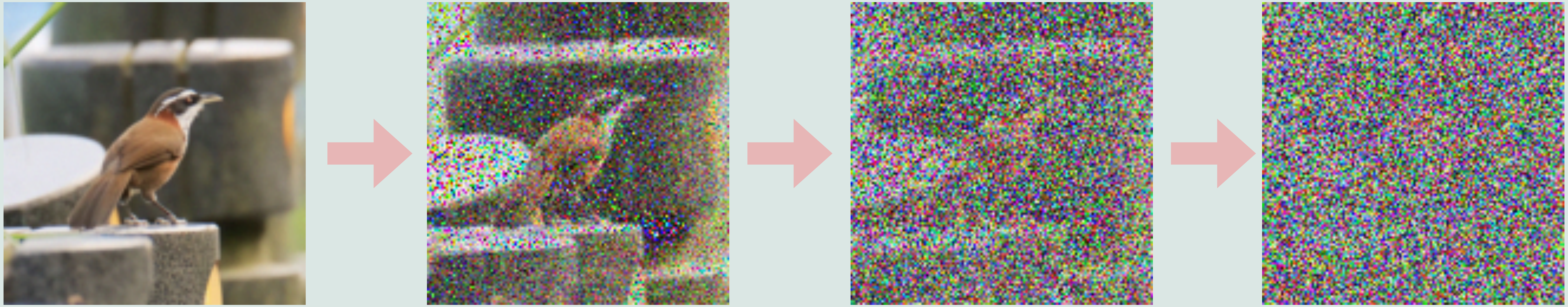
Diffusion Models 基本上就是 Autoencoder!!



只差前面 encoder 是算出來的!!

Diffusion

用相同的計算方式，一步步加上高斯雜訊。



$\mathbf{x}_0$ $\mathbf{x}_{50}$ $\mathbf{x}_{100}$ $\mathbf{x}_{150}$

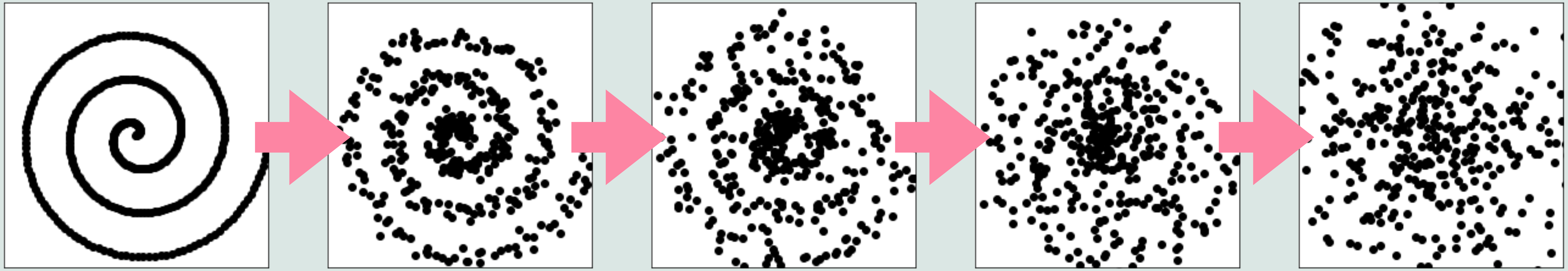
就是一路加上雜訊

β_t 我們就是取個很小的數字, 而 $\alpha_t = 1 - \beta_t$

$$x_t = \sqrt{\alpha_t}\, x_{t-1} + \sqrt{1 - \alpha_t}\, \varepsilon_{t-1}$$

一般我們會取 $\beta_1 < \beta_2 < \cdots < \beta_T$

一次擴散一點點



為什麼要做這樣的事呢? 因為每一個點就會是一個常態分佈抽樣出來的, 我們很容易生出這樣的 latent tensor, 然後應該會對應一張圖!

一個小技巧

事實上我們不用 $x_0, x_1, x_2, \ldots$ **這樣算下去, 可以一次到位!**

令

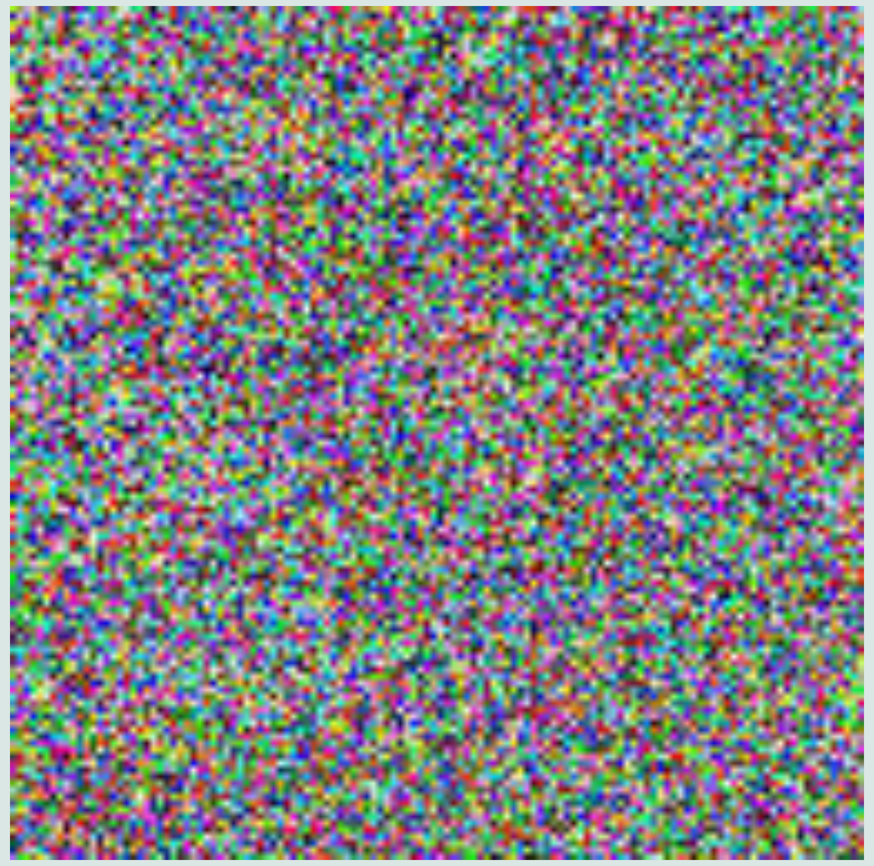
$$\overline{\alpha}_t = \prod_{i=1}^{t} \alpha_i$$

$$x_t = \sqrt{\overline{\alpha}_t}\, x_0 + \sqrt{1 - \overline{\alpha}_t}\, \varepsilon$$

也就是 encoder 我們不用學!

