

手把手教你深度學習實務

張鈞閔

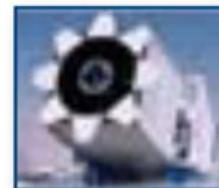
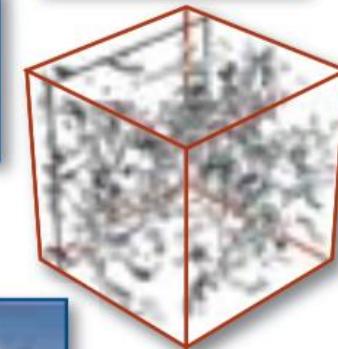
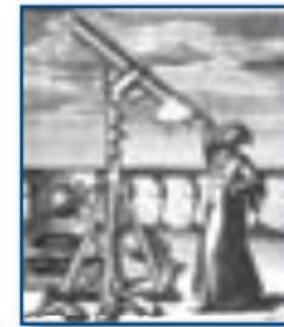
中央研究院資訊科學研究所資料洞察實驗室



Science Paradigms

- Thousand years ago:
science was empirical
describing natural phenomena
- Last few hundred years:
theoretical branch
using models, generalizations
- Last few decades:
a computational branch
simulating complex phenomena
- Today: **data exploration (eScience)**
unify theory, experiment, and simulation
 - Data captured by instruments
or generated by simulator
 - Processed by software
 - Information/knowledge stored in computer
 - Scientist analyzes database/files
using data management and statistics

$$\left(\frac{\dot{a}}{a}\right)^2 = \frac{4\pi G p}{3} - K \frac{c^2}{a^2}$$



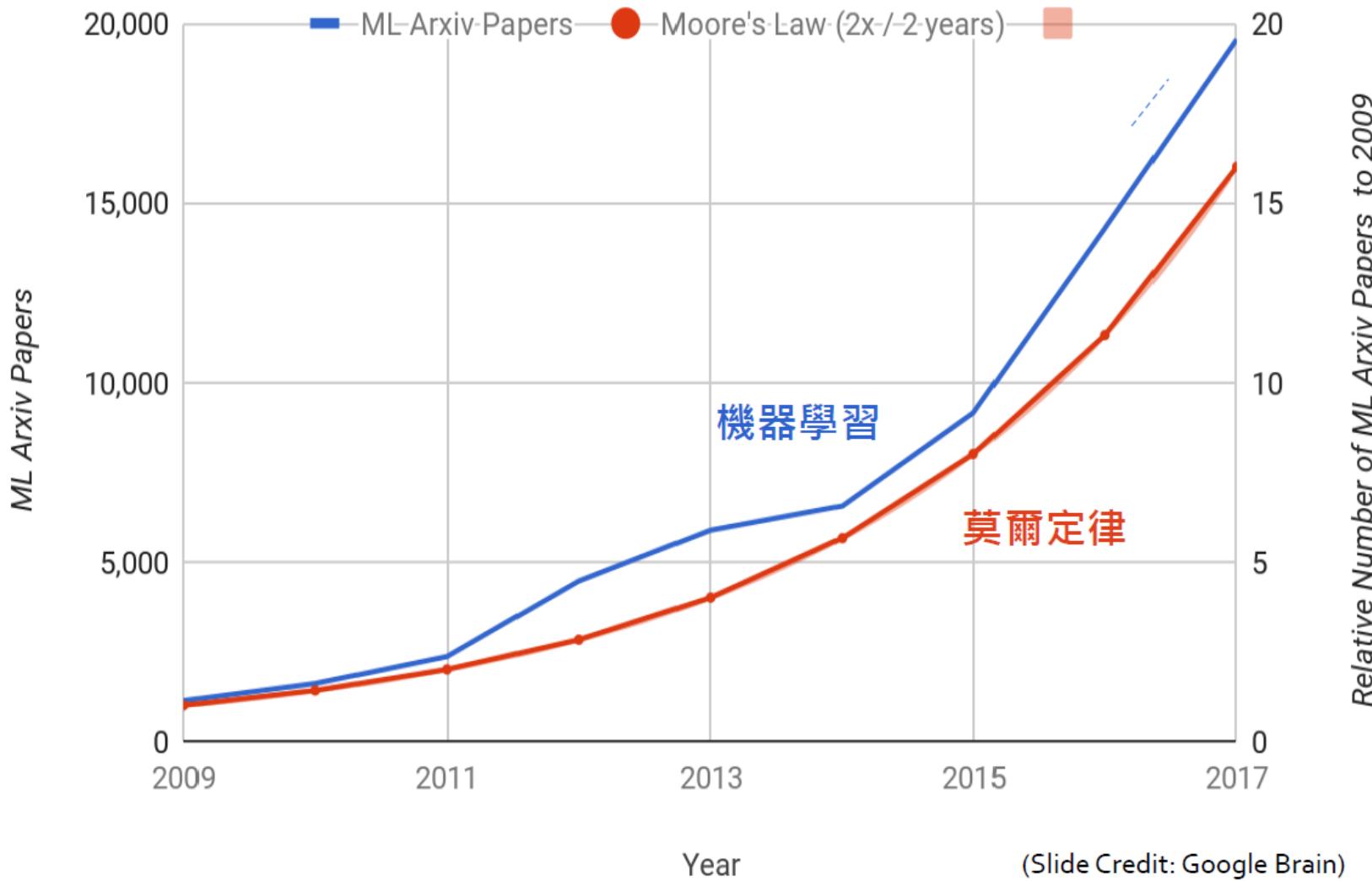


Outline

- What is Deep Learning?
- Hands-on Tutorial of Deep Learning
- Tips for Training DL Models
- Variants - Convolutional Neural Network
- Transfer Learning and Semi-supervised Learning



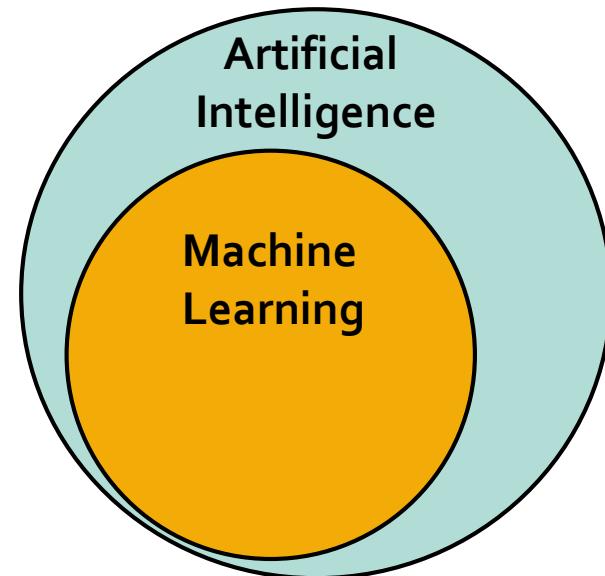
機器學習論文成長曲線



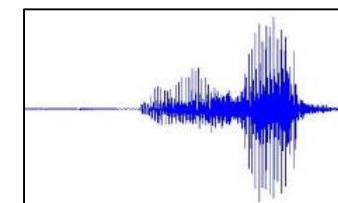
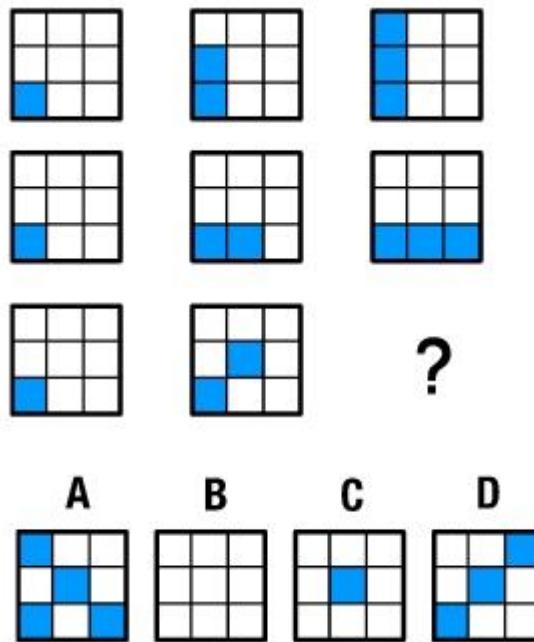
(Slide Credit: Google Brain)

Machine Learning vs Artificial Intelligence

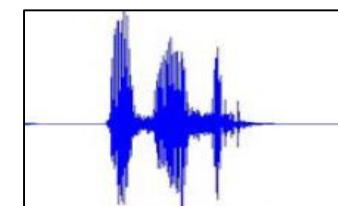
- AI is the simulation of human intelligence processes
 - Outcome-based: 從結果來看，是否有 human intelligence
 - 一個擁有非常詳盡的 rule-based 系統也可以是 AI
- Machine learning 是達成 AI 的一種技術
- 從資料當中學習出 rules
 - 找到一個夠好的 function 能解決特定的問題



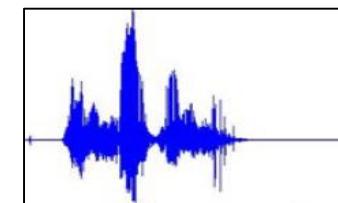
讓機器從資料裡淬取出規則的 演算法



你好



大家好



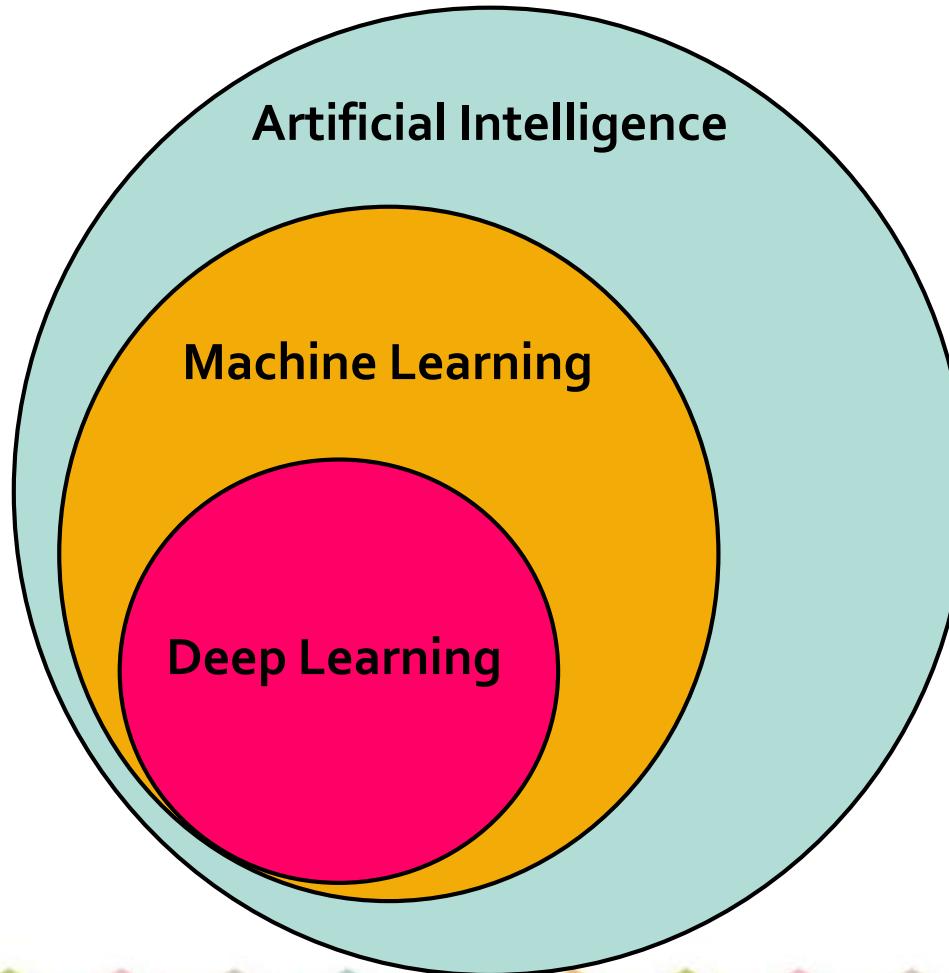
人帥真好

Slide credit: [Sheng-Wei Chen](#)



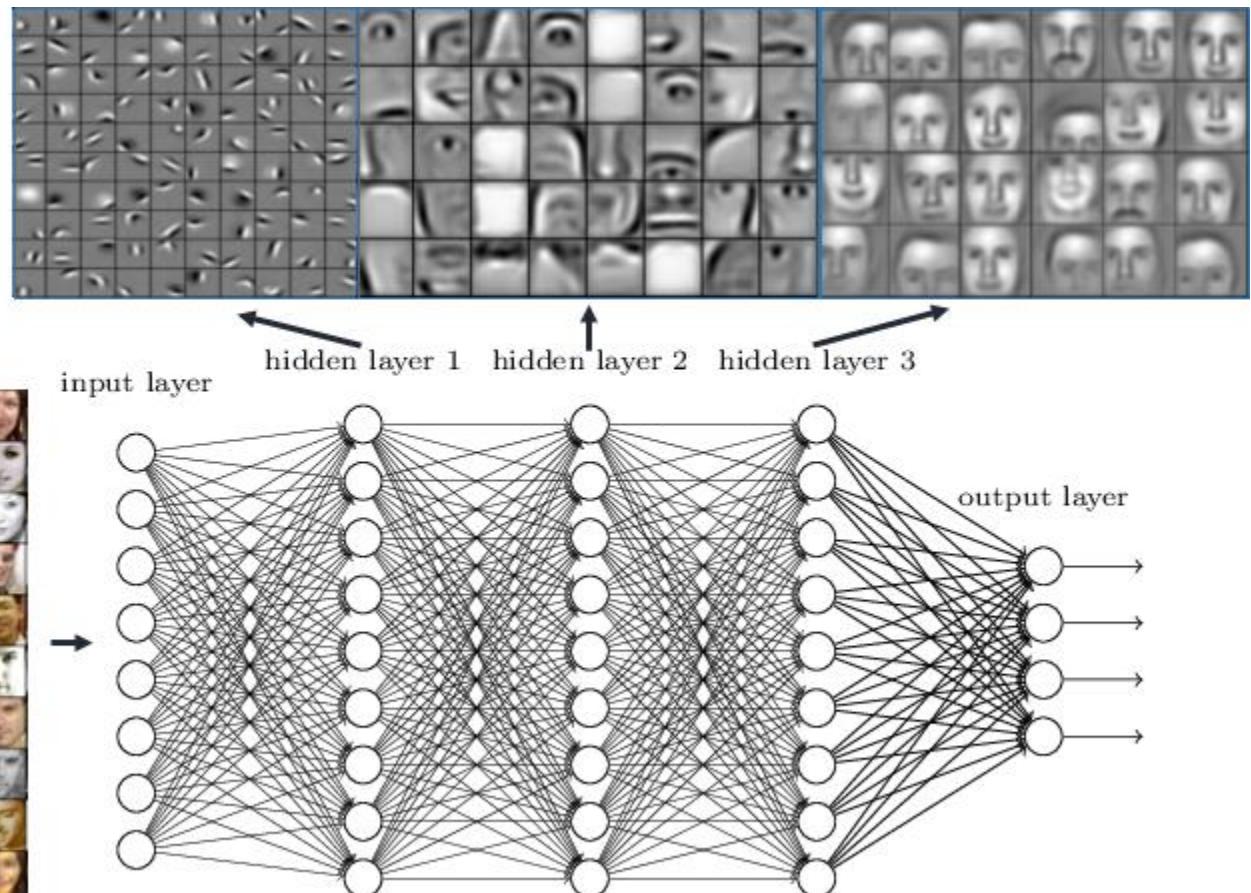
Deep Learning vs Machine Learning

- Deep learning is a subset of machine learning



機器學習算法學習到的複雜規則

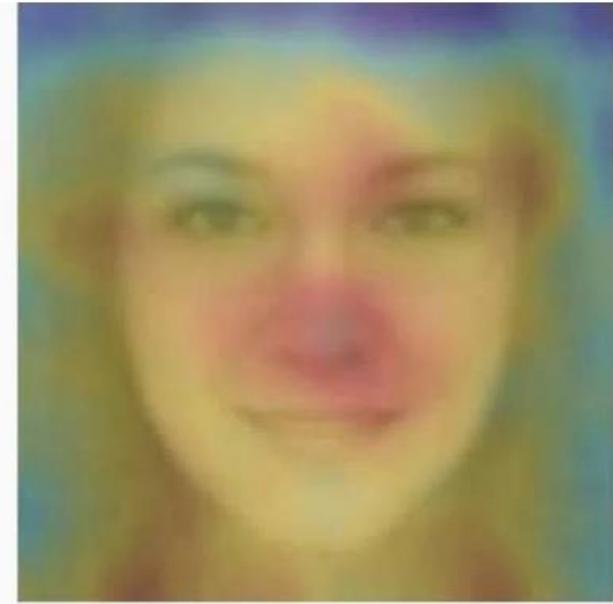
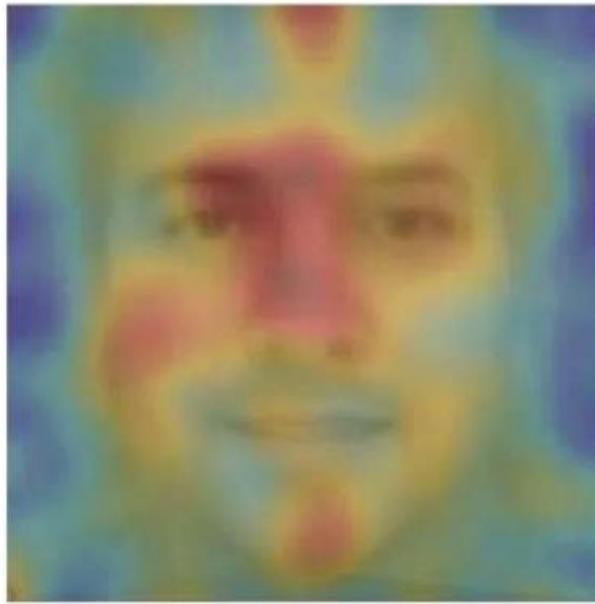
Deep neural networks learn hierarchical feature representations



Ref: [Deep Learning and Convolutional Neural Networks: RSIP Vision Blogs](#)

機器學習算法得到的特徵熱點

- Gaydar: detect signs of homosexuality
- Human judges: 61% for men, 54% for women
- AI judges: 91% for men, 83% for women

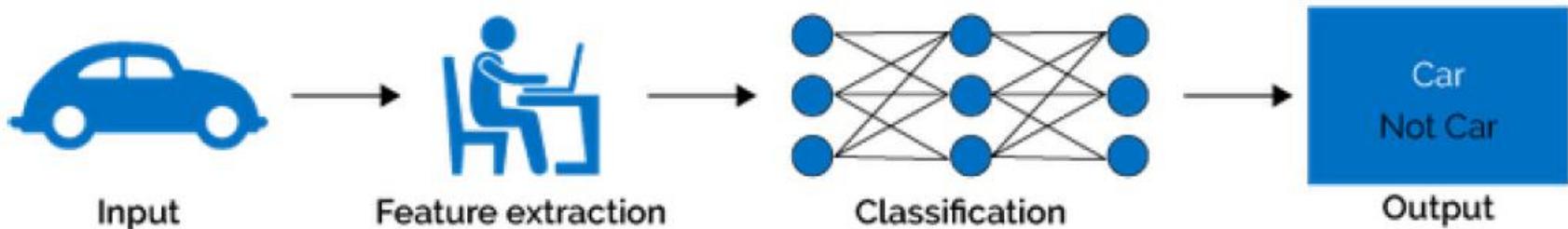


<https://osf.io/zn79k/>

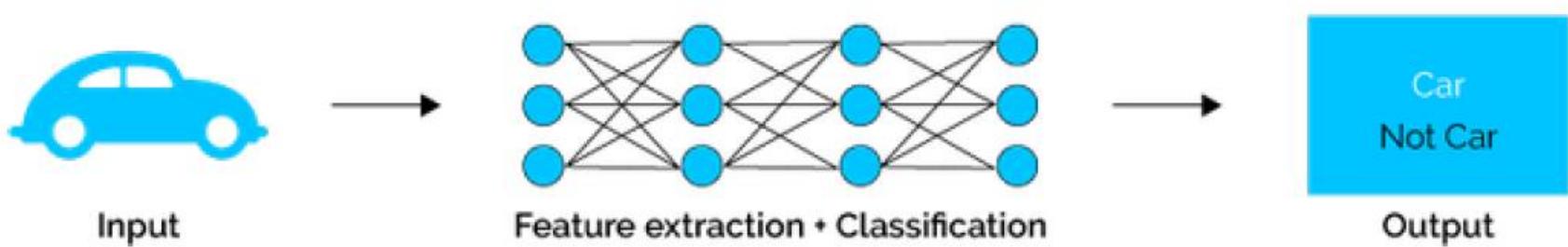
Rule-based System



Classical Machine Learning



Deep Learning



Slide credit: [Sheng-Wei Chen](#)

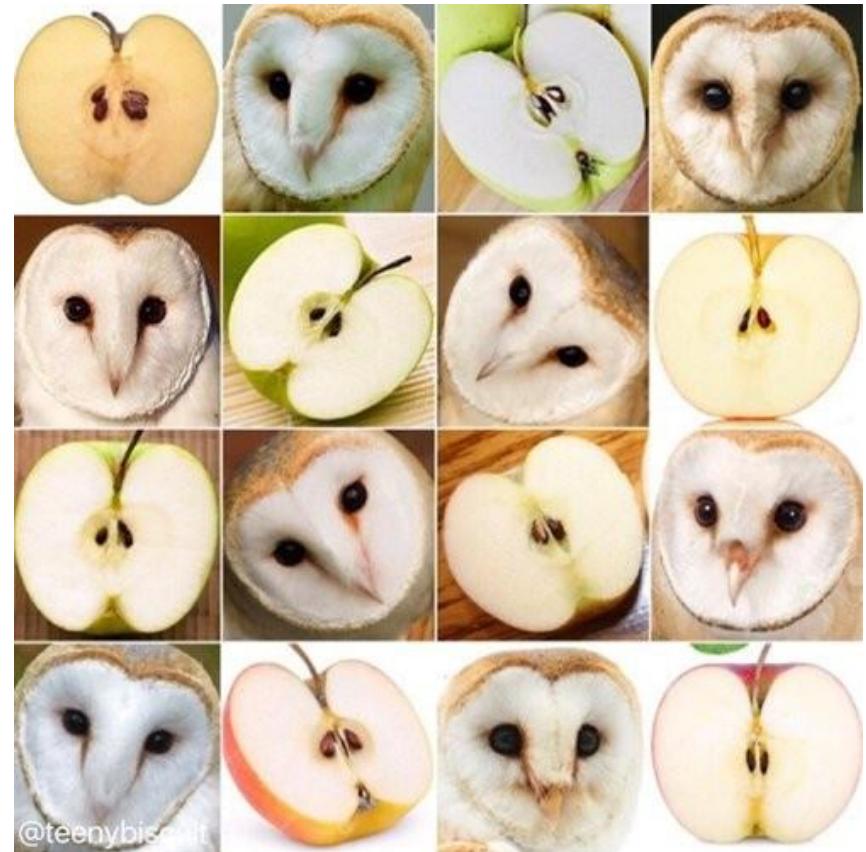


圖像辨識的進步

吉娃娃與馬芬



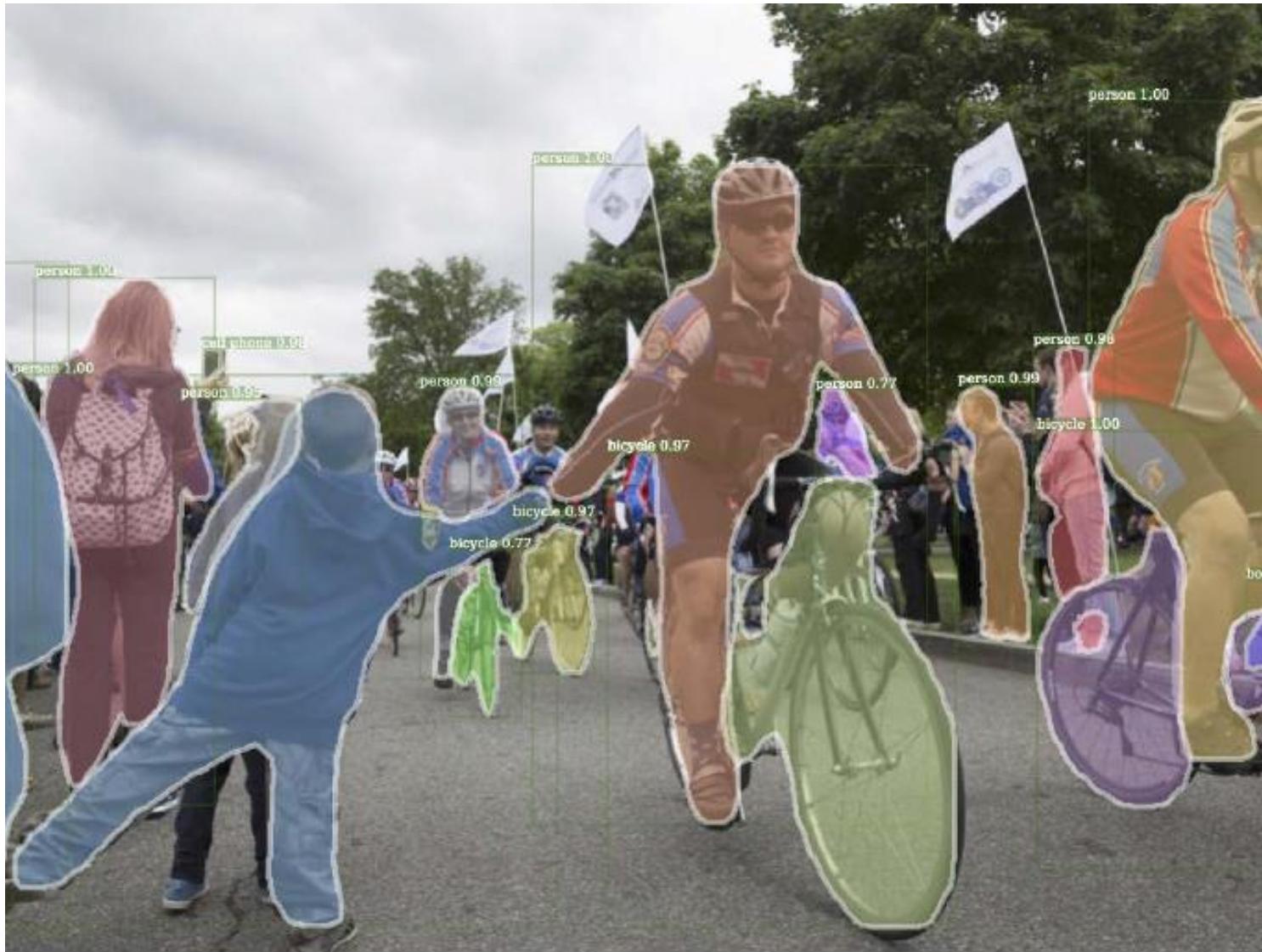
貓頭鷹與蘋果



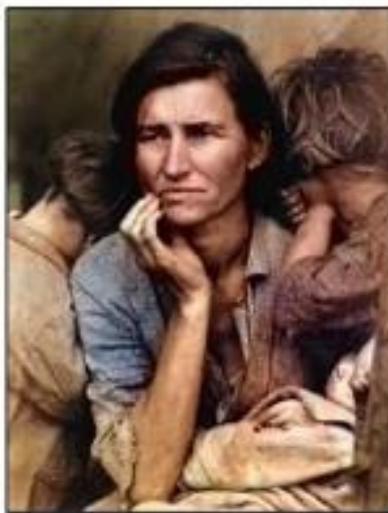
Source: teenybiscuit - twitter



Semantic Segmentation

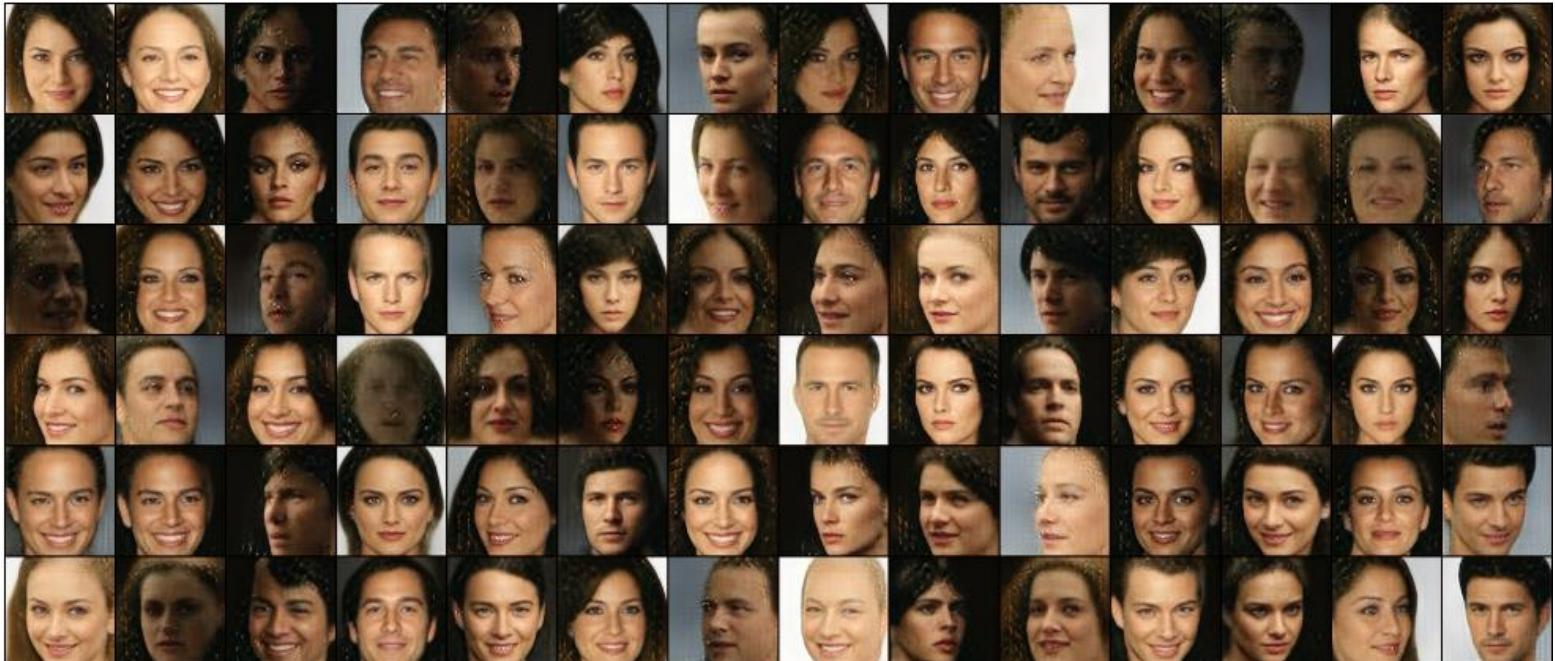


Colorizing Legacy Photos



其他有趣的應用

□ Image Generation



<https://...>

Video Captioning



Video sequence



Answer: a woman is carefully slicing tofu.

Generated caption: a woman is cutting a block of tofu.

Text-To-Image

the flower has petals that
are bright pinkish purple
with white stigma



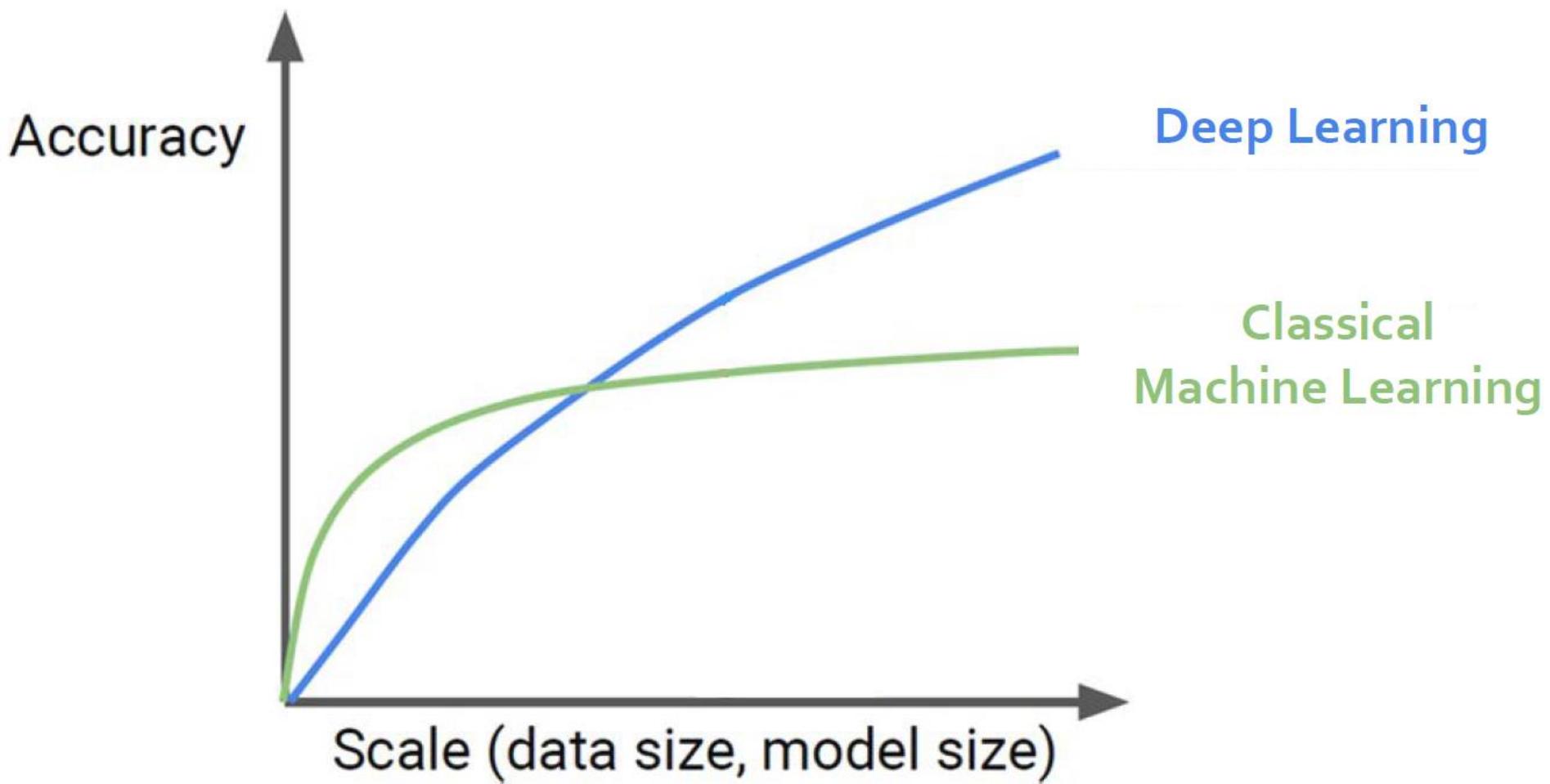
this white and yellow flower
have thin white petals and a
round yellow stamen



<https://arxiv.org/pdf/1701.00160.pdf>



There is no free lunch



Slide credit: [Sheng-Wei Chen](#)



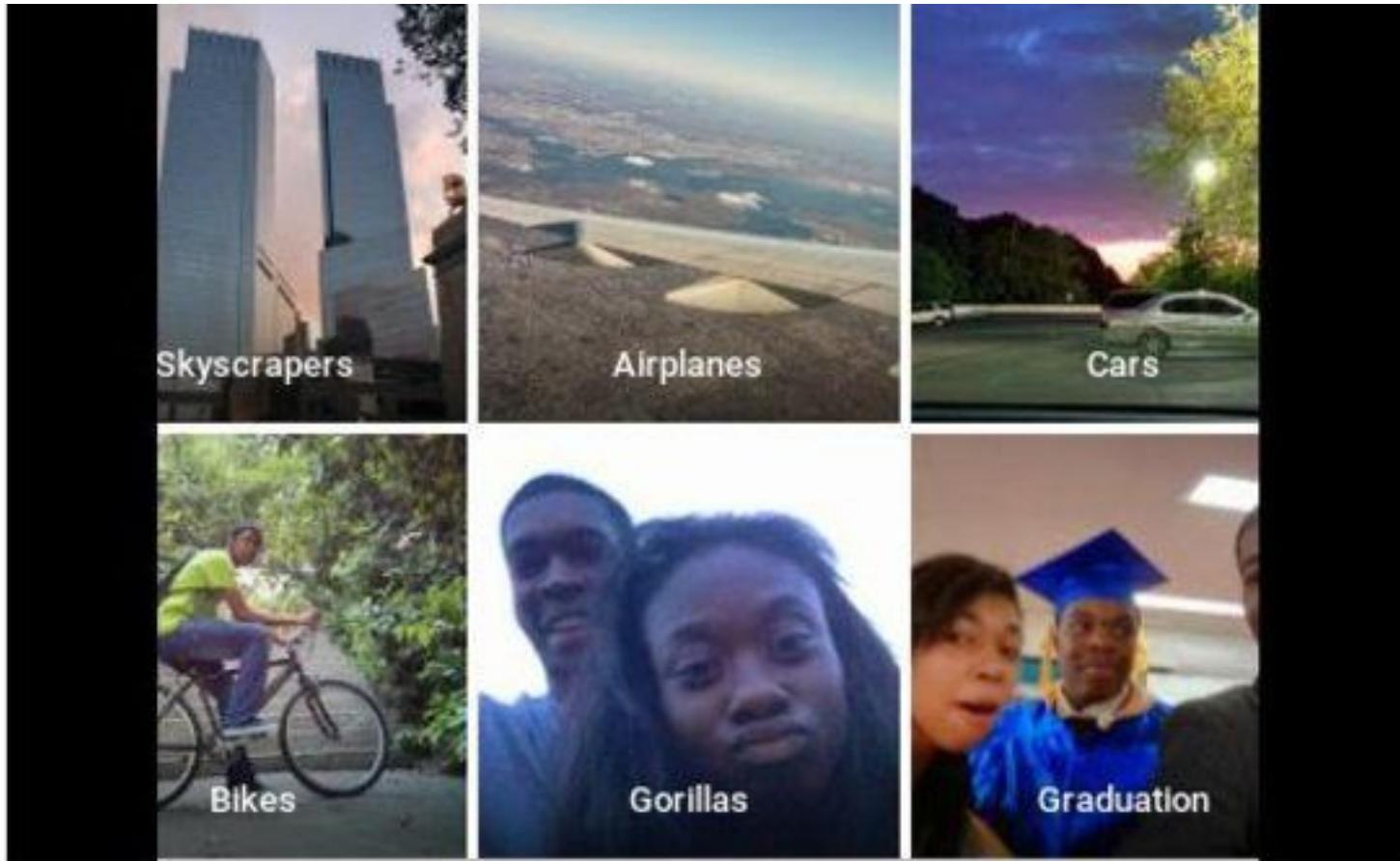


One year on this
planet is 7 years on
Earth

Great!
I will train my model
here

Slide credit: [Sheng-Wei Chen](#)

Machines may make mistakes as well



diri noir avec banan @jackyalcine · Jun 29

Google Photos, y'all [REDACTED] My friend's not a gorilla.



