介紹

圖片的辨識大家已經做得有點膩了！

還有沒有什麼可以玩的呢？人類的智慧活動除了『判斷』(辨識) 好像還有另外一個，那就是『創作』，『創作』才是最終極的智慧活動的形式

如何創作呢？還是要從『人』出發，思考『人類』創作的過程

人類創作過程

? 我們如何把人類創作的過程套用到深度學習呢？

AI創作過程

隨機數列 (突發靈感)

反向神經網路

成品

之前我們在辨別事物的時候神經數目都是『縮小』，因為是『組合』的概念，但是現在要『增加』，因為是將靈感『擴充』

?

深度學習的精髓是透過與『正確答案』的距離來調整『模型參數』，但是我們在創作的情況下，如何定義『正確』？無法定義正確，我們就無法調整參數

救星

GAN之父 - Ian Goodfellow

這個人真的得好好介紹一下！就是他想出了GAN，讓整個深度學習界為之瘋狂，整個過程也曲折離奇，Ian Goodfellow 跟朋友在一個酒館慶祝的時候，剛好討論到生成圖片的困境，那時候一個衝擊性的主意突然閃過他的腦袋！於是他馬上回家開始寫程式一直到天亮，那天的那個主意就是GAN

GAN

隨機數列 (突發靈感)

反向神經網路

我們喜歡叫他
Generator(創作家)

成品

正向神經網路

我們喜歡叫他
Discriminator(鑑賞家)

0 (爛)
1 (好)

GAN精髓

用另外一個深度網路來判定成品到底好或者壞，這也是為何我們稱它為GAN(Generative Adversarial Network)，生成式對抗模型，由兩個神經網路互相對抗而組成

訓練步驟

步驟1 訓練鑑賞家

訓練目標 **鑑賞家能分辨出真假**

操作方式 從訓練資料集抽出真正樣本以及讓現在的創作家創作出假樣本，真正樣本標注答案1(真)，假樣本標註答案0(假)

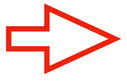
步驟2 訓練創作家

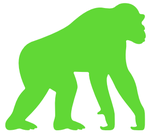
訓練目標 **創作家的作品能被評斷為真**

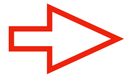
操作方式 讓創作家創作出假樣本，並且給出答案1(真)，利用和真答案的距離回頭調整創作家的參數

GAN訓練過程

隨著創作家和鑑賞家的訓練， 整個過程會經歷什麼狀態呢？







狀態1. 創作家的作品是跟真的作品差很多的，所以在訓練的時候鑑賞家可以輕易訓練出『真』和『偽』的差別

狀態2. 創作家進步了，但是還是跟真正的作品有差距，所以鑑賞家也一起進步了，還是可以找到『真』和『偽』的差別

狀態3. 最終狀態，偽作跟真正的作品幾乎沒差距了，所以到這個平衡狀態，你的鑑賞家已經無法再訓練了，這時候候創作家的作品就像真的作品一樣

GAN注意事項

跟其他深度學習的訓練不一樣，其他深度學習的訓練Loss會持續下降，但是GAN最後是到一個穩定狀態，創作家的圖(標註為假)和真圖(標註為真)幾乎長得一樣，這時候鑑賞家的訓練已經停滯了，所以訓練Loss會停留在某個數字區間，不會無止盡往下降