$\mathbf{x}_0$

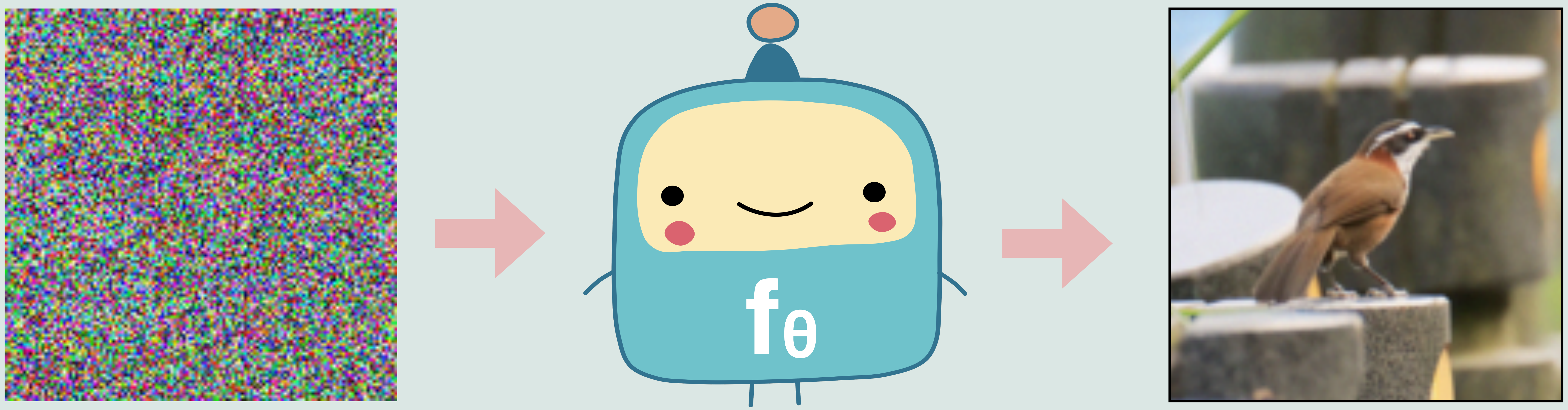
encoder



$\mathbf{x}_T$

這算出來的 (當然有從常態分佈抽樣的部份)

然後我們就用神經網路, 訓練一個 decoder

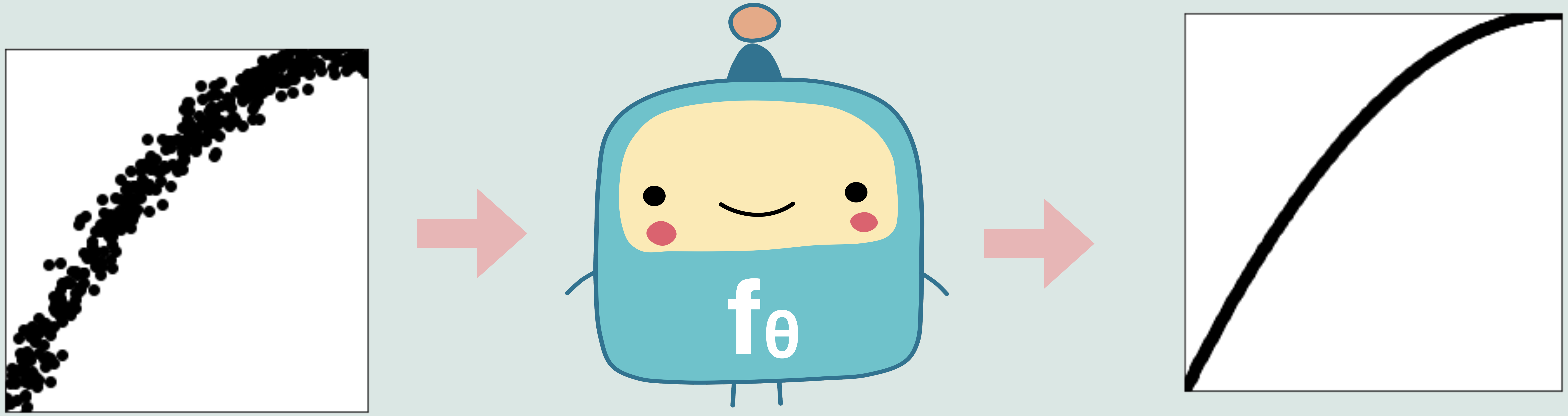


$\mathbf{x}_T$ $\mathbf{x}_0$

然後我們就用神經網路, 訓練一個 decoder

你可能會問, encoder 是用算的, 為什麼 decoder 不能算回來呢?