813



394



...

TWITTER



Deep Learning 的優勢

□ 學習非結構化資料

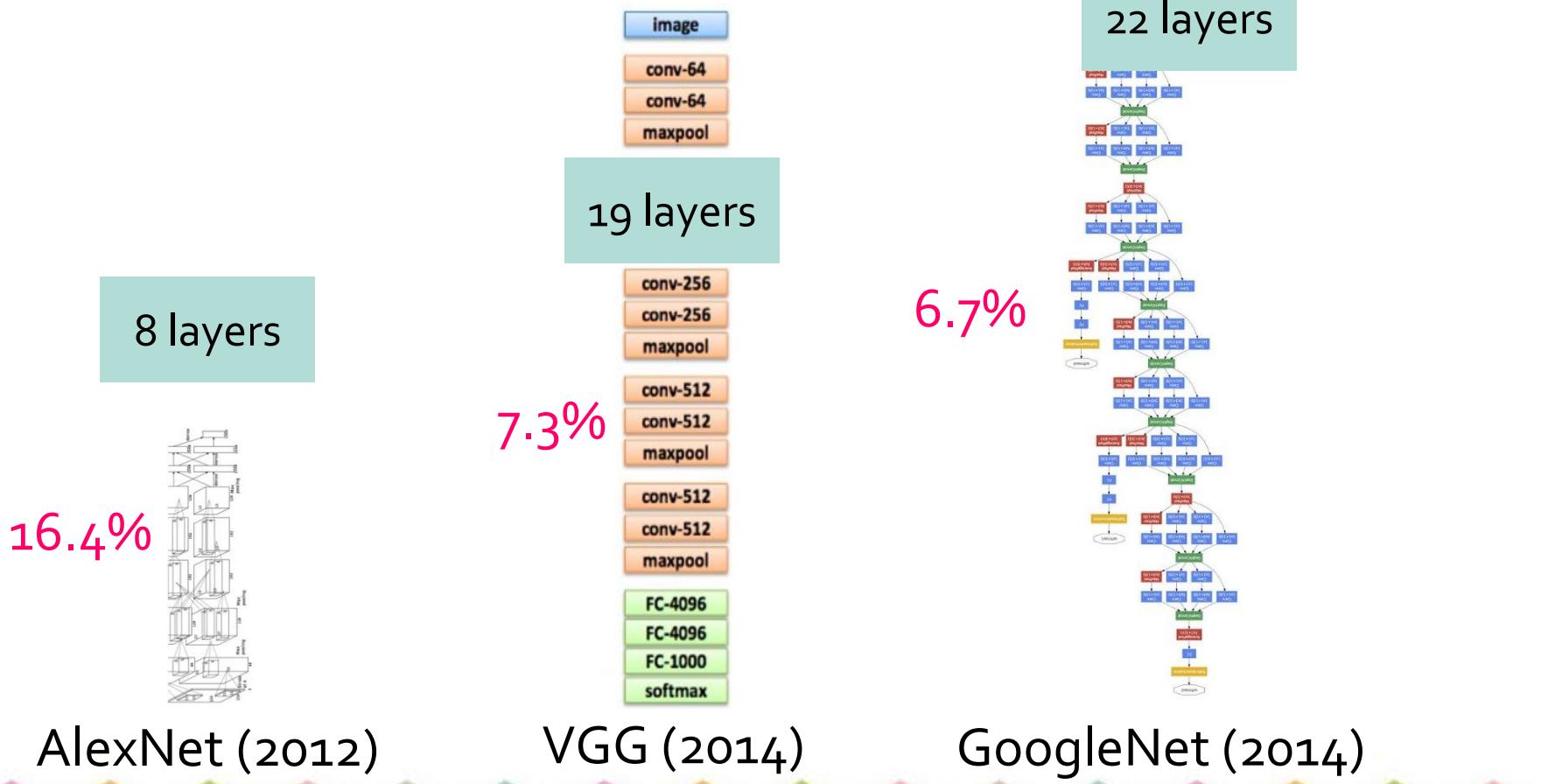
- 圖片 → Convolutional neural network (CNN)
(2D) → Residual neural network
- 序列 → Recurrent neural network (RNN)
→ Long short-term memory (LSTM)
→ Gated recurrent units (GRU)

□ 學習特徵表示 representation learning

- 取代過去的特徵工程 feature engineering
- Learning from data (O), learning from human (X)

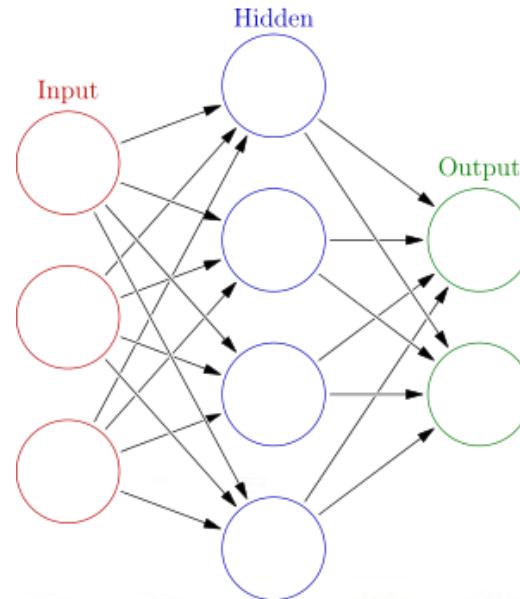
為什麼叫做 Deep Learning ?

- 當 hidden layers 層數夠多
就稱為 Deep neural network

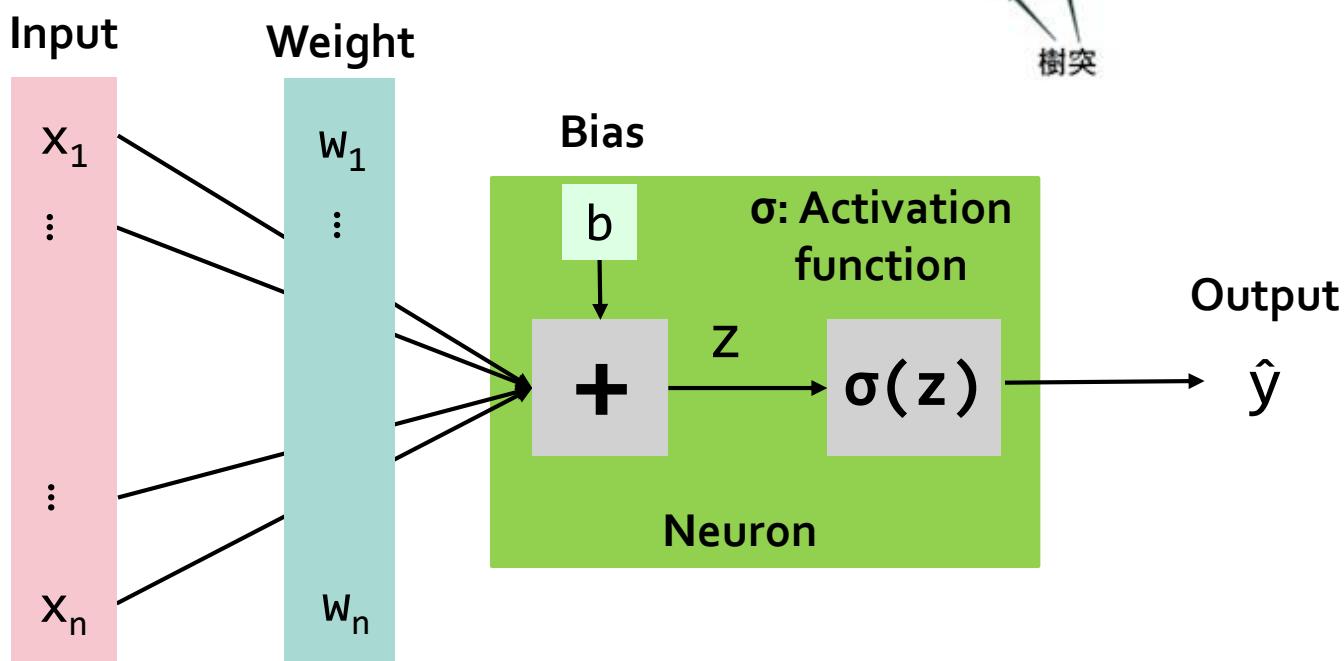


Fundamentals of Deep Learning

- Artificial neural network (ANN, 1943)
Multi-layer perceptron
- 模擬人類神經傳導機制的設計
 - 由許多層的 neurons 互相連結而形成 neural network

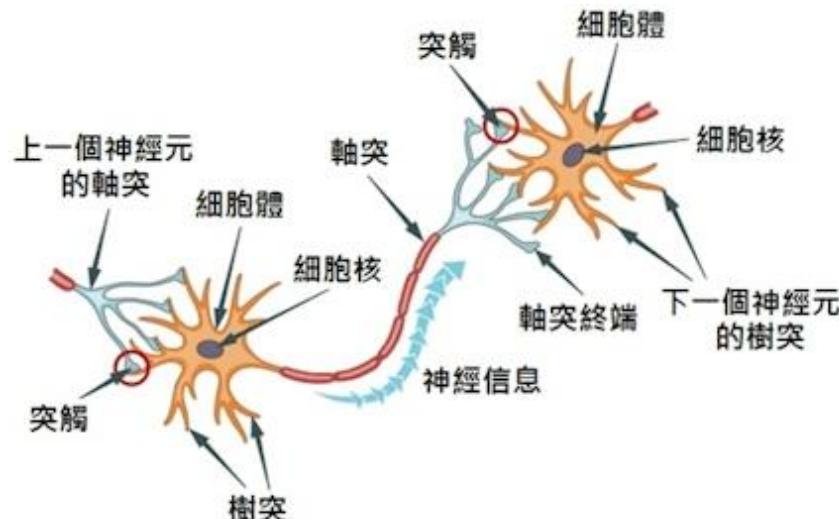


A Neuron



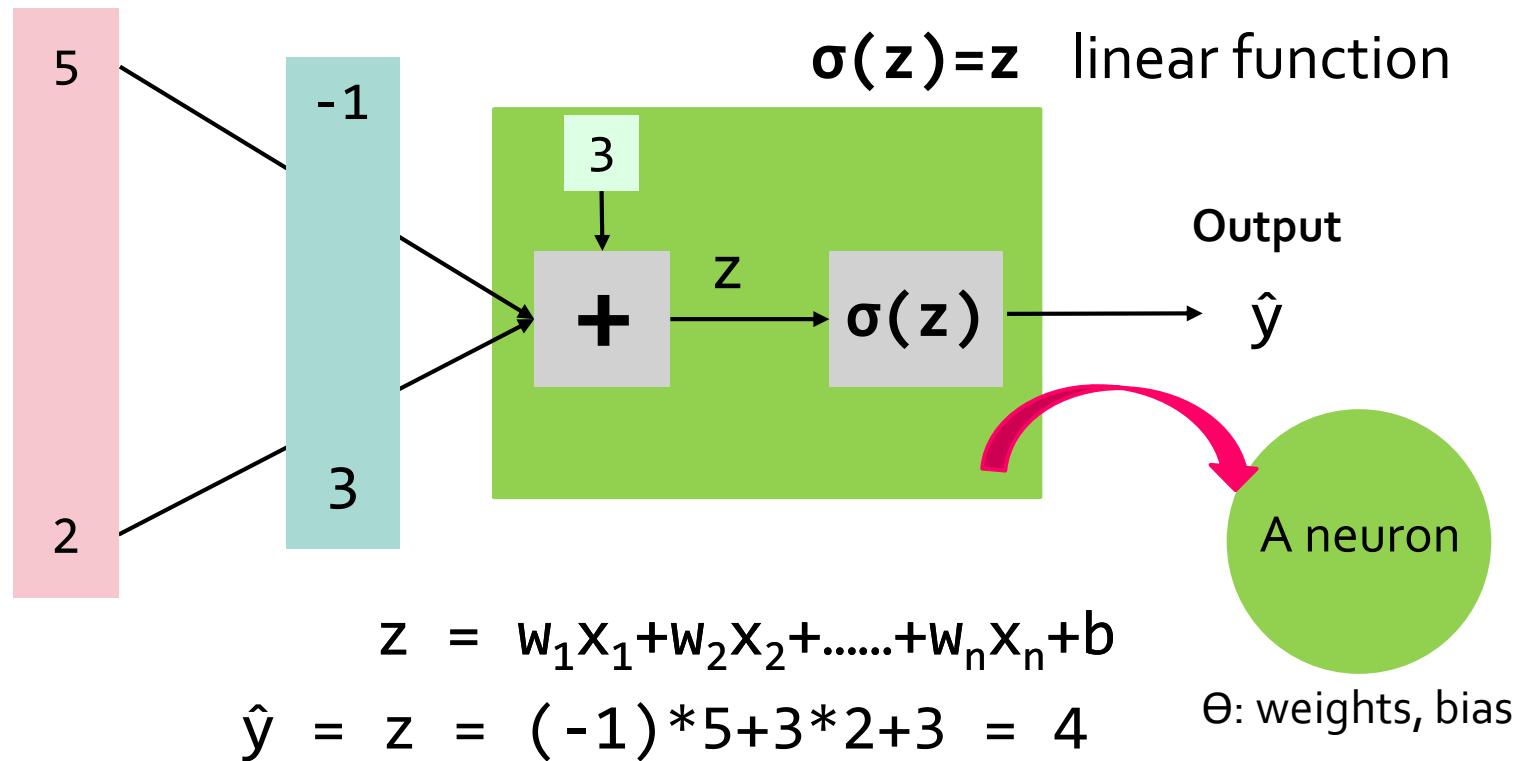
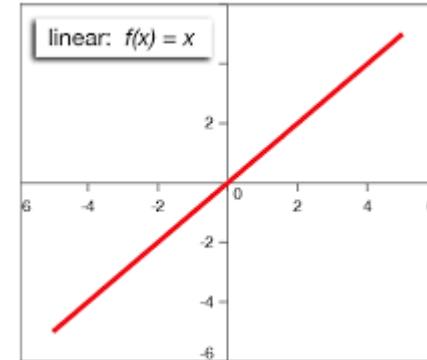
$$z = w_1x_1 + w_2x_2 + \dots + w_nx_n + b$$

$$\hat{y} = \sigma(z)$$



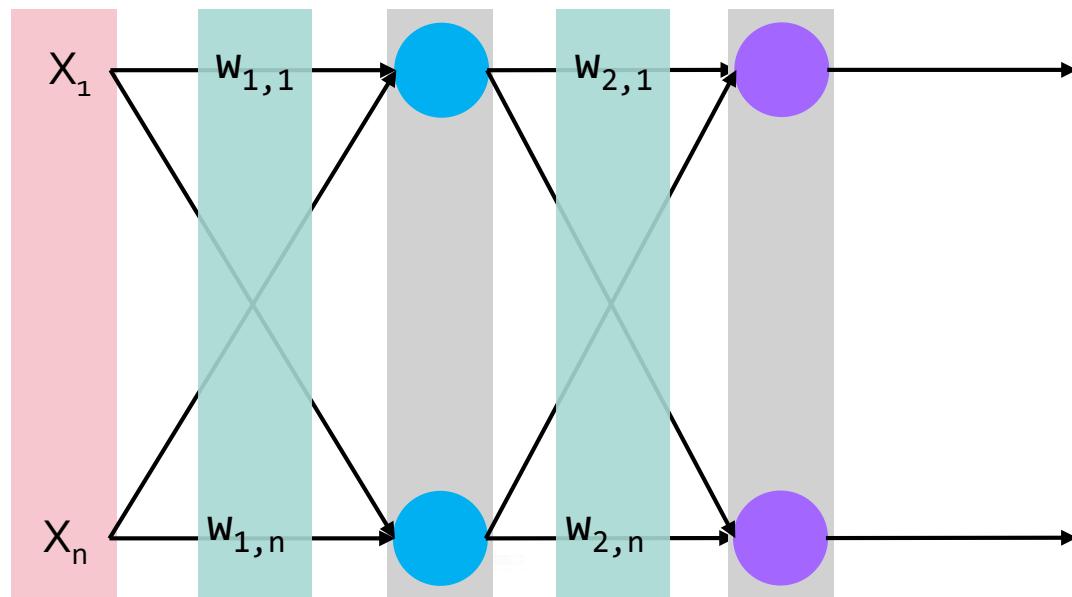
Neuron 的運作

Example



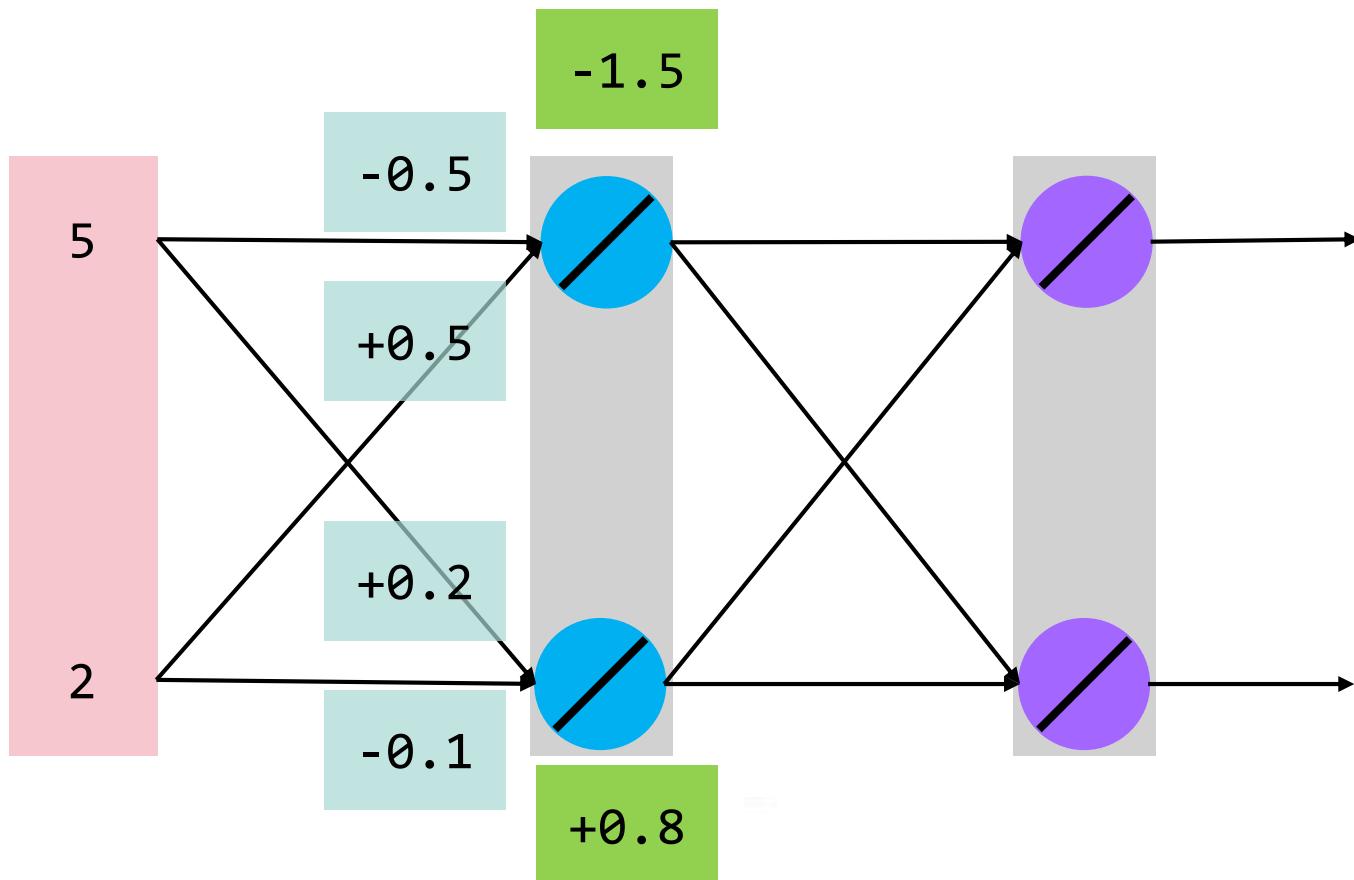
Fully Connected Neural Network

- 很多個 neurons 連接成 network
- Universality theorem: a network with enough number of neurons can present any function



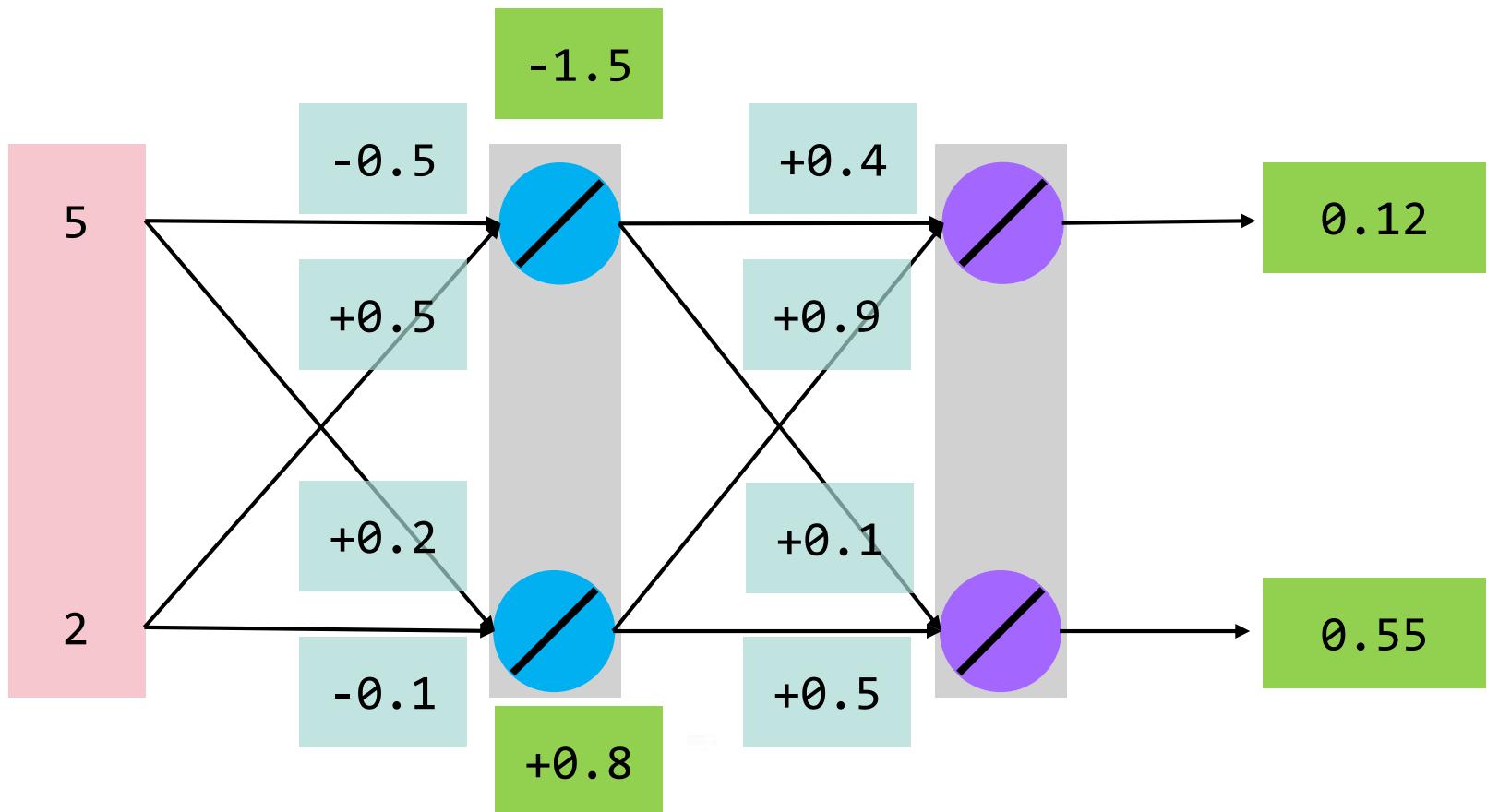
Fully Connected Neural Network

- A simple network with linear activation functions

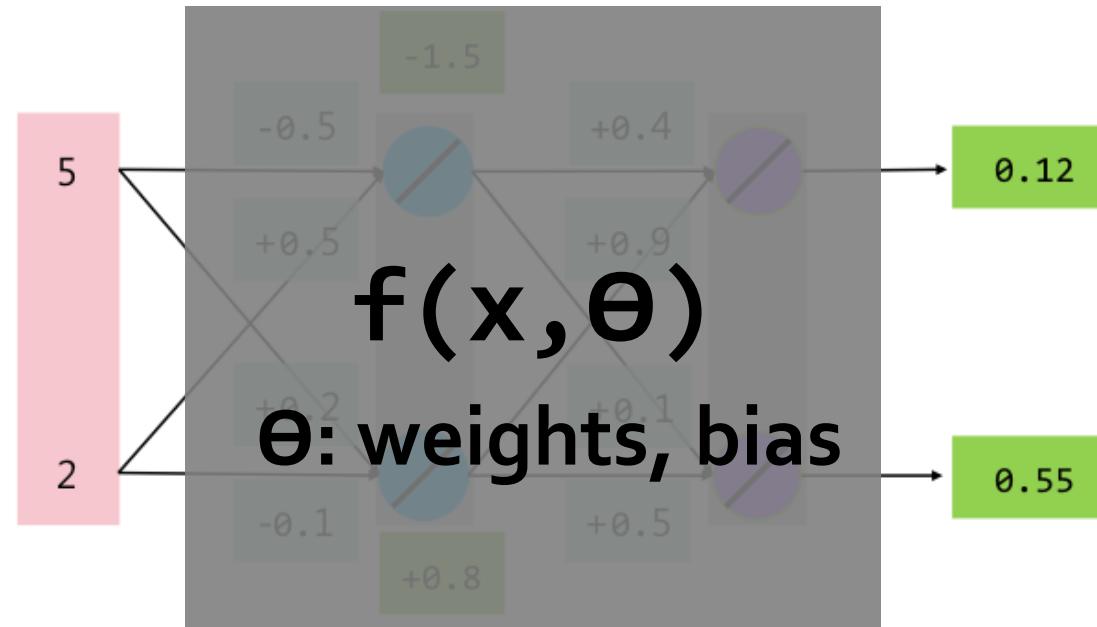


Fully Connected Neural Network

- A simple network with linear activation functions



給定 Network Weights



Given

$$\begin{matrix} -0.5 \\ +0.2 \\ +0.5 \\ -0.1 \end{matrix} \quad \& \quad \begin{matrix} +0.4 \\ +0.9 \\ +0.1 \\ +0.5 \end{matrix}$$

$$f(\begin{matrix} 5 \\ 2 \end{matrix}) = \begin{matrix} 0.12 \\ 0.55 \end{matrix}$$

A Neural Network = A Function

Deep Learning Framework

Define a set
of functions

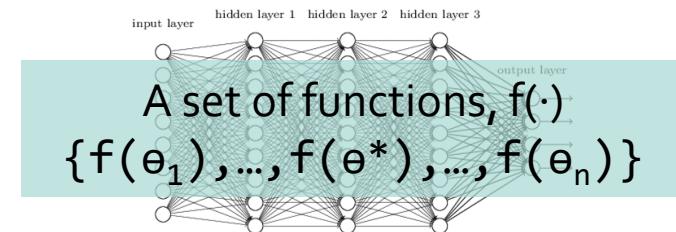


Evaluate
and
Search



Pick the best
function

特定的網絡架構



不斷修正 f 的參數

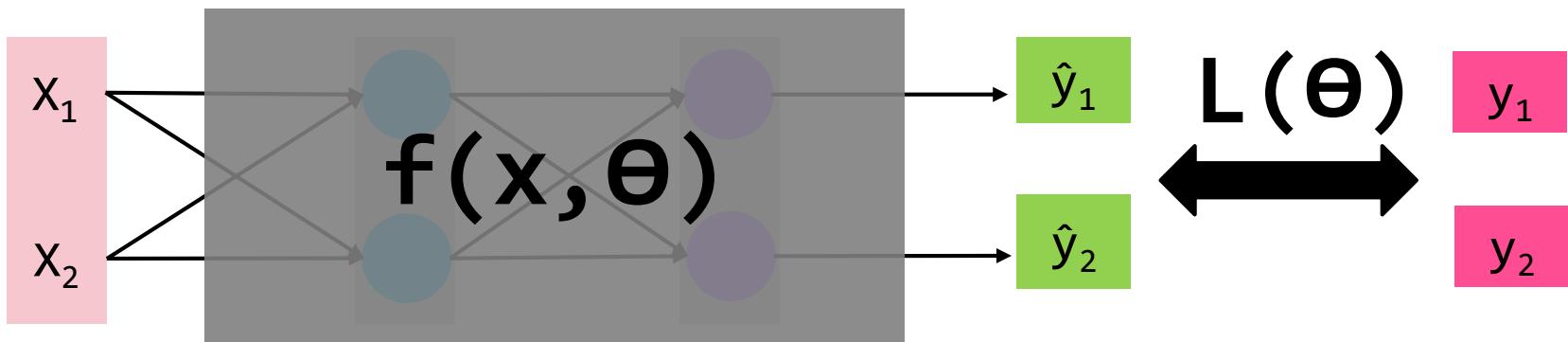
$f(\theta_{94}) \rightarrow$
 $f(\theta_{87}) \rightarrow$
 $f(\theta_{945}) \dots$

找到最適合的參數

$f(\theta^*)$

如何評估模型好不好？

- output values 跟 actual values 越一致越好



- A loss function is to quantify the gap between network outputs and actual values
- Loss function is a function of Θ

目標：最佳化 Total Loss

- Find **the best function** that minimize total loss
 - Find the best network weights, θ^*
 - $\theta^* = \underset{\theta}{\operatorname{argmin}} L(\theta)$
- 最重要的問題: 該如何找到 θ^* 呢 ?
 - 踏破鐵鞋無覓處 (enumerate all possible values)
 - 假設 weights 限制只能 0.0, 0.1, ..., 0.9 , 有 500 個 weights 全部組合就有 10^{500} 組
 - 評估 1 秒可以做 10^6 組，要約 10^{486} 年
 - 宇宙大爆炸到現在才 10^{10} 年
 - Impossible to enumerate