因此訓練該到什麼時候為止呢？我建議觀察兩點

1. 鑑賞家和創作家的Loss都差不多平穩
2. 真的觀察一下創作家創作圖看看合不合理

手寫數字

第一個練習先從手寫數字開始，
MNIST資料集是 28 x 28 的手寫數字

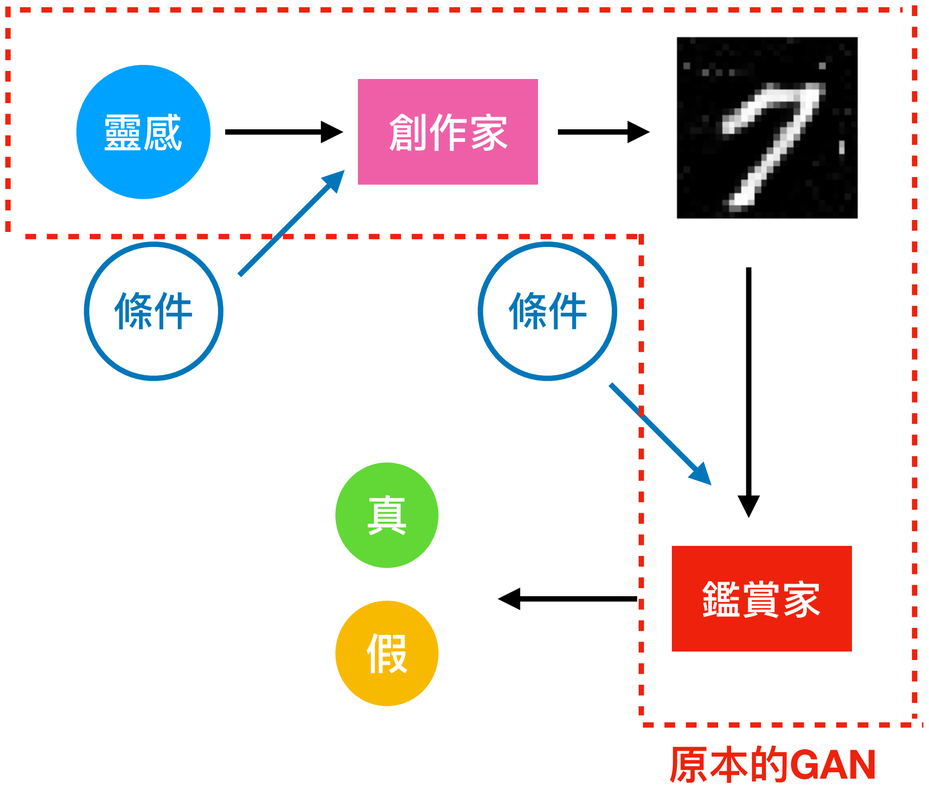
最後成果

條件式GAN

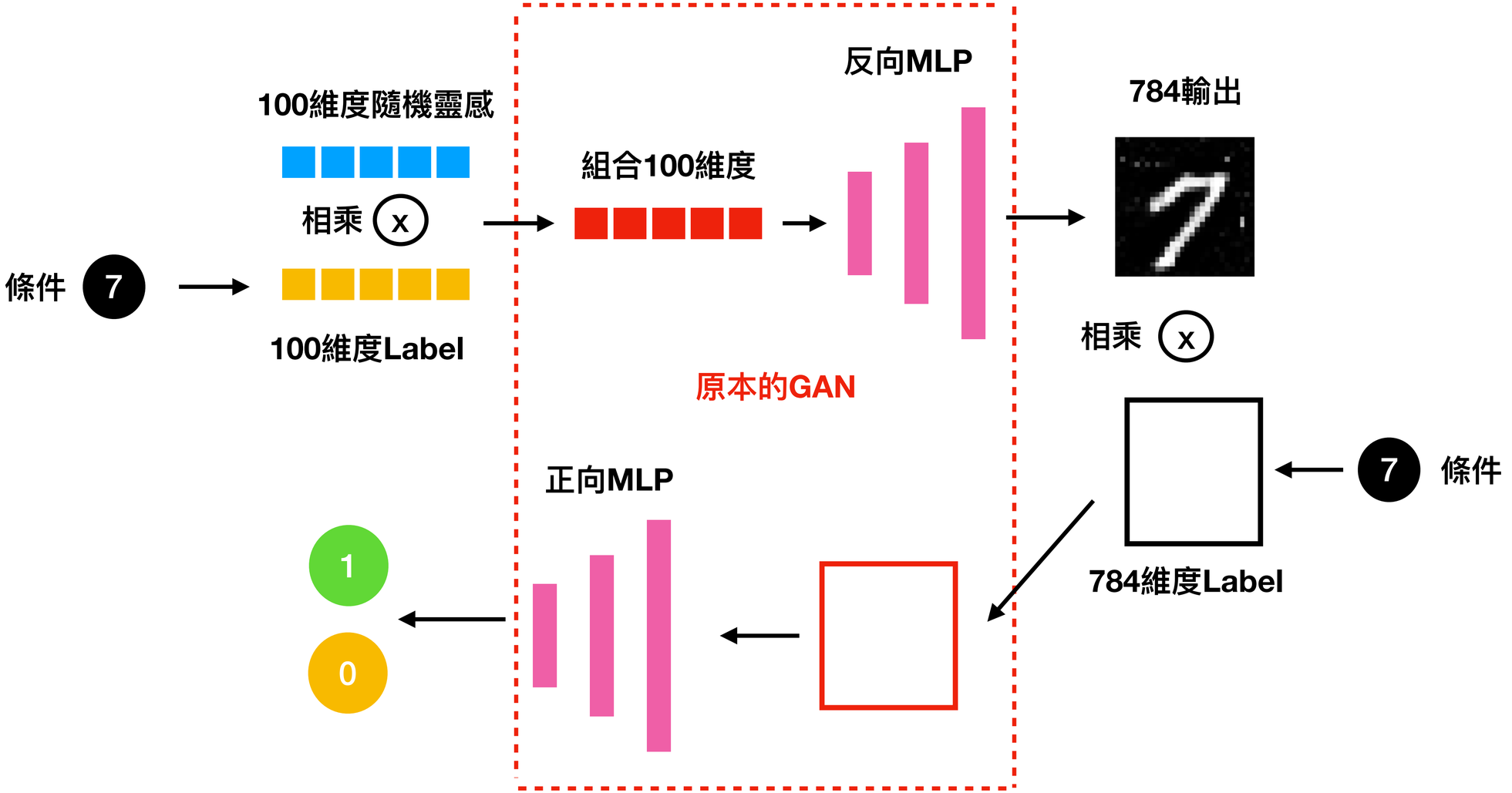
我們已經學會基本的GAN了，但我有個小問題，就是我們的創作家基本上只管真假，盡量創作出『真』，但並不能控制他到底要寫 0 - 9 的哪一個

條件式 GAN (conditional GAN) 就是來解決這個問題的，在帶入靈感的時候，我們就會額外限制要生成什麼樣的作品！

模型



模型詳細


反向MLP
100維度隨機靈感
組合100維度
784輸出
相乘
x
條件
7
100維度Label
原本的GAN
相乘
x
7
條件
正向MLP
784維度Label
1
0

其實

如果要簡單的了解CGAN，你會發現乘上 Label 的動作不就像『卷積網路』過濾的概念嗎！？針對該出現的給予『正分數』，不該出現的點給予『負分數』，對於『靈感』我們過濾出我們針對那個數字該考慮的靈感位置，對於『判斷』我們也給予每個位置一個過濾！

GAN

隨機數列 (突發靈感)

反向神經網路

我們喜歡叫他
Generator(創作家)

成品

正向神經網路

我們喜歡叫他
Discriminator(鑑賞家)

0 (爛)
1 (好)

GAN精髓

用另外一個深度網路來判定成品到底好或者壞，這也是為何我們稱它為GAN(Generative Adversarial Network)，生成式對抗模型，由兩個神經網路互相對抗而組成

訓練步驟

步驟1 訓練鑑賞家

訓練目標 **鑑賞家能分辨出真假**

操作方式 從訓練資料集抽出真正樣本以及讓現在的創作家創作出假樣本，真正樣本標注答案1(真)，假樣本標註答案0(假)

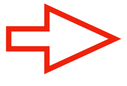
步驟2 訓練創作家

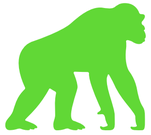
訓練目標 **創作家的作品能被評斷為真**

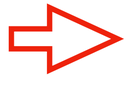
操作方式 讓創作家創作出假樣本，並且給出答案1(真)，利用和真答案的距離回頭調整創作家的參數

GAN訓練過程

隨著創作家和訓練家的訓練， 整個過程會經歷什麼狀態呢？







狀態1. 創作家的作品是跟真的作品差很多的，所以在訓練的時候鑑賞家可以輕易訓練出『真』和『偽』的差別

狀態2. 創作家進步了，但是還是跟真正的作品有差距，所以鑑賞家也一起進步了，還是可以找到『真』和『偽』的差別

狀態3. 最終狀態，偽作跟真正的作品幾乎沒差距了，所以到這個平衡狀態，你的鑑賞家已經無法再訓練了，這時候候創作家的作品就像真的作品一樣

GAN注意事項

跟其他深度學習的訓練不一樣，其他深度學習的訓練Loss會持續下降，但是GAN最後是到一個穩定狀態，創作家的圖(標註為假)和真圖(標註為真)幾乎長得一樣，這時候鑑賞家的訓練已經停滯了，所以訓練Loss會停留在某個數字區間，不會無止盡往下降

因此訓練該到什麼時候為止呢？我建議觀察兩點

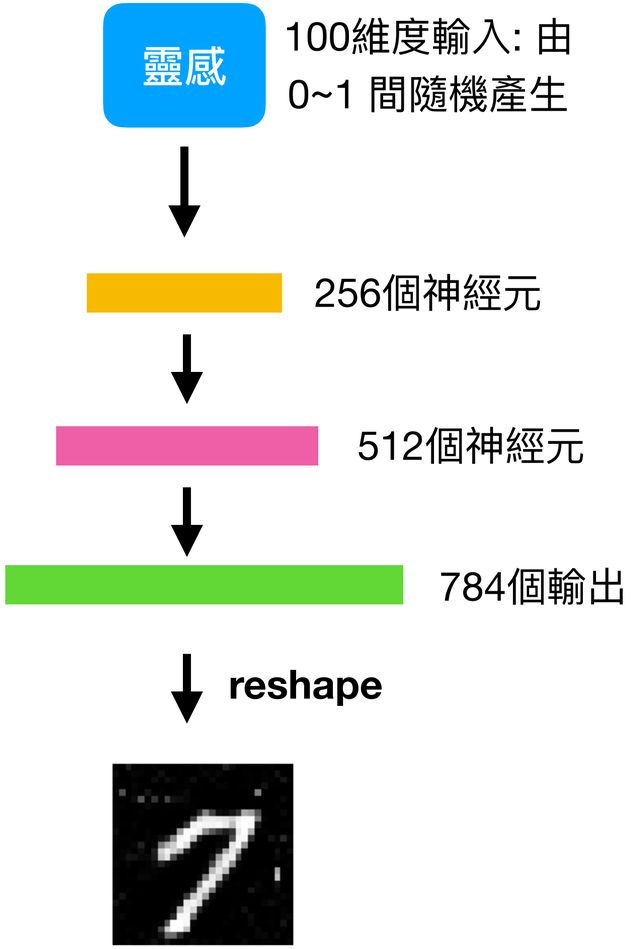
1. 鑑賞家和創作家的Loss都差不多平穩
2. 真的觀察一下創作家創作圖看看合不合理