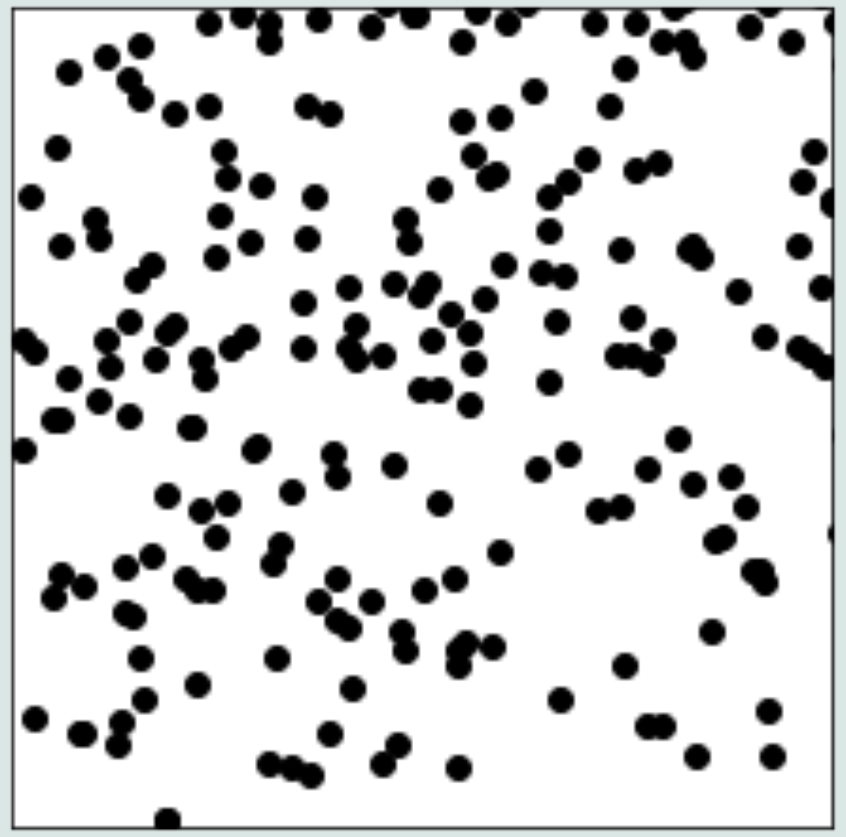
整個過程其實就是迴歸一般

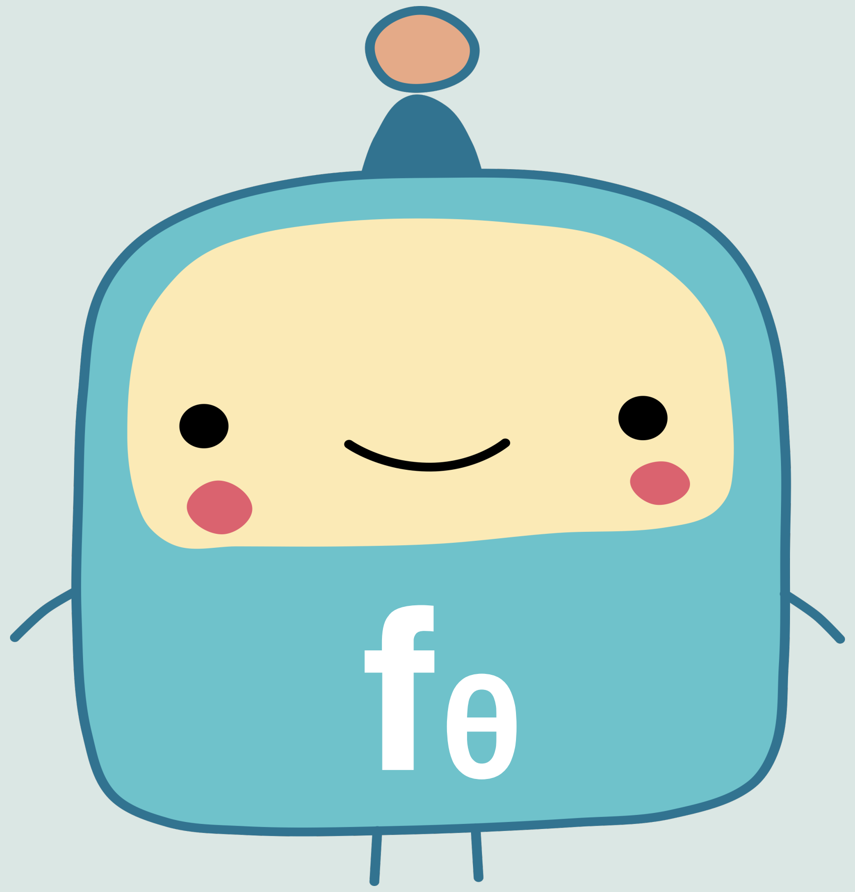


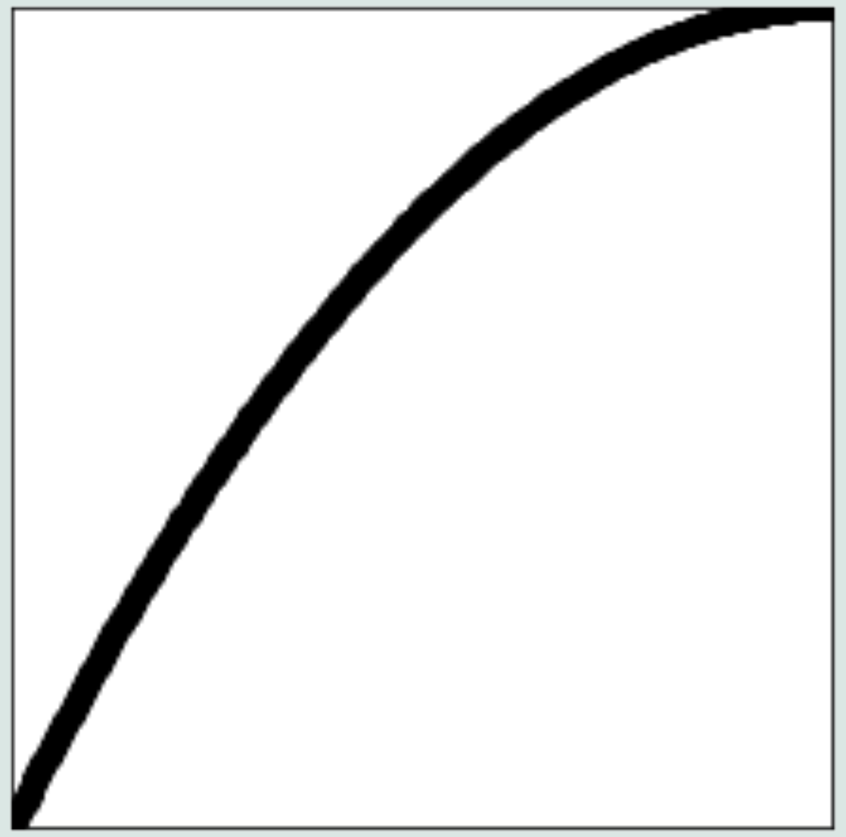
$\mathbf{x}_T$ $\mathbf{x}_0$

只是難度更高

更亂而且我們也不知道目標是什麼形式的函數。

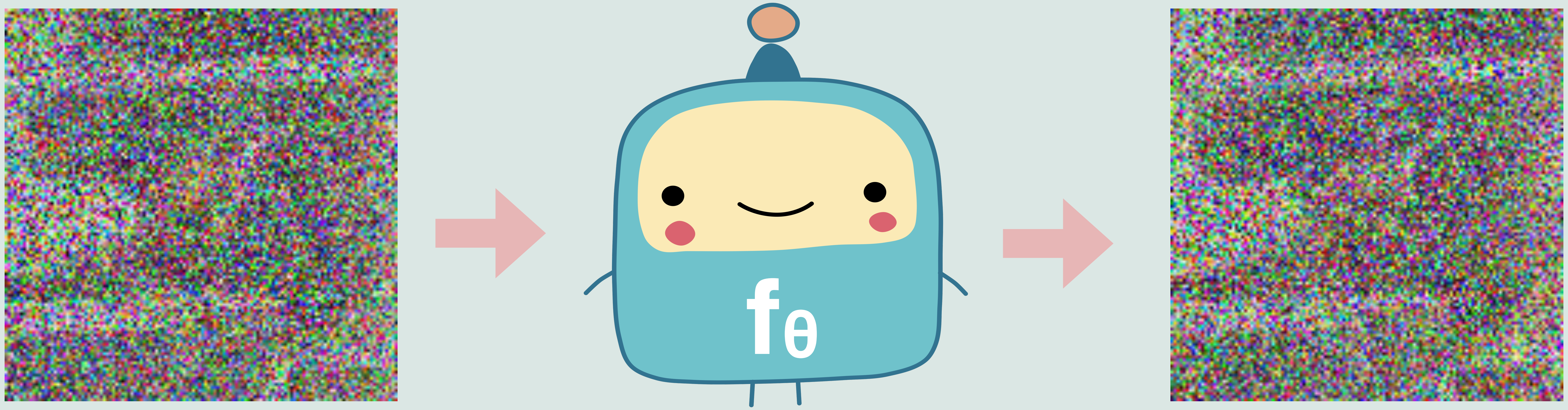






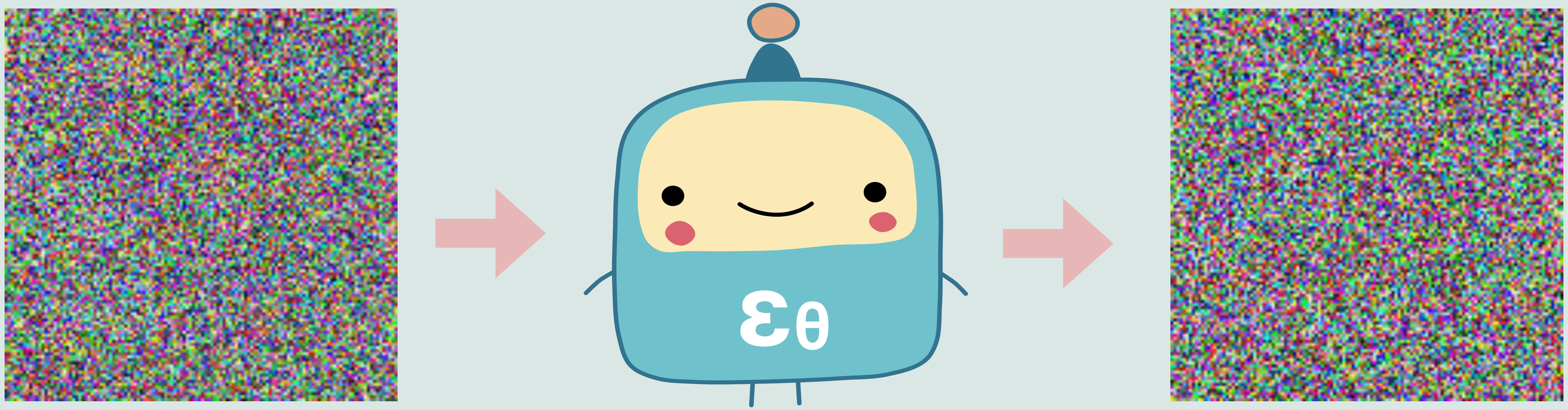