Gradient Descent

- 一種 heuristic 最佳化方法，適用於連續、可微的目標函數
- 核心精神

每一步都朝著進步的方向，直到沒辦法再進步



當有選擇的時候，國家還是
要往**進步的方向**前進。

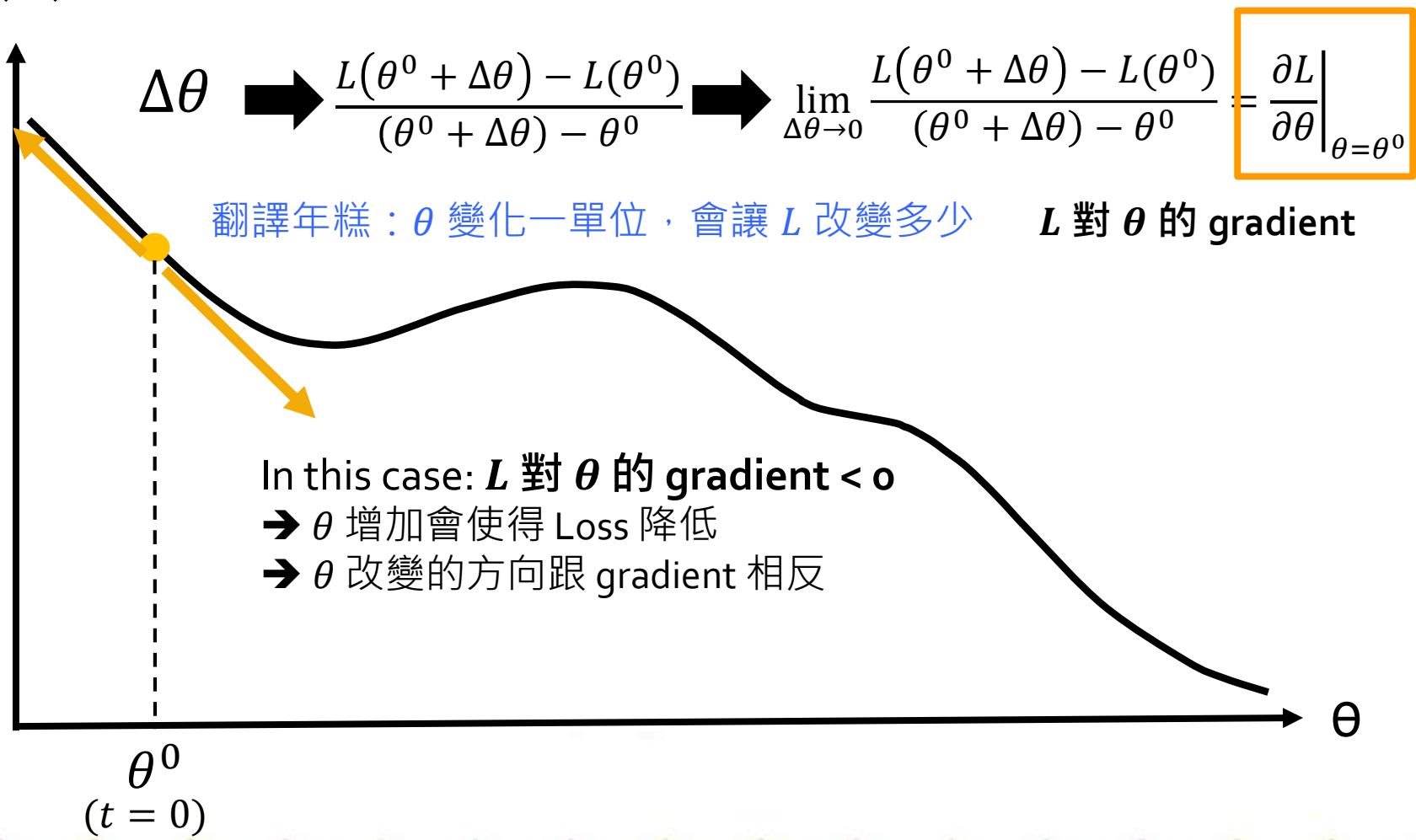


<http://i.imgur.com/xxzpPFN.jpg>

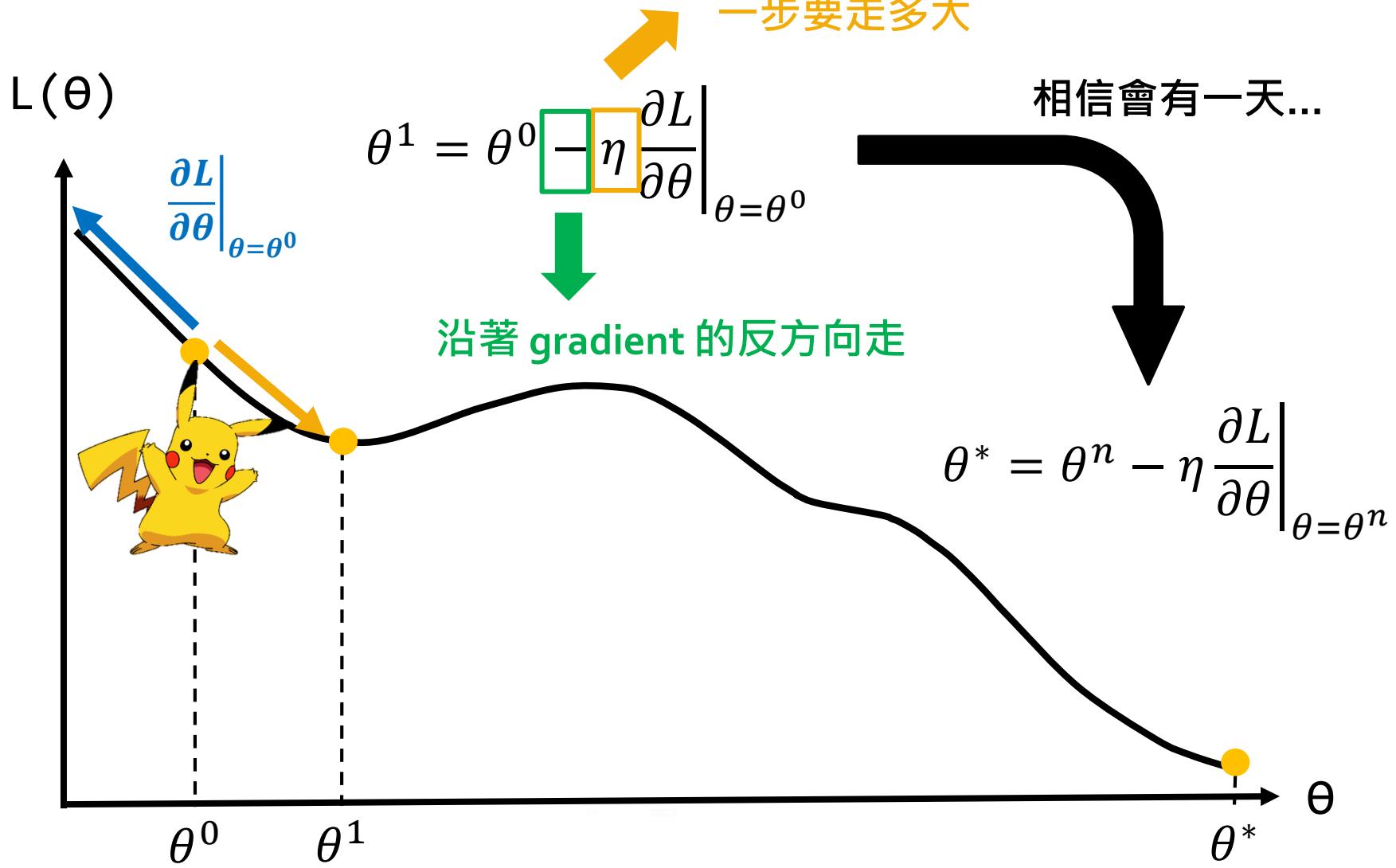


Gradient Descent

$L(\theta)$ 想知道在 θ^0 這個點時， L 隨著 θ 的變化

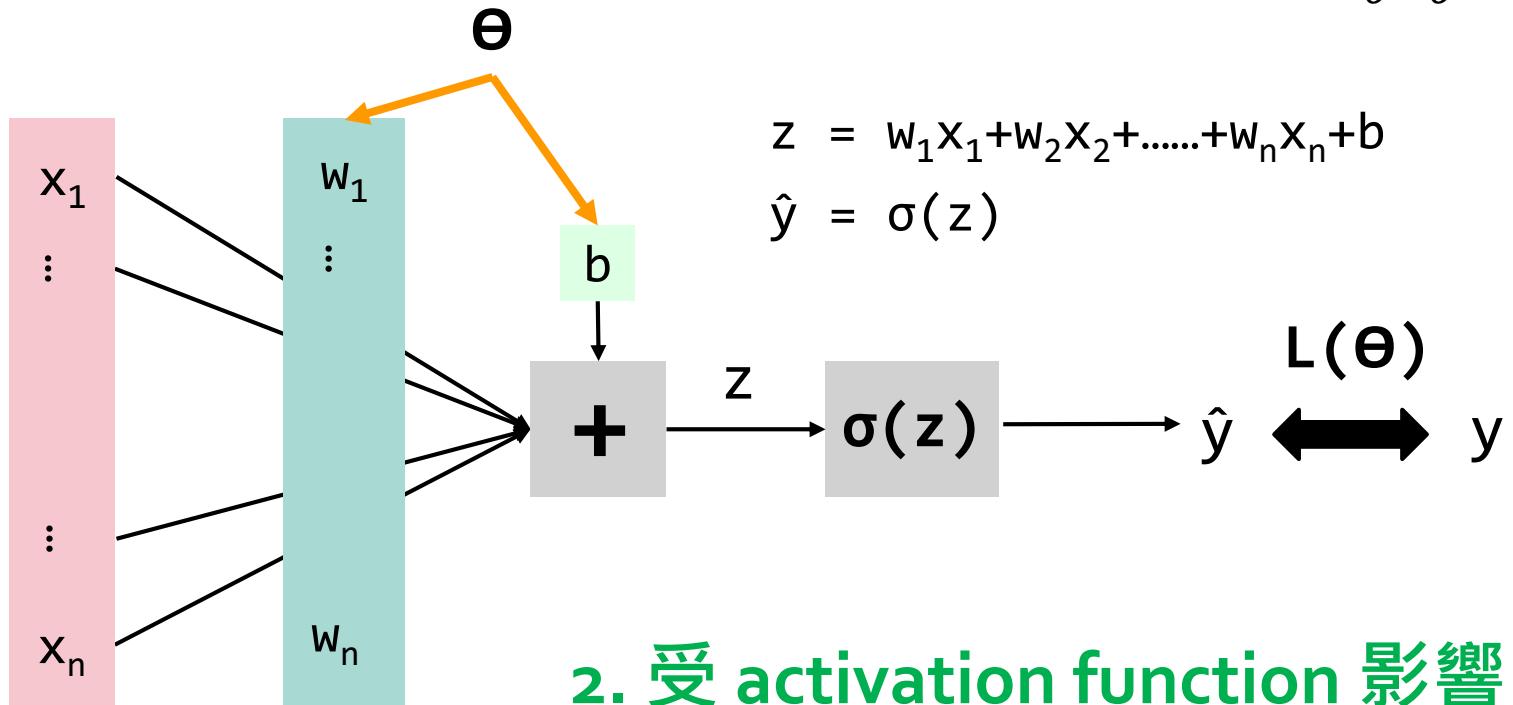


Gradient Descent



影響 Gradient 的因素

$$\theta^1 = \theta^0 - \eta \frac{\partial L}{\partial \theta} \Big|_{\theta=\theta^0}$$



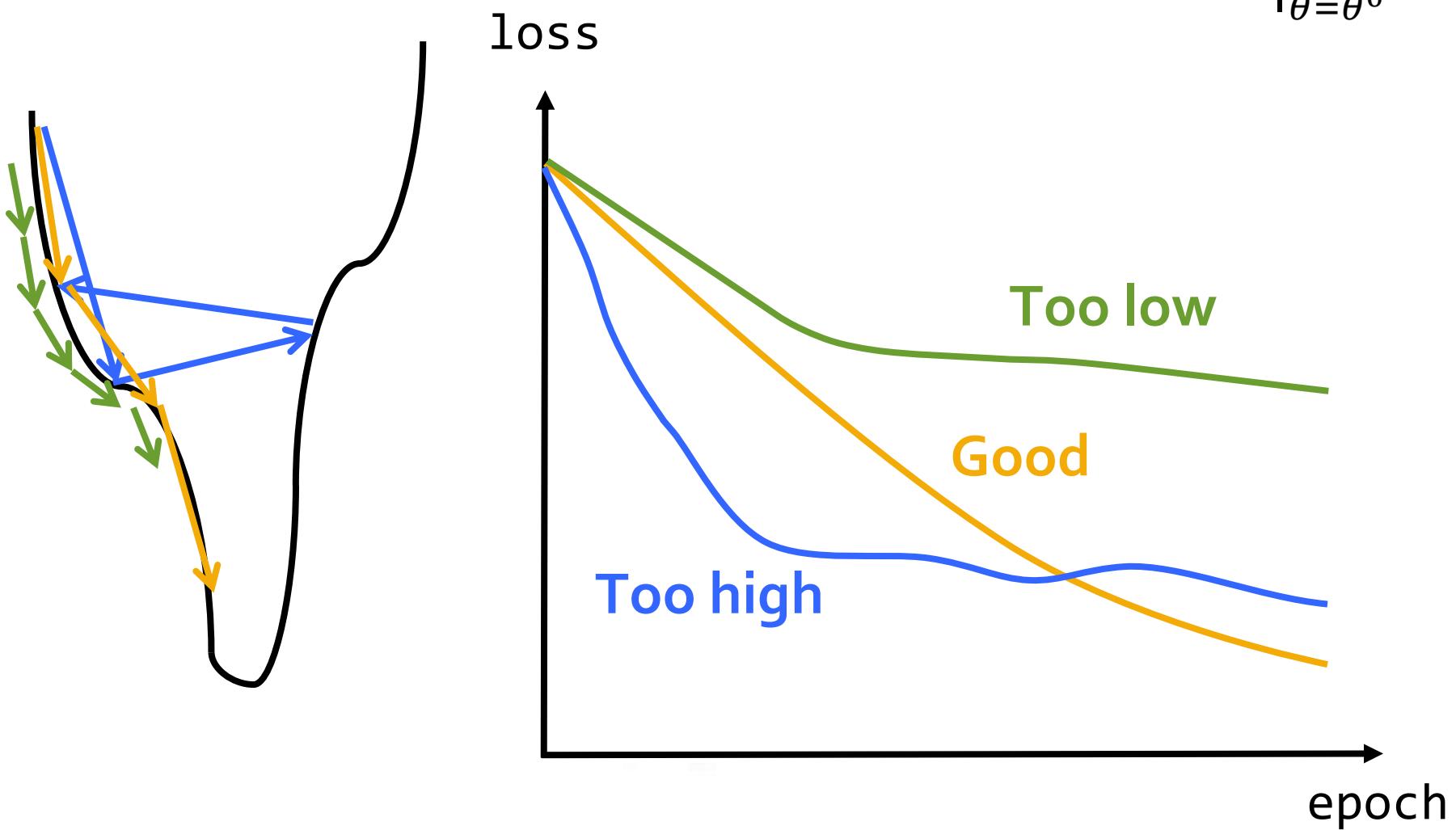
2. 受 activation function 影響

$$\frac{\partial L}{\partial \theta} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \theta} = \boxed{\frac{\partial L}{\partial \hat{y}}} \boxed{\frac{\partial \hat{y}}{\partial z}} \frac{\partial z}{\partial \theta}$$

1. 受 loss function 影響

Learning Rate 的影響

$$\theta^1 = \theta^0 - \eta \frac{\partial L}{\partial \theta} \Big|_{\theta=\theta^0}$$



Summary – Gradient Descent

- 用來最佳化一個連續的目標函數
- 朝著進步的方向前進
- Gradient descent
 - Gradient 受 loss function 影響
 - Gradient 受 activation function 影響
 - 受 learning rate 影響

Gradient Descent 的缺點

- 一個 epoch 更新一次，收斂速度很慢
 - ▣ 一個 epoch 等於看過所有 training data 一次

□ Problem 1

有辦法加速嗎？

A solution: stochastic gradient descent (SGD)

□ Problem 2

Gradient based method 不能保證找到全域最佳解

- ▣ 可以利用 momentum 降低困在 local minimum 的機率



Gradient Descent 的缺點

- 一個 epoch 更新一次，收斂速度很慢
 - ▣ 一個 epoch 等於看過所有 training data 一次

□ Problem 1

有辦法加速嗎？

A solution: stochastic gradient descent (SGD)

□ Problem 2

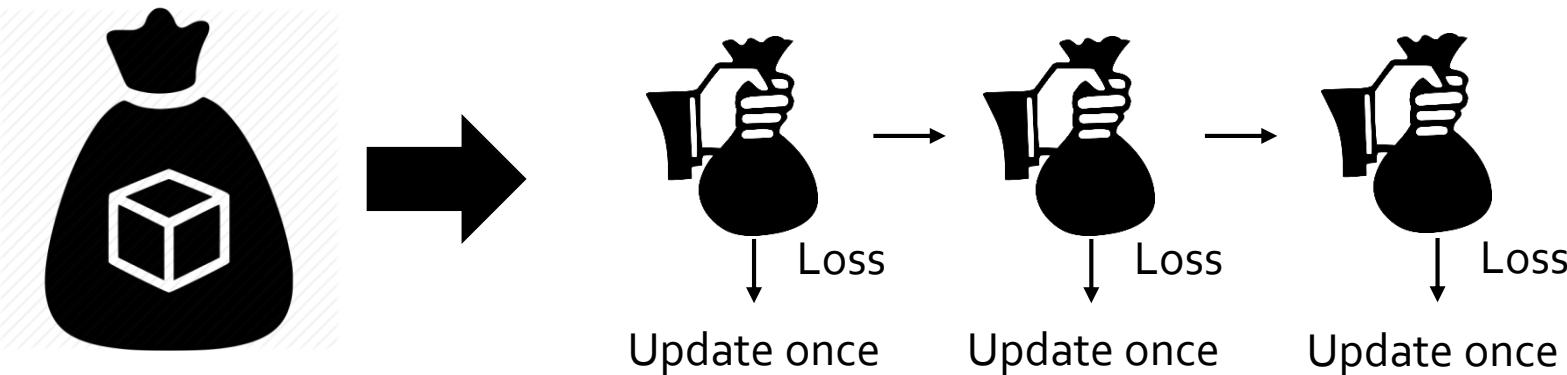
Gradient based method 不能保證找到全域最佳解

- ▣ 可以利用 momentum 降低困在 local minimum 的機率



Stochastic Gradient Descent

- 隨機抽一筆 training sample，依照其 loss 更新一次
- 另一個問題，一筆一筆更新也很慢
- Mini-batch: **每一個 mini-batch 更新一次**



- Benefits of mini-batch
 - 相較於 SGD: faster to complete one epoch
 - 相較於 GD: faster to converge (to optimum)

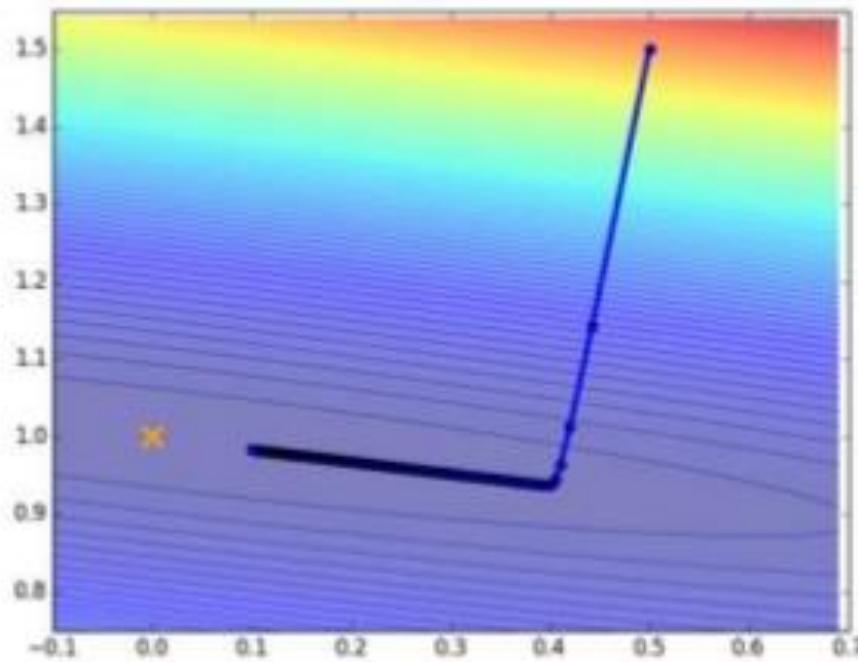
Mini-batch vs. Epoch

- 一個 epoch = 看完所有 training data 一次
- 依照 mini-batch 把所有 training data 拆成多份
- 假設全部有 1000 筆資料
 - Batch size = 100 可拆成 10 份 → 一個 epoch 內會更新 10 次
 - Batch size = 10 可拆成 100 份 → 一個 epoch 內會更新 100 次
- 如何設定 batch size?
 - 不要設太大，常用 28, 32, 128, 256, ...

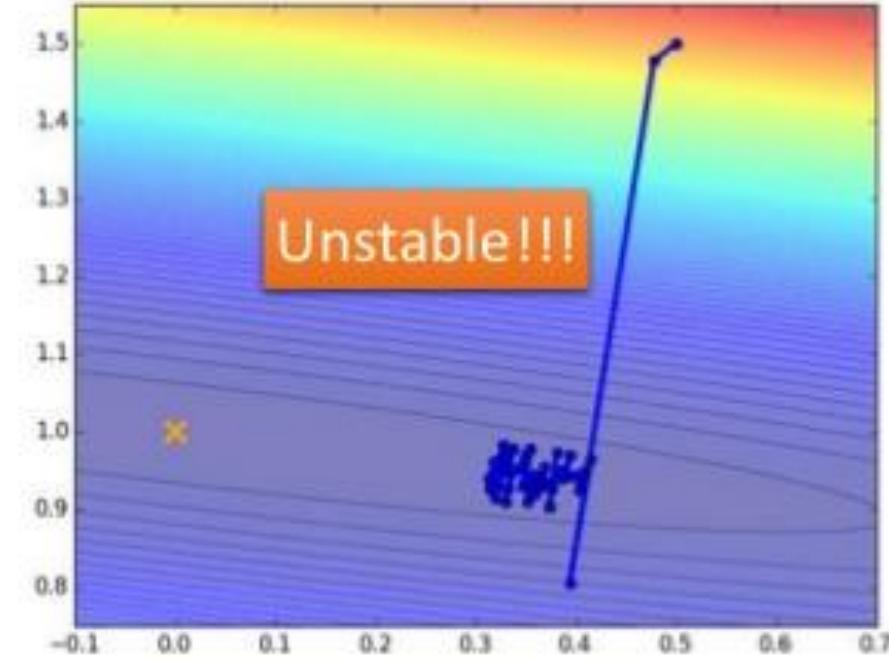
Mini-batch 的影響

- 本質上已經不是最佳化 total loss 而是在最佳化 batch loss

Gradient Descent



Mini-batch



Gradient Descent 的缺點

- 一個 epoch 更新一次，收斂速度很慢
 - ▣ 一個 epoch 等於看過所有 training data 一次

□ Problem 1

有辦法加速嗎？

A solution: stochastic gradient descent (SGD)

□ Problem 2

Gradient based method 不能保證找到全域最佳解

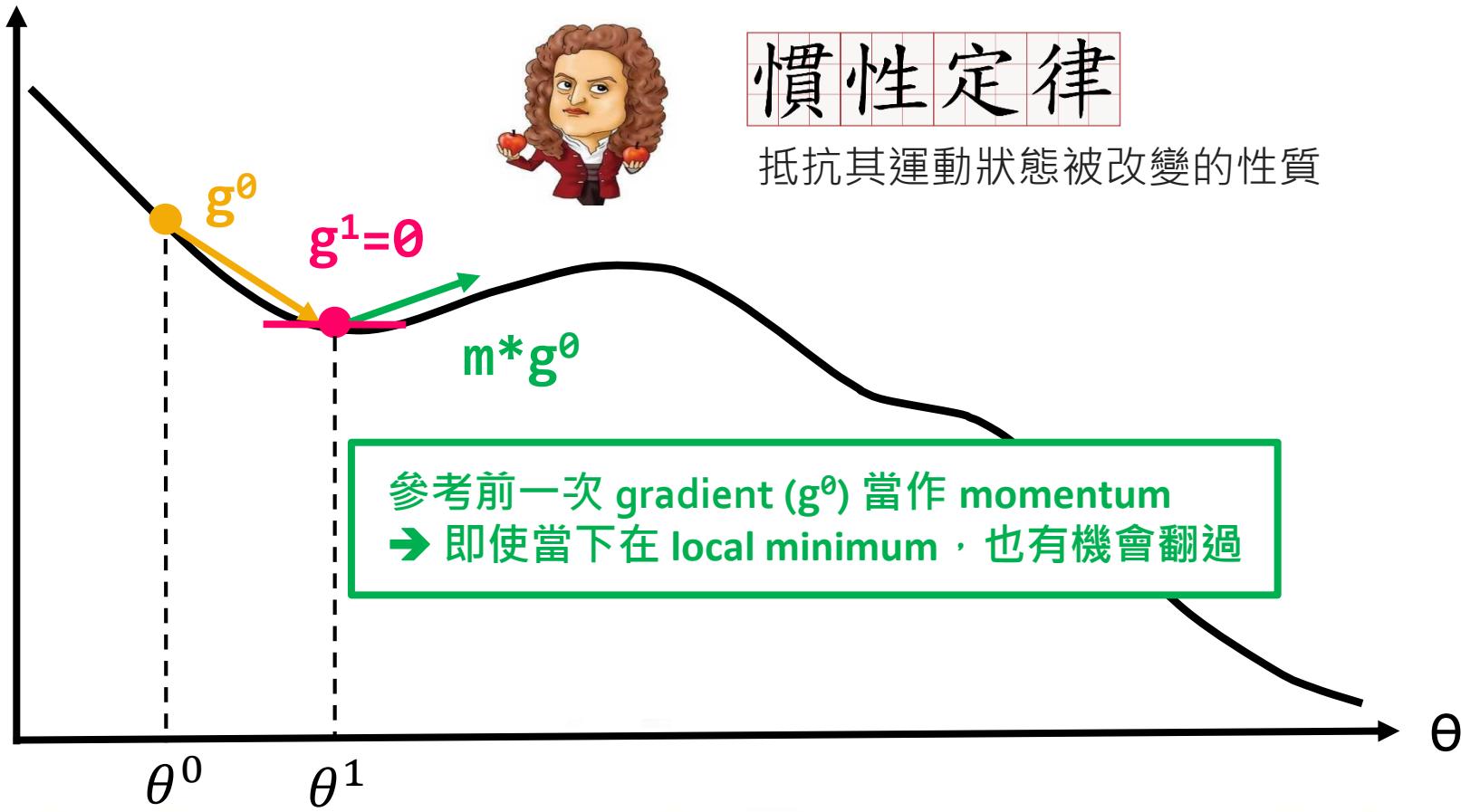
- ▣ 可以利用 momentum 降低困在 local minimum 的機率



Momentum

$L(\theta)$

Gradient=0 → 不更新，陷在 local minimum



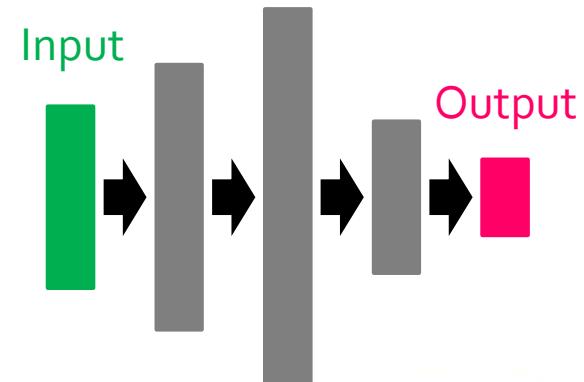
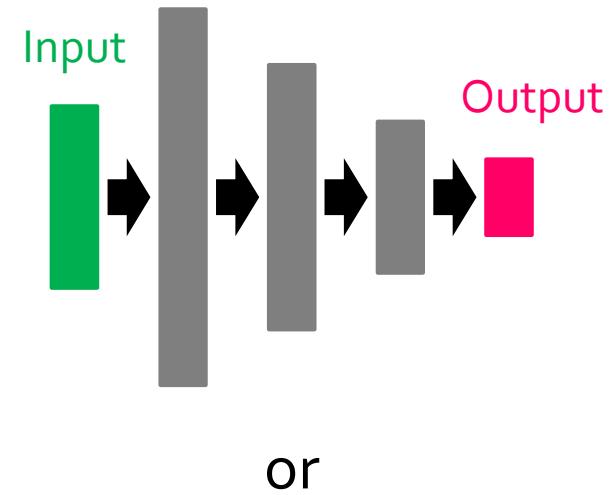


Introduction of Deep Learning

- Artificial neural network
- Activation functions
- Loss functions
- Gradient descent
 - Loss function, activation function, learning rate
- Stochastic gradient descent
- Mini-batch
- Momentum

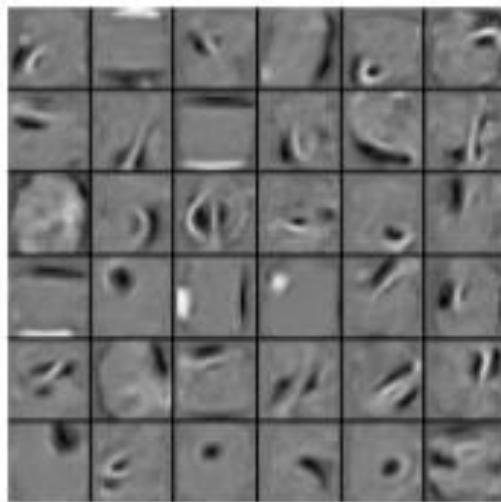
Frequently Asked Questions

- 要有幾層 hidden layers ?
- 每層幾個 neurons ?
 - Neurons 多寡跟資料多寡有關
 - Intuition + trial and error
- 深會比較好嗎 ?
 - Deep for modulation

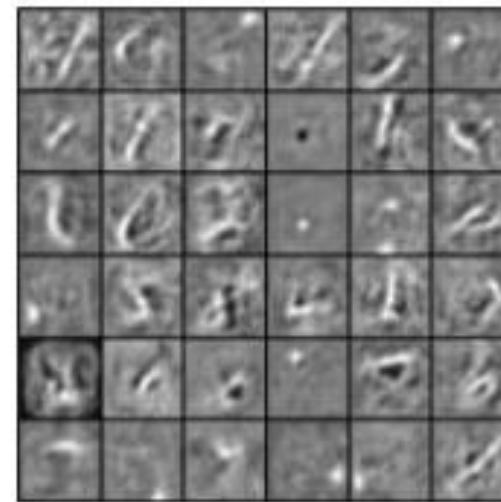


Visualization of Modulation

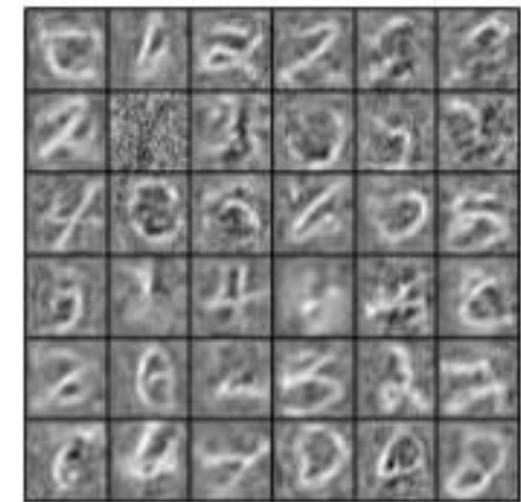
1st hidden layer



2nd hidden layer



3rd hidden layer



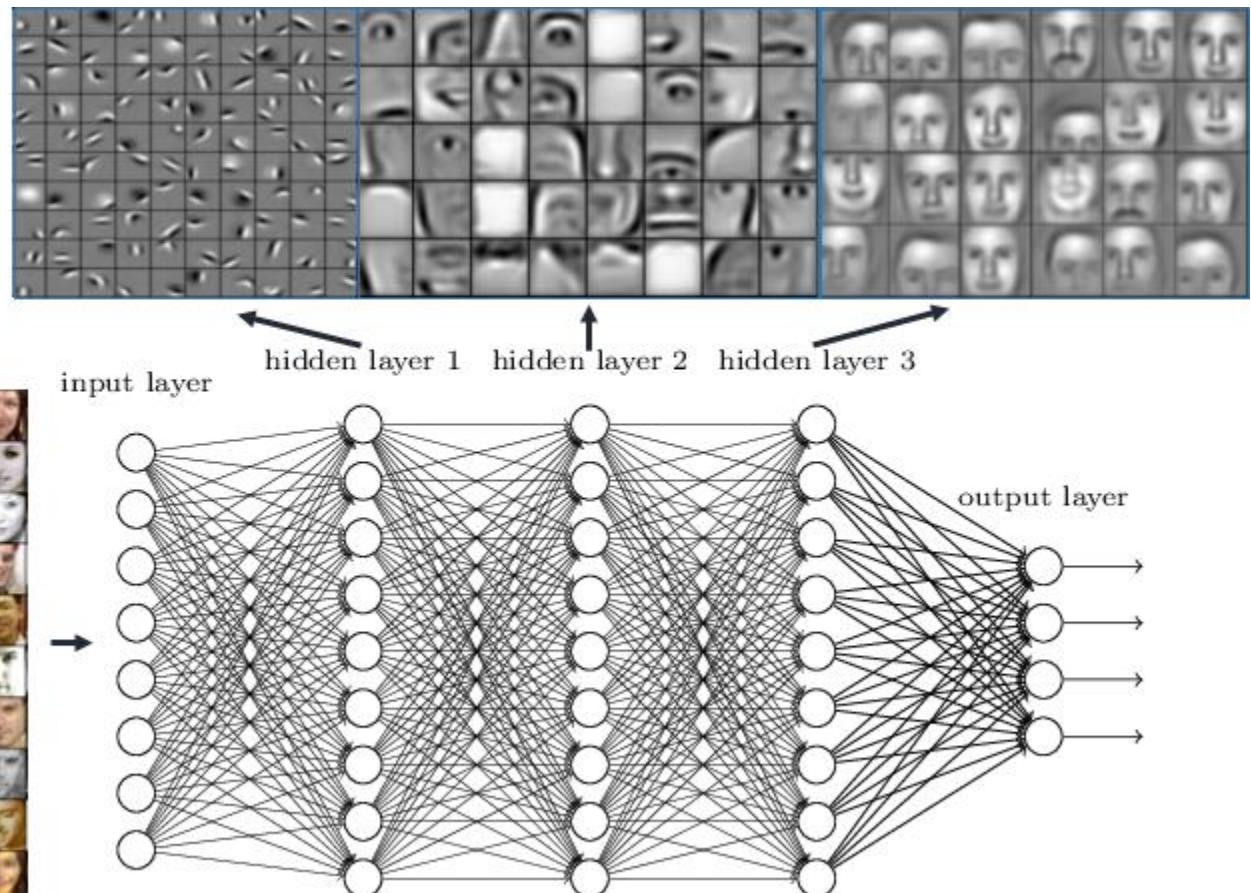
各司其職、由簡馭繁，組織出越來越複雜的 feature extractors

Ref: [Visualizing Higher-Layer Features of a Deep Network](#)



Visualization of Modulation

Deep neural networks learn hierarchical feature representations



Ref: [Deep Learning and Convolutional Neural Networks: RSIP Vision Blogs](#)

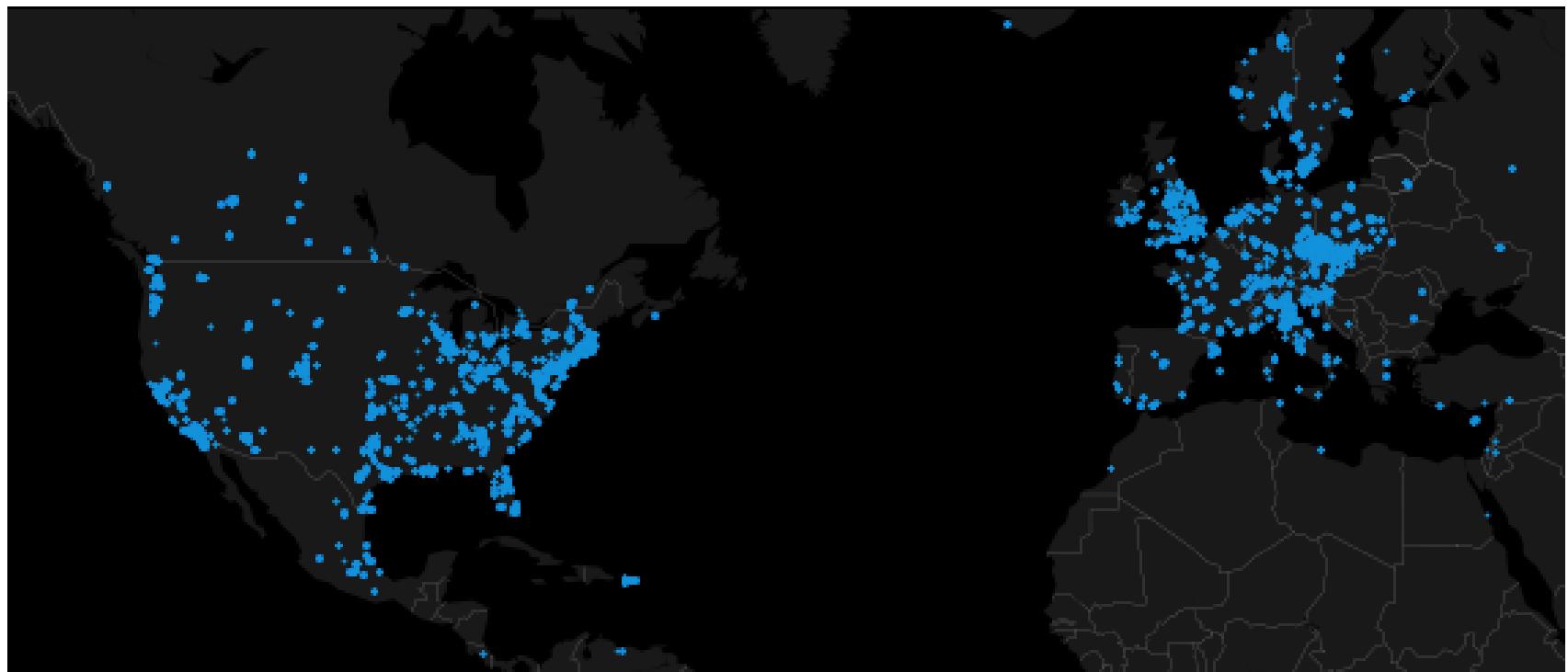


Hands-on Tutorial

寶可夢雷達 using Pokemon Go Dataset on Kaggle

範例資料

- 寶可夢過去出現的時間與地點紀錄 (dataset from Kaggle)



Ref: <https://www.kaggle.com/kostyabahshetsyan/d/seminiy/predictmall/pokemon-geolocation-visualisations/notebook>