創作家

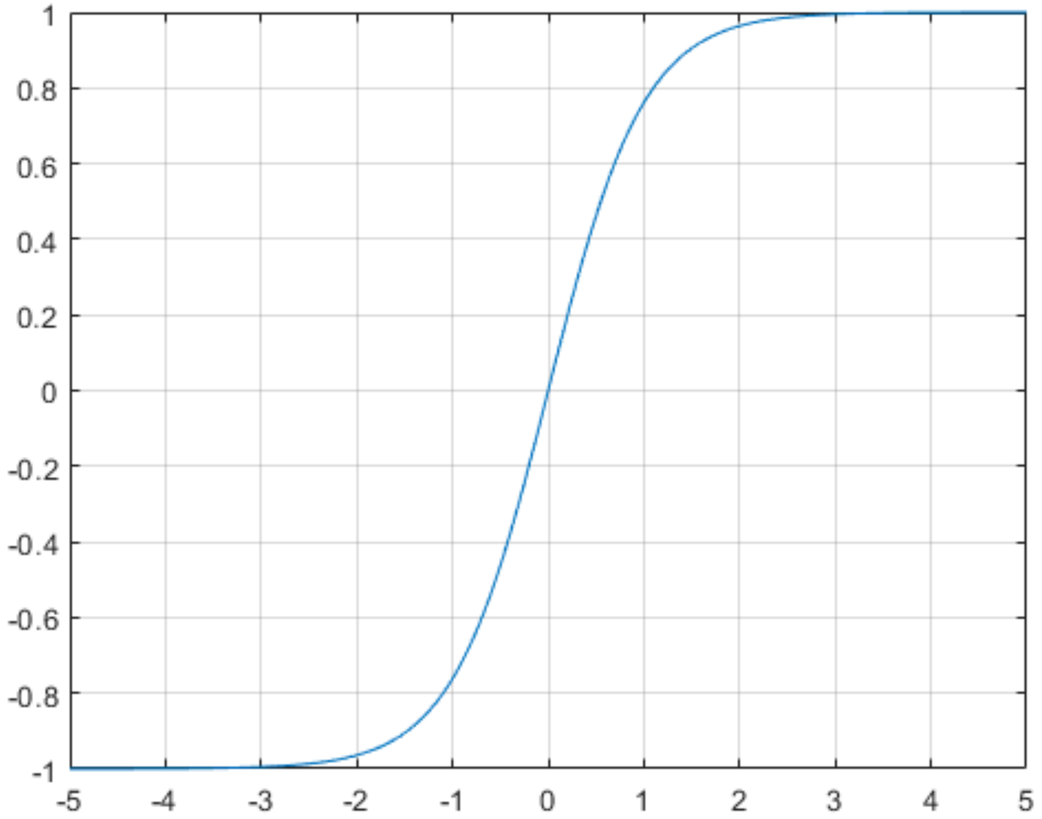
我們使用最基礎的MLP來當我們的深度模型基礎

Layer (type)	Output Shape	Param #
dense_8 (Dense)	(None, 256)	25856
batch_normalization_3 (Batch	(None, 256)	1024
dense_9 (Dense)	(None, 512)	131584
batch_normalization_4 (Batch	(None, 512)	2048
dense_10 (Dense)	(None, 784)	402192

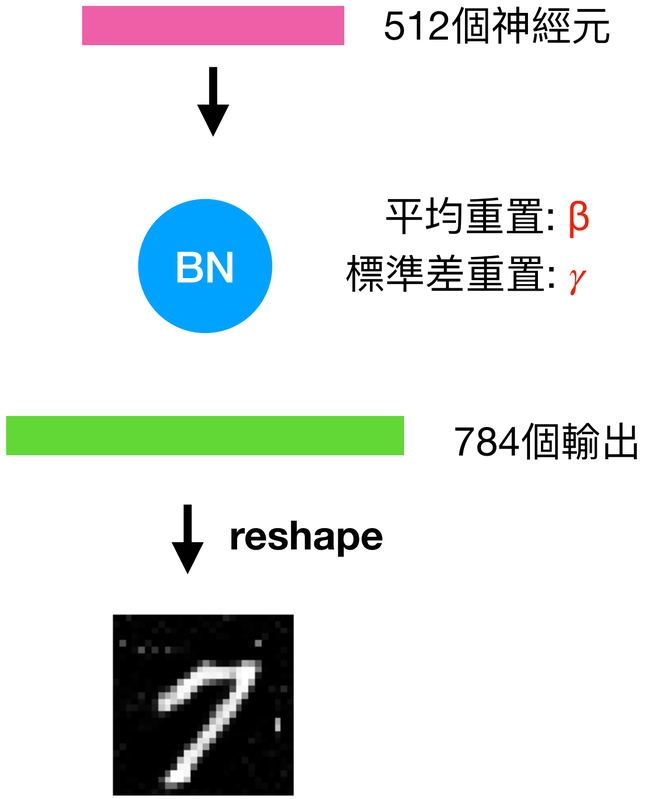
Total params: 562,704
Trainable params: 561,168
Non-trainable params: 1,536