$\mathbf{x}_T$ $\mathbf{x}_0$

理論上這還原的動作應該也是一步步來



$\mathbf{x}_t$ $\mathbf{x}_{t-1}$

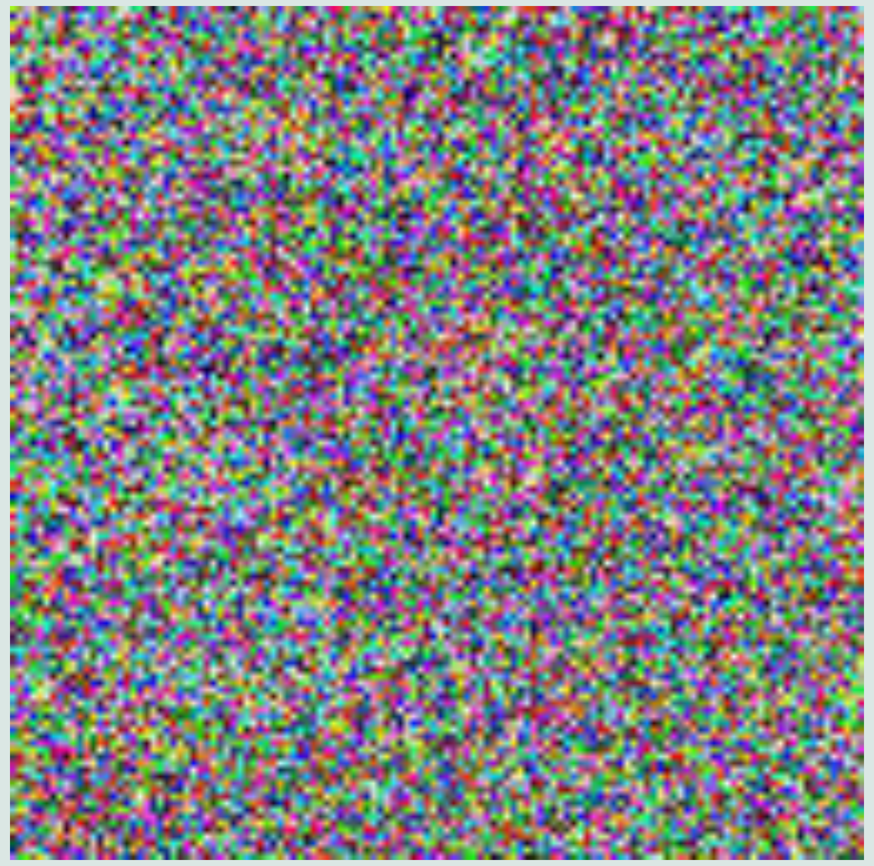
好消息是只學 noise 的部份比較容易

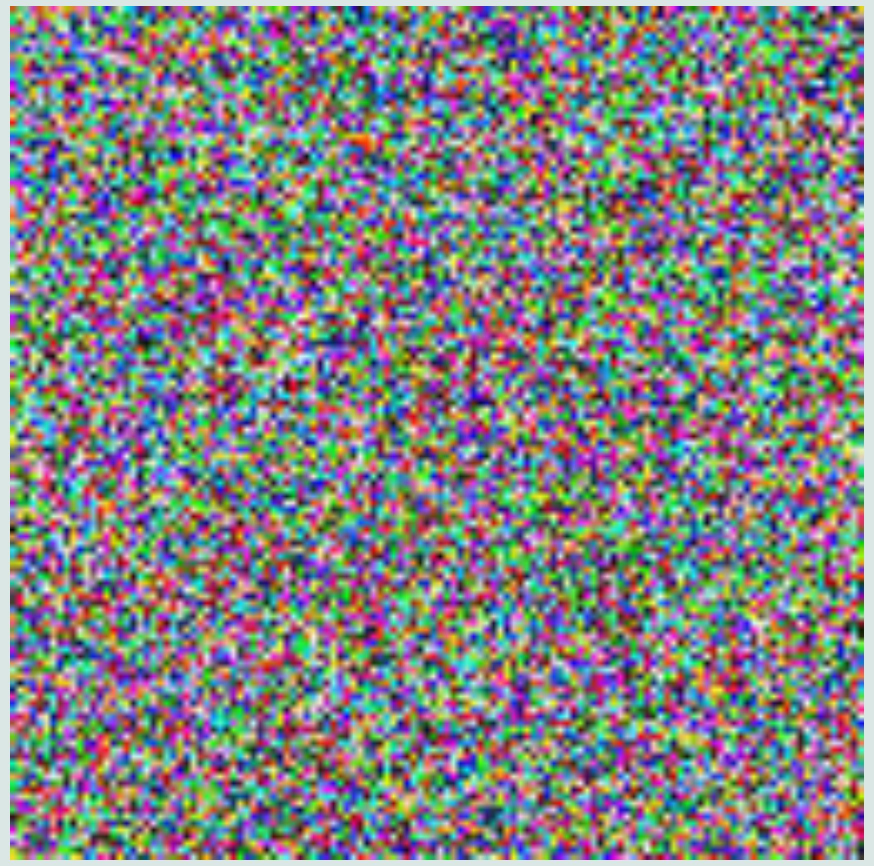


$\mathbf{x}_N$

noise $\mathcal{E}$

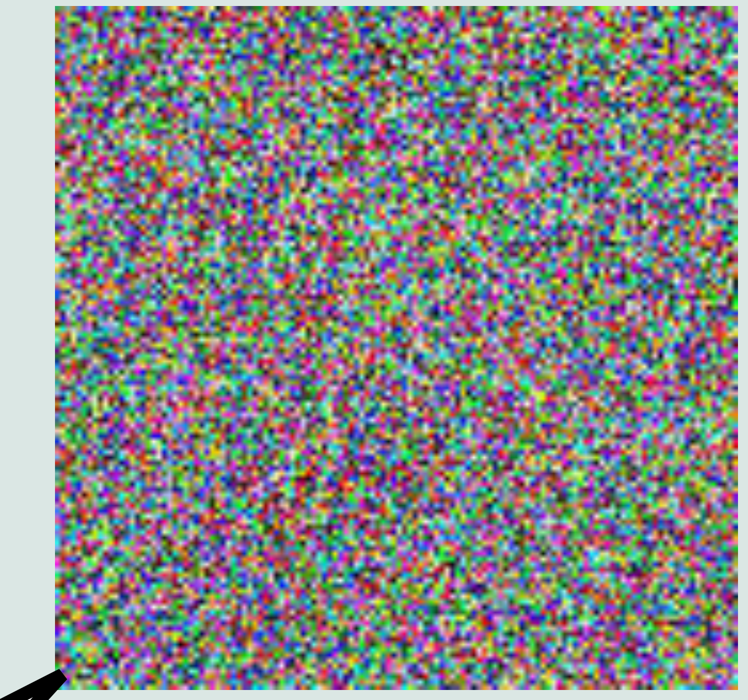
訓練成功我們圖就可以生出來了!