Raw Data Overview

latitude	longitude	appearedTimeOfDay	appearedYear	terrainType	closeToWater	city	weather	temperature	windSpeed	cooc_151	class
20.525745	-97.460829	night	2016	14	0	Mexico_City	Foggy	25.5	4.79	0	16
20.523695	-97.461167	night	2016	14	0	Mexico_City	Foggy	25.5	4.79	0	133
38.90359	-77.19978	night	2016	13	0	New_York	Clear	24.2	4.29	0	16
47.665903	-122.312561	night	2016	0	1	Los_Angeles	PartlyCloudy	15.6	5.84	0	13
47.666454	-122.311628	night	2016	0	1	Los_Angeles	PartlyCloudy	15.6	5.84	0	133
-31.95498	115.853609	night	2016	13	0	Perth	PartlyCloudy	16.5	6.39	0	21
-31.954245	115.852038	night	2016	13	0	Perth	PartlyCloudy	16.5	6.4	0	66
26.235257	-98.197591	night	2016	13	0	Chicago	Clear	28	11.26	0	27
20.525554	-97.4588	night	2016	14	0	Mexico_City	Foggy	25.5	4.79	0	35
32.928558	-84.340278	night	2016	8	0	New_York	Clear	23.7	3.94	0	19
32.930646	-84.339867	night	2016	8	0	New_York	Clear	23.7	3.94	0	116
32.943651	-84.334443	night	2016	8	0	New_York	Clear	23.7	3.94	0	74
26.235552	-98.197249	night	2016	13	0	Chicago	Clear	28	11.26	0	16
20.52577	-97.460237	night	2016	14	0	Mexico_City	Foggy	25.5	4.79	0	19
26.236029	-98.196908	night	2016	13	0	Chicago	Clear	28	11.26	0	19
47.664333	-122.312645	night	2016	0	1	Los_Angeles	PartlyCloudy	15.6	5.84	0	19
20.526489	-97.460745	night	2016	14	0	Mexico_City	Foggy	25.5	4.79	0	16
53.611417	-113.369528	night	2016	12	0	Edmonton	Clear	8.9	1.47	0	13

問題：會出現哪一隻神奇寶貝呢？

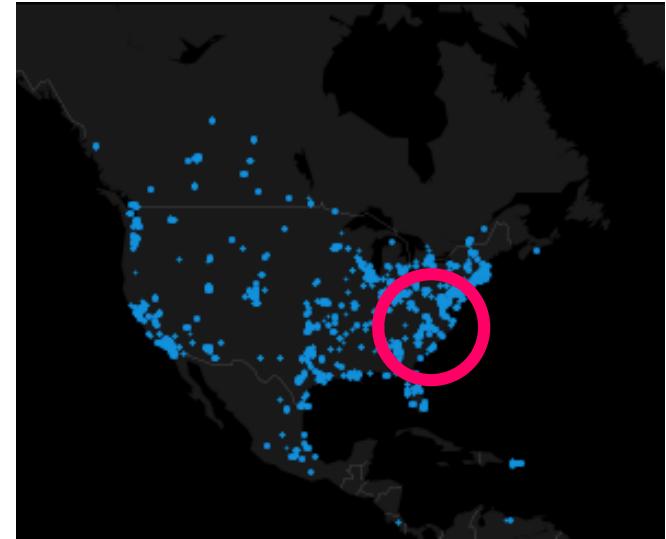


寶可夢雷達 Data Field Overview

- 時間: local.hour, local.month, DayofWeek...
- 天氣: temperature, windSpeed, pressure...
- 位置: longitude, latitude, pokestop...
- 環境: closeToWater, terrainType...
- 十分鐘前有無出現其他寶可夢
 - 例如: cooc_1=1 十分鐘前出現過 class=1 之寶可夢
- class 就是我們要預測目標

Sampled Dataset for Fast Training

□ 挑選在 New York City 出現的紀錄



□ 挑選下列五隻常見的寶可夢

No.4 小火龍



No.43 走路草



No.56 火爆猴



No.71 喇叭芽



No.98 大鉗蟹





開始動手囉！Keras Go！



Input 前處理

- 因為必須跟 weights 做運算
Neural network 的輸入**必須為數值 (numeric)**
- 如何處理非數值資料？
 - 順序資料
 - 名目資料
- 不同 features 的數值範圍差異會有影響嗎？
 - 溫度：最低 0 度、最高 40 度
 - 距離：最近 0 公尺、最遠 10000 公尺

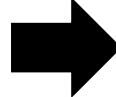
處理順序資料

□ Ordinal variables (順序資料)

- For example: {Low, Medium, High}
- Encoding in order
 - {Low, Medium, High} → {1, 2, 3}

□ Create a new feature using mean or median

UID	Age
P1	0-17
P2	0-17
P3	55+
P4	26-35



UID	Age
P1	15
P2	15
P3	70
P4	30

處理名目資料

□ Nominal variables (名目資料)

- `{"SugarFree", "Half", "Regular"}`

- One-hot encoding

- 假設有三個類別

- Category 1 → [1, 0, 0]

- Category 2 → [0, 1, 0]

- 紿予類別上的解釋 → Ordinal variables

- `{"SugarFree", "Half", "Regular"}` → 1, 2, 3

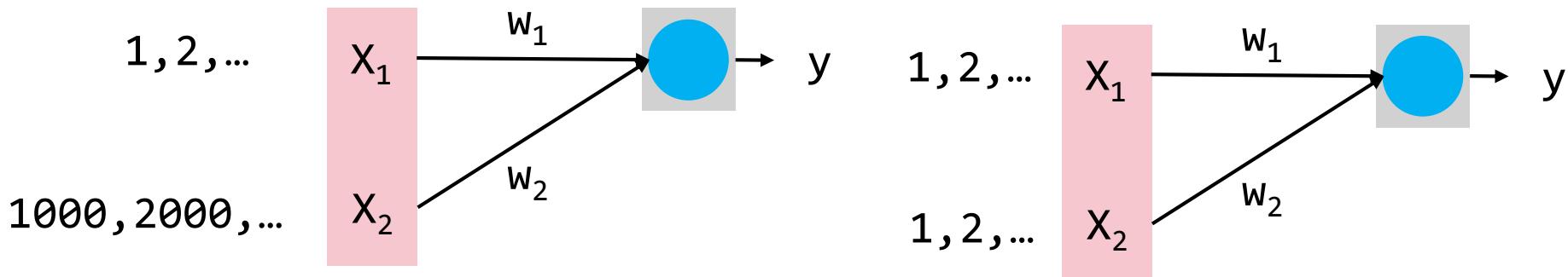
- 特殊的名目資料：地址

- 台北市南港區研究院路二段128號

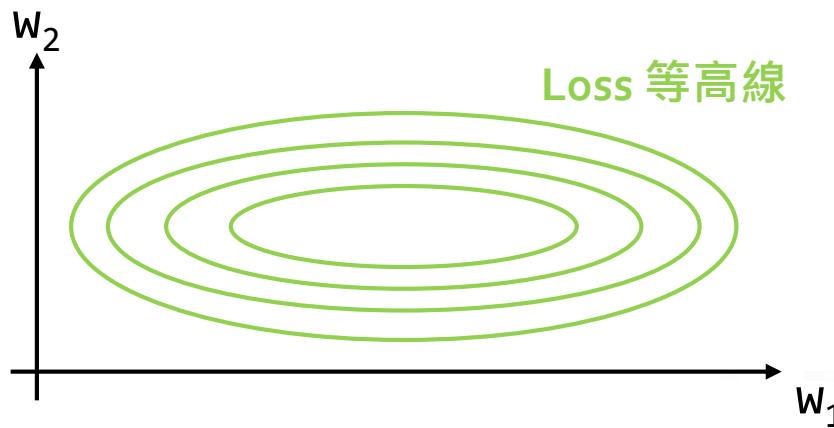
- 轉成經緯度 {25.04, 121.61}

處理不同的數值範圍

□ 先說結論：建議 re-scale！但為什麼？



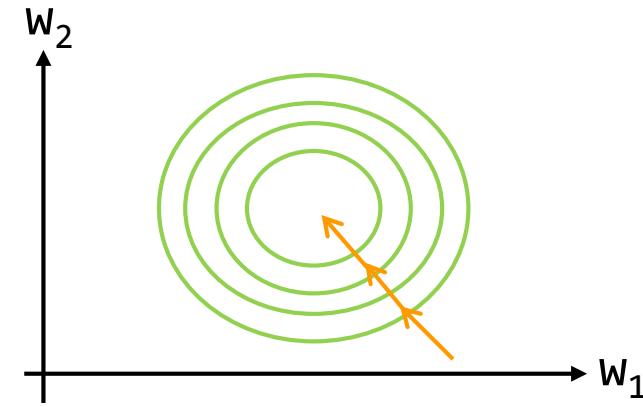
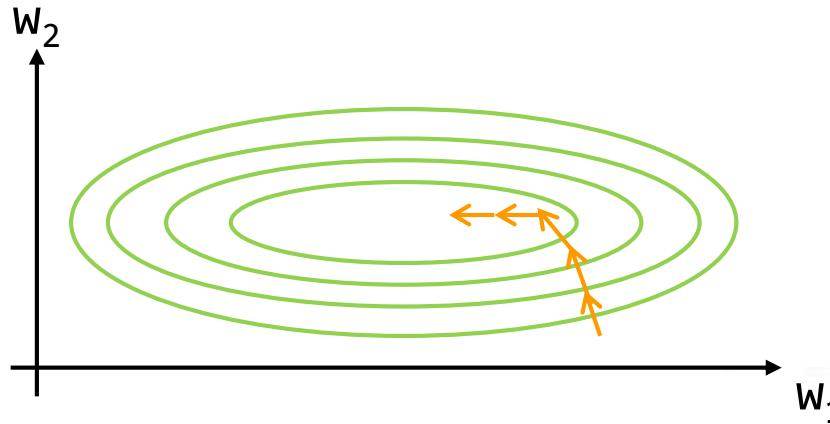
w_2 的修正(Δw)對於 loss 的影響比較大



處理不同的數值範圍

影響訓練的過程

- 不同 scale 的 weights 修正時會需要不同的 learning rates
 - 不用 adaptive learning rate 是做不好的
- 在同個 scale 下，loss 的等高線會較接近圓形
→ gradient 的方向會指向圓心 (最低點)



小提醒

- 輸入 (input) 只能是數值
- 名目資料、順序資料
 - One-hot encoding
 - 順序轉成數值
- 建議 re-scale 到接近的數值範圍
- 今天的資料都已經先幫大家做好了 ☺

Read Input File

```
import numpy as np

# 讀進檔案，以 , (逗號)分隔的 csv 檔，不包含第一行的欄位定義
my_data = np.genfromtext('pkgo_city66_class5_v1.csv',
                        delimiter=',',
                        skip_header=1)

# Input 是有 200 個欄位(index 從 0 - 199)
X_train = my_data[:,0:200]

# Output 是第 201 個欄位(index 為 200)
y_train = my_data[:,200]

# 確保資料型態正確
X_train = X_train.astype('float32')
y_train = y_train.astype('int')
```

Input

```
# 觀察一筆 X_train  
print(X_train[1,:32])
```

```
print(X_train[1,:32])  
  
[ 7.32567608e-02 -7.03223109e-01 9.00000000e+00 8.00000000e+00  
 2.00000000e+00 5.00000000e+01 4.50000000e+01 4.00000000e+00  
 4.00000000e+00 5.00000000e+01 1.00000000e+00 8.00000000e+00  
 8.00000000e+00 1.30000000e+01 0.00000000e+00 6.60000000e+01  
 2.00000000e+00 5.00000000e+00 5.11728227e-01 -1.19994007e-01  
 2.61039048e-01 4.33402471e-02 1.20537996e+00 1.83238339e+00  
 -1.39905763e+00 1.12362671e+00 6.24226868e-01 -8.81169885e-02  
 1.41916251e+00 -1.14249873e+00 -4.79164541e-01 0.00000000e+00 ]
```

Output 前處理

- Keras 預定的 class 數量與值有關
 - 挑選出的寶可夢中，最大 Pokemon ID = 98
Keras 會認為『有 99 個 classes 分別為 Class 0, 1, 2, ..., 98 class』
 - zero-based indexing (python)
- 把下面的五隻寶可夢轉換成

No.4 小火龍



Class 0

No.43 走路草



Class 1

No.56 火爆猴



Class 2

No.71 喇叭芽



Class 3

No.98 大鉗蟹



Class 4



Output

```
# 觀察一筆 y_train  
print(y_train[0])
```

```
y_train[0]
```

```
1
```

```
# [重要] 將 Output 從特定類別轉換成 one-hot encoding 的形式
```

```
from keras.utils import np_utils  
Y_train = np_utils.to_categorical(y_train, 5)
```

```
# 轉換成 one-hot encoding 後的 Y_train
```

```
print(Y_train[1,:])
```

```
Y_train[0,:]
```

```
array([ 0.,  1.,  0.,  0.,  0.])
```

接下來的流程

- 先建立一個深度學習模型



就像開始冒險前要先選一隻寶可夢

- 邊移動邊開火

六步完模 – 建立深度學習模型

1. 決定 hidden layers 層數與其中的 neurons 數量
2. 決定該層使用的 activation function
3. 決定模型的 loss function
4. 決定 optimizer
 - Parameters: learning rate, momentum, decay
5. 編譯模型 (Compile model)
6. 開始訓練囉！(Fit model)

步驟 1+2: 模型架構

```
from keras.models import Sequential
from keras.layers.core import Dense, Activation
from keras.optimizers import SGD

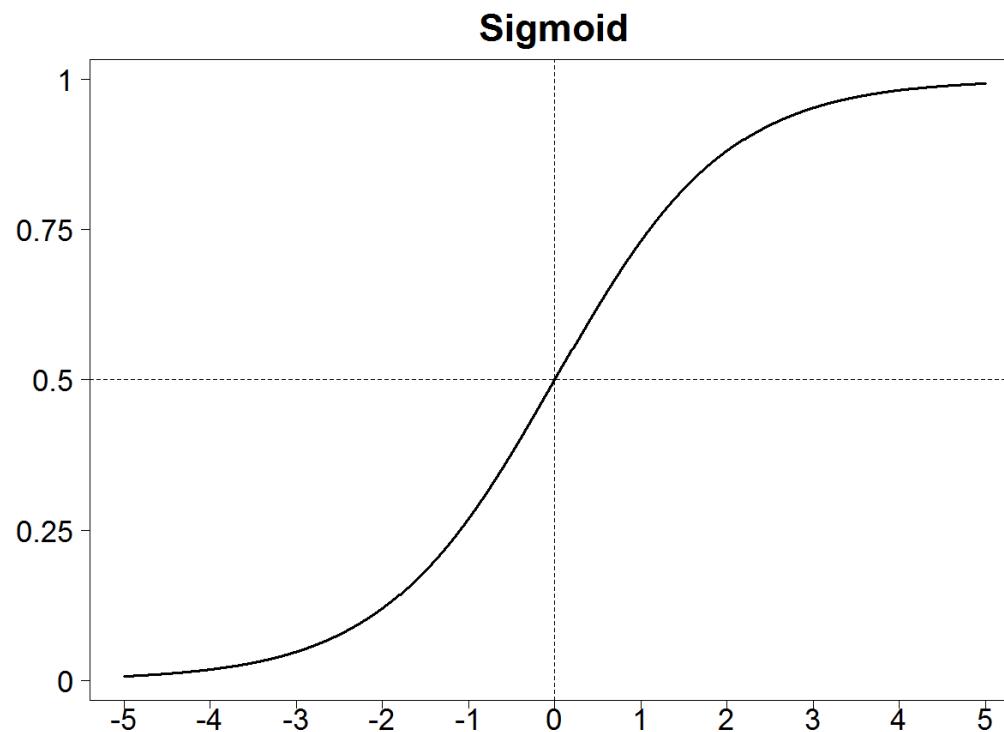
# 宣告這是一個 Sequential 次序性的深度學習模型
model = Sequential()

# 加入第一層 hidden layer (128 neurons)
# [重要] 因為第一層 hidden layer 需連接 input vector
# 故需要在此指定 input_dim
model.add(Dense(128, input_dim=200))
```

Model 建構時，是以次序性的疊加 (add) 上去

基本款 activation function

□ Sigmoid function



步驟 1+2: 模型架構 (Cont.)

```
# 宣告這是一個 Sequential 次序性的深度學習模型
model = Sequential()

# 加入第一層 hidden layer (128 neurons) 與指定 input 的維度
model.add(Dense(128, input_dim=200))
# 指定 activation function
model.add(Activation('sigmoid'))

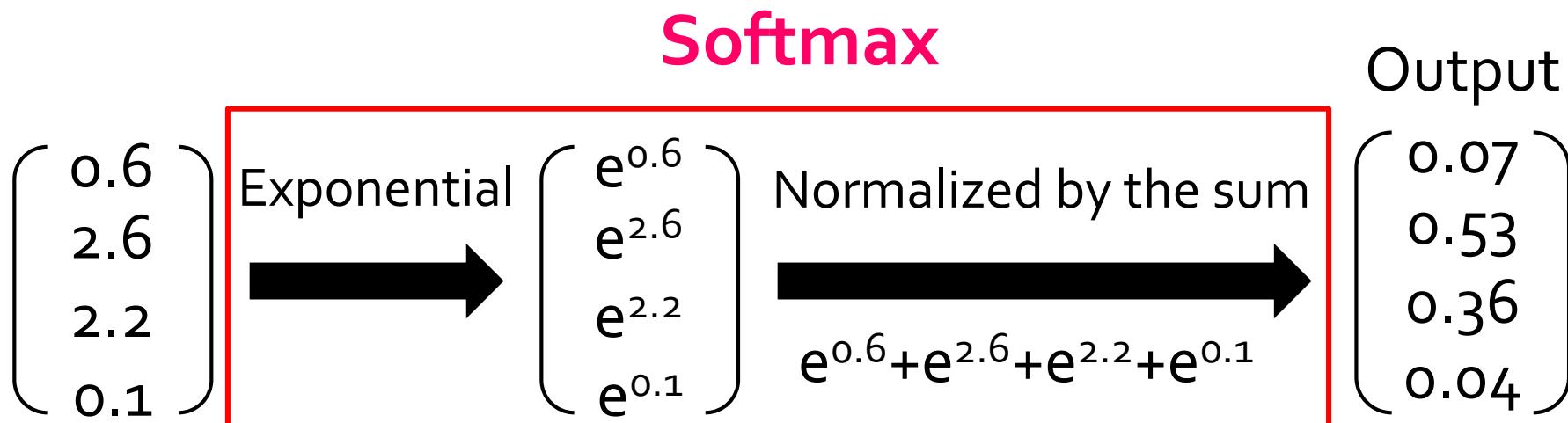
# 加入第二層 hidden layer (256 neurons)
model.add(Dense(256))
model.add(Activation('sigmoid'))

# 加入 output layer (5 neurons)
model.add(Dense(5))
model.add(Activation('softmax'))

# 觀察 model summary
model.summary()
```

Softmax

- Classification 常用 softmax 當 output 的 activation function



- Normalization: network output 轉換到 $[0,1]$ 之間且 softmax output 相加為 1 → 像“機率”
- 保留對其他 classes 的 prediction error

Model Summary

Layer (type)	Output Shape	Param #	Connected to
dense_10 (Dense)	(None, 128)	25728	dense_input_4[0][0]
activation_10 (Activation)	(None, 128)	0	dense_10[0][0]
dense_11 (Dense)	(None, 256)	33024	activation_10[0][0]
activation_11 (Activation)	(None, 256)	0	dense_11[0][0]
dense_12 (Dense)	(None, 5)	1285	activation_11[0][0]
activation_12 (Activation)	(None, 5)	0	dense_12[0][0]
Total params: 60037			

可以設定 Layer 名稱

```
# 另外一種寫法  
model.add(Dense(5,activation='softmax',name='output'))  
  
# 觀察 model summary  
model.summary()
```

Layer (type)	Output Shape	Param #	Connected to
1st hidden layer (Dense)	(None, 128)	25728	dense_input_8[0][0]
2nd hidden layer (Dense)	(None, 256)	33024	1st hidden layer[0][0]
output (Dense)	(None, 5)	1285	2nd hidden layer[0][0]
Total params: 60037			

步驟 3: 選擇 loss function

Prediction	Answer
0.8	0.9
0.2	0.1

□ Mean_squared_error

$$\frac{(0.9 - 0.8)^2 + (0.1 - 0.2)^2}{2} = 0.01$$

□ Mean_absolute_error

$$\frac{|0.9 - 0.8| + |0.1 - 0.2|}{2} = 0.1$$

□ Mean_absolute_percentage_error

$$\frac{|0.9 - 0.8|/|0.9| + |0.1 - 0.2|/|0.1|}{2} * 100 = 55$$

□ Mean_squared_logarithmic_error

$$\frac{[\log(0.9) - \log(0.8)]^2 + [\log(0.1) - \log(0.2)]^2}{2} * 100 = 0.247$$

常用於 Regression



Loss Function

Prediction	Answer
0.9	0
0.1	1

□ binary_crossentropy (logloss)

$$-\frac{1}{N} \sum_{n=1}^N [y_n \log(\hat{y}_n) + (1 - y_n) \log(1 - \hat{y}_n)]$$

$$-\frac{1}{2} [0 \log(0.9) + (1 - 0) \log(1 - 0.9) + 1 \log(0.1) + 0 \log(1 - 0.1)]$$

$$= -\frac{1}{2} [\log(0.1) + \log(0.1)] = -\log(0.1) = 2.302585$$

□ categorical_crossentropy

- 需要將 class 的表示方法改成 one-hot encoding

Category 1 → [0, 1, 0, 0, 0]

- 用簡單的函數 keras.utils.to_categorical

□ 常用於 classification



步驟 4: 選擇 optimizer

- SGD – Stochastic Gradient Descent
- Adagrad – Adaptive Learning Rate
- RMSprop – Similar with Adagrad
- Adam – Similar with RMSprop + Momentum
- Nadam – Adam + Nesterov Momentum

SGD: 基本款 optimizer

- Stochastic gradient descent
- 設定 learning rate, momentum, learning rate decay, Nesterov momentum

```
# 指定 optimizier
from keras.optimizers import SGD, Adam, RMSprop, Adagrad
sgd = SGD(lr=0.01, momentum=0.0, decay=0.0, nesterov=False)
```

- 設定 Learning rate by experiments (later)

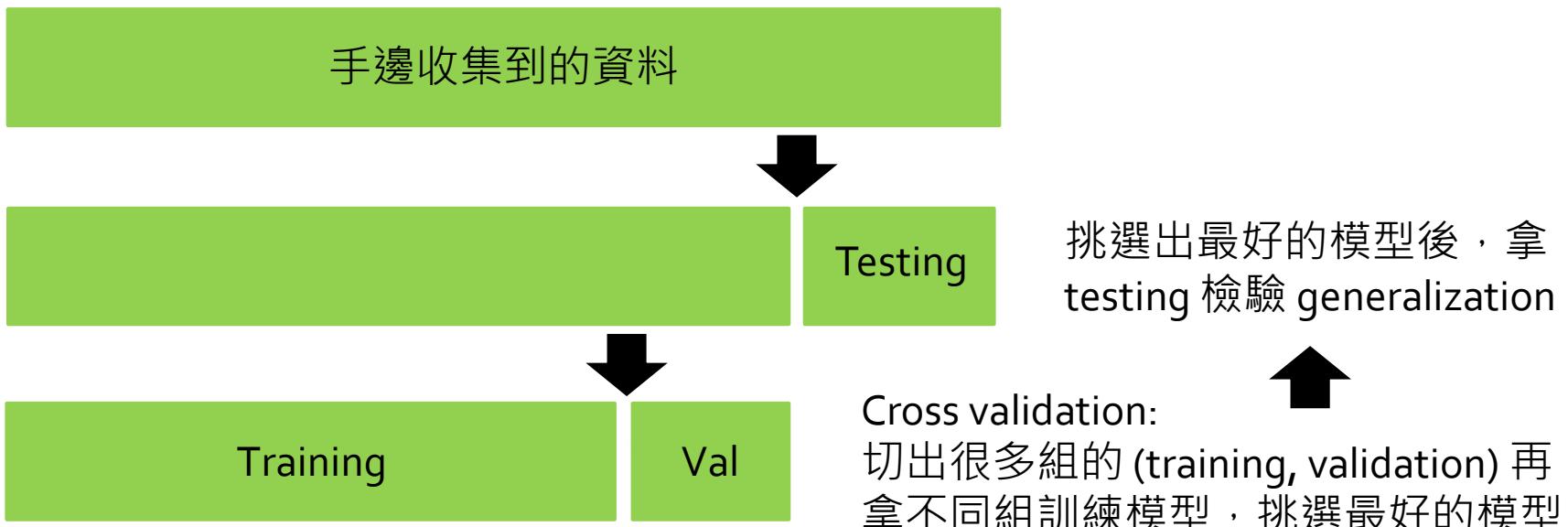
就決定是你了！

```
# 指定 loss function 和 optimizier  
model.compile(loss='categorical_crossentropy',  
                optimizer=sgd)
```

Validation Dataset

- Validation dataset 用來挑選模型
- Testing dataset 檢驗模型的普遍性 (generalization)
避免模型過度學習 training dataset

理論上



Validation Dataset

- 利用 model.fit 的參數 validation_split
 - 從輸入(X_train,Y_train) 取固定比例的資料作為 validation
 - 不會先 shuffle 再取 validation dataset
 - 固定從資料尾端開始取
 - 每個 epoch 所使用的 validation dataset 都相同

- 手動加入 validation dataset

validation_data=(X_valid, Y_valid)

Fit Model

```
# 指定 batch_size, nb_epoch, validation 後，開始訓練模型!!!
history = model.fit( X_train,
                      Y_train,
                      batch_size=16,
                      verbose=0,
                      epochs=30,
                      shuffle=True,
                      validation_split=0.1)
```

- batch_size: min-batch 的大小
- nb_epoch: epoch 數量
 - 1 epoch 表示看過全部的 training dataset 一次
- shuffle: 每次 epoch 結束後是否要打亂 training dataset
- verbose: 是否要顯示目前的訓練進度，0 為不顯示

練習 oo_firstModel.py (5 minutes)



Alternative: Functional API

- ❑ The way to go for defining a complex model
 - ❑ For example: multiple outputs, multiple input source
- ❑ Why “Functional API” ?
 - ❑ All layers and models are callable (like function call)

```
from keras.layers import Input, Dense  
  
input = Input(shape=(200,))  
output = Dense(10)(input)
```

- ❑ Example

```
# Sequential (依序的)深度學習模型
model = Sequential()
model.add(Dense(128, input_dim=200))
model.add(Activation('sigmoid'))
model.add(Dense(256))
model.add(Activation('sigmoid'))
model.add(Dense(5))
model.add(Activation('softmax'))
model.summary()
```

```
# Functional API
from keras.layers import Input, Dense
from keras.models import Model

input = Input(shape=(200,))
x = Dense(128,activation='sigmoid')(input)
x = Dense(256,activation='sigmoid')(x)
output = Dense(5,activation='softmax')(x)

# 定義 Model (function-like)
model = Model(inputs=[input], outputs=[output])
```

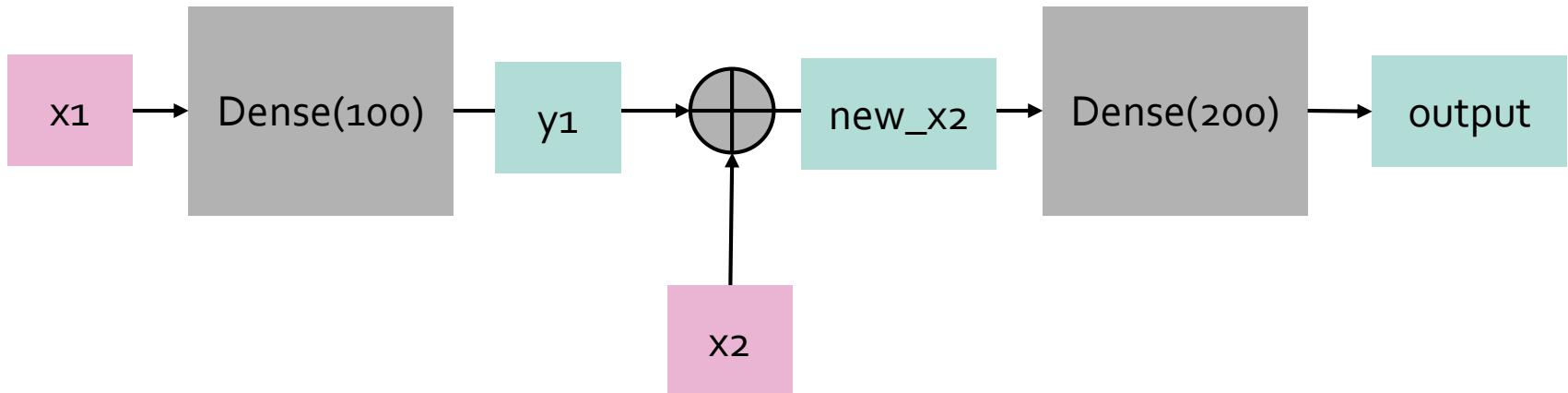
Good Use Case for Functional API (1)

- ❑ Model is callable as well, so it is easy to re-use the trained model
 - ❑ Re-use the architecture and weights as well

```
# If model and input is defined already  
# re-use the same architecture of the above model  
y1 = model(input)
```

Good Use Case for Functional API (2)

- ❑ Easy to manipulate various input sources



```
x1 = input(shape=(10,))
y1 = Dense(100)(x1)

x2 = input(shape=(20,))
new_x2 = keras.layers.concatenate([y1,x2])
output = Dense(200)(new_x2)

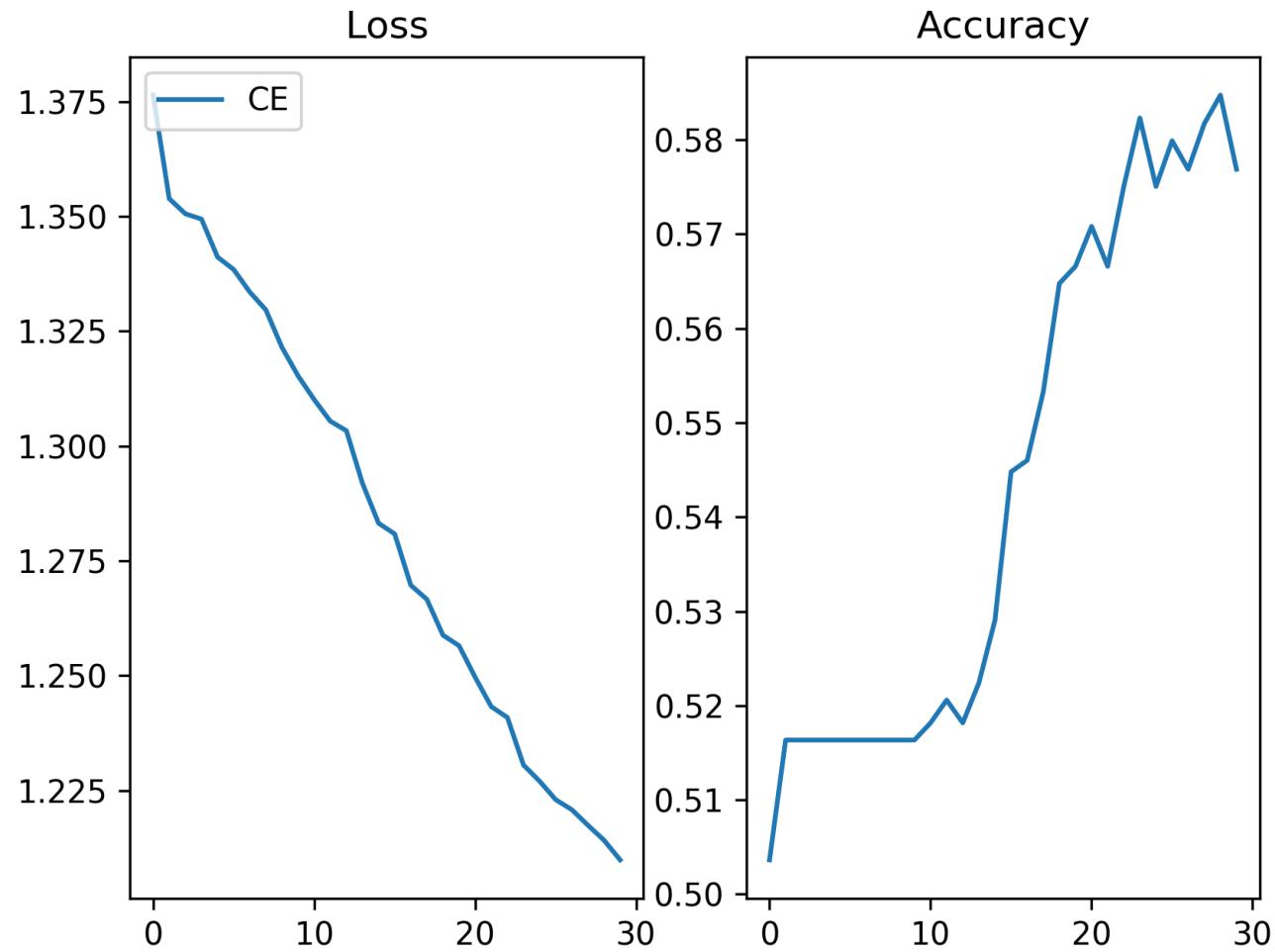
Model = Model(inputs=[x1,x2],outputs=[output])
```



Today

- Our exercise uses “Sequential” model because it is more straight-forward to understand the details of stacking layers

Result



這樣是好是壞？

- 我們選用最常見的

Component	Selection
Loss function	categorical_crossentropy
Activation function	sigmoid + softmax
Optimizer	SGD