tanh

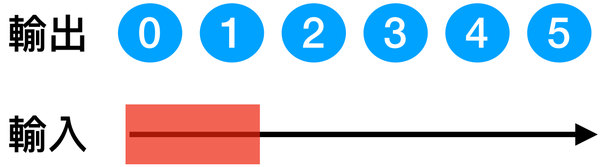


BatchNormalization



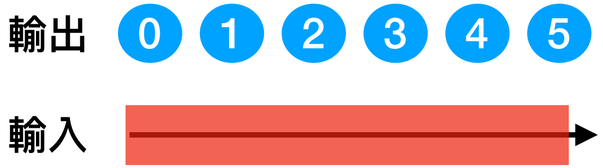
without BN

模型也會像人類一樣，盡挑一些容易的事情做，譬如只輸出好寫的0和1



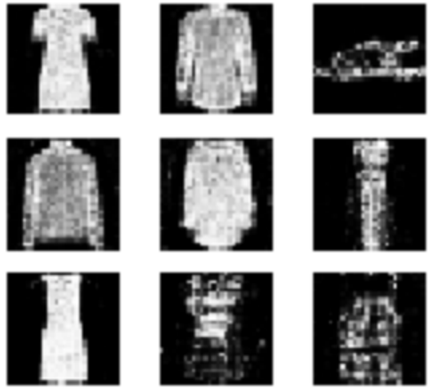
with BN

藉由把輸入調整回正常區間，強迫模型一定要寫出別的數字



流行衣物

作業我們來練習一下流行衣物如何
請載入Keras的Fashion MNIST資料集



最後成果

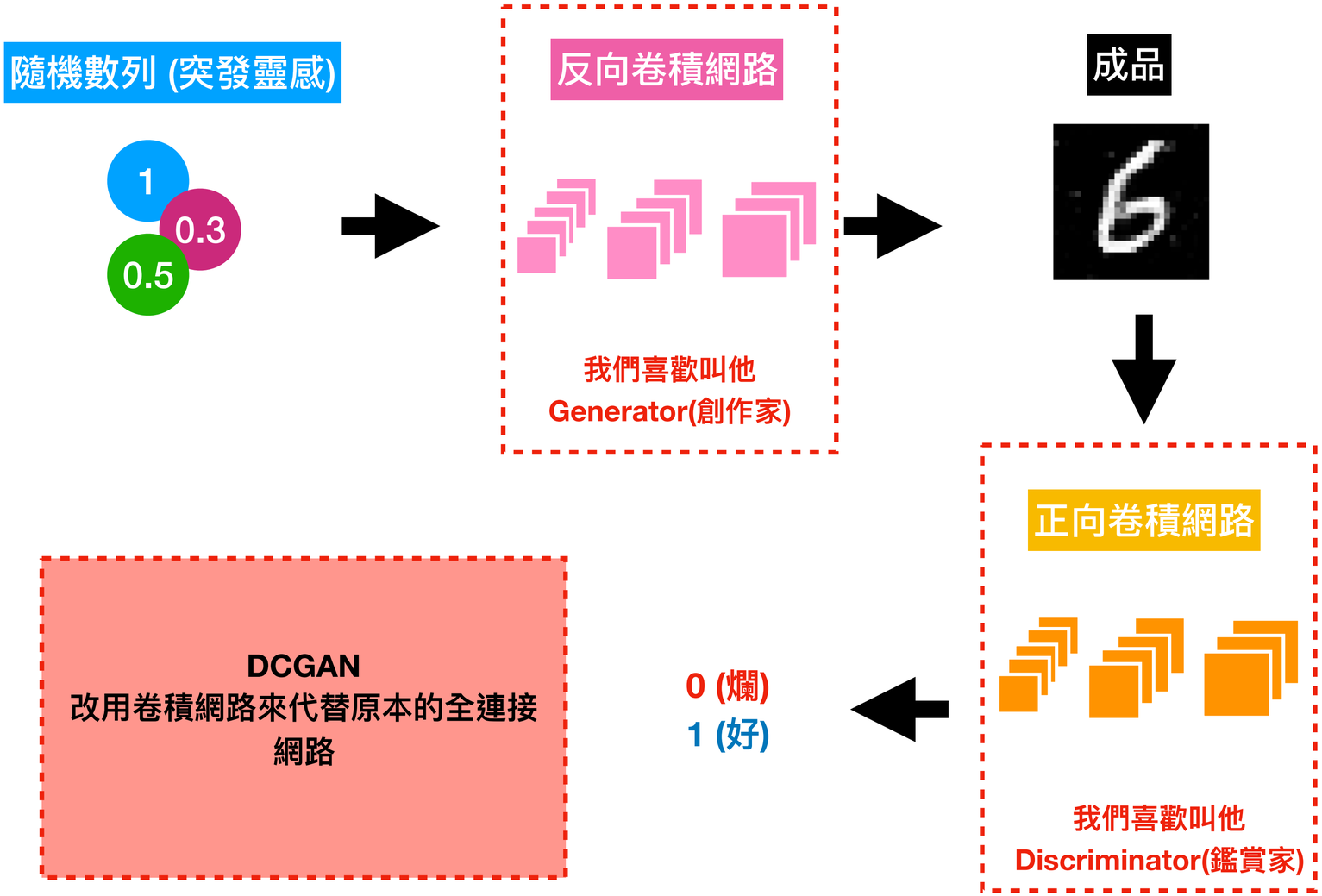
DCGAN

這是我們之前ＧＡＮ的成果，不知道你有沒有發現，旁邊很多雜點，原因是這些雜點並不影響 Discriminator 分辨真假，所以對於 Generator也不會用畢生力氣去抑制這些雜點

但也意味著你的 Generator 是以 像素(點) 的 Level 在生產，而不是抓住物品的 精髓 在生產

記得我們在 MLP 和 CNN 的時候也聊過這話題，CNN 比 MLP 厲害的地方就是他是真的抓住物品的精髓在判定東西，那能不能把 Discriminator 和 Generator 都換成 卷積網路 呢?

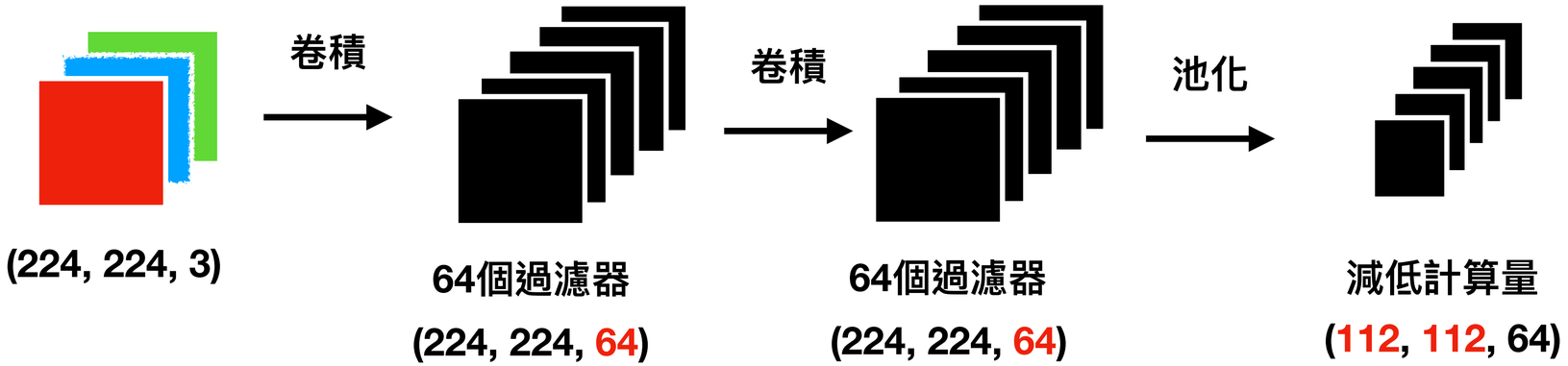
DCGAN


隨機數列 (突發靈感)
1
0.3
0.5
反向卷積網路
我們喜歡叫他
Generator(創作家)
成品
正向卷積網路
我們喜歡叫他
Discriminator(鑑賞家)
0 (爛)
1 (好)
DCGAN
改用卷積網路來代替原本的全連接網路

卷積小複習

VGG16為例

卷積：特徵變多
池化：寬高變小



反向卷積

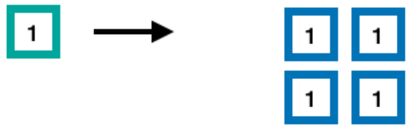
反向卷積(又稱轉置卷積)

反向卷積：特徵變少
反向池化：寬高變大

假設輸入
14 x 14 x 32

(2, 2)上採樣
28 x 28 x 32

16個filters的卷積
28 x 28 x 16

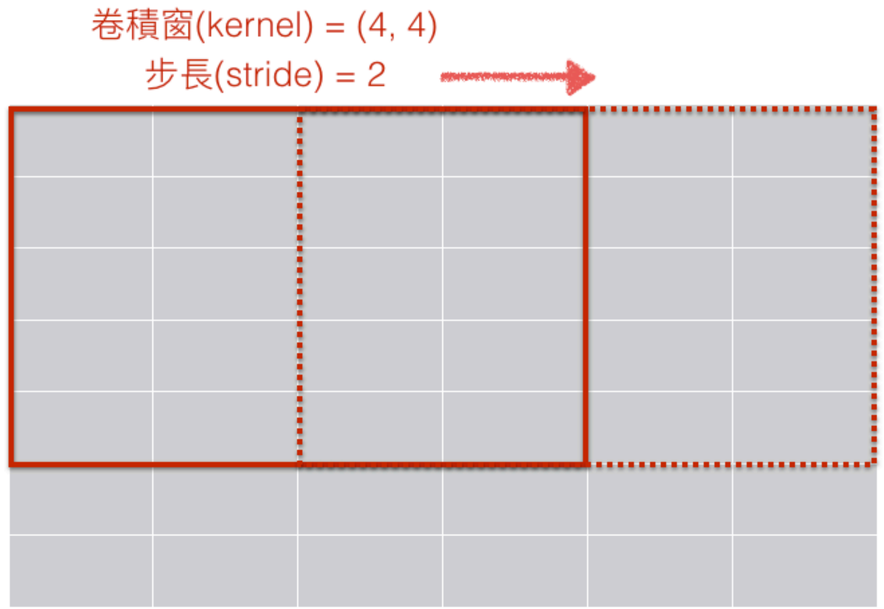


卷積: filters(特徵)增加
轉置卷積: filters(特徵)減少

卷積步長

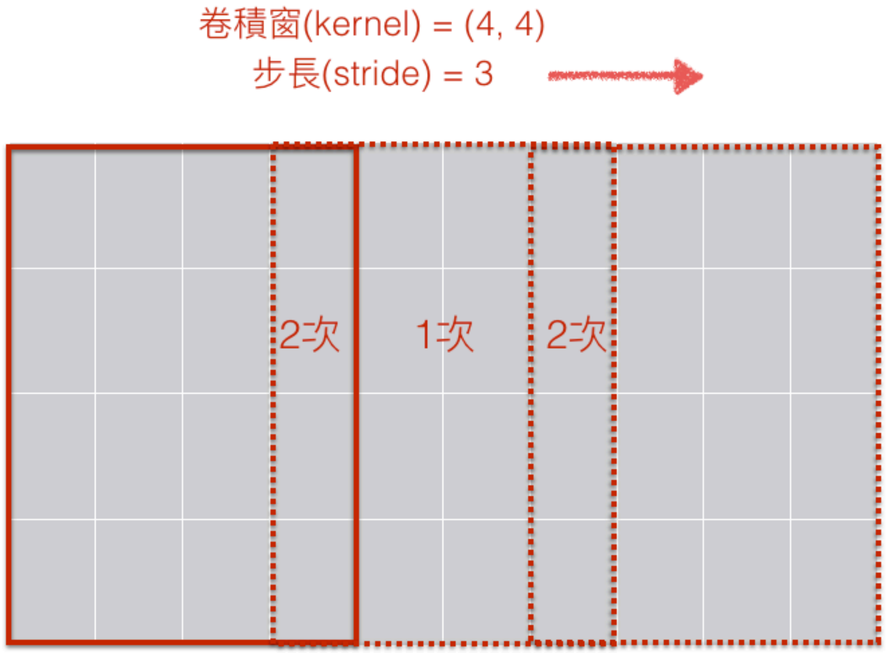
以前我們都靠著池化來縮小圖，但是你仔細想想，如果卷積的時候不要一格一格移，兩格兩格移不就可以達到池化的效果了嗎？