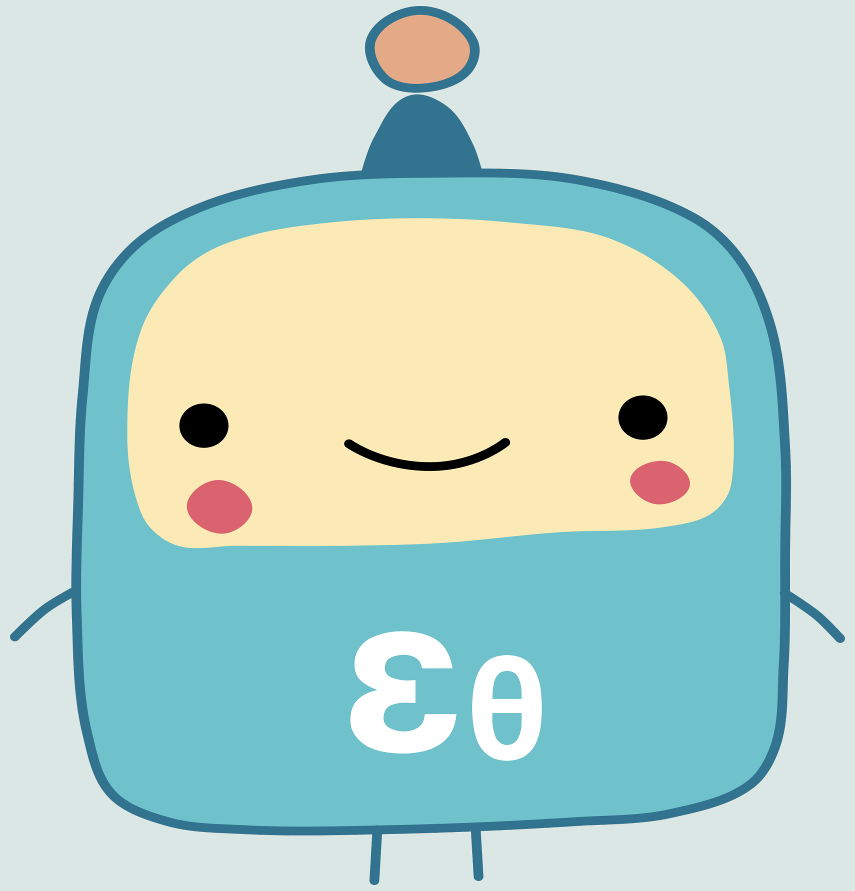
$$\mathbf{x}_N - \varepsilon = \mathbf{x}_0$$

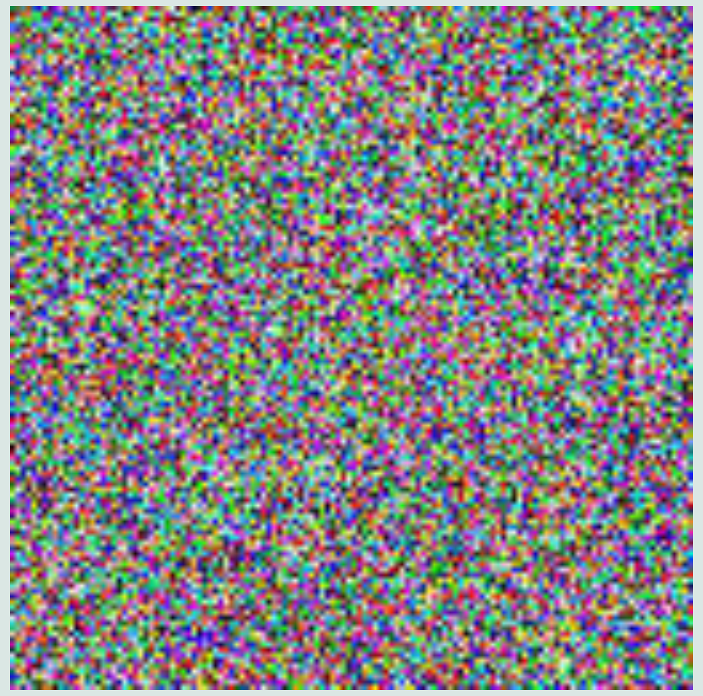
於是我們就可以生圖了!



$\mathbf{z}$

隨機生出來!





ε

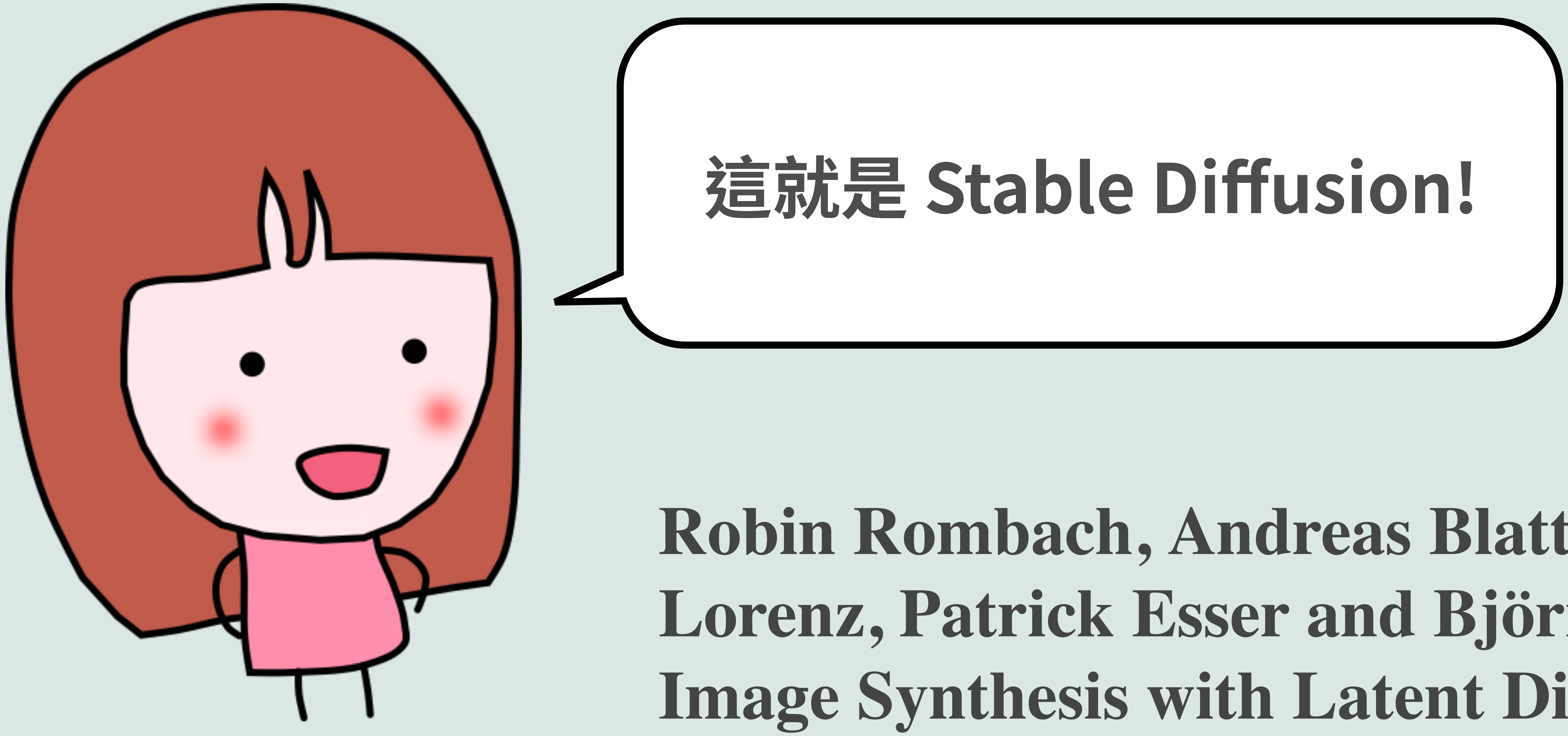
$$\mathbf{z} - \varepsilon =$$

其實可能會重覆去雜訊的工作

06.