用下面的招式讓模型更好吧



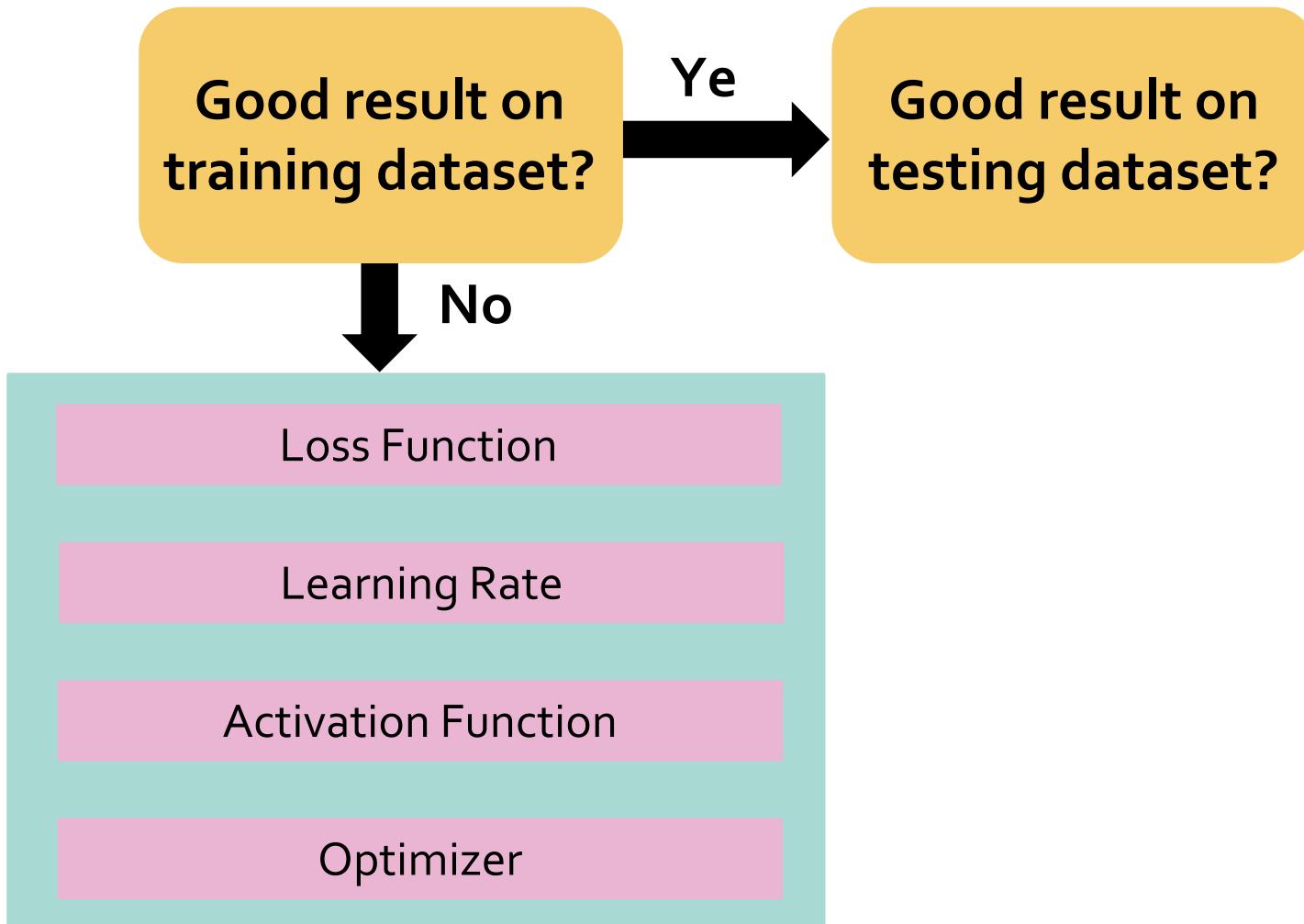


Tips for Training DL Models

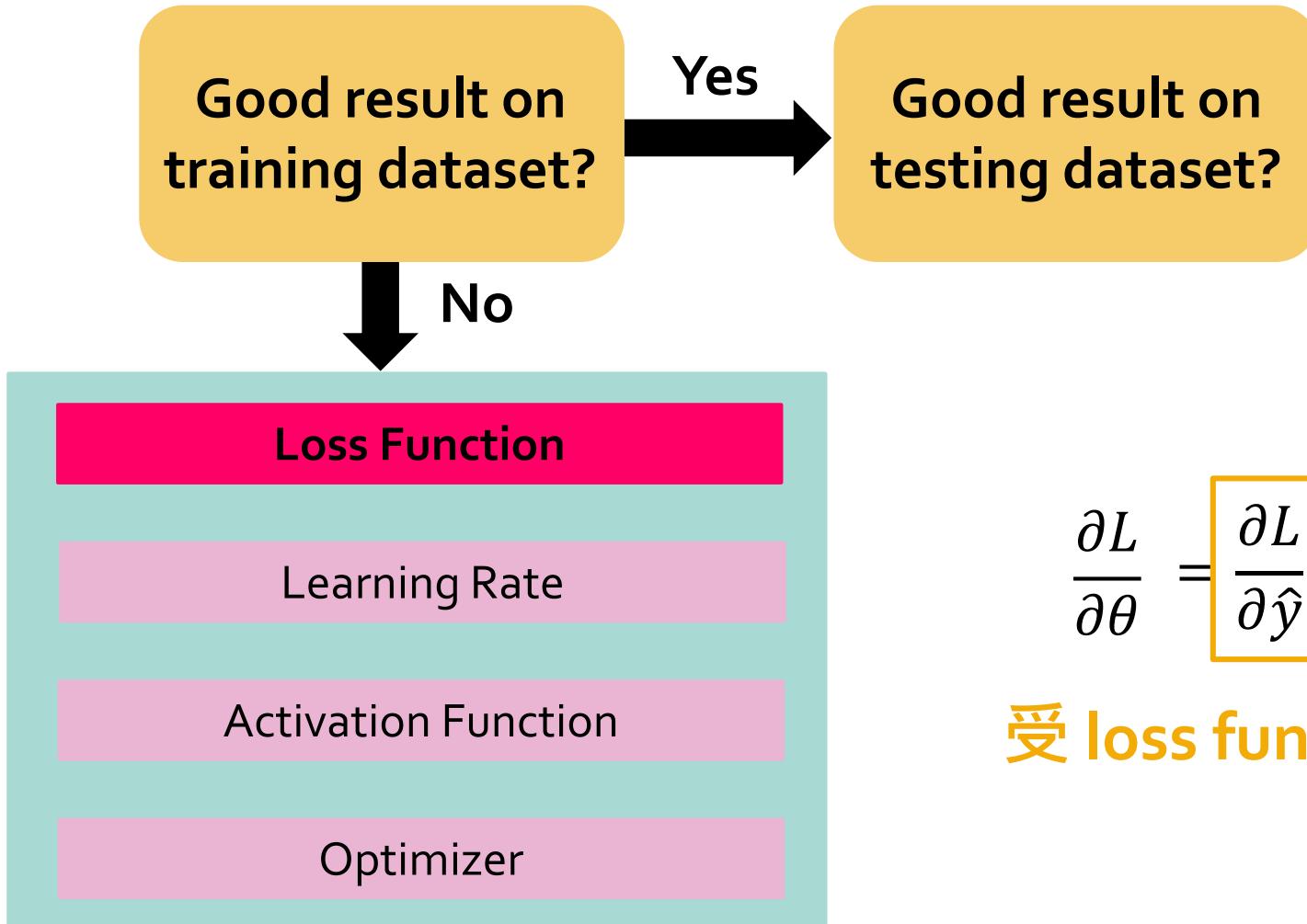
不過盲目的使用招式，會讓你的寶可夢失去戰鬥意識



Tips for Deep Learning



Tips for Deep Learning



$$\frac{\partial L}{\partial \theta} = \boxed{\frac{\partial L}{\partial \hat{y}}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial \theta}$$

受 loss function 影響



Using MSE

□ 在指定 loss function 時

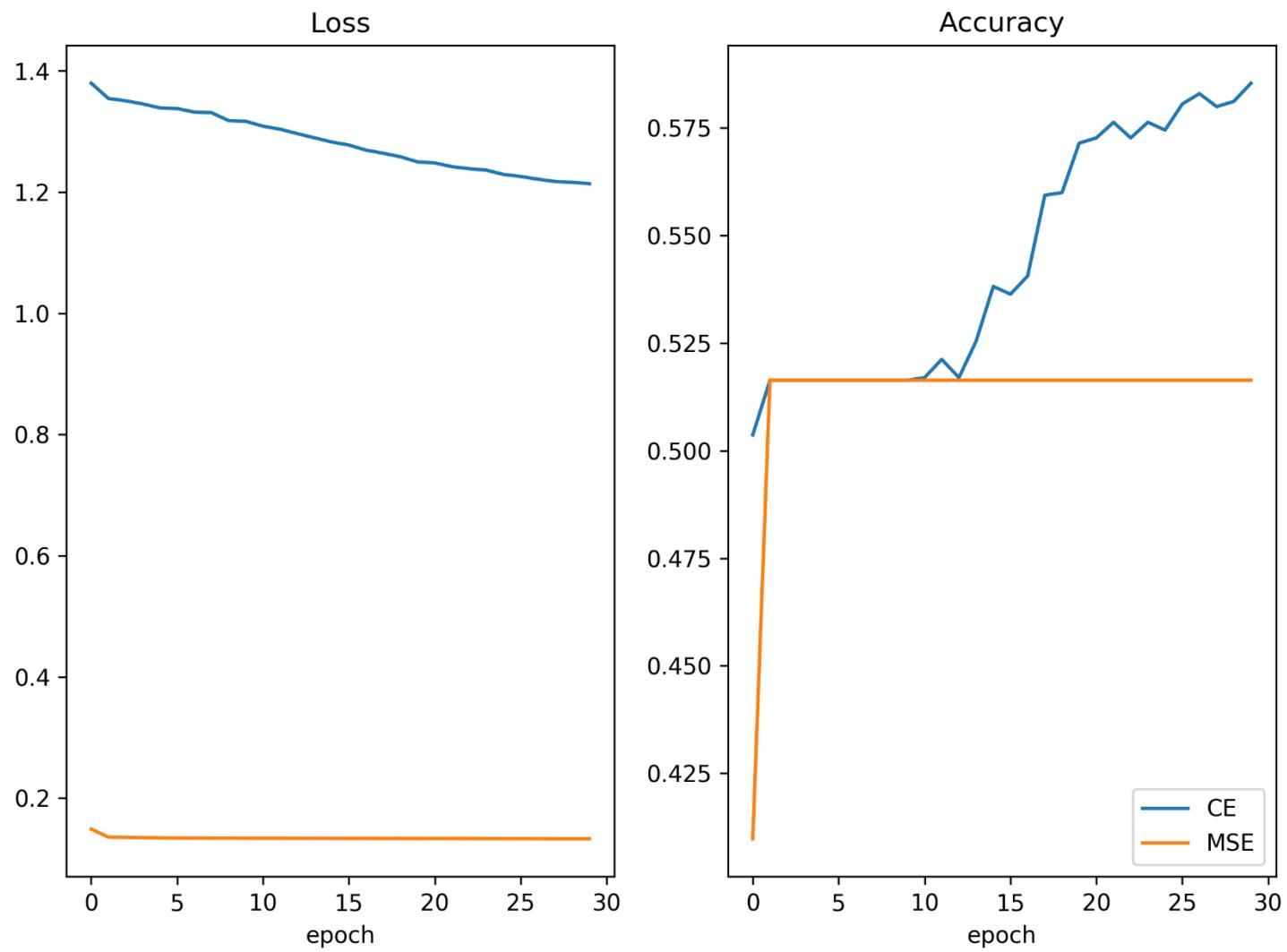
```
# 指定 loss function 和 optimizier  
model.compile(loss='categorical_crossentropy',  
                optimizer=sgd)
```



```
# 指定 loss function 和 optimizier  
model.compile(loss='mean_squared_error',  
                optimizer=sgd)
```

練習 01_lossFuncSelection.py (10 minutes)

Result – CE vs MSE



為什麼 Cross-entropy 比較好？

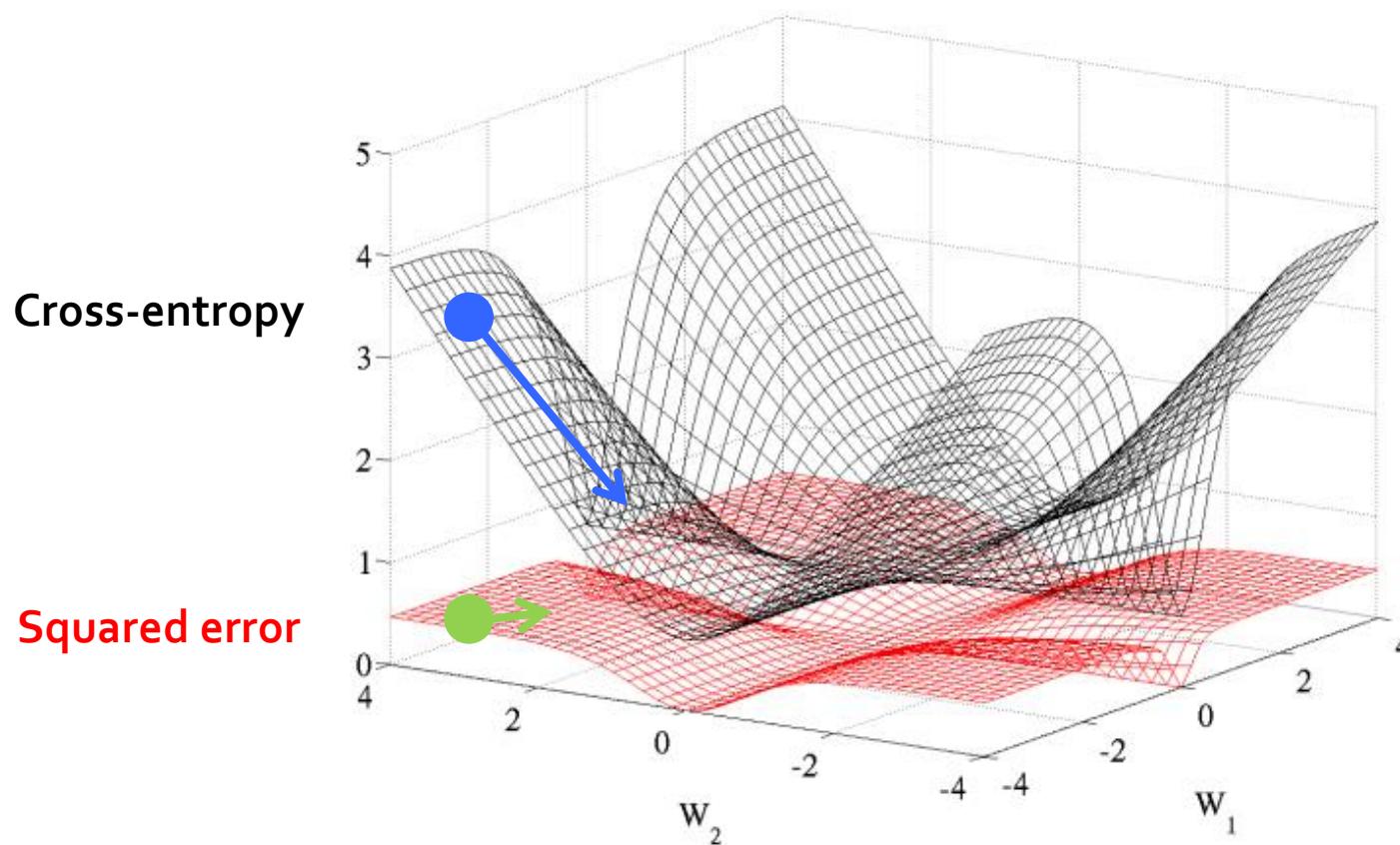


Figure source

The error surface of logarithmic functions is steeper than that of quadratic functions. [[ref](#)]



How to Select Loss function

- Classification 常用 cross-entropy
 - 搭配 softmax 當作 output layer 的 activation function
- Regression 常用 mean absolute/squared error
- 對特定問題定義 loss function
 - Unbalanced dataset, class 0 : class 1 = 99 : 1

Self-defined loss function

Loss	Class 0	Class 1
Class 0	0	99
Class 1	1	0

Current Best Model Configuration

Component	Selection
Loss function	categorical_crossentropy
Activation function	sigmoid + softmax
Optimizer	SGD

Tips for Deep Learning

Good result on
training data?

Ye

Good result on
testing data?

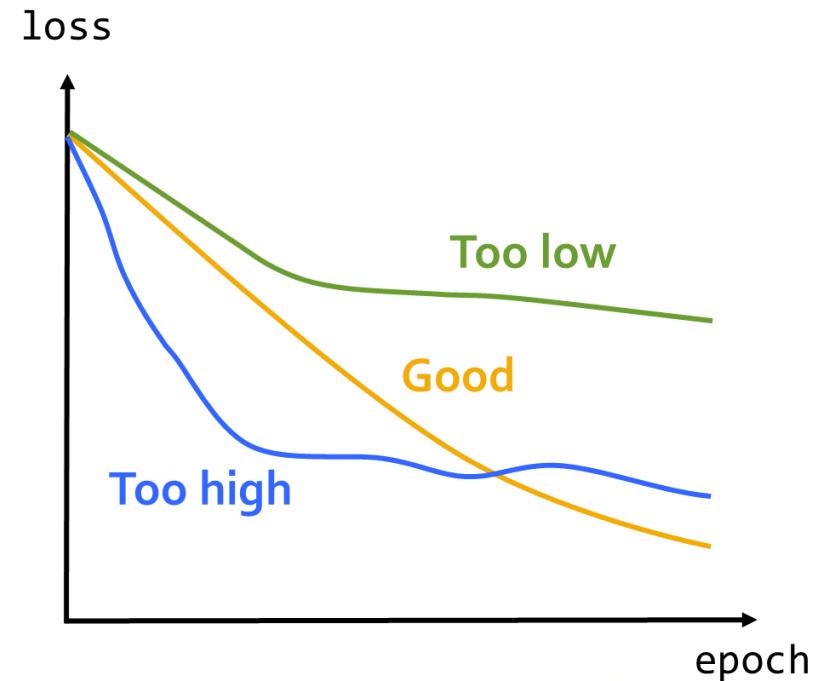
No

Loss Function

Learning Rate

Activation Function

Optimizer



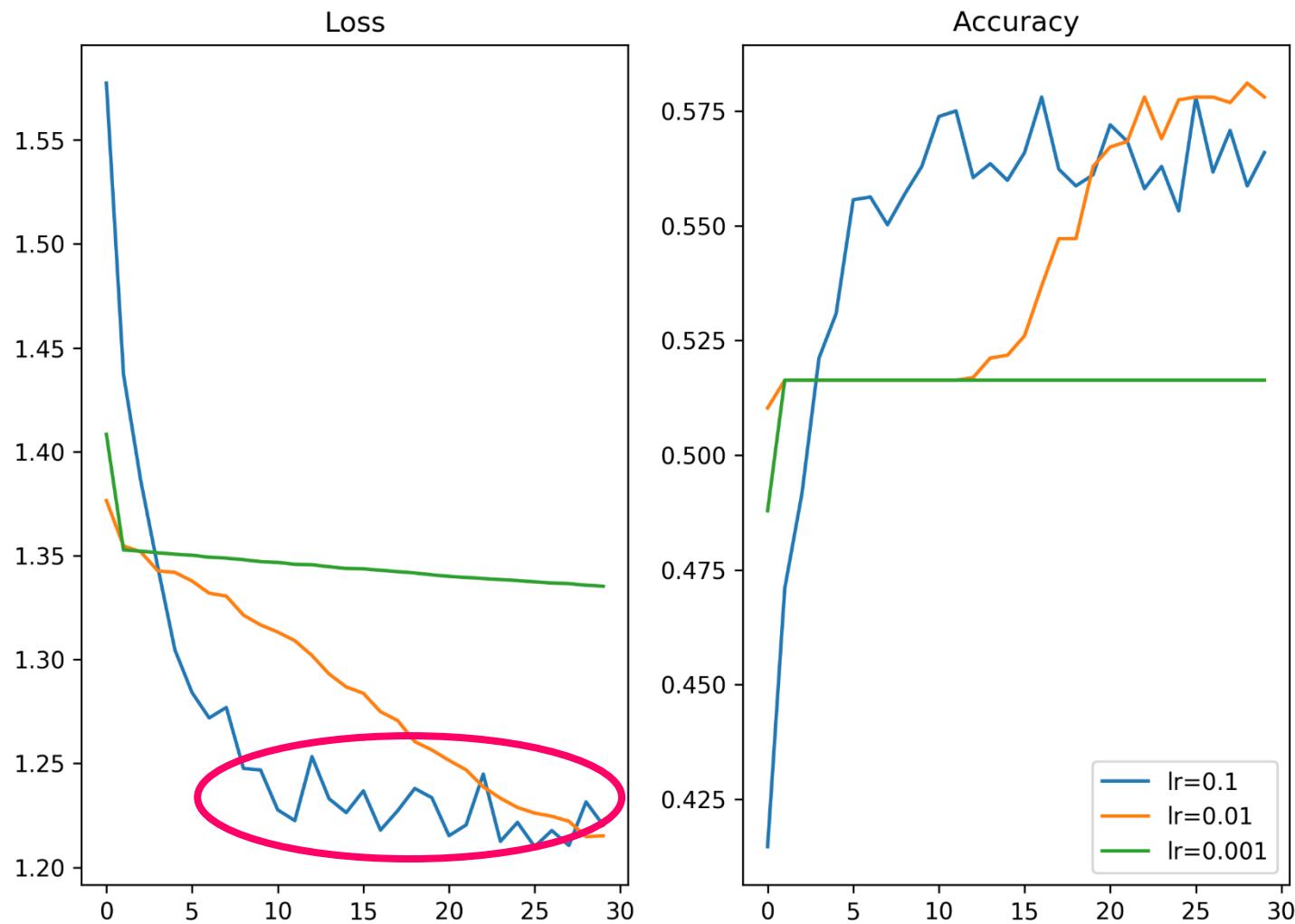
練習 o2_learningRateSelection.py (5-8 minutes)

```
# 指定 optimizier
from keras.optimizers import SGD, Adam, RMSprop, Adagrad
sgd = SGD(lr=0.01, momentum=0.0, decay=0.0, nesterov=False)
```



試試看改變 learning rate，挑選出最好的 learning rate。
建議一次降一個數量級，如: 0.1 vs 0.01 vs 0.001

Result – Learning Rate Selection

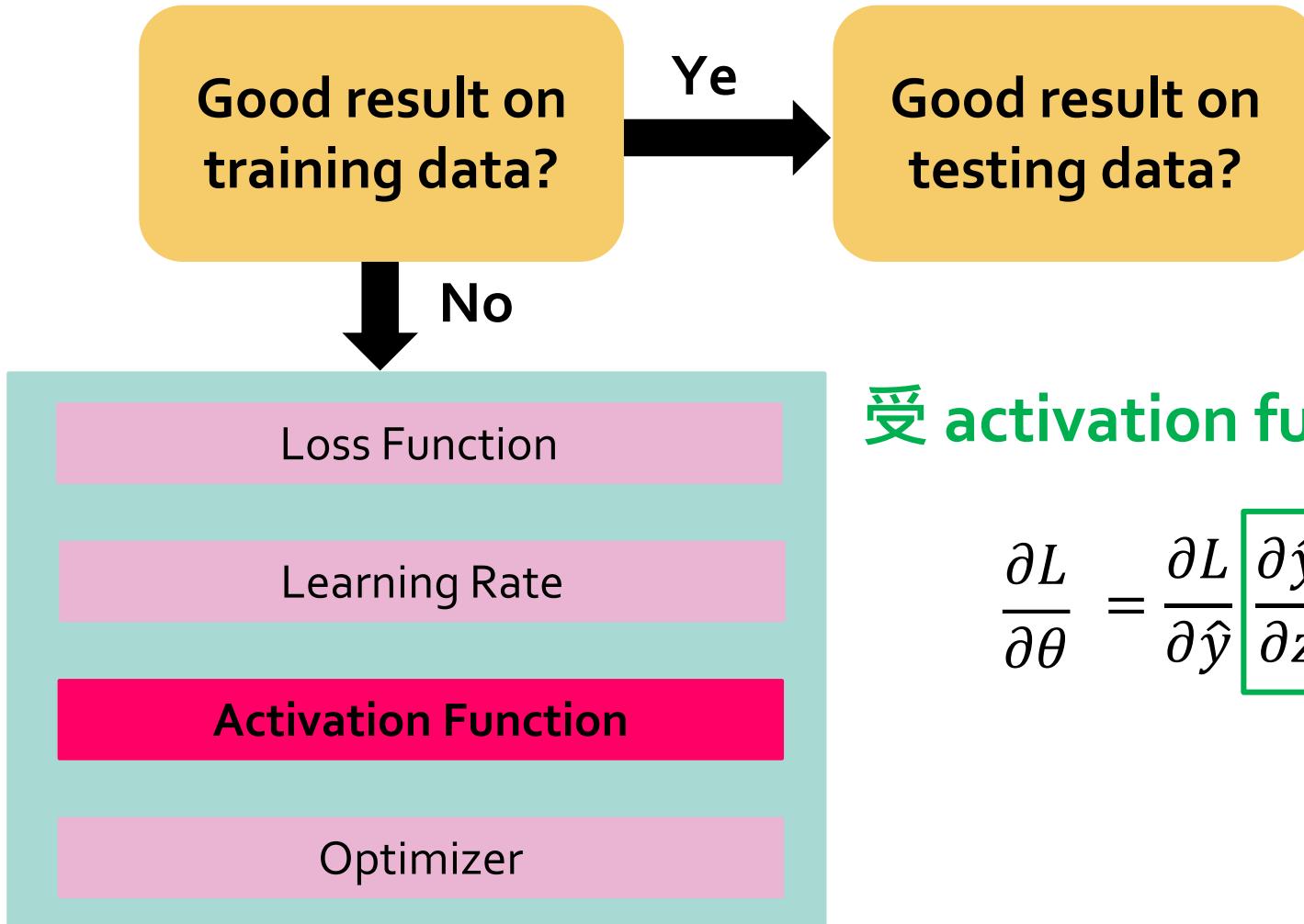


觀察 loss，這樣的震盪表示 learning rate 可能太大

How to Set Learning Rate

- 大多要試試看才知道，通常不會大於 0.1
- 一次調一個數量級
 - $0.1 \rightarrow 0.01 \rightarrow 0.001$
 - ~~$0.01 \rightarrow 0.012 \rightarrow 0.015 \rightarrow 0.018 \dots$~~
- 幸運數字！

Tips for Deep Learning



受 activation function 影響

$$\frac{\partial L}{\partial \theta} = \frac{\partial L}{\partial \hat{y}} \left[\frac{\partial \hat{y}}{\partial z} \right] \frac{\partial z}{\partial \theta}$$

Sigmoid, Tanh, Softsign

□ Sigmoid

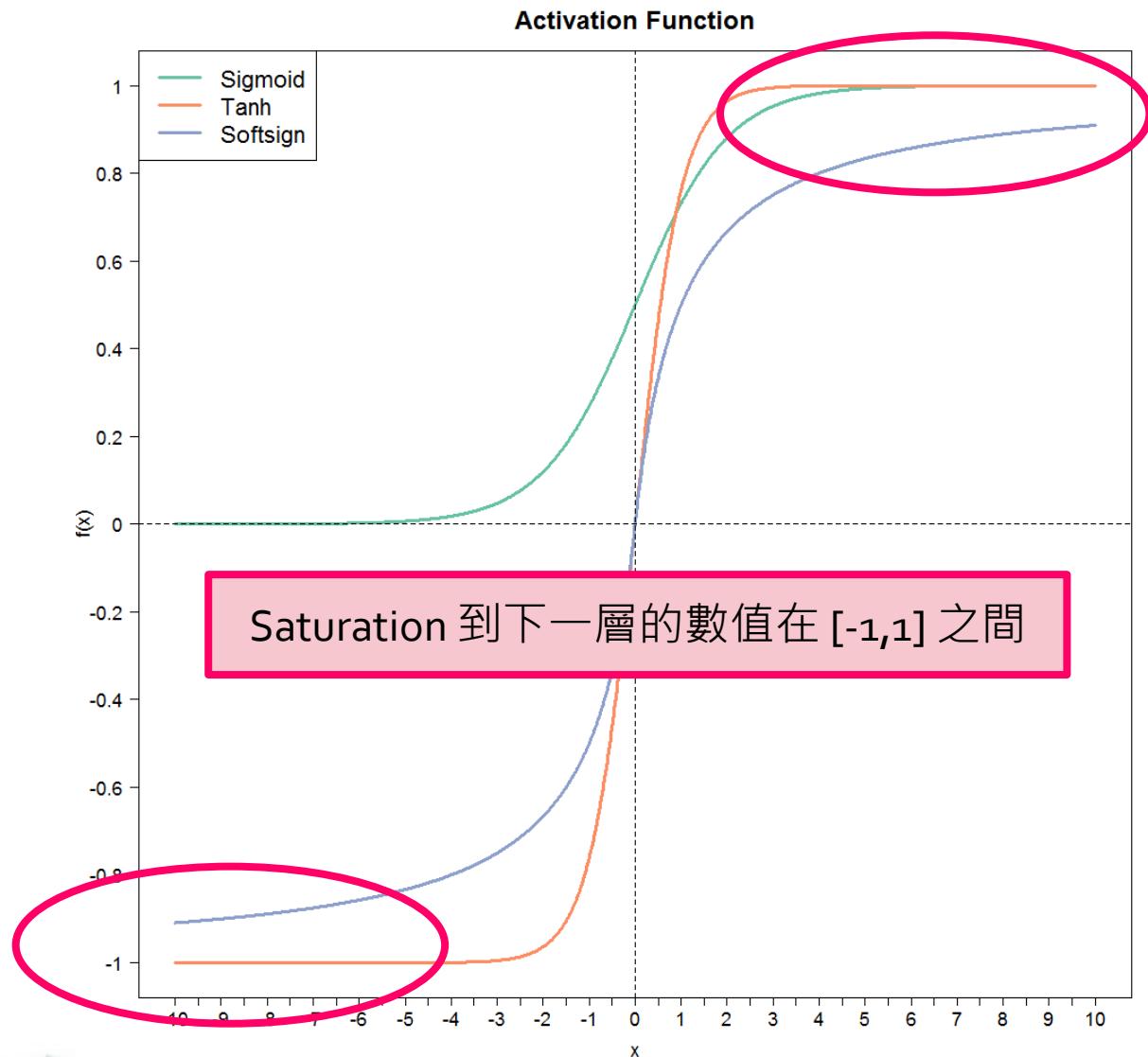
$$\square f(x) = \frac{1}{(1+e^{-x})}$$

□ Tanh

$$\square f(x) = \frac{(1-e^{-2x})}{(1+e^{-2x})}$$

□ Softsign

$$\square f(x) = \frac{x}{(1+|x|)}$$



Derivatives of Sigmoid, Tanh, Softsign

□ Sigmoid

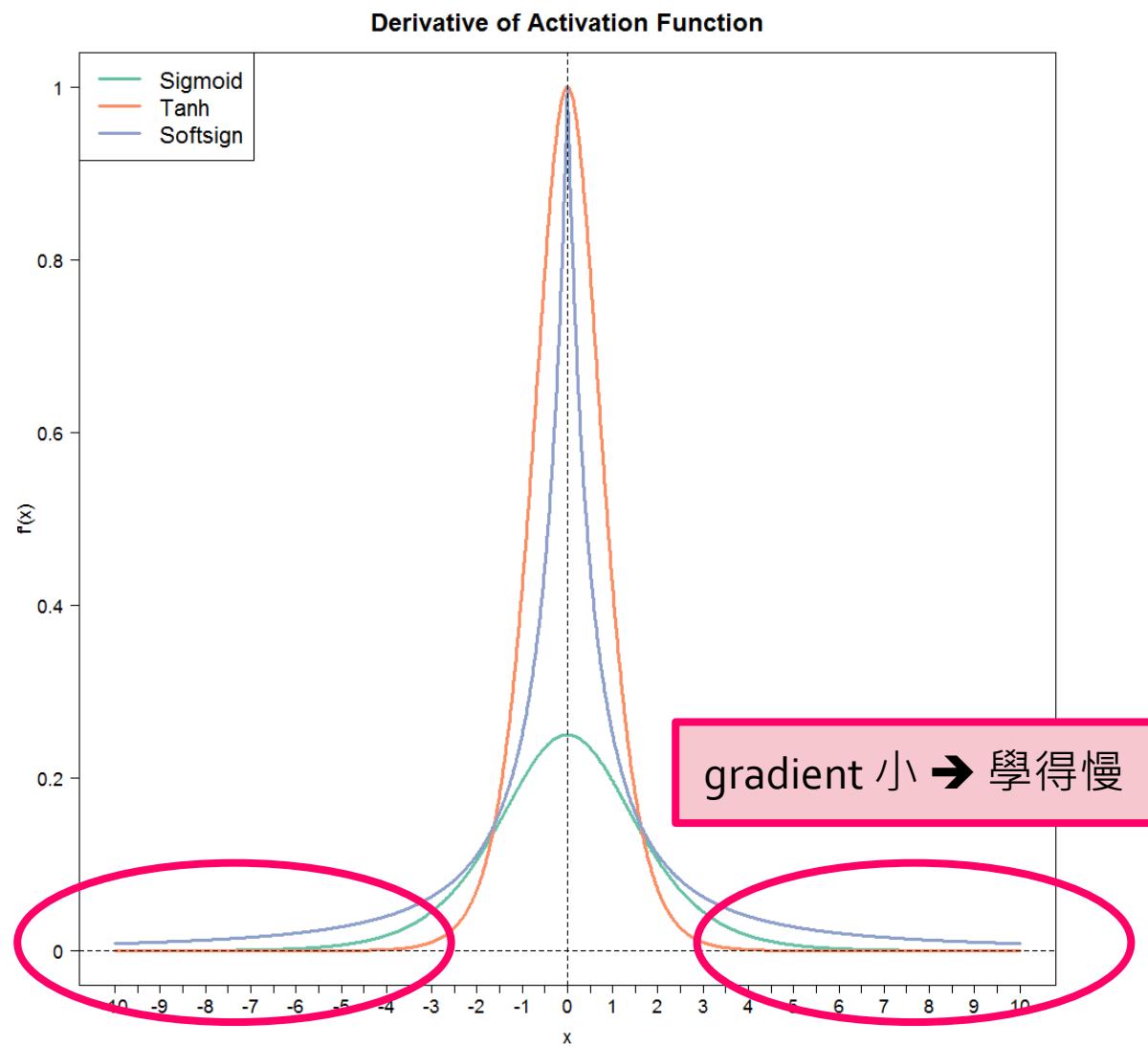
□ $\frac{df}{dx} = \frac{e^{-x}}{(1+e^{-x})^2}$

□ Tanh

□ $\frac{df}{dx} = 1-f(x)^2$

□ Softsign

□ $\frac{df}{dx} = \frac{1}{(1+|x|)^2}$



Drawbacks of Sigmoid, Tanh, Softsign

□ Vanishing gradient problem

- 原因: input 被壓縮到一個相對很小的 output range
- 結果: 很大的 input 變化只能產生很小的 output 變化
→ Gradient 小 → 無法有效地學習
- Sigmoid, Tanh, Softsign 都有這樣的特性

□ 特別不適用於深的深度學習模型

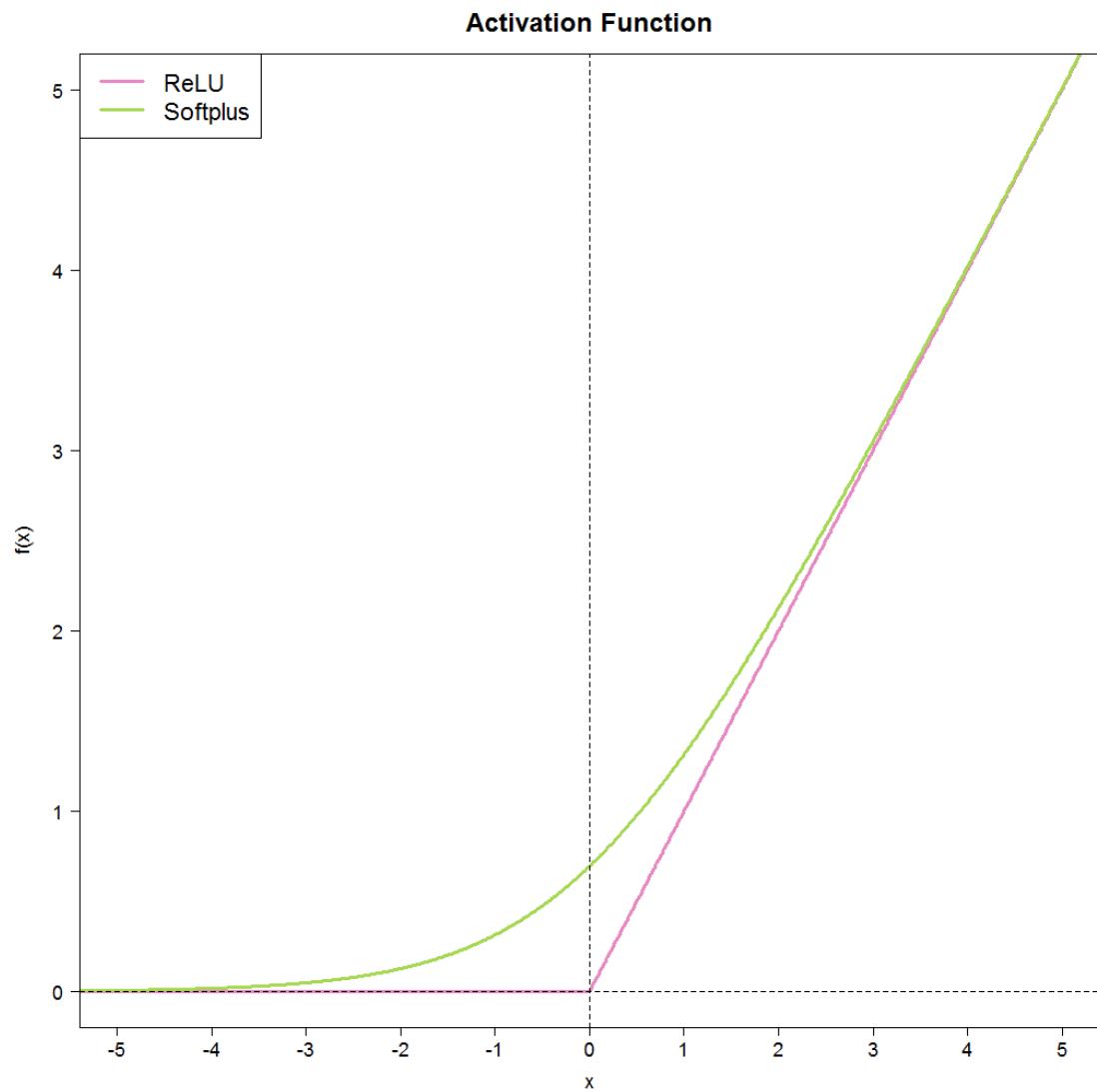
ReLU, Softplus

□ ReLU

- $f(x) = \max(0, x)$
- $df/dx = 1 \text{ if } x > 0,$
 0 otherwise.