說的好，步長=2的卷積 其實就像 卷積 + 池化 的效果，但你發現我的卷積窗從以前常用的 (3, 3) 變成 (4, 4)了，為什麼呢？



步長 + 卷積窗

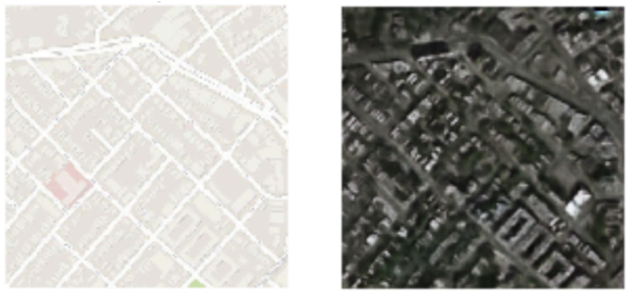
但是如果你的卷積窗大小要開始變得不太一樣了，你會發現如果你的卷積窗大小如果不是**步長的整數倍**的話，會發生有些格子你考慮了多次，但有些格子考慮比較少次，**下面就是一個不整除的例子**



圖像翻譯

圖像翻譯的領域是把某一張圖像轉成為另外一個條件下的圖像，譬如把冬天轉換成夏天，把白天轉換成夜晚，我們今天來做個例子：把Google地圖轉換成對應的地形圖

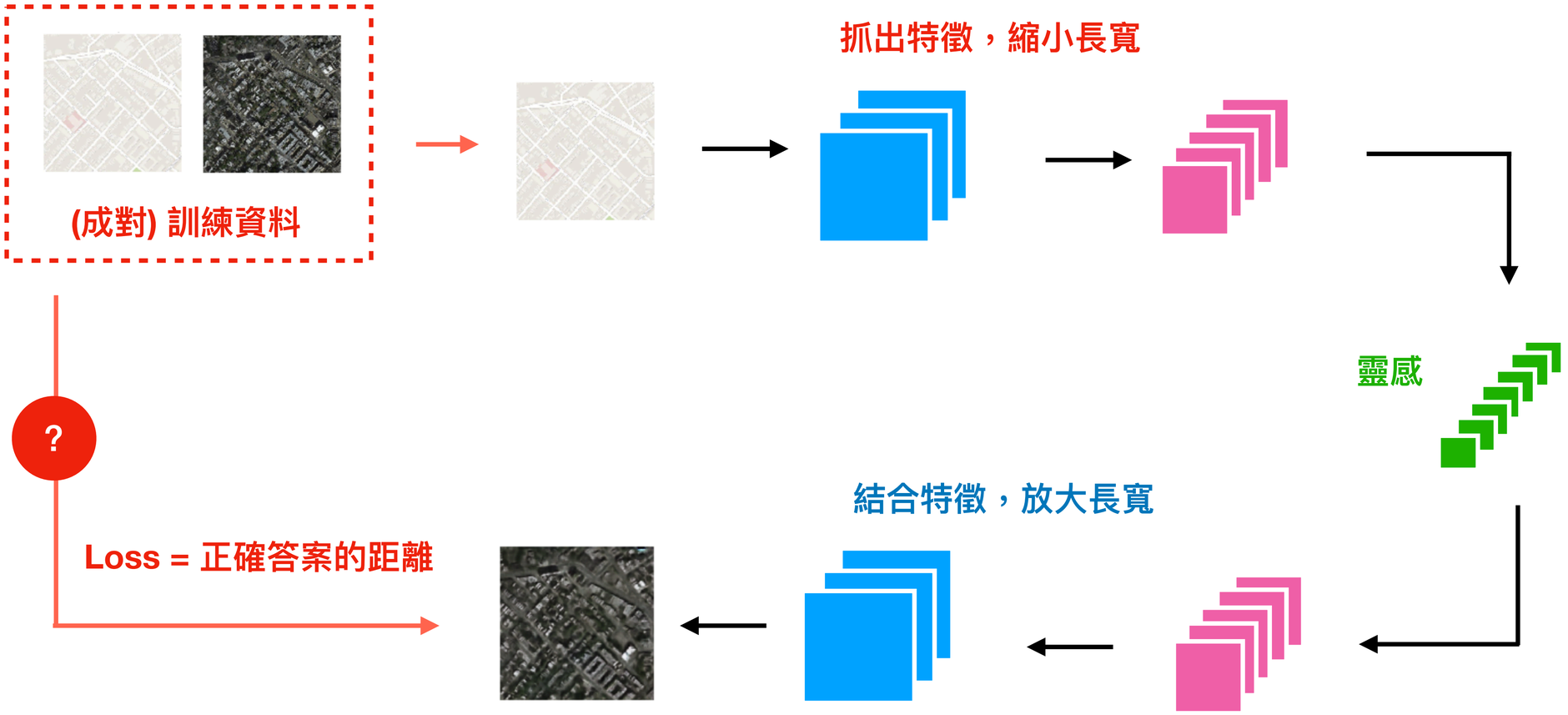
在圖像翻譯的主題上使用GAN，我們給他一個特別的名字 PIX2PIX



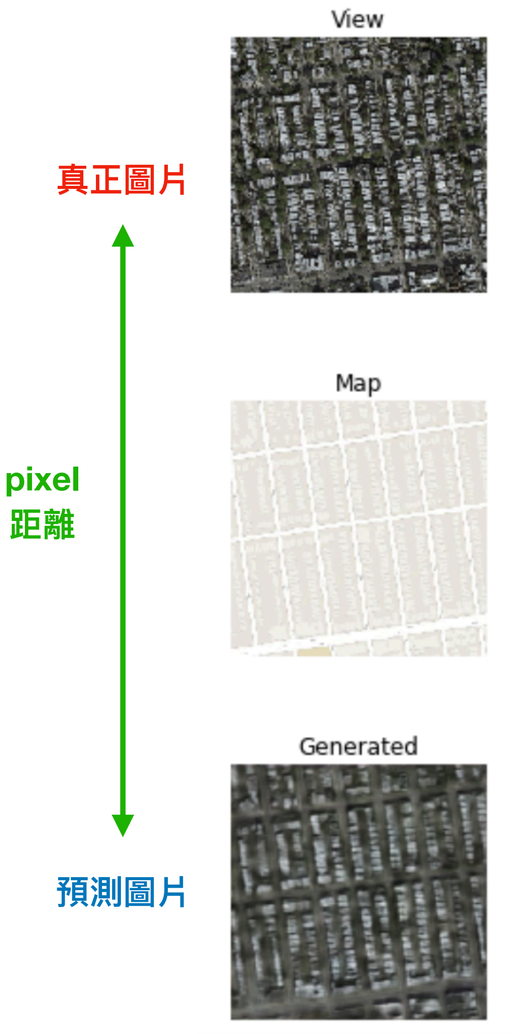
轉換成果:

原圖(左)　　轉換後(右)