Latent Diffusion Models

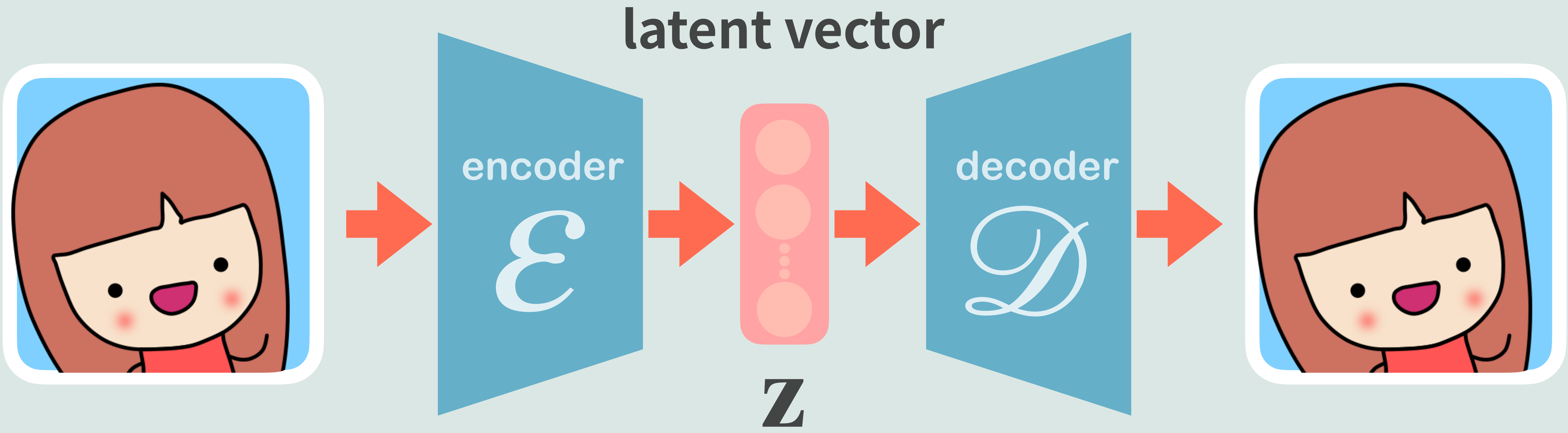
Stable Diffusion



Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser and Björn Ommer "High-Resolution Image Synthesis with Latent Diffusion Models," 2022.