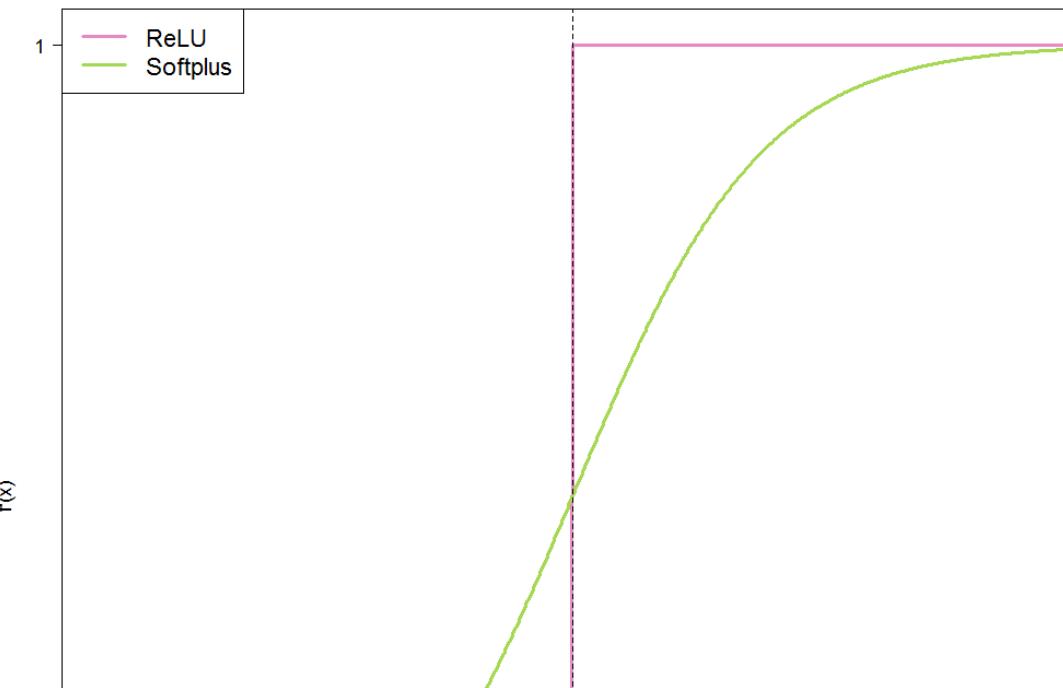
□ Softplus

- $f(x) = \ln(1+e^x)$
- $df/dx = e^x/(1+e^x)$

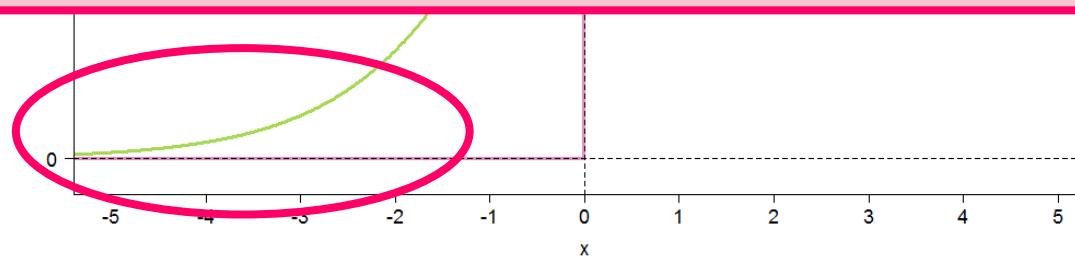


Derivatives of ReLU, Softplus

Derivative of Activation Function



ReLU 在輸入小於零時，gradient 等於零，會有問題嗎？

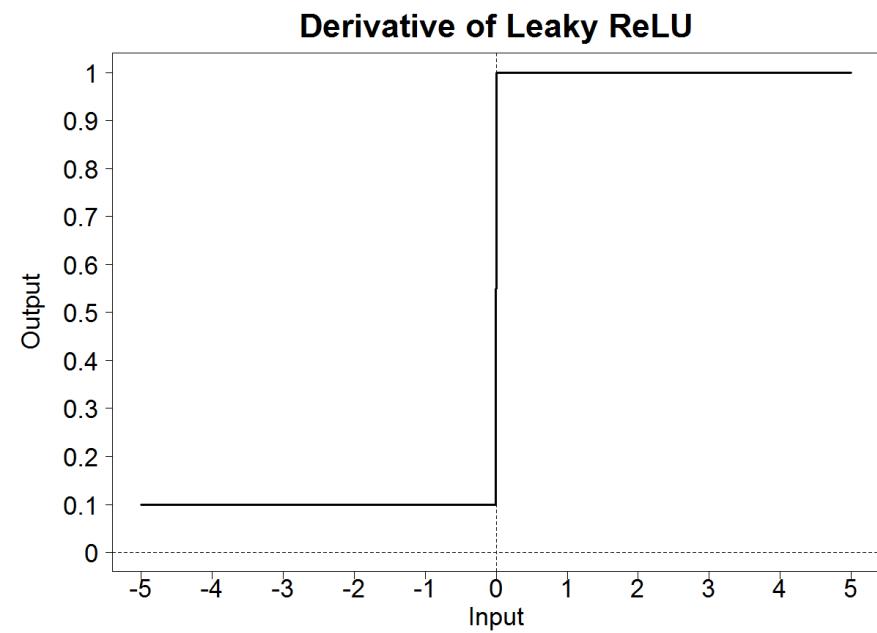
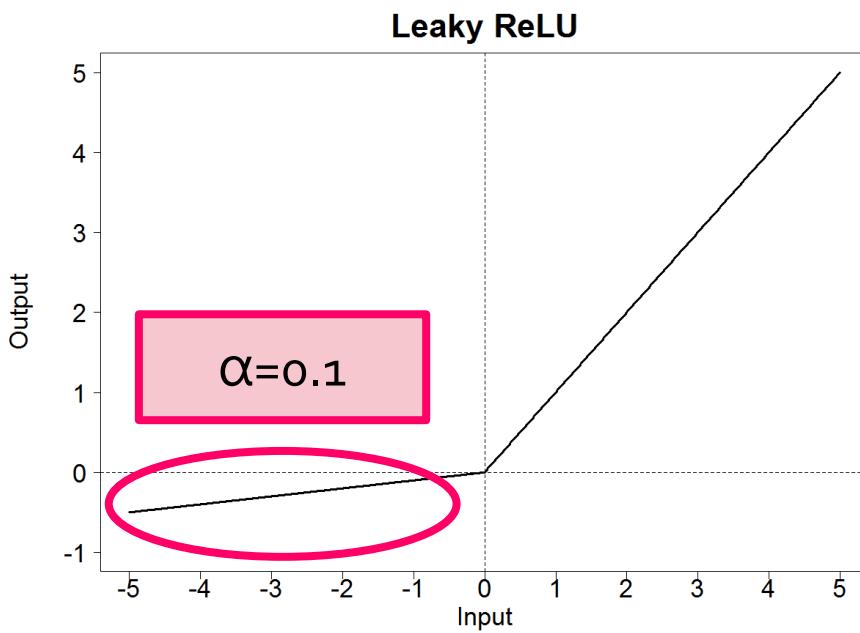


Leaky ReLU

- Allow a small gradient while the input to activation function smaller than 0

$$f(x) = \begin{cases} x & \text{if } x > 0, \\ \alpha x & \text{otherwise.} \end{cases}$$

$$\frac{df}{dx} = \begin{cases} 1 & \text{if } x > 0, \\ \alpha & \text{otherwise.} \end{cases}$$



Leaky ReLU in Keras

```
# For example
From keras.layers.advanced_activation import LeakyReLU
lrelu = LeakyReLU(alpha = 0.02)
model.add(Dense(128, input_dim = 200))
# 指定 activation function
model.add(lrelu)
```

- 更多其他的 activation functions

<https://keras.io/layers/advanced-activations/>

嘗試其他的 activation functions

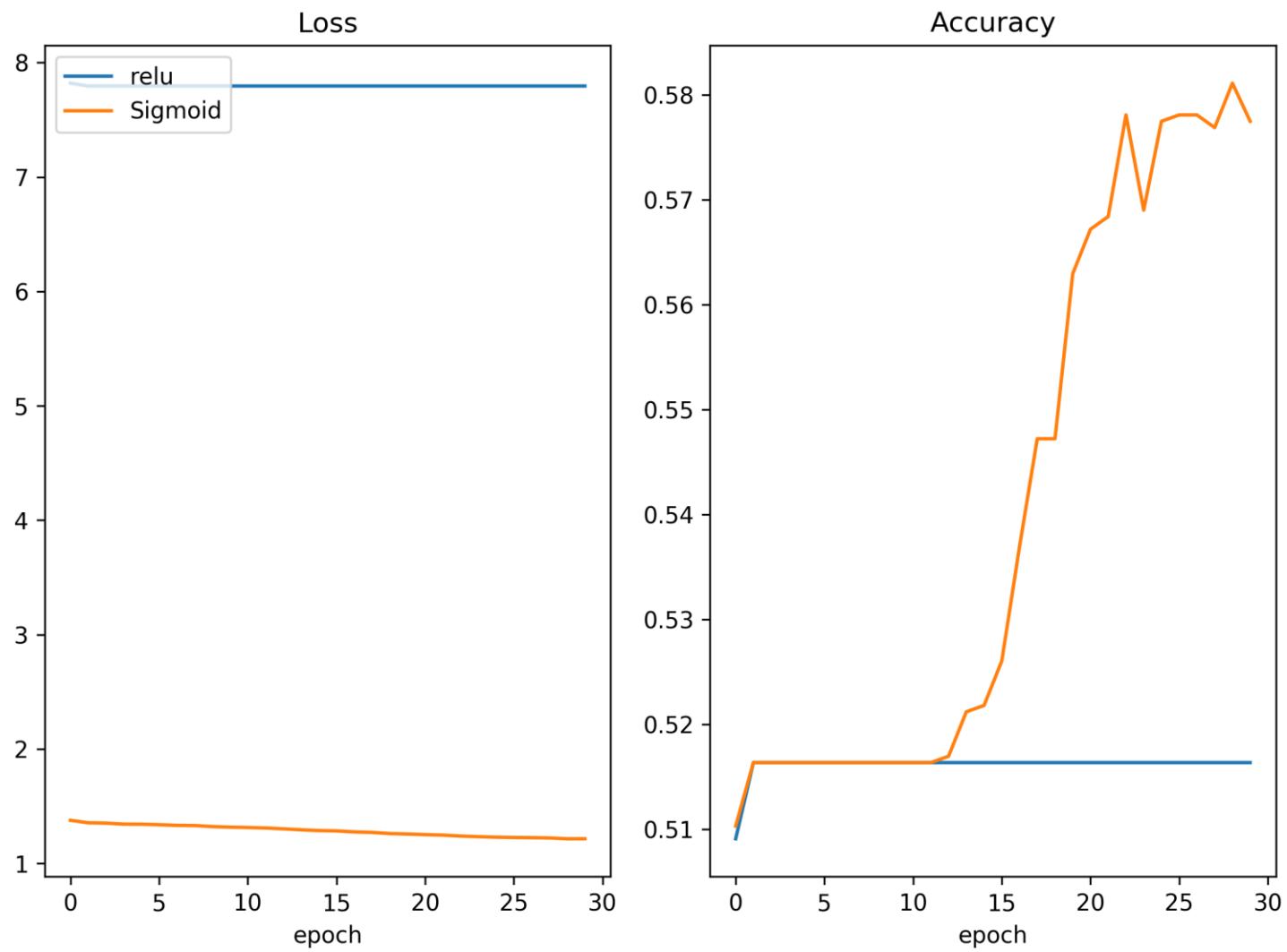
```
# 宣告這是一個 Sequential 次序性的深度學習模型
model = Sequential()

# 加入第一層 hidden layer (128 neurons) 與指定 input 的維度
model.add(Dense(128, input_dim=200))
# 指定 activation function
model.add(Activation('relu'))  
  
# 加入第二層 hidden layer (256 neurons)
model.add(Dense(256))
model.add(Activation('relu'))  
  
# 加入 output layer (5 neurons)
model.add(Dense(5))
model.add(Activation('softmax'))

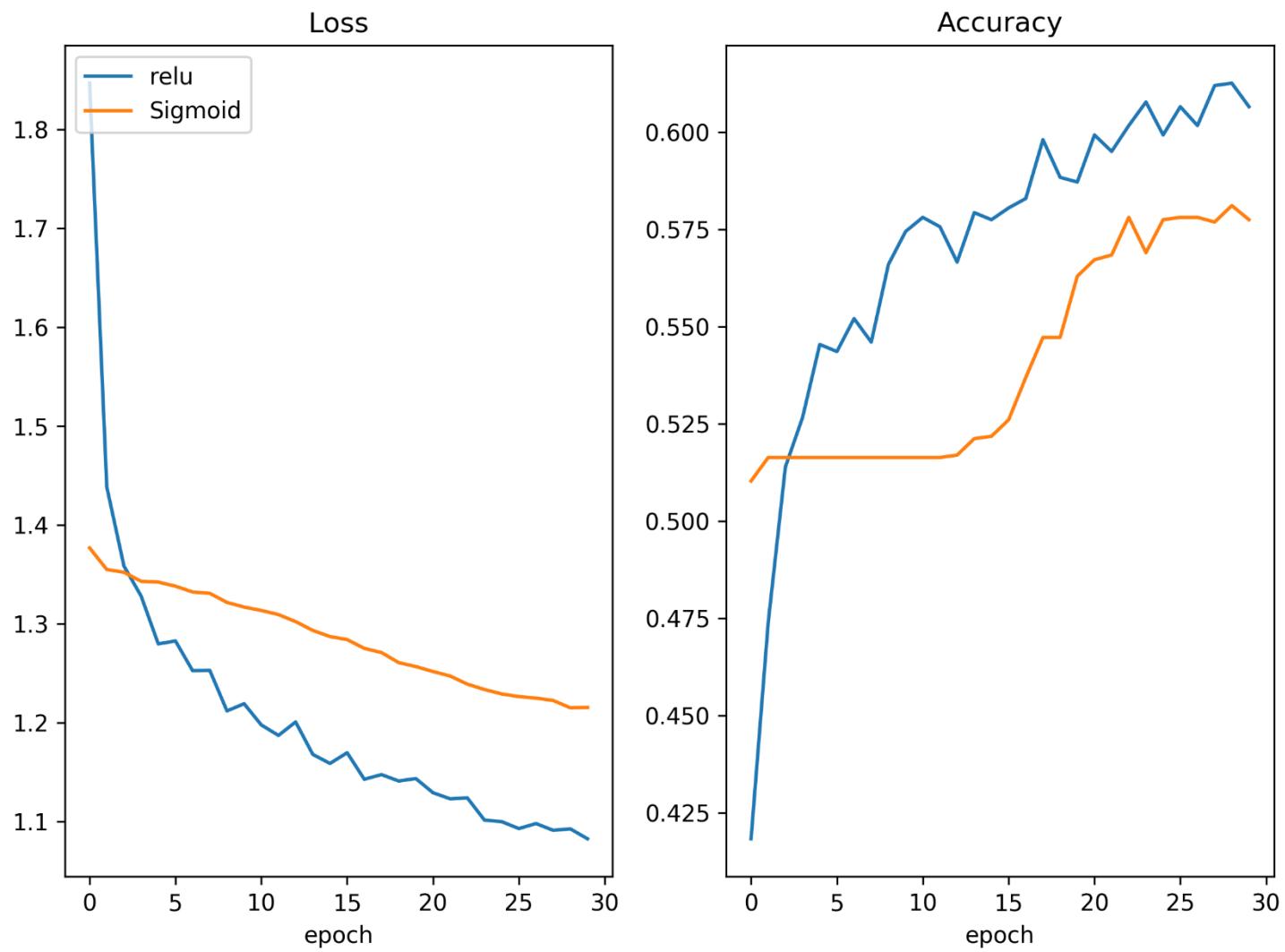
# 觀察 model summary
model.summary()
```

練習 03_activationFuncSelection.py (5-8 minutes)

Result – ReLU versus Sigmoid



Result – ReLU versus Sigmoid





How to Select Activation Functions

□ Hidden layers

- 通常會用 ReLU
- Sigmoid 有 vanishing gradient 的問題較不推薦

□ Output layer

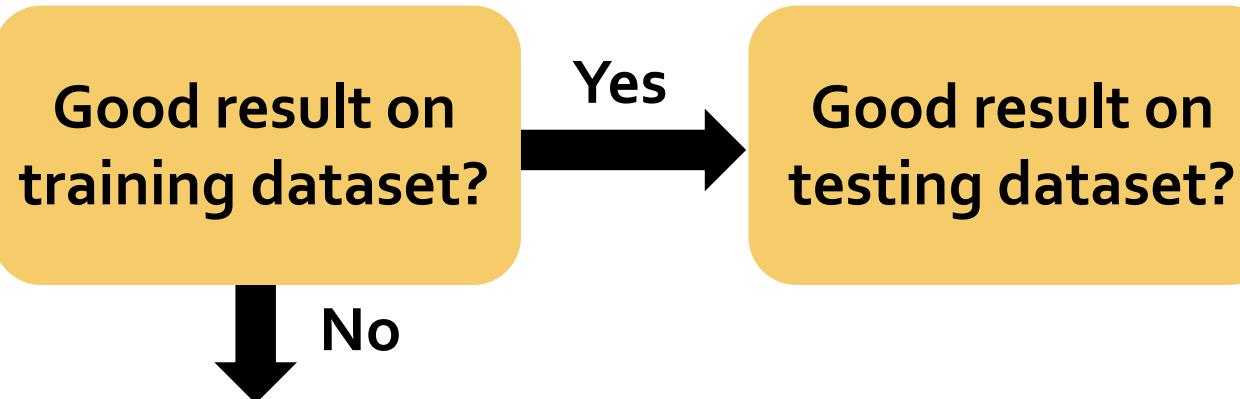
- Regression: linear
- Classification: softmax



Current Best Model Configuration

Component	Selection
Loss function	categorical_crossentropy
Activation function	relu + softmax
Optimizer	SGD

Tips for Deep Learning



Activation Function

Optimizer

Optimizers in Keras

- ❑ SGD – Stochastic Gradient Descent
- ❑ Adagrad – Adaptive Learning Rate
- ❑ RMSprop – Similar with Adagrad
- ❑ Adam – Similar with RMSprop + Momentum
- ❑ Nadam – Adam + Nesterov Momentum

Optimizer – SGD

- ❑ Stochastic gradient descent
- ❑ 支援 momentum, learning rate decay, Nesterov momentum

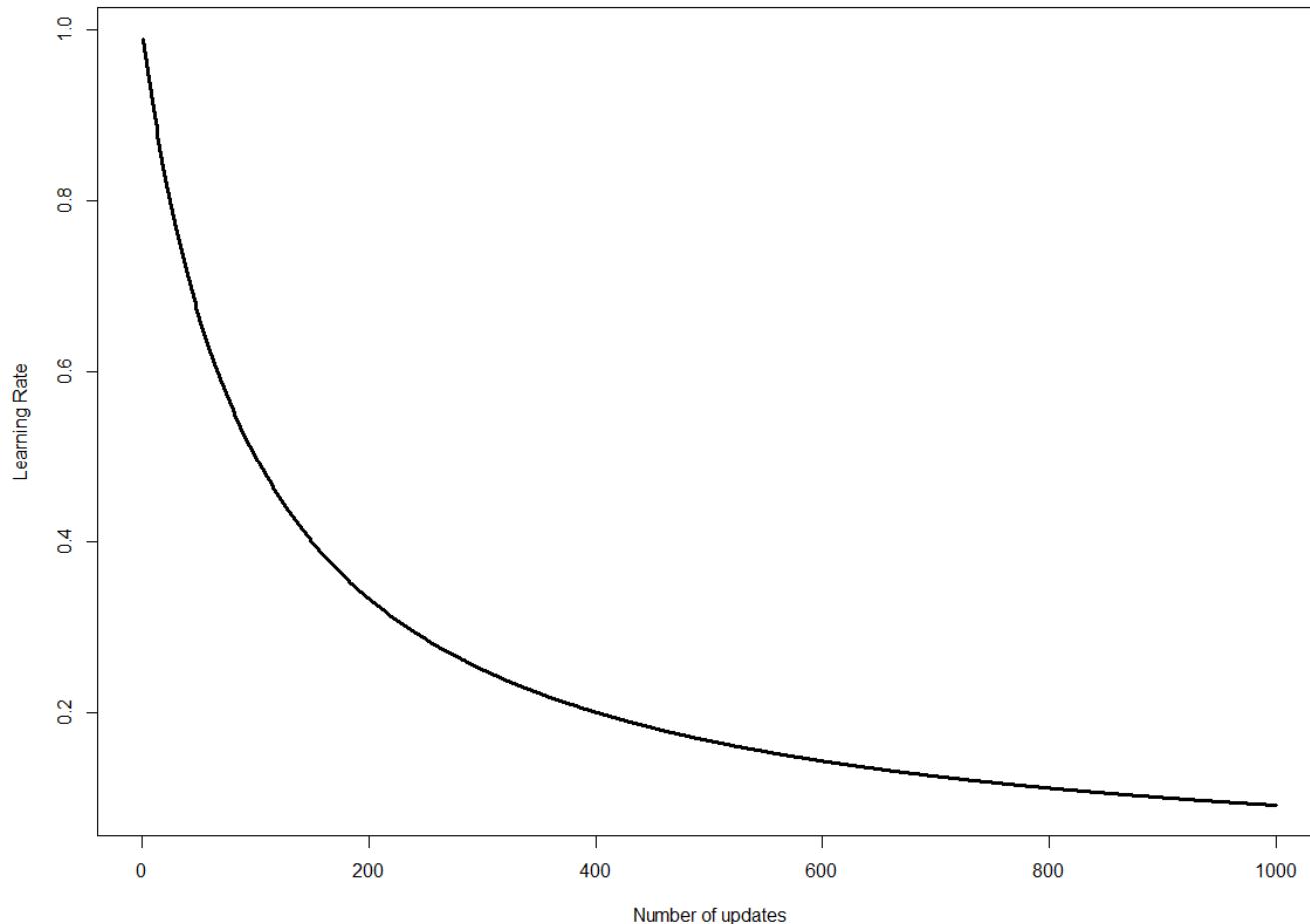
```
keras.optimizers.SGD(lr=0.01, momentum=0.0, decay=0.0, nesterov=False)
```

- ❑ Momentum 的影響
 - ❑ 無 momentum: **update = -lr*gradient**
 - ❑ 有 momentum: **update = -lr*gradient + m*last_update**
- ❑ Learning rate decay after update once
 - ❑ 屬於 $1/t$ decay → **lr = lr / (1 + decay*t)**
 - ❑ t: number of done updates

Learning Rate with $1/t$ Decay

$$\text{lr} = \text{lr} / (1 + \text{decay} * t)$$

Learning rate=1; Decay=0.01



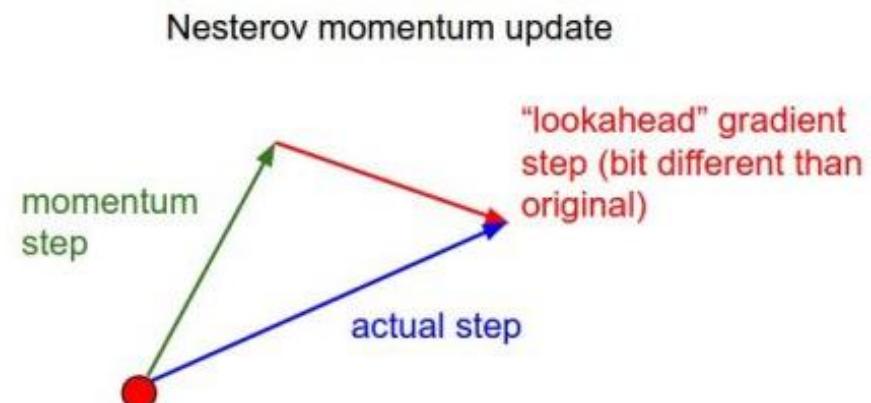
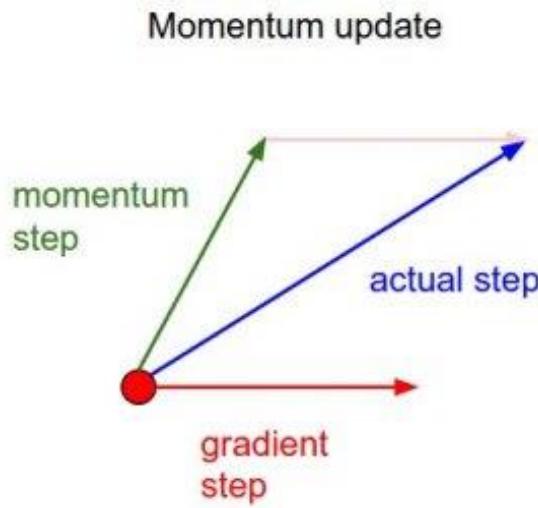
Nesterov Momentum

Momentum

- 先算 gradient
- 加上 momentum
- 更新

Nesterov momentum

- 加上 momentum
- 再算 gradient
- 更新



Optimizer – Adagrad

- 因材施教：每個參數都有不同的 learning rate
- 根據之前所有 gradient 的 root mean square 修改

第 t 次更新

$$g^t = \frac{\partial L}{\partial \theta} \Big|_{\theta=\theta^t}$$

Gradient descent

$$\theta^{t+1} = \theta^t - \eta g^t$$

Adagrad

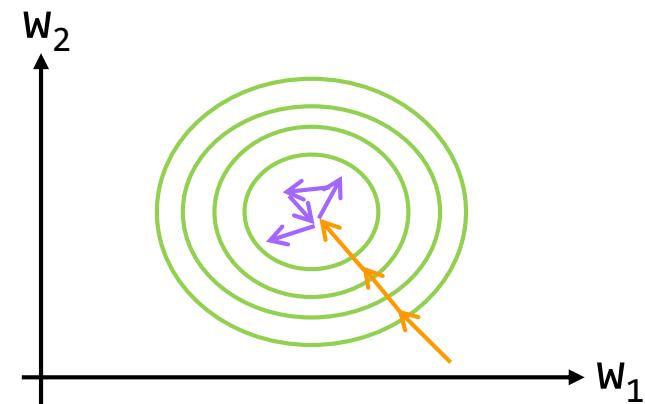
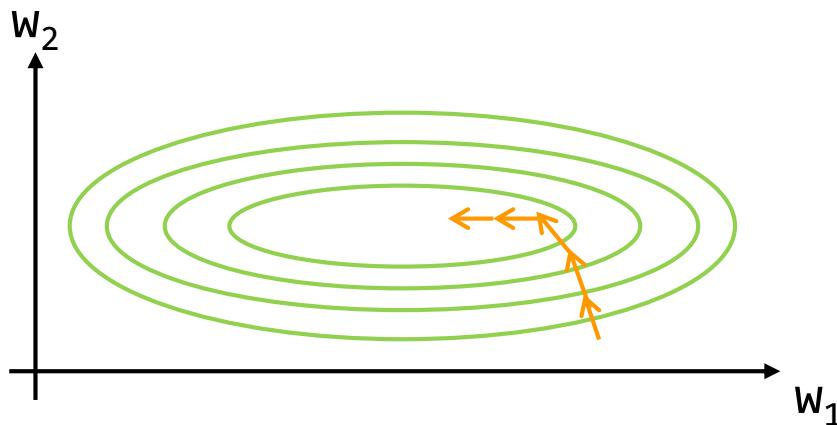
$$\theta^{t+1} = \theta^t - \frac{\eta}{\sigma^t} g^t$$

所有 gradient 的 root mean square

$$\sigma^t = \sqrt{\frac{(g^0)^2 + \dots + (g^t)^2}{t + 1}}$$

Adaptive Learning Rate

- Feature scales 不同，需要不同的 learning rates



- 每個 weight 收斂的速度不一致
 - 但 learning rate 沒有隨著減少的話 → **bumpy**
- 因材施教：每個參數都有不同的 learning rate



Optimizer – Adagrad

- 根据之前所有 gradient 的 root mean square 修改

第 t 次更新

$$g^t = \frac{\partial L}{\partial \theta} \Big|_{\theta=\theta^t}$$

Gradient descent

$$\theta^{t+1} = \theta^t - \eta g^t$$

Adagrad

$$\theta^{t+1} = \theta^t - \frac{\eta}{\sigma^t} g^t$$



所有 gradient 的 root mean square

$$\sigma^t = \sqrt{\frac{(g^0)^2 + \dots + (g^t)^2}{t + 1}}$$

Step by Step – Adagrad

$$\theta^1 = \theta^0 - \frac{\eta}{\sigma^0} g^0$$

$$\theta^2 = \theta^1 - \frac{\eta}{\sigma^1} g^1$$

$$\theta^t = \theta^{t-1} - \frac{\eta}{\sigma^{t-1}} g^{t-1}$$

$$\sigma^0 = \sqrt{(g^0)^2}$$

$$\sigma^1 = \sqrt{\frac{(g^0)^2 + (g^1)^2}{2}}$$

$$\sigma^t = \sqrt{\frac{(g^0)^2 + (g^1)^2 + \dots + (g^t)^2}{t + 1}}$$

□ g^t 是一階微分，那 σ^t 隱含什麼資訊？

An Example of Adagrad

g^t	g^0	g^1	g^2	g^3
W_1	0.001	0.003	0.002	0.1
W_2	1.8	2.1	1.5	0.1
σ^t	σ^0	σ^1	σ^2	σ^3
W_1	0.001	0.002	0.002	0.05
W_2	1.8	1.956	1.817	1.57
g^t/σ^t	t=0	t=1	t=2	t=3
W_1	1	1.364	0.952	2
W_2	1	1.073	0.826	0.064

- 老馬識途，參考之前的經驗修正現在的步伐
- 不完全相信當下的 gradient

Optimizer – RMSprop

Adagrad

$$\theta^{t+1} = \theta^t - \frac{\eta}{\sigma^t} g^t$$

$$\sigma^t = \sqrt{\frac{(g^0)^2 + \dots + (g^t)^2}{t + 1}}$$

RMSprop

$$\theta^{t+1} = \theta^t - \frac{\eta}{\sqrt{r^t}} g^t$$

$$r^t = (1 - \rho)(g^t)^2 + \rho r^{t-1}$$

- 另一種參考過去 gradient 的方式

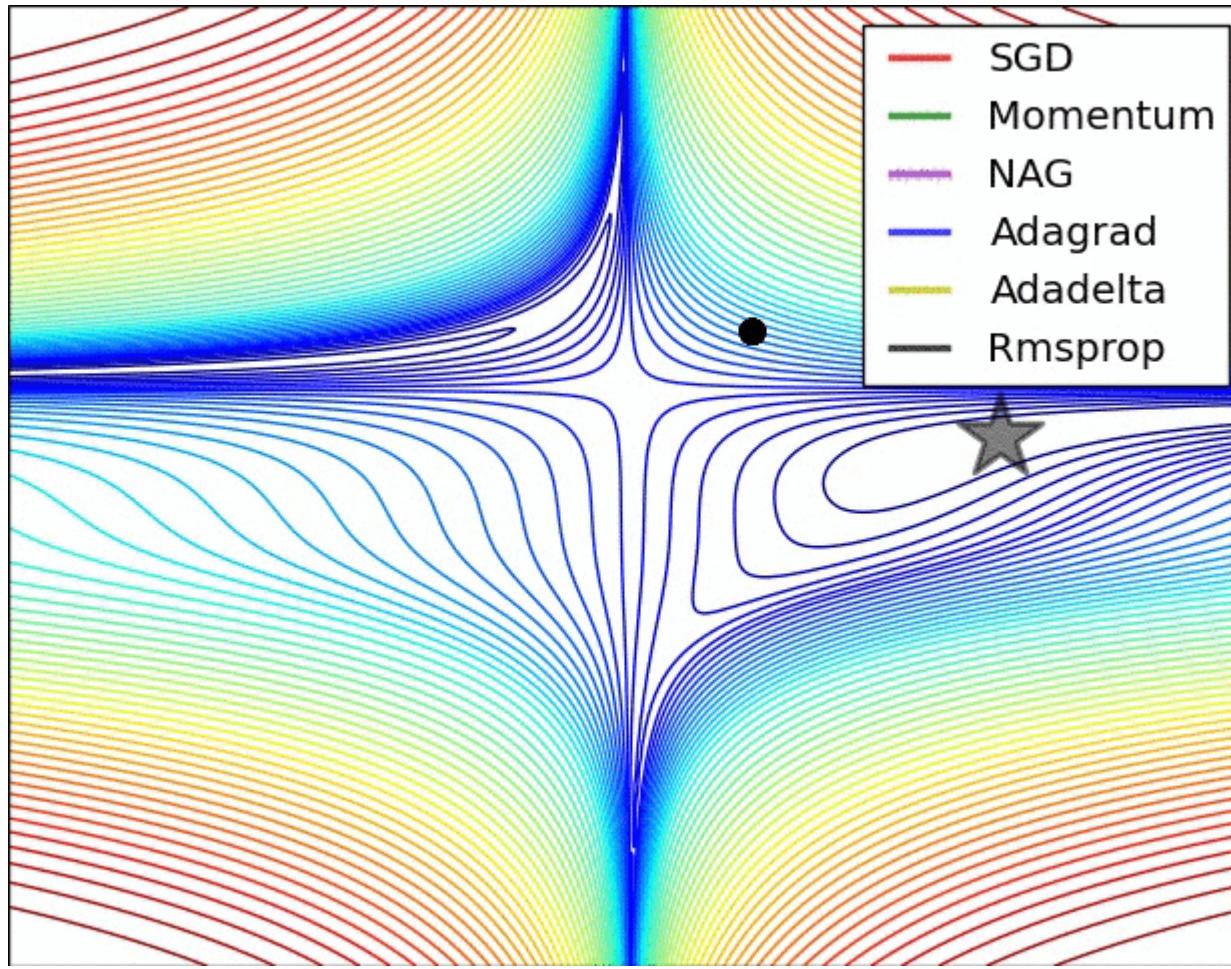
```
keras.optimizers.RMSprop(lr=0.001, rho=0.9, epsilon=1e-08, decay=0.0)
```

Optimizer – Adam

- Close to RMSprop + Momentum
- ADAM: A Method For Stochastic Optimization
- In practice, 不改參數也會做得很好

```
keras.optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-08, decay=0.0)
```

比較 Optimizers



來源



練習 04_optimizerSelection.py (5-8 minutes)