希望模型



以前Loss



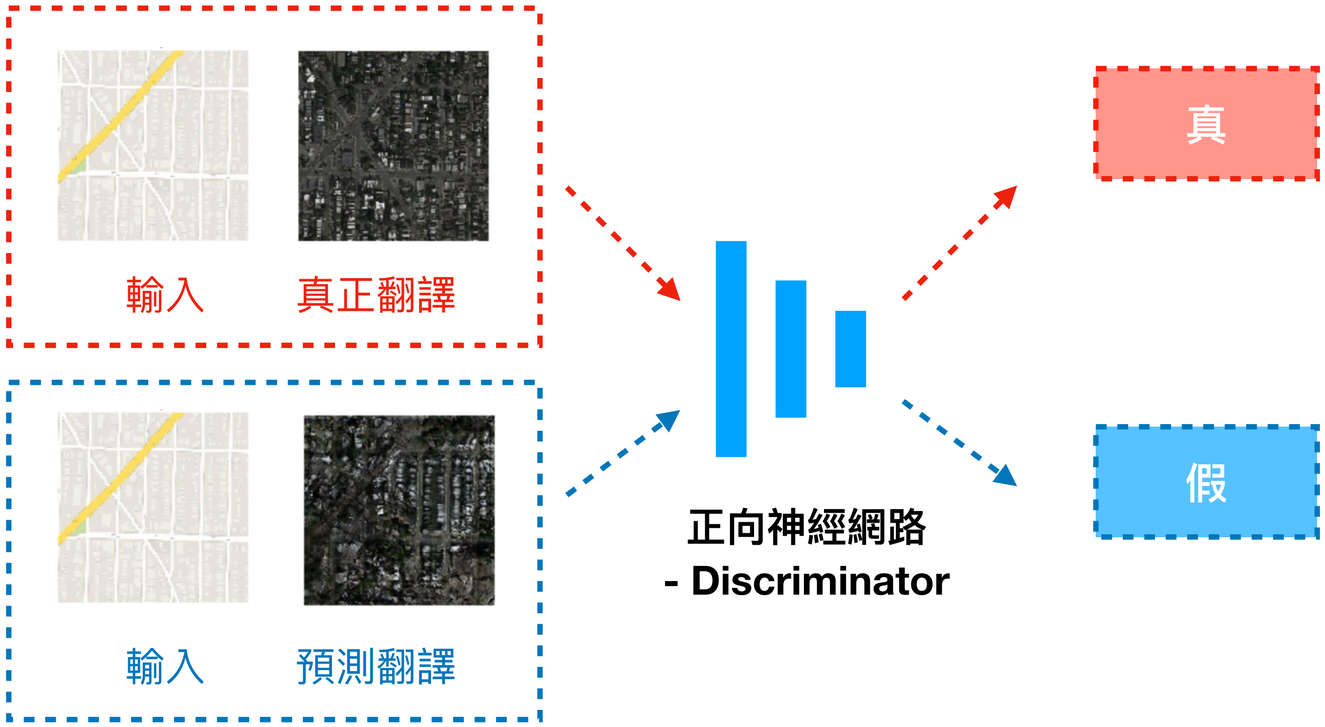
以前這類模型我們叫 Encoder-Decoder，
使用的 Loss 就是把真實的圖片和生產出來
的圖片一個一個 pixel 比較差異，這種 Loss
我們叫它 MSE Loss 或 MAE Loss

MSE Loss = (預測的pixel值 - 正確的pixel值) ^ 2

MAE Loss = | 預測的pixel值 - 正確的pixel值 |

使用 MSE Loss 或 MAE Loss 最大的問題
是：假設一個pixel對應到n種真實的草地，
但是我們要優化他啊，找到預測最小的誤
差，你會發現，最小的誤差就是n種的『平
均』，但你想像一下，把不同的草地的圖疊
一起做平均，對於人眼的感覺，就一個字
『模糊』！

GAN Loss



GAN 考慮的基礎是整張圖，而不是一個一個像素，所以你會發現遠看真的超像一個真品的，不過細節就不太行了！

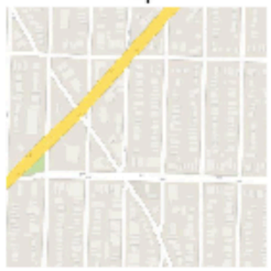
Loss 大比拼

GAN Loss

我們看出只用GAN Loss的模型的色調是非常像原圖的！甚至是無所不用其極地讓整張圖接近原圖，但細節就完全不行了！

View

Map



Generated

傳統(MAE) Loss

傳統Loss就如同我們所說，因為太過關注一個一個pixel，所以導致雖然細節很棒，但過度模糊

View

Map

Generated

GAN Loss + MAE Loss

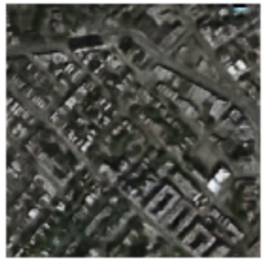
那把兩種方式混在一起就好了！！！你會發現雖然還是有點模糊，但整張圖看起來非常像是真實圖片