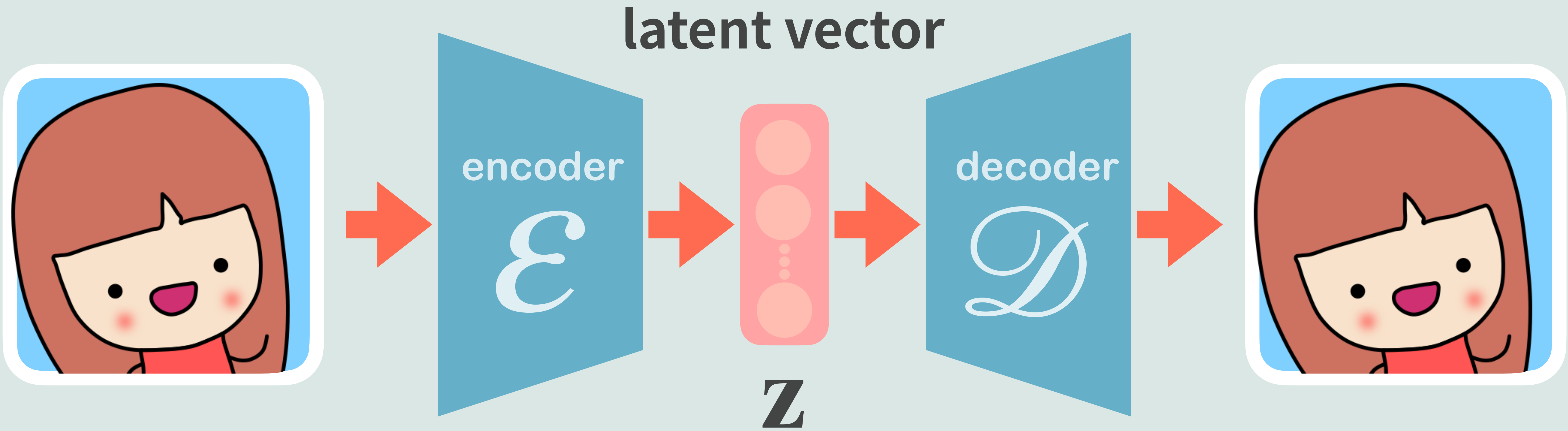
https://arxiv.org/abs/2112.10752

先訓練個前面說的 VAE



真正用 diffusion model 的是中間 latent vector。

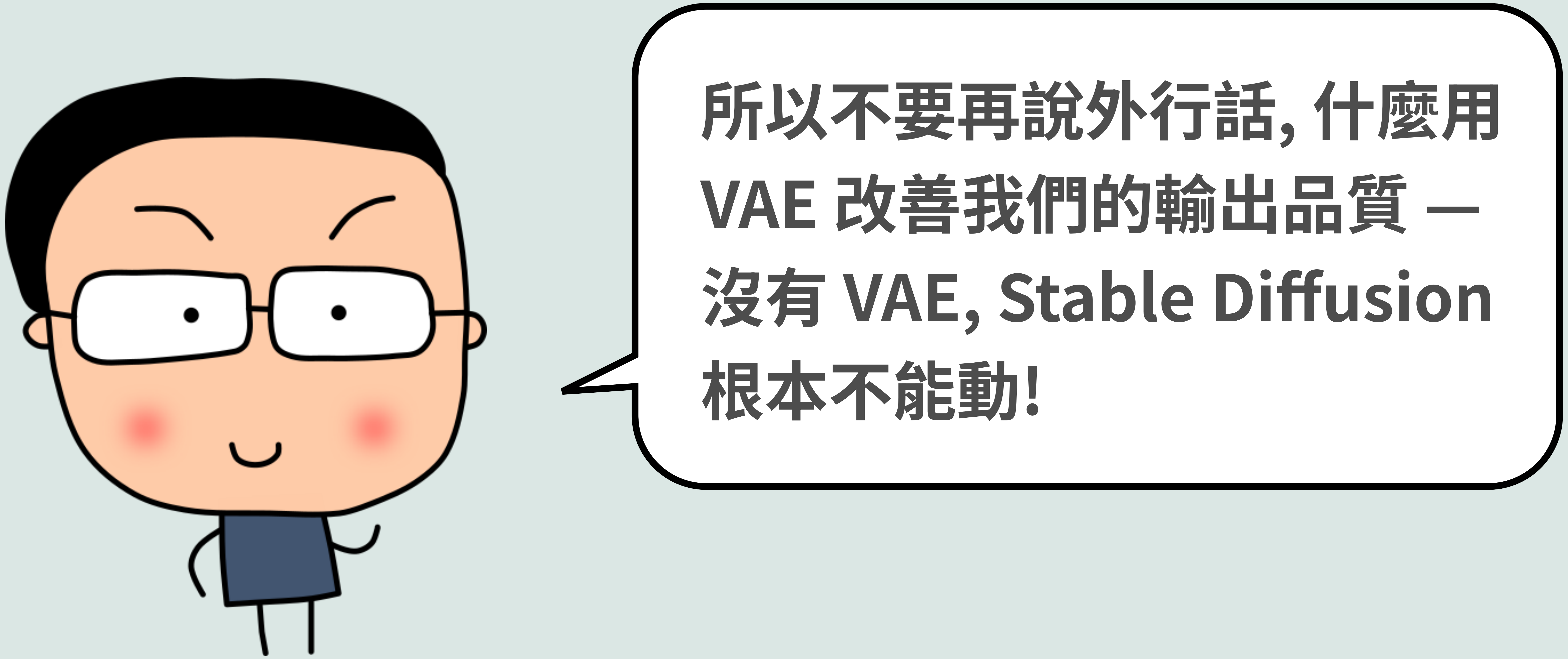
標準 Stable Diffusion 是縮小 8 倍



512×512 64×64 512×512

這種先用 VAE 的模型叫 LDM

Latent Diffusion Models



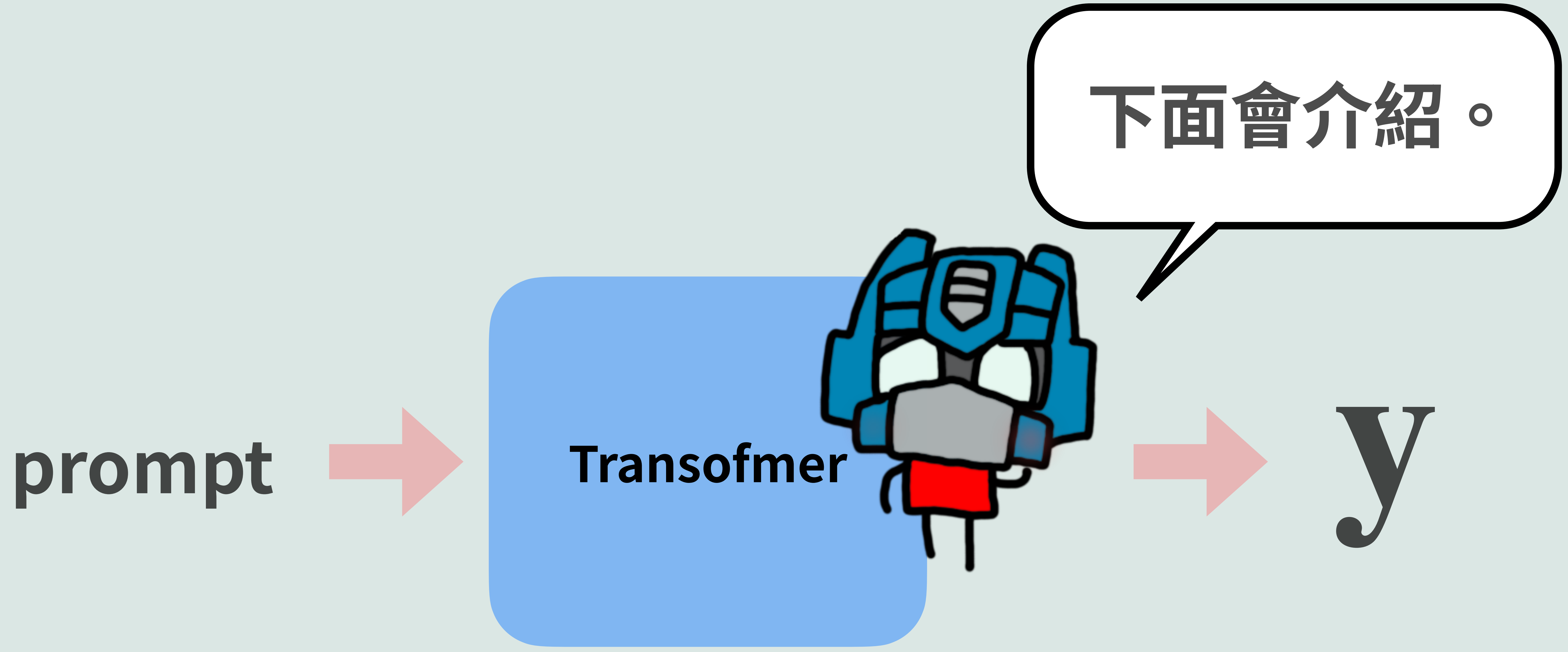
但我們可以不用原本預設的 VAE

預設

EMA

MSE

那文字生圖是怎麼做到的呢?

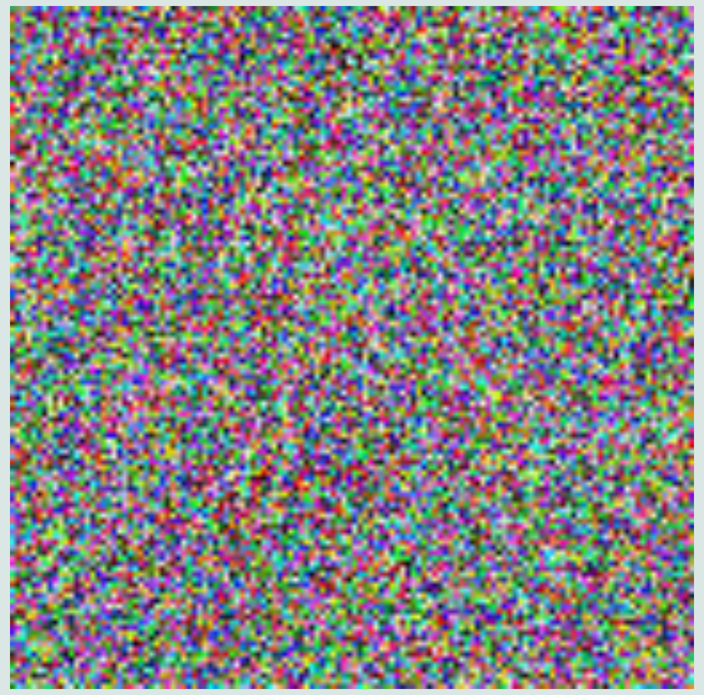


一般就是用很會處理文字的 transformer model, 把文字化為特徵代表 tensor (想成一堆數字或向量就好)。

然後「加到」我們隨機生成的那些 noise

latent vector

$$z = \quad z_T + y$$

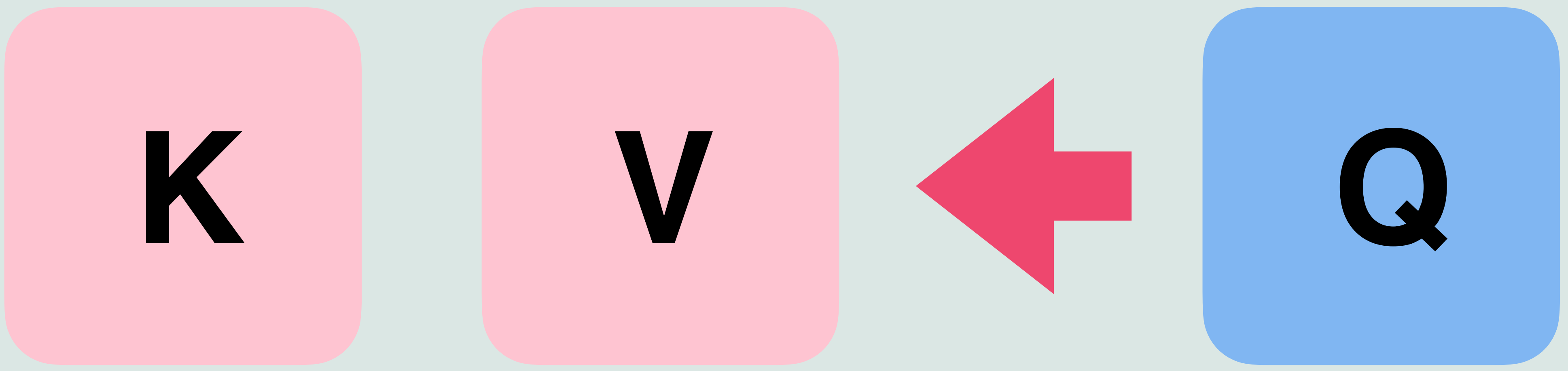


隨機生出的

代表文字意思的

是不是和 styleGAN 很像? 這裡「加」也不一定是真的加...