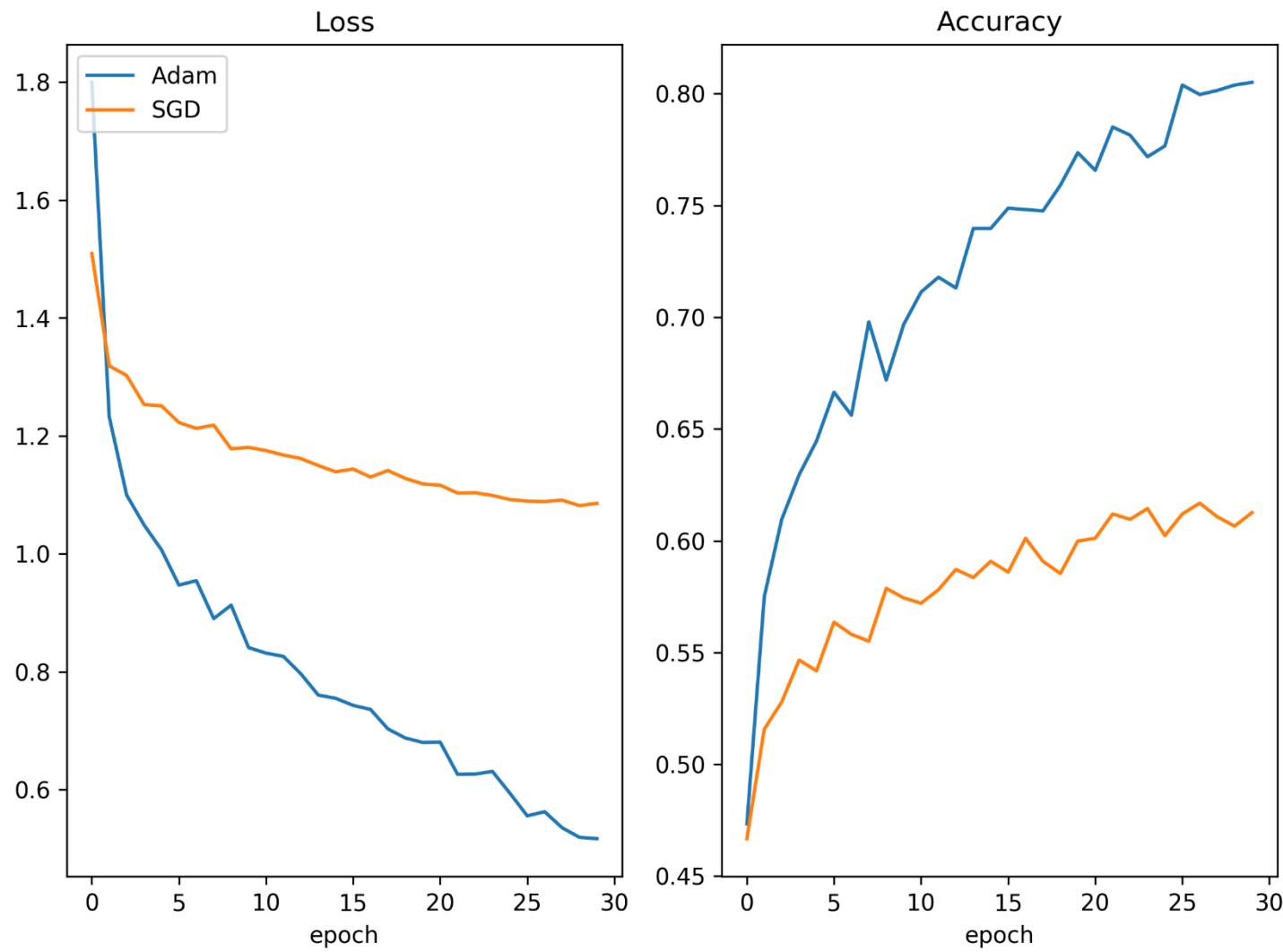
1. 設定選用的 optimizer

```
# 指定 optimizier
from keras.optimizers import SGD, Adam, RMSprop, Adagrad
sgd = SGD(lr=0.01, momentum=0.0, decay=0.0, nesterov=False)
```

2. 修改 model compilation

```
# 指定 loss function 和 optimizier
model.compile(loss='categorical_crossentropy',
                optimizer=sgd)
```

Result – Adam versus SGD



How to Select Optimizers

□ 一般的起手式：Adam

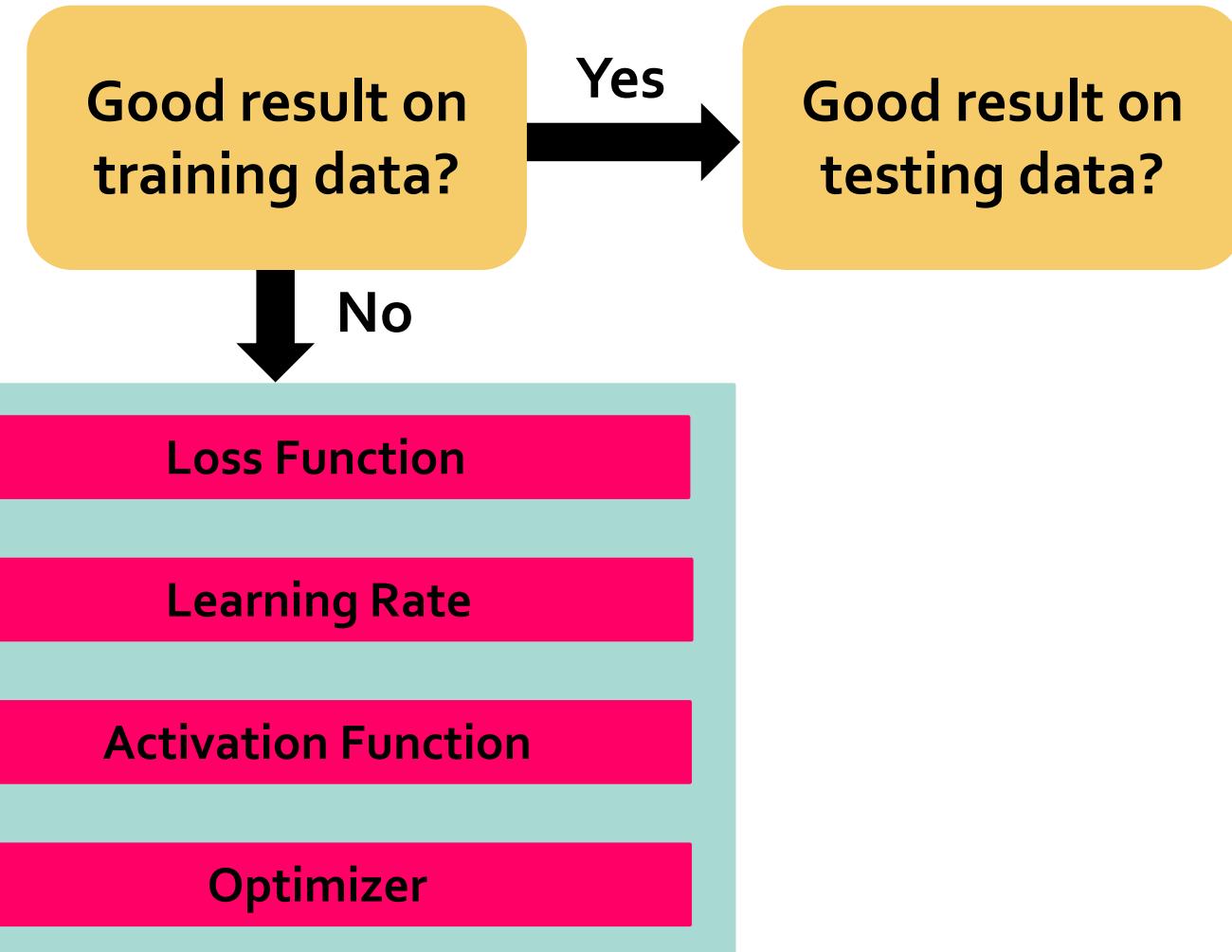
- Adaptive learning rate for every weights
- Momentum included

□ Keras 推薦 RNN 使用 RMSProp

- 在訓練 RNN 需要注意 explosive gradient 的問題
→ clip gradient 的暴力美學

□ RMSProp 與 Adam 的戰爭仍在延燒

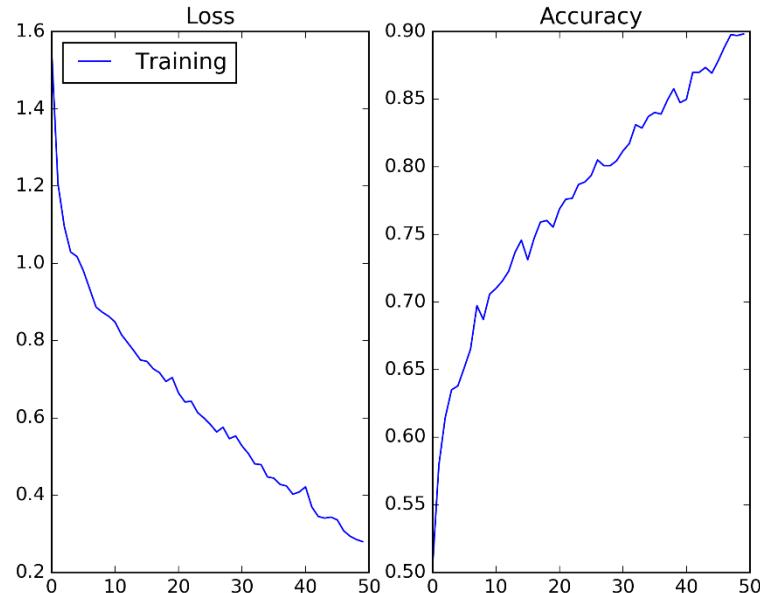
Tips for Deep Learning



Current Best Model Configuration

Component	Selection
Loss function	categorical_crossentropy
Activation function	relu + softmax
Optimizer	Adam

50 epochs 後
90% 準確率！

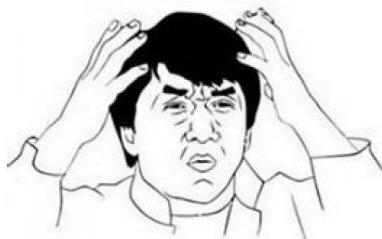


進度報告



我們有90%準確率了！

但是在 training dataset 上的表現

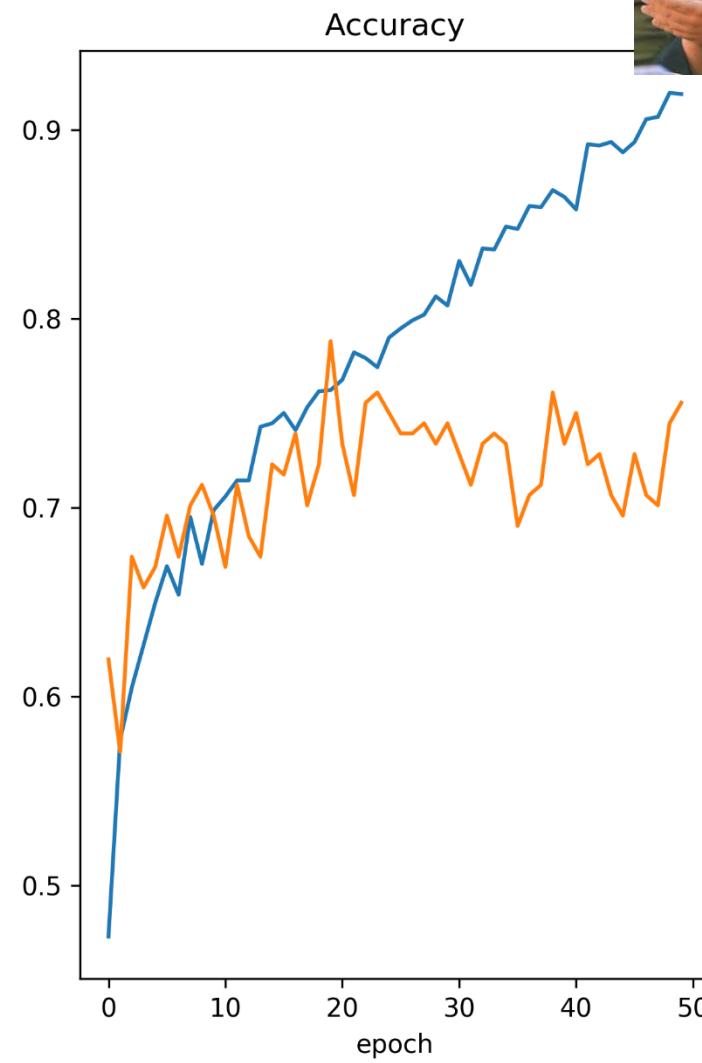
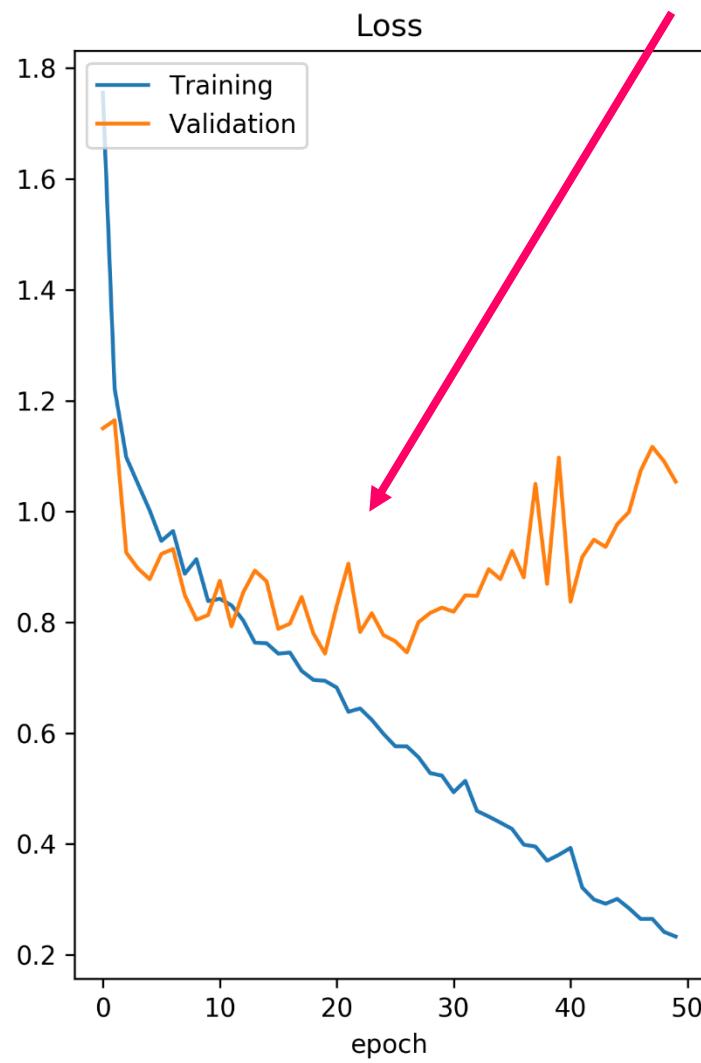
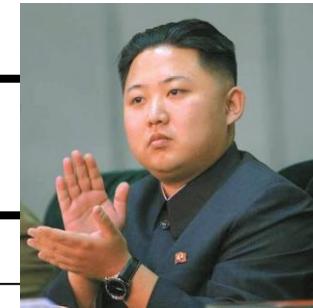


這會不會只是一場美夢？



見真章

Overfitting 啦!



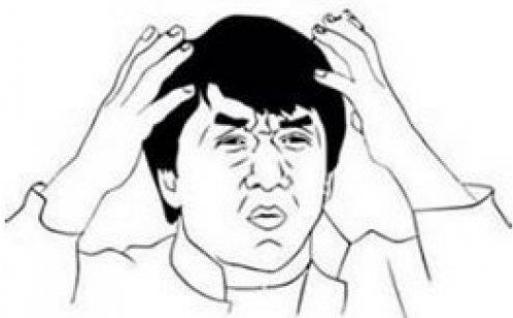
Tips for Deep Learning

Good result on
training dataset?

Yes

Good result on
testing dataset?

Ye



Regularization

Early Stopping

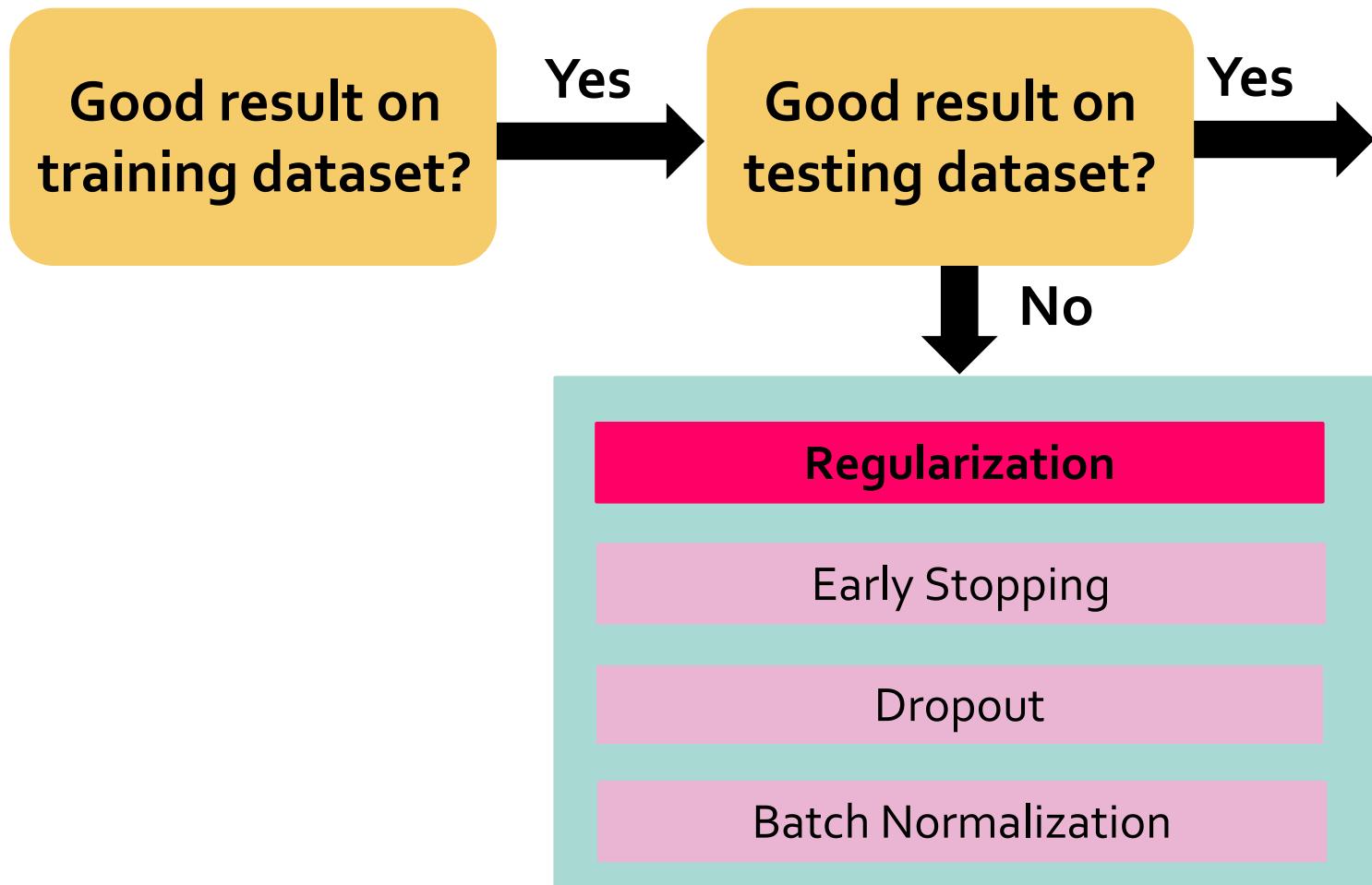
Dropout

Batch Normalization

什麼是 overfitting ?

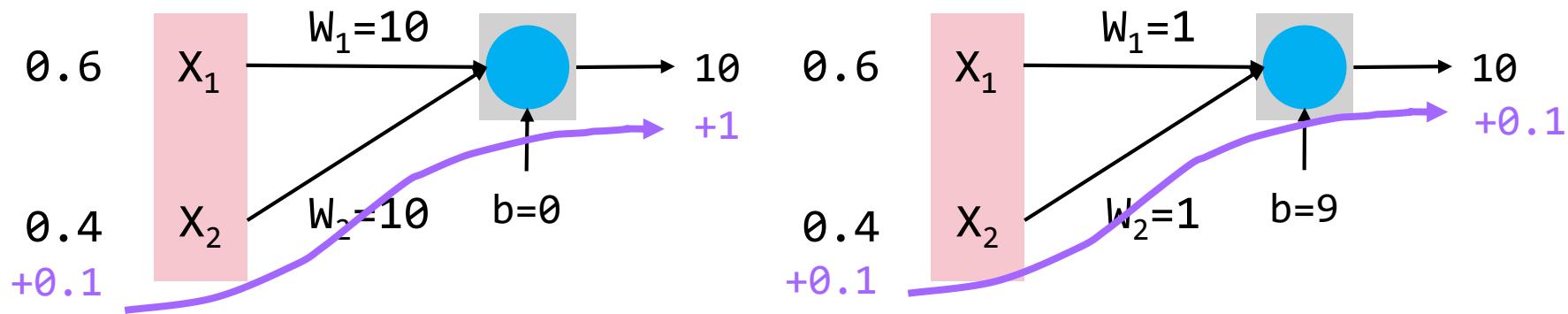
training result 進步，但 testing result 反而變差

Tips for Deep Learning



Regularization

- 限制 weights 的大小讓 output 曲線比較平滑
- 為什麼要限制呢？



w_i 較小 $\rightarrow \Delta x_i$ 對 \hat{y} 造成的影響($\Delta \hat{y}$)較小
 \rightarrow 對 input 變化比較不敏感 \rightarrow generalization 好

Regularization

□ 怎麼限制 weights 的大小呢？

加入目標函數中，一起優化

$$\text{Loss}_{\text{reg}} = \sum (y - (\hat{y})) + \alpha(\text{regularizer})$$

\hat{y}

□ α 是用來調整 regularization 的比重

□ 避免顧此失彼 (降低 weights 的大小而犧牲模型準確性)

L₁ and L₂ Regularizers

□ L₁ norm

$$L_1 = \sum_{i=1}^N |W_i|$$

Sum of absolute values

□ L₂ norm

$$L_2 = \sqrt{\sum_{i=1}^N |W_i|^2}$$

Root mean square of
absolute values



Regularization in Keras

```
''' Import l1,l2 (regularizer) '''
from keras.regularizers import l1,l2

model_l2 = Sequential()

# 加入第一層 hidden layer 並加入 regularizer (alpha=0.01)
Model_l2.add(Dense(128, input_dim=200, kernel_regularizer=l2(0.01)))
Model_l2.add(Activation('relu'))

# 加入第二層 hidden layer 並加入 regularizer
model_l2.add(Dense(256, kernel_regularizer=l2(0.01)))
model_l2.add(Activation('relu'))

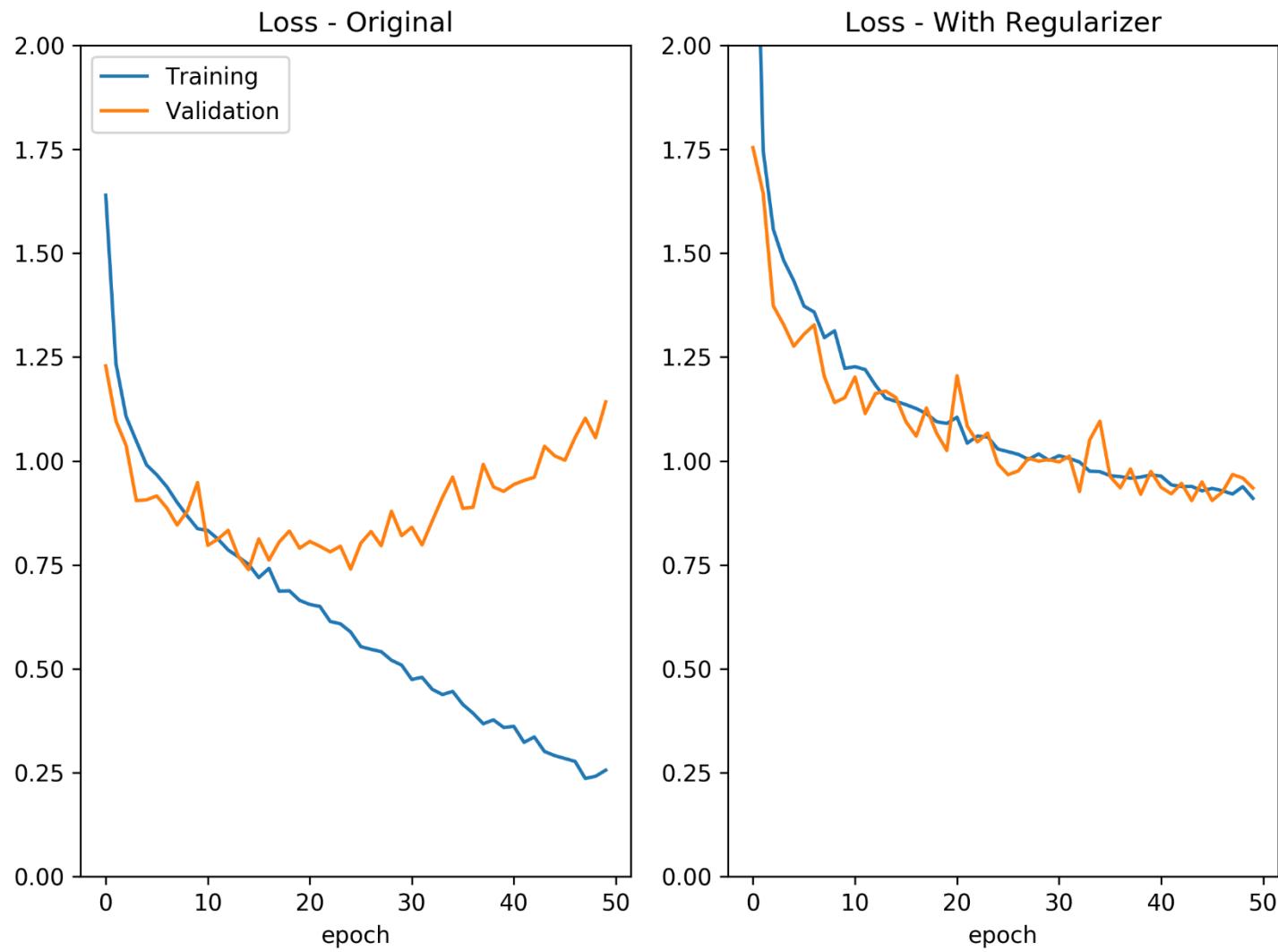
# 加入 Output layer
model_l2.add(Dense(5, kernel_regularizer=l2(0.01)))
model_l2.add(Activation('relu'))
```

練習 o6_regularizer.py (5-8 minutes)

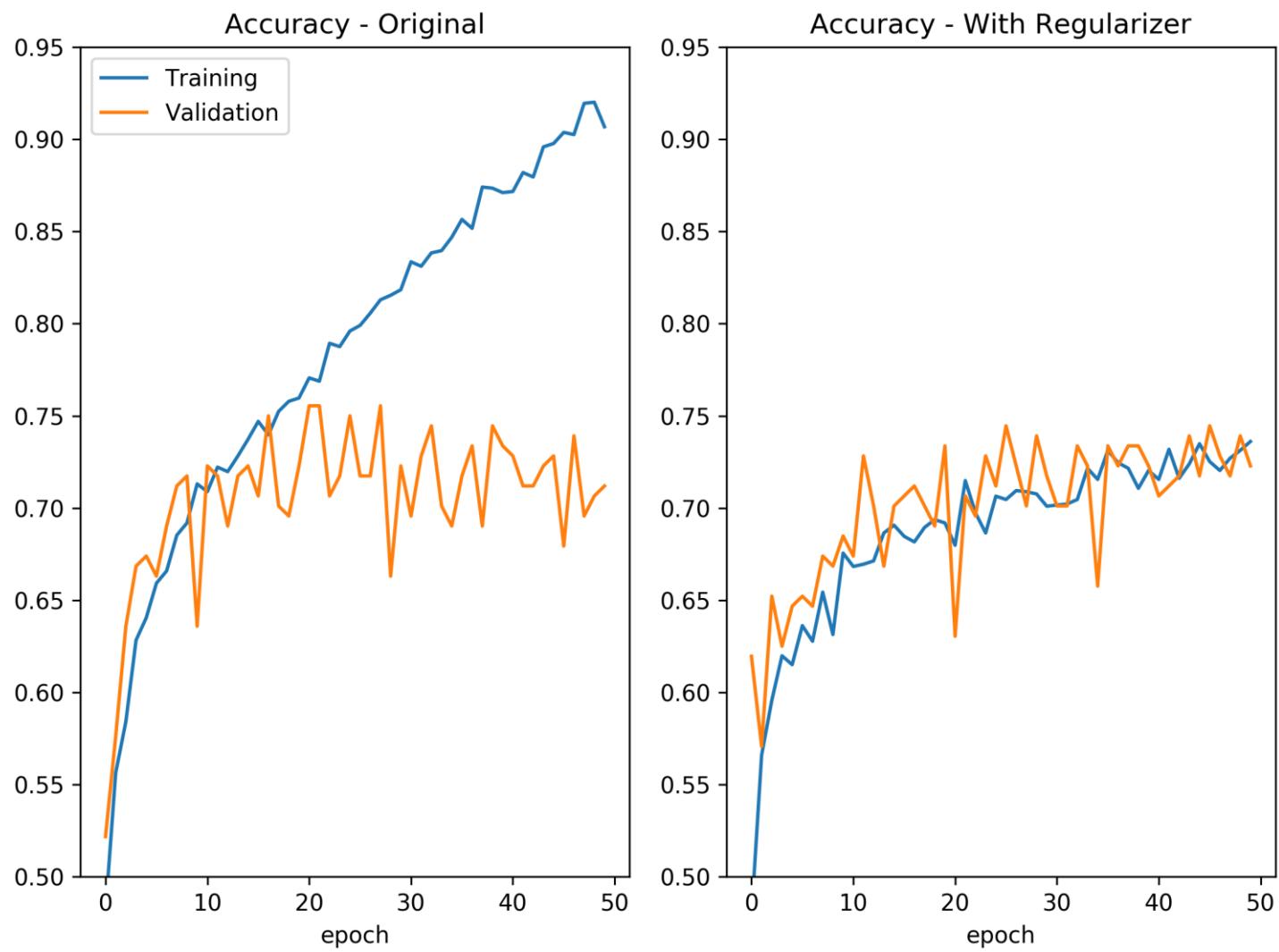
```
''' Import l1,l2 (regularizer) '''
from keras.regularizers import l1,l2,l1l2
# 加入第一層 hidden layer 並加入 regularizer (alpha=0.01)
Model_l2.add(Dense(128, input_dim=200, kernel_regularizer=l2(0.01)))
Model_l2.add(Activation('softplus'))
```

1. alpha = 0.01 會太大嗎？該怎麼觀察呢？
2. alpha = 0.001 再試試看

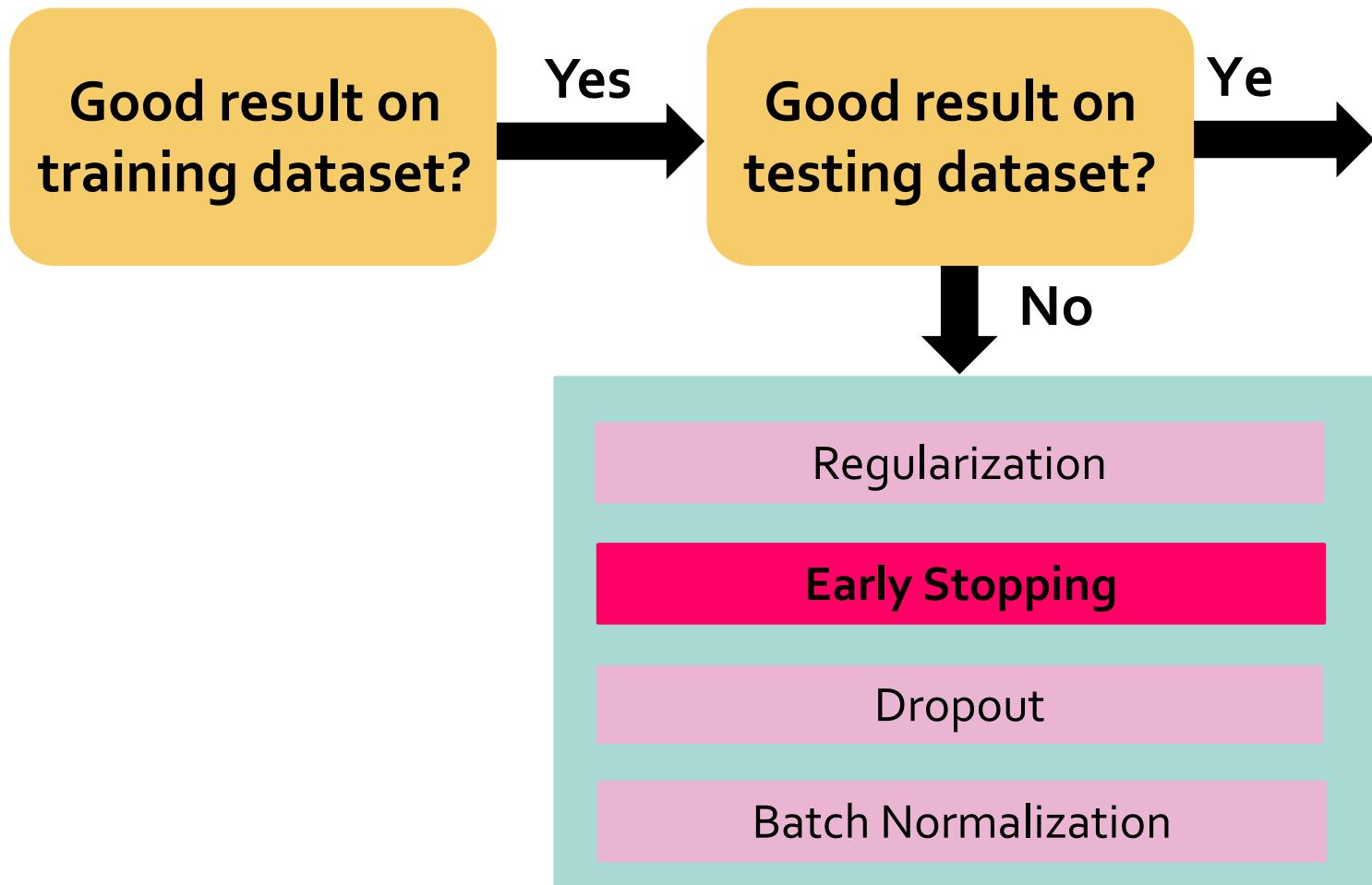
Result – L2 Regularizer (alpha=0.001)



Result – L2 Regularizer (alpha=0.001)

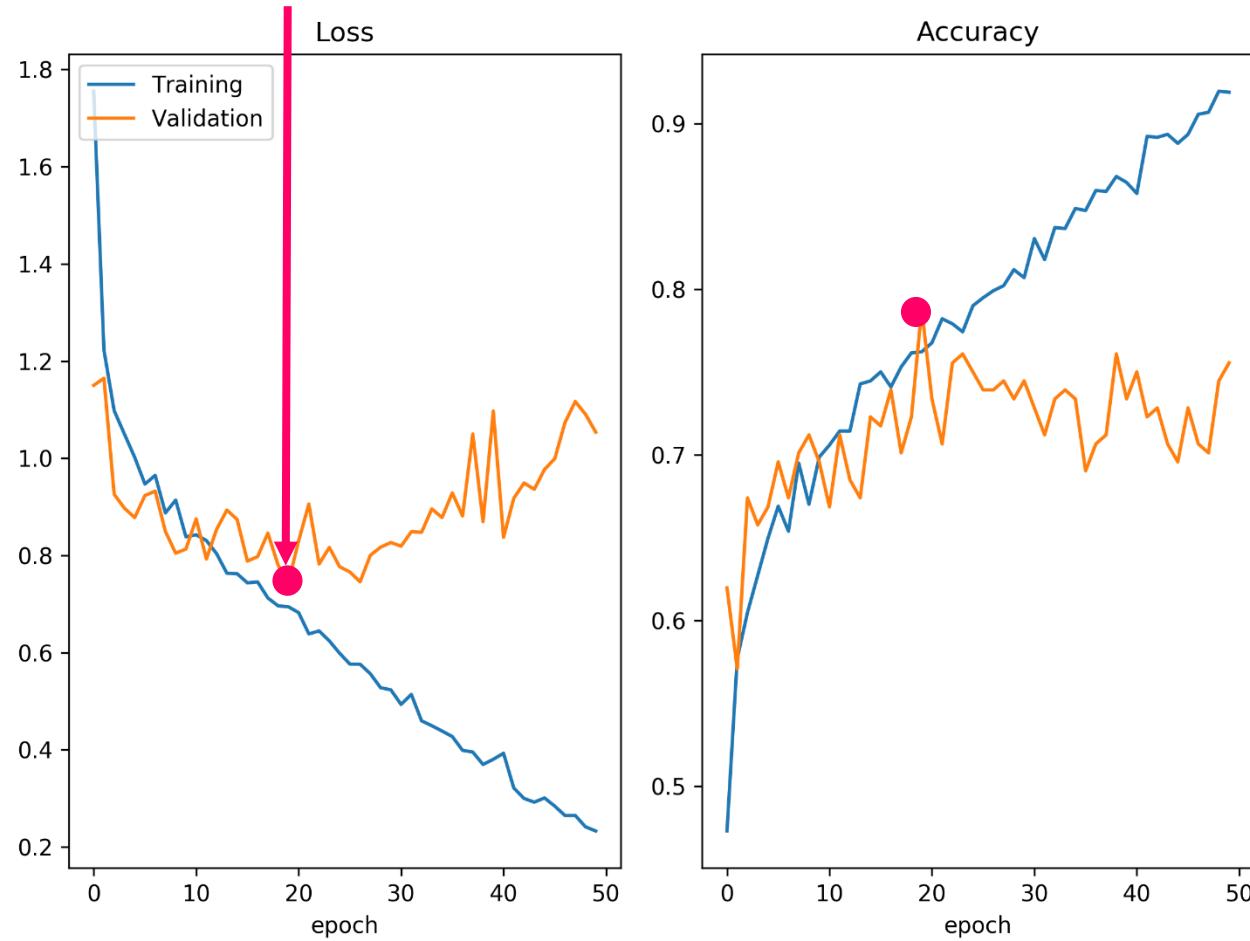


Tips for Deep Learning



Early Stopping

□ 假如能早點停下來就好了...