勝

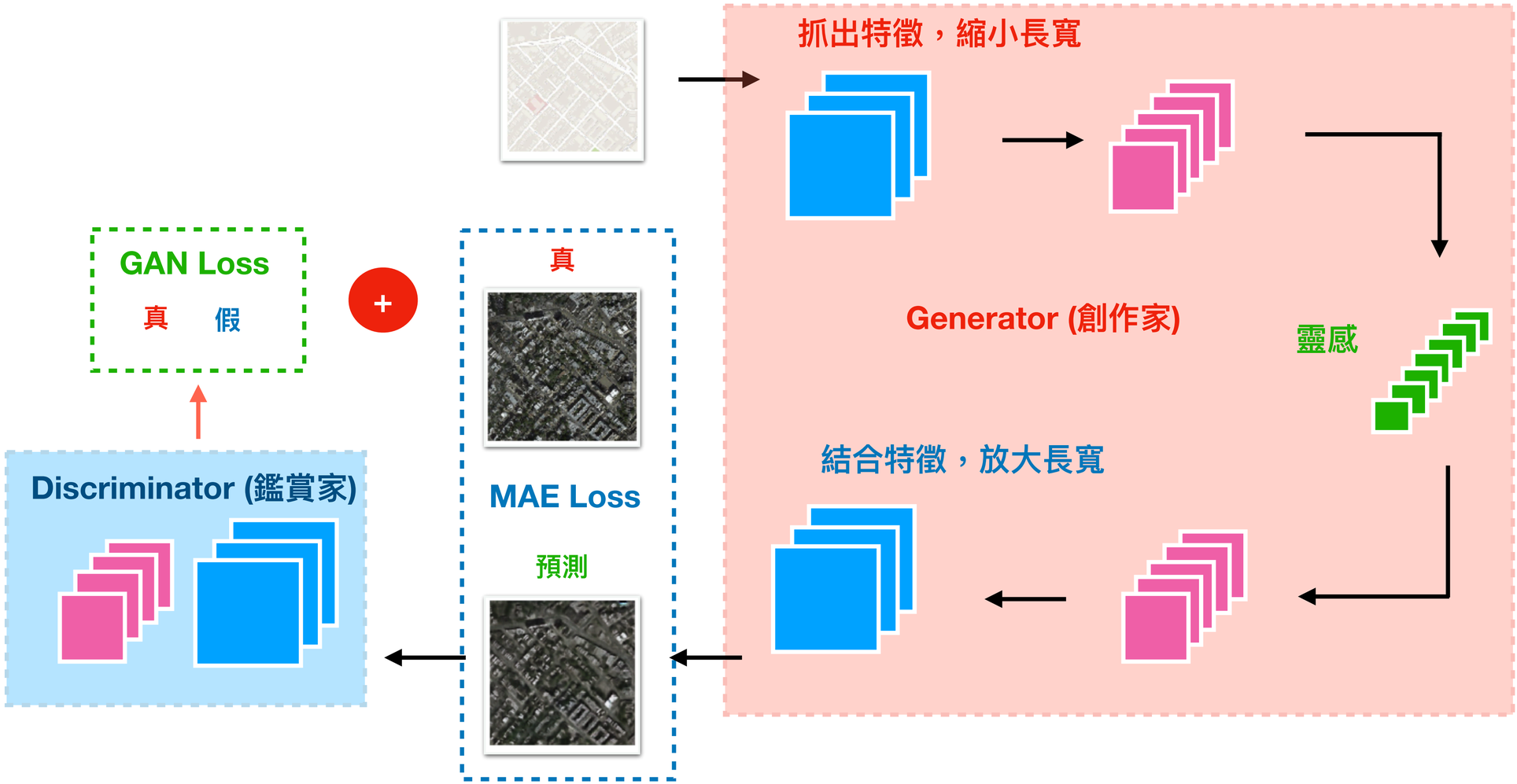
View

Map

Generated



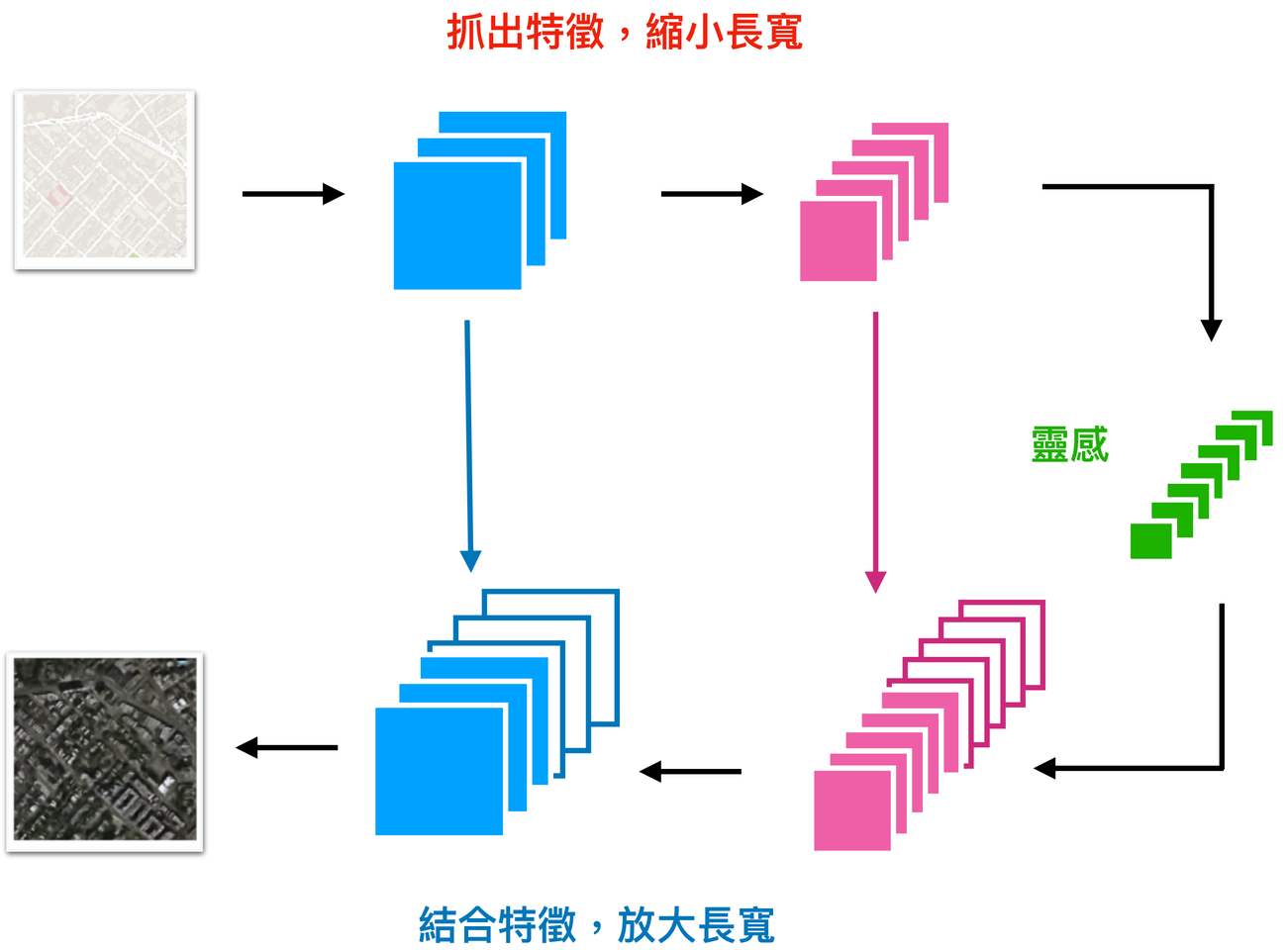
完整模型


抓出特徵，縮小長寬
Generator (創作家)
靈感
結合特徵，放大長寬
真
MAE Loss
預測
GAN Loss
真 假
+
Discriminator (鑑賞家)

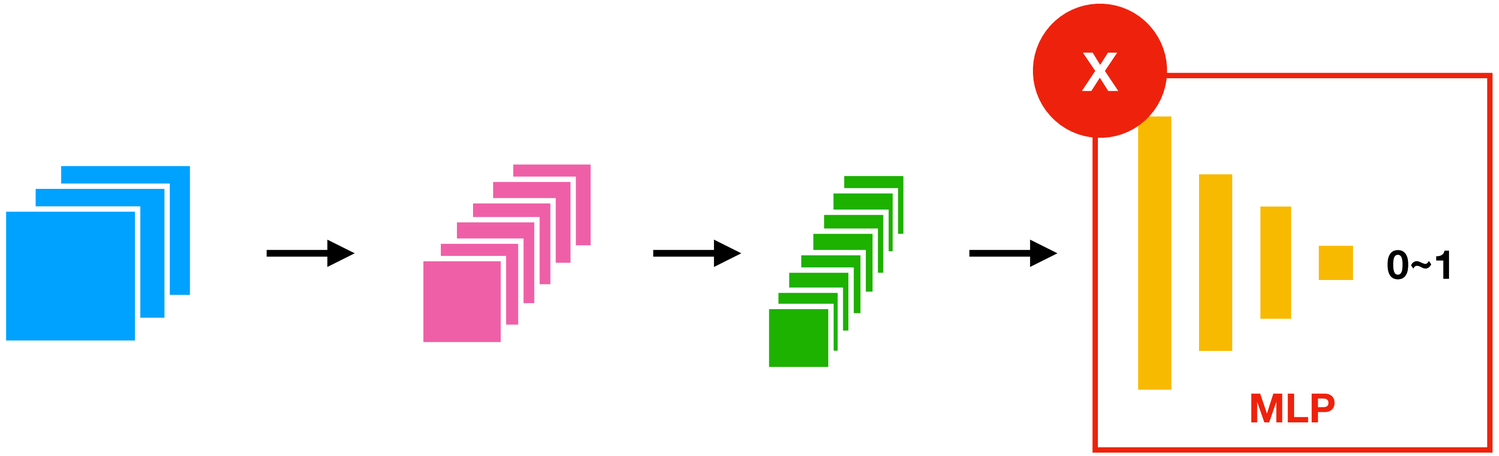
創作家 Trick

我們在創作的過程發現一個現象，如果任由創作家從靈感創作，有可能創作出跟原圖差距非常多的圖

所以我們採取了一個方式，把前面抓出的原圖特徵當成一個條件加到後面，強迫靈感在變成作圖的時候要參照原圖條件，這方式我們叫做 U-Net



鑑賞家 Trick



我們如果只再和以前一樣，針對整張圖輸出真假，就有可能會造成局部的不真，於是我們改成用 每一個區域 給出一個分數的方式來打分

事實上就是把 MLP 拿掉，因為卷積後的一個位置就是前面一個區域的給分！這個我們叫做 FCN(全卷積)，利用在 GAN 上我們會叫做 PatchGAN，針對一個Patch(區域) 打分

感受野

提到 全卷積(FCN) 或者 PatchGAN 就得提到感受野(Receptive Field)，也就是一個區域的『真/假』到底對應到原圖的多少區域，其實就是所謂的『視野』啦下面我們以 (4, 4) 卷積窗 以及 步長 2 的卷積來說