標準「融合」通常是用 transformer...

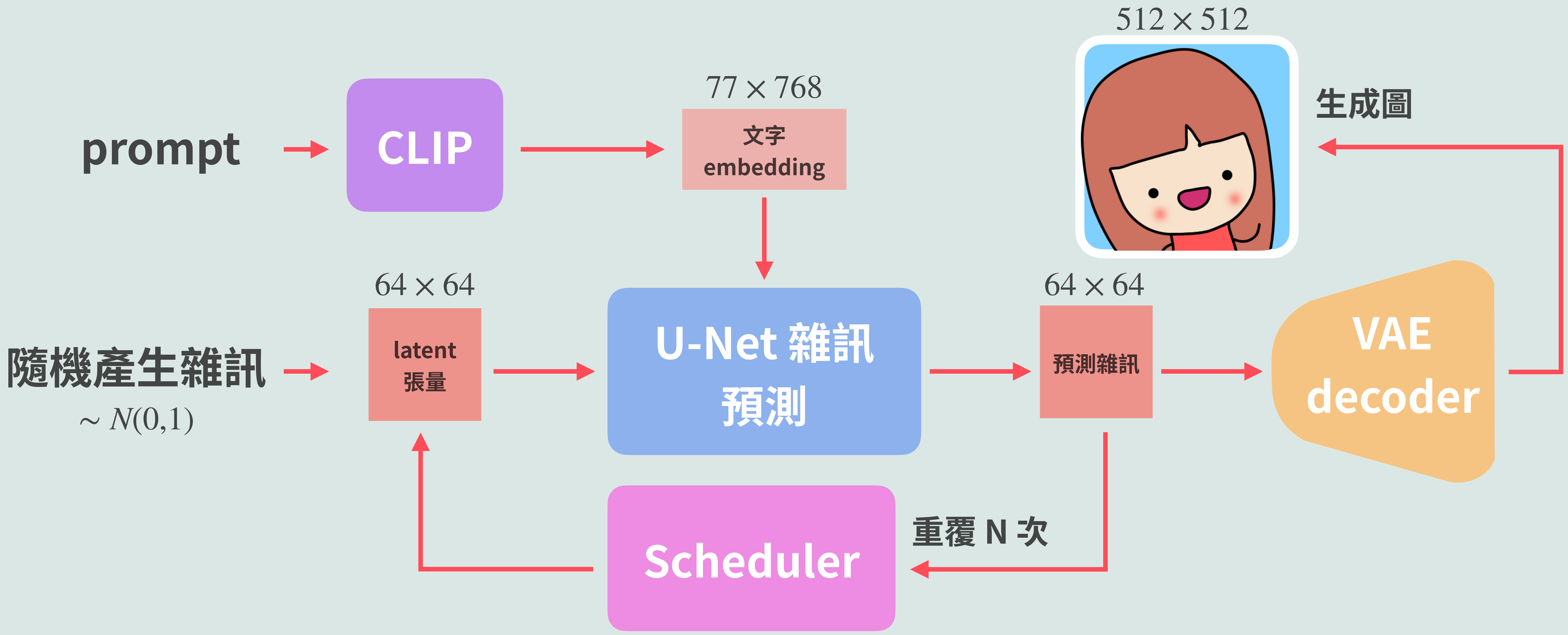


$\mathbf{y}$ $\qquad$ $\mathbf{z}_T$

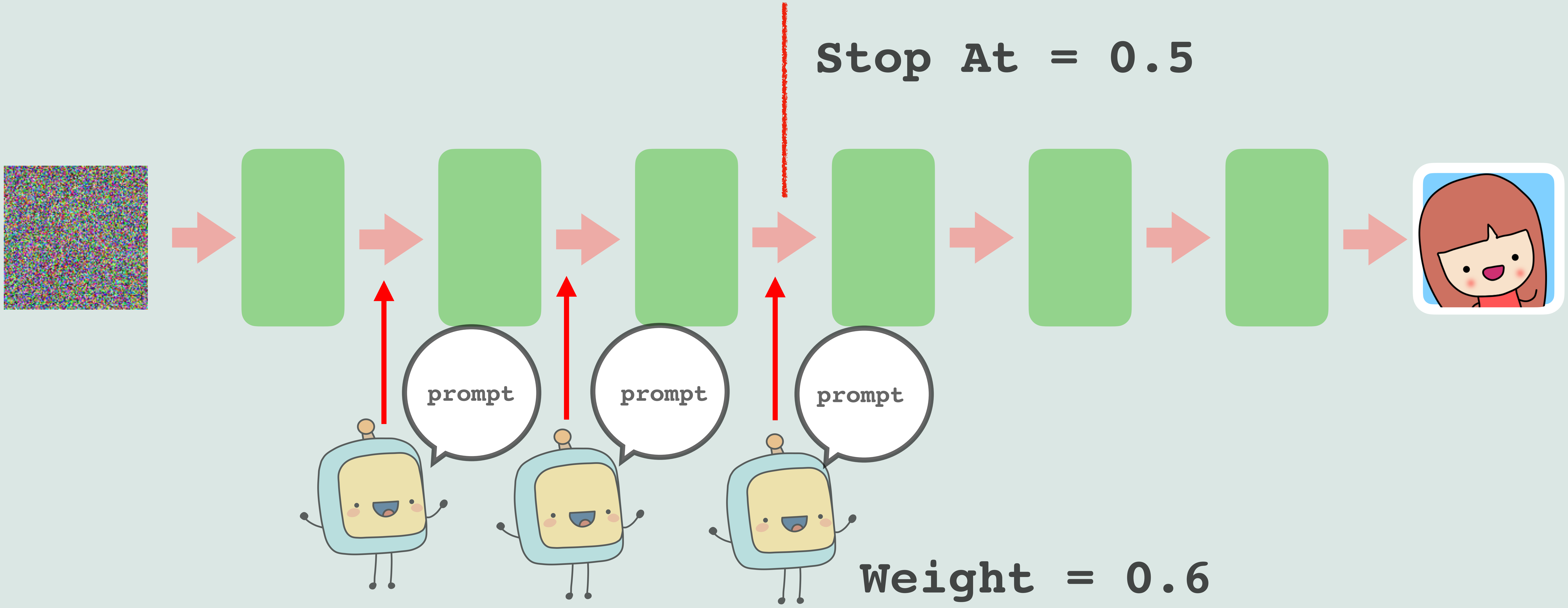
K, V 是文字這邊來的

Q 是隨機生出原始 latent vector 來的

Stable Diffusion 架構圖



解碼 (生成) 最主要是用 U-Net



我們的「想法」至少有兩個參數可以調整