Early Stopping in Keras

□ Early Stopping

```
''' EarlyStopping '''
from keras.callbacks import EarlyStopping
earlyStopping=EarlyStopping( monitor = 'val_loss',
                            min_delta = 0.0001,
                            patience = 3)
```

- monitor: 要監控的 performance index
- patience: 可以容忍連續幾次的不思長進

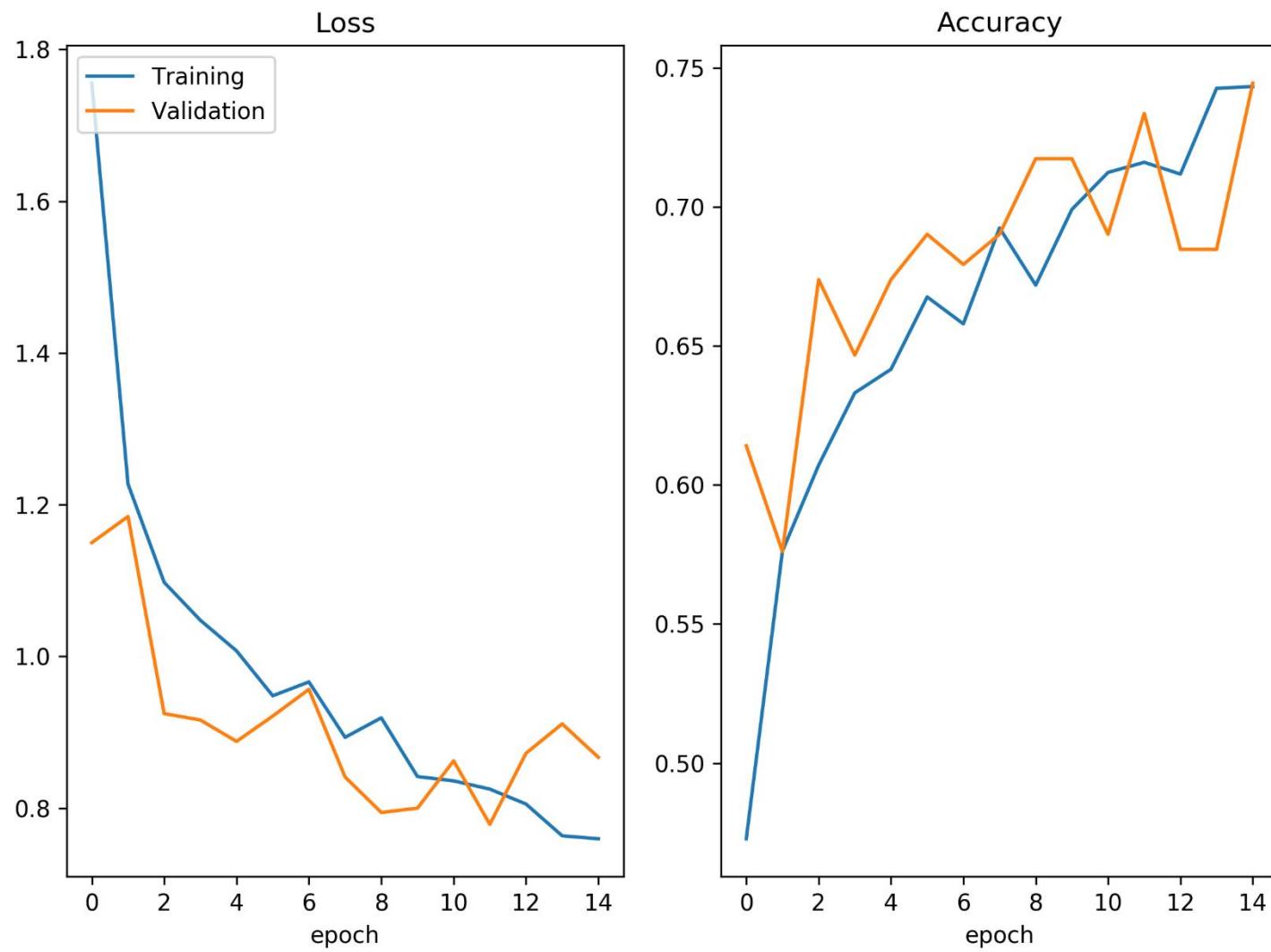
加入 Early Stopping

```
''' EarlyStopping '''
from keras.callbacks import EarlyStopping
earlyStopping=EarlyStopping( monitor = 'val_loss',
                            min_delta = 0.0001,
                            patience = 3)
```

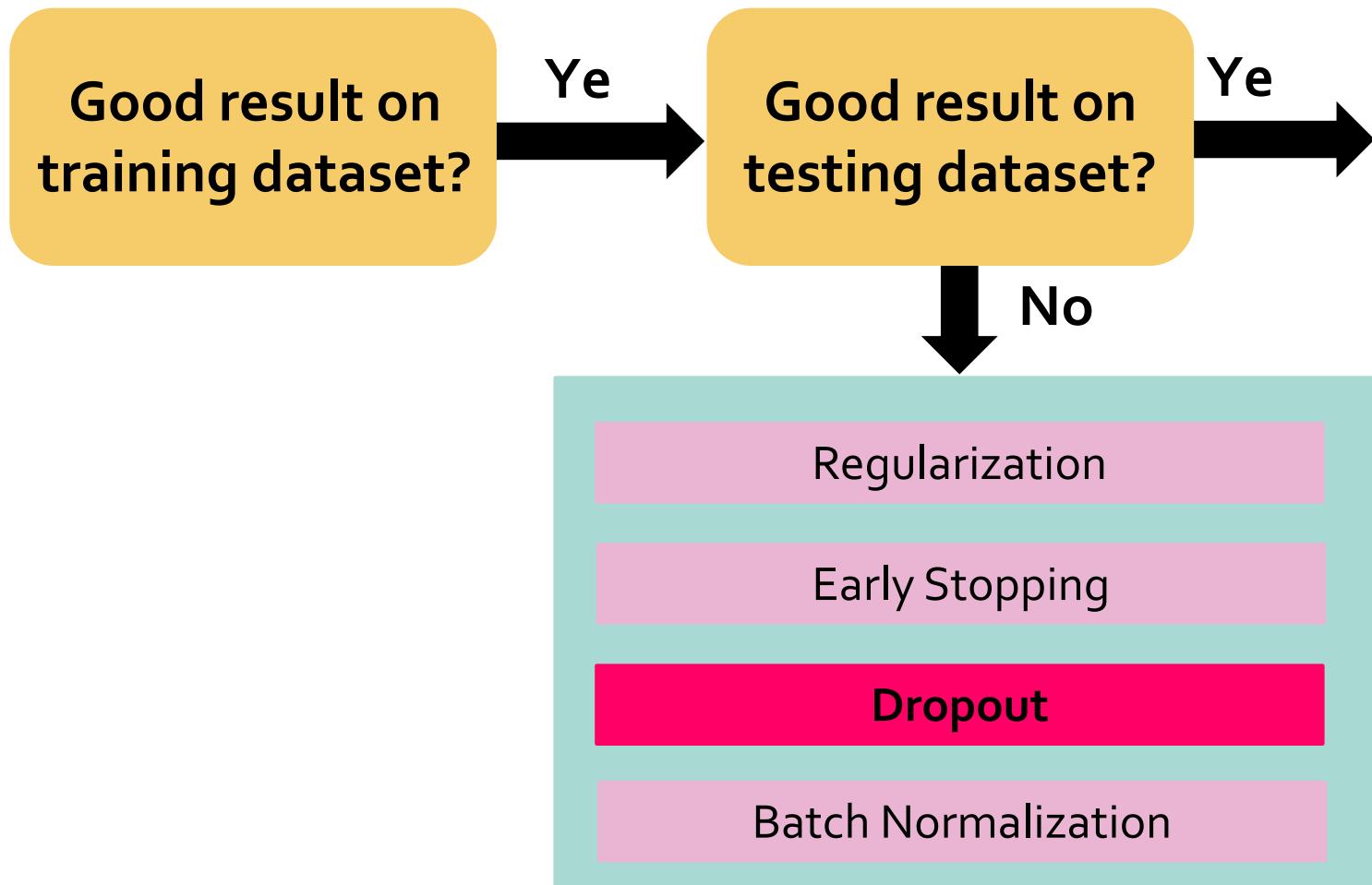
```
# 指定 batch_size, nb_epoch, validation 後，開始訓練模型!!!
history = model.fit( X_train,
                      Y_train,
                      batch_size=16,
                      verbose=0,
                      epochs=30,
                      shuffle=True,
                      validation_split=0.1,
                      callbacks=[earlyStopping])
```

練習 07_earlyStopping.py (5-8 minutes)

Result – EarlyStopping (patience=3)



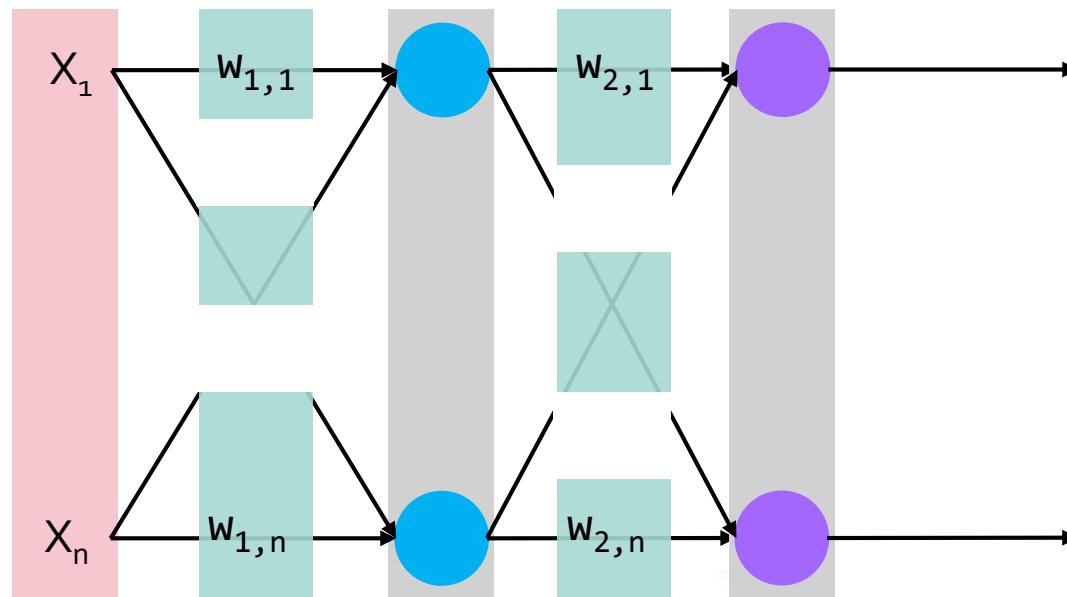
Tips for Deep Learning



Dropout

□ What is Dropout?

- 原本為 neurons 跟 neurons 之間為 fully connected
- 在訓練過程中，隨機拿掉一些連結 (weight 設為 0)



Dropout 的結果

- 會造成 training performance 變差
 - 用全部的 neurons 原本可以做到 $(\hat{y} - y) < \epsilon$
 - 只用某部分的 neurons 只能做到 $(\hat{y}' - y) < \epsilon + \Delta\epsilon$

Implications

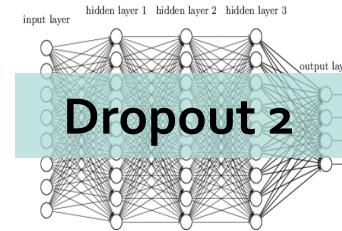
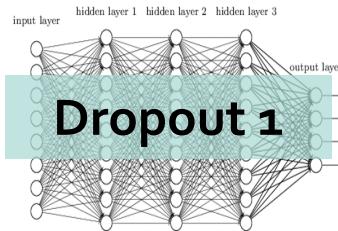
1. 增加訓練的難度



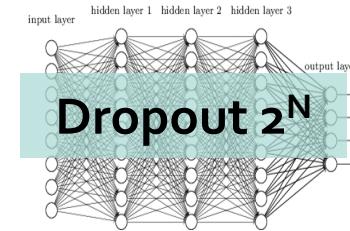
在真正的考驗時爆發



2. Dropout 可視為一種終極的 ensemble 方法 N 個 weights 會有 2^N 種 network structures



.....



Dropout in Keras

```
from keras.layers.core import Dropout
model = Sequential()

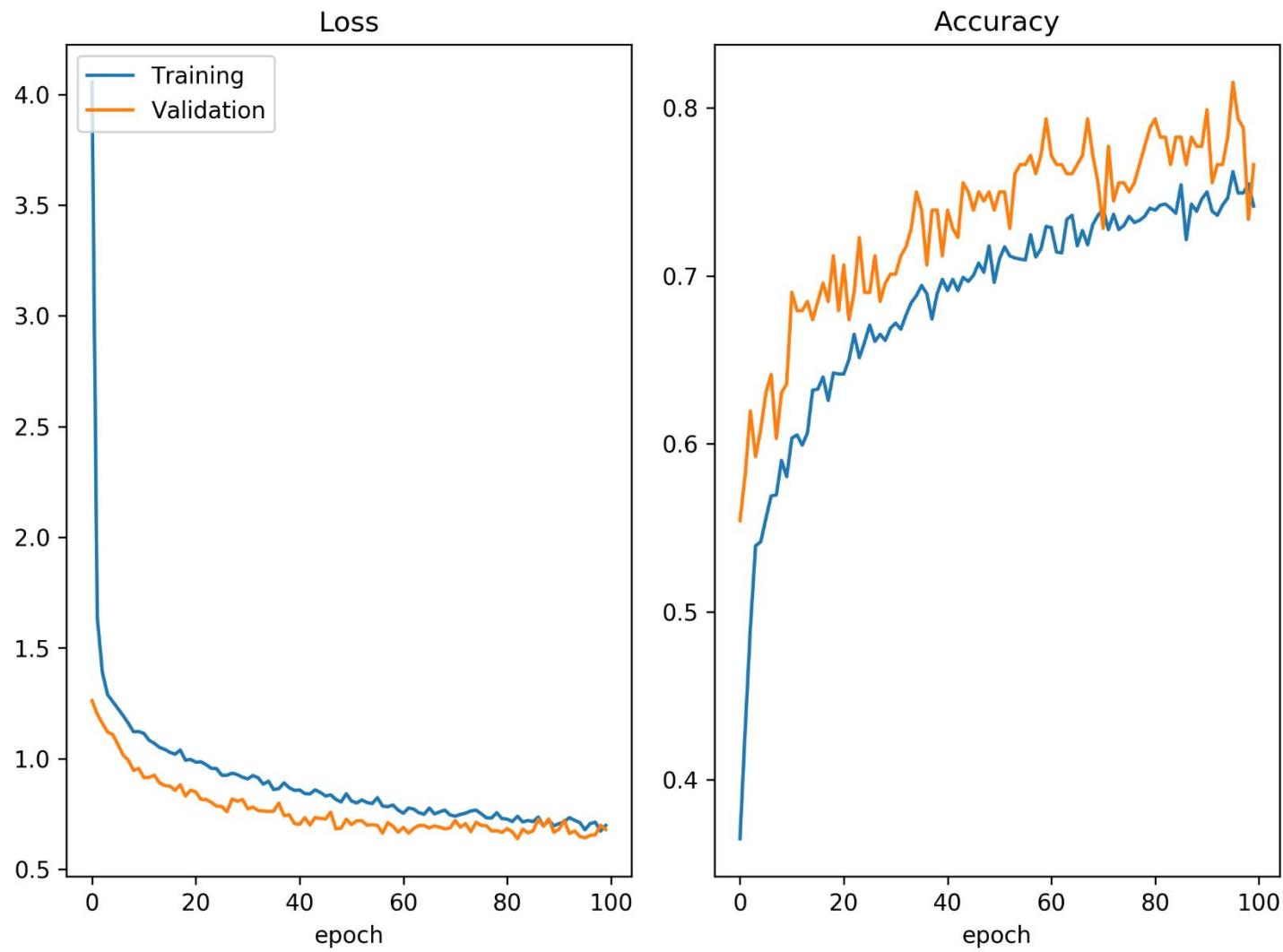
# 加入第一層 hidden layer 與 dropout=0.4
model.add(Dense(128, input_dim=200))
model.add(Activation('relu'))
model.add(Dropout(0.4))

# 加入第二層 hidden layer 與 dropout=0.4
model.add(Dense(256))
model.add(Activation('relu'))
model.add(Dropout(0.4))

# 加入 output layer (5 neurons)
model.add(Dense(5))
model.add(Activation('softmax'))
```

練習 08_dropout.py (5-8 minutes)

Result – Dropout or not

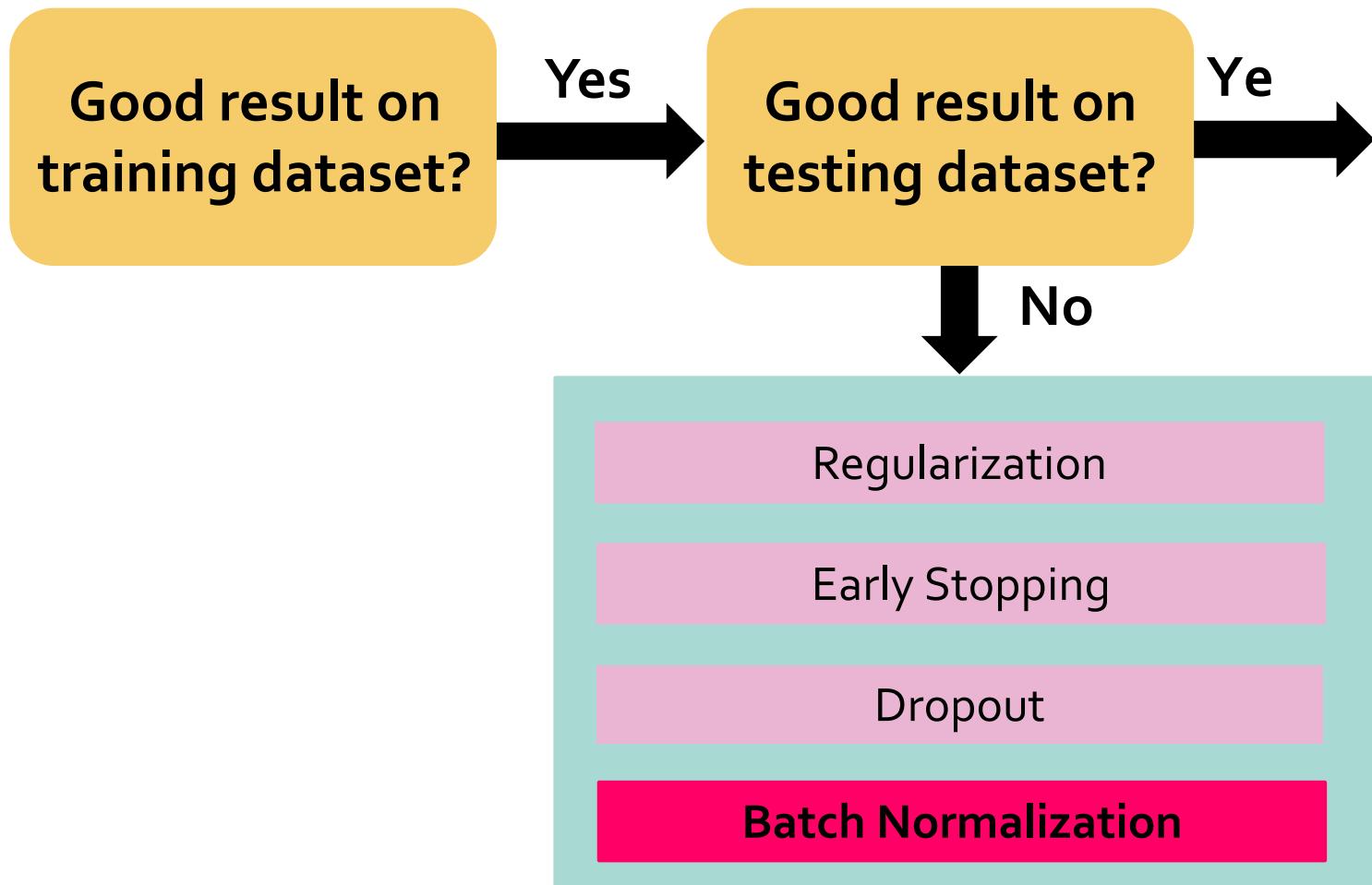


How to Set Dropout

- 不要一開始就加入 Dropout
 - 不要一開始就加入 Dropout
 - 不要一開始就加入 Dropout
-
- a) Dropout 會讓 training performance 變差
 - b) Dropout 是在避免 overfitting，不是萬靈丹
 - c) 參數少時，regularization

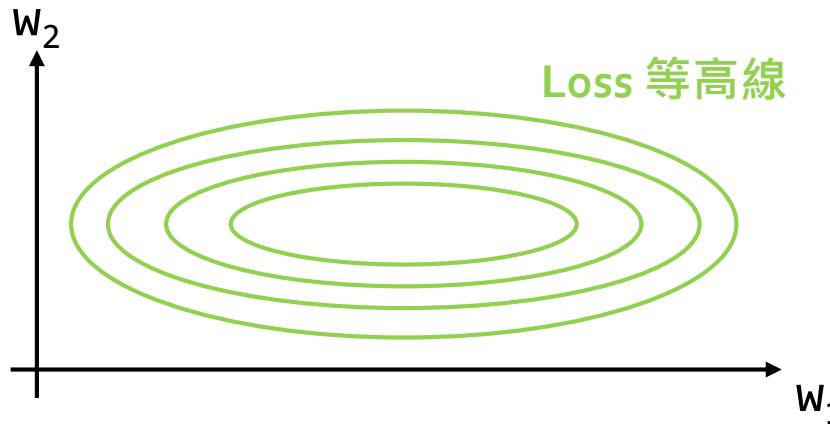


Tips for Deep Learning



回顧一下

- 對於 Input 的數值，前面提到建議要 re-scale
 - Weights 修正的路徑比較會在同心圓山谷中往下滑



- But ..., 這是只有在 inputs 啊? 如果神經網路很深, 中間每一層的輸出會變得無法控制 (due to nonlinear functions inside networks)

回顧一下

- 對於 Input 的數值，前面提到建議要 re-scale
 - Weights 修正的路徑比較會在同心圓山谷中往下滑



- But ..., 這是只有在 inputs 啊? 如果神經網路很深, 中間每一層的輸出會變得無法控制 (due to nonlinear functions inside networks)

Batch Normalization

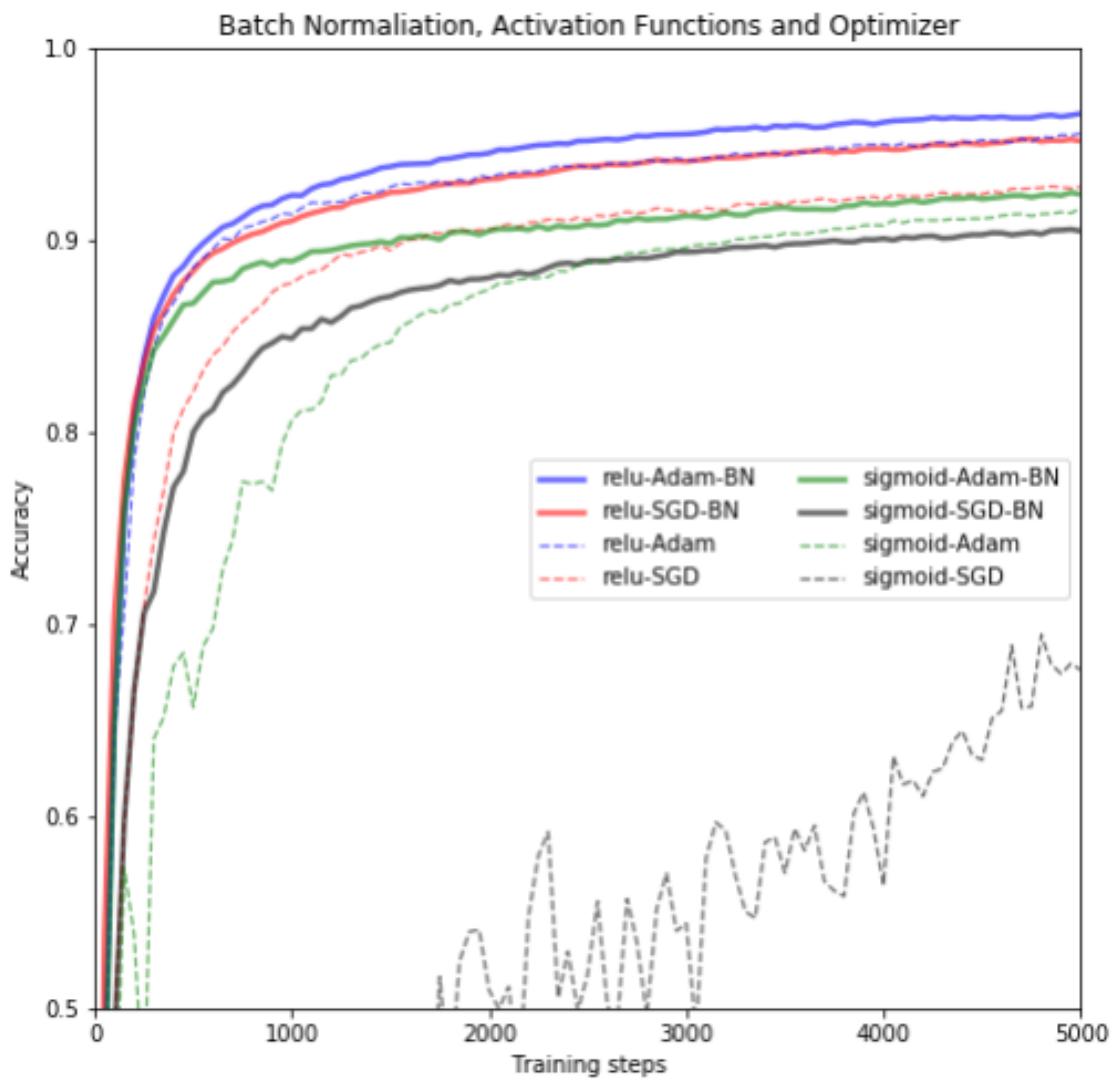
- 每個 input feature 獨立做 normalization
- 利用 batch statistics 做 normalization 而非整份資料
 - 同一筆資料在不同的 batch 中會有些微不同 (a kind of data augmentation)

$$\begin{aligned}\mu_{\mathcal{B}} &\leftarrow \frac{1}{m} \sum_{i=1}^m x_i && // \text{mini-batch mean} \\ \sigma_{\mathcal{B}}^2 &\leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 && // \text{mini-batch variance} \\ \hat{x}_i &\leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} && // \text{normalize} \\ y_i &\leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) && // \text{scale and shift}\end{aligned}$$

Batch Normalization 好處多多

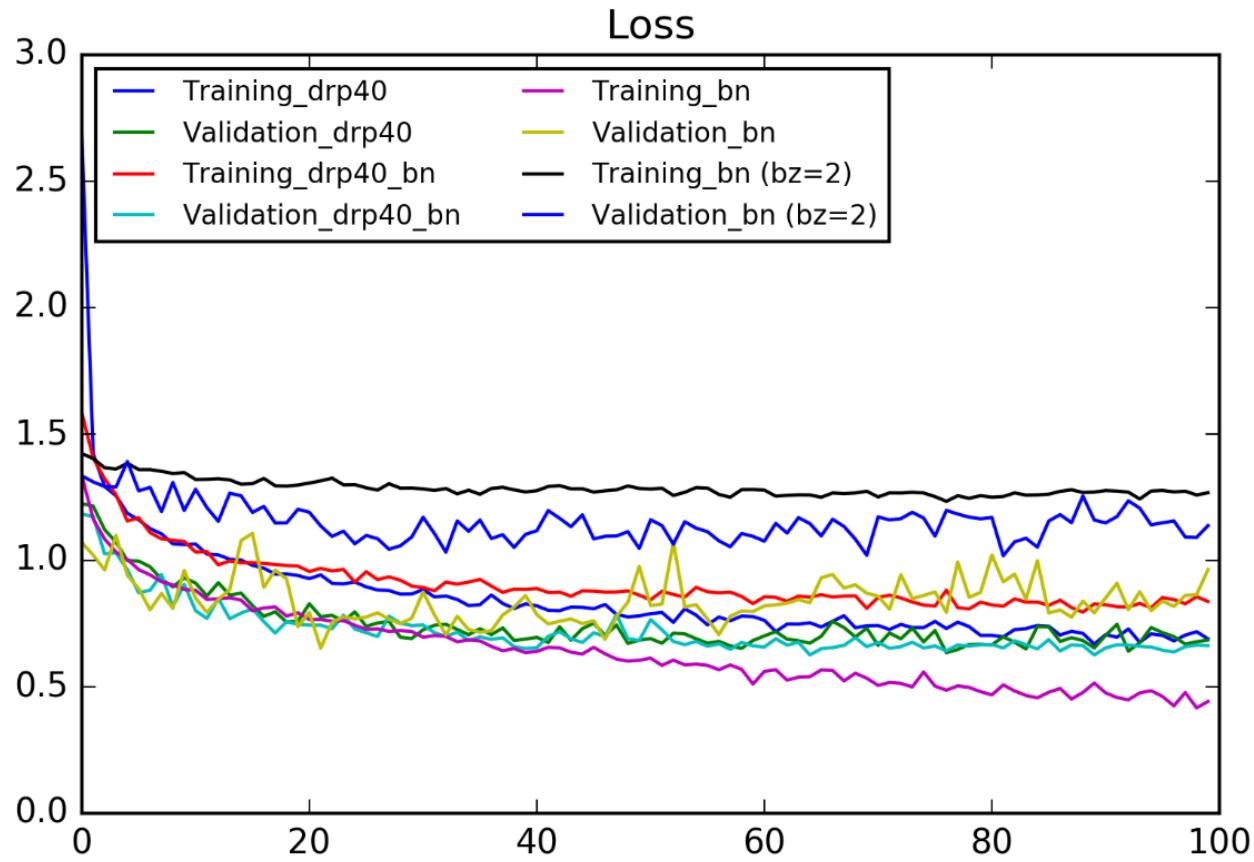
- 可以解決 Gradient vanishing 的問題
- 可以用比較大的 learning rate
- 加速訓練
- 取代 dropout & regularizes
- 目前大多數的 Deep neural network 都會加
- 大多加在 activation function 前 (Pre-activation!)

Comparisons

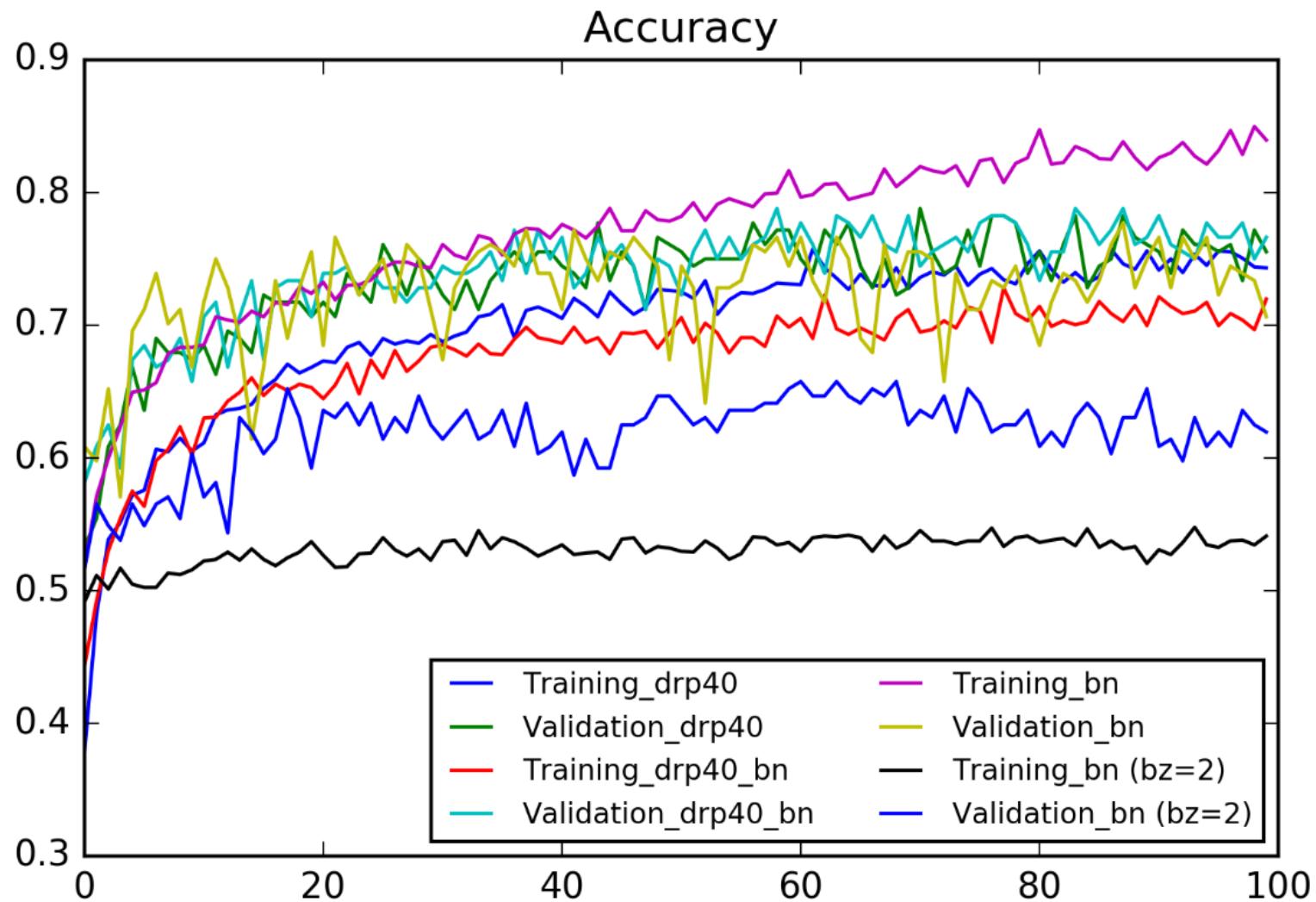


Shortcomings of BN

- 當 batch size 設定很小時不要用! 會爆炸 (general speaking, batch size should at least $> 16 / 32$)



Result of BN



BN in Keras

```
from keras.layers import BatchNormalization
model = Sequential()

model.add(Dense(128, input_dim=200))
model.add(BatchNormalization())
model.add(Activation('relu'))

model.add(Dense(256))
model.add(BatchNormalization())
model.add(Activation('relu'))

# 加入 output layer (5 neurons)
model.add(Dense(5))
model.add(Activation('softmax'))

# 觀察 model summary
model.summary()
```

(Answer) 09_batchnorm.py

(5-8 minutes)



大家的好朋友 Callbacks

善用 Callbacks 幫助你躺著 train models

Callbacks Class

```
from keras.callbacks import Callbacks
Class LossHistory(Callbacks):
    def on_train_begin(self, logs={}):
        self.loss = []
        self.acc = []
        self.val_loss = []
        self.val_acc = []
    def on_batch_end(self, batch, logs={}):
        self.loss.append(logs.get('loss'))
        self.acc.append(logs.get('acc'))
        self.val_loss.append(logs.get('val_loss'))
        self.val_acc.append(logs.get('val_acc'))
loss_history = LossHistory()
```

Callback 的時機

- on_train_begin
- on_train_end
- on_batch_begin
- on_batch_end
- on_epoch_begin
- on_epoch_end

ModelCheckpoint