感受野公式: 4 + (邊長 - 1) * 2

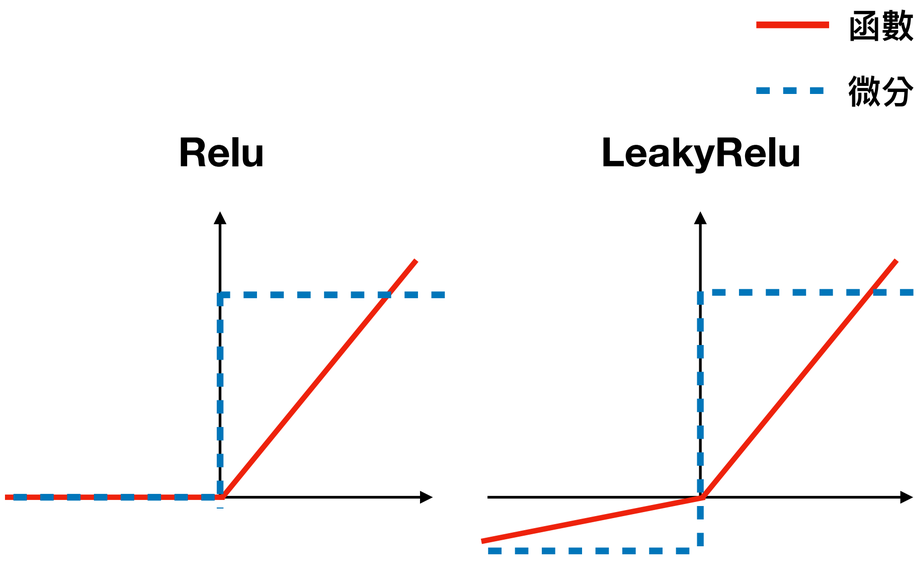
層數	感受野大小	公式
1	4	
2	10	4 + (4 - 1) * 2
3	22	4 + (10 - 1) * 2
4	46	4 + (22 - 1) * 2
5	94	4 + (46 - 1) * 2

以256的圖片大小，我們通常選四到五層即可！

GAN Hacks

在這種複雜題目的時候，我們需要面面俱到，步步優化，才有機會
生成優秀的圖片，老前輩們總結出了一些小小的優化經驗

Hack 1. Relu 可以換成 LeakyRelu



還記得我們是在算 梯度: Loss 對 W 的微分 吧
總體梯度 = 第一層微分 * 激活微分 * ...
所以Relu最大的問題是在『非激活區』微分是0，所以總體梯度 = 0，也就是不更新的意思，更慘的是，如果這個神經元總是落入『不更新區』，那就意味著他永遠沒機會調整成正確的參數，我們稱之為『神經元死去』

Leaky Relu的改進就在，『非激活區』我們會給出『負斜率』，所以就算你進入『非激活區』也會更新參數，至於斜率經過實驗後，0.2 的負斜率表現得不錯！

Hack 2. Adam參數設置

Adam [source]

lr = 0.0002　　beta_1 = 0.5

```
keras.optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=N
```

lr (學習速率)

實驗結果：比較小的學習速率有助於兩個深度網路的平衡，實驗後我們通常會設定成0.0002

beta 1(速度剩餘)

實驗結果：Adam有累積速度，但是速度累積的時候當然要考慮『摩擦力』，新速度 = beta_1 * 舊原有速度 + (1 - beta_1) * 新瞬時速度，我們會想要設定成0.5，你可以當成前一秒的速度會因為摩擦力在這一秒只剩下 50%

不對稱圖像翻譯

我們在之前已經學會了 PIX2PIX，利用 GAN 讓圖片能夠翻譯成另外的圖片，但你有沒有發現我們的訓練資料都是『成對』出現，因為如果沒『成對』出現的話，我們無法 對答案 (傳統loss部分)！

但現實生活我們有時候不太可能找到『成對』的翻譯，譬如：我要做出 馬 <-> 斑馬 的翻譯 或者 風景 <-> 油畫 的翻譯，我們不可能找出一樣位置的馬和斑馬或者跟風景一模一樣的油畫，那該怎麼辦呢？ 讓 CycleGAN 教你如何做！

天才一般的想法

! 假設我的輸入是馬，我想要轉換成斑馬，但我又沒有對稱圖片對答案

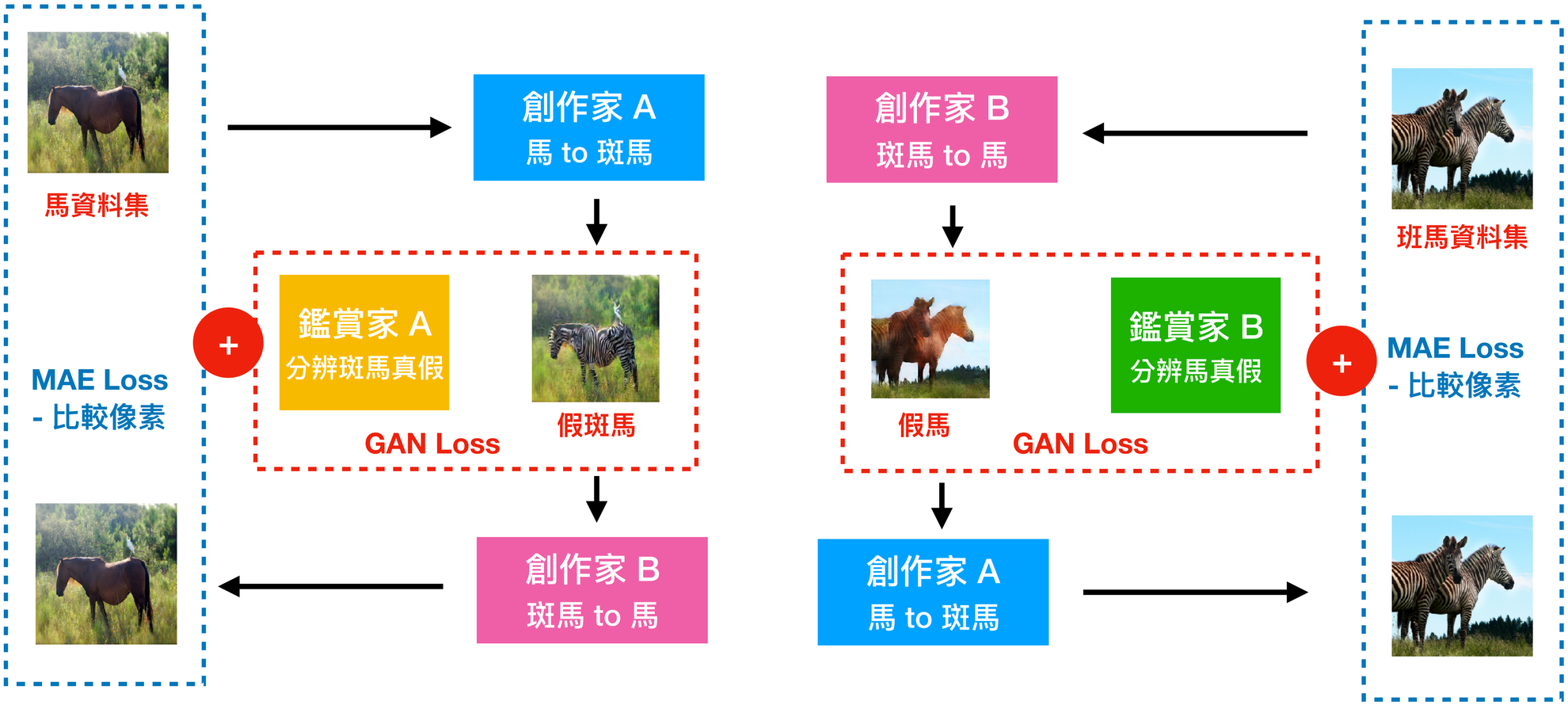
! 我難道不能用另外一套 Generator 轉換回 馬，來對答案嗎！

! 天才！這就是CycleGAN的精髓，運用兩套PIX2PIX組合起來對答案

轉換前

轉換後

完整模型


馬資料集
MAE Loss
- 比較像素
創作家 A
馬 to 斑馬
鑑賞家 A
分辨斑馬真假
假斑馬
GAN Loss
創作家 B
斑馬 to 馬
創作家 B
斑馬 to 馬
鑑賞家 B
分辨馬真假
假馬
GAN Loss
創作家 A
馬 to 斑馬
斑馬資料集
MAE Loss
- 比較像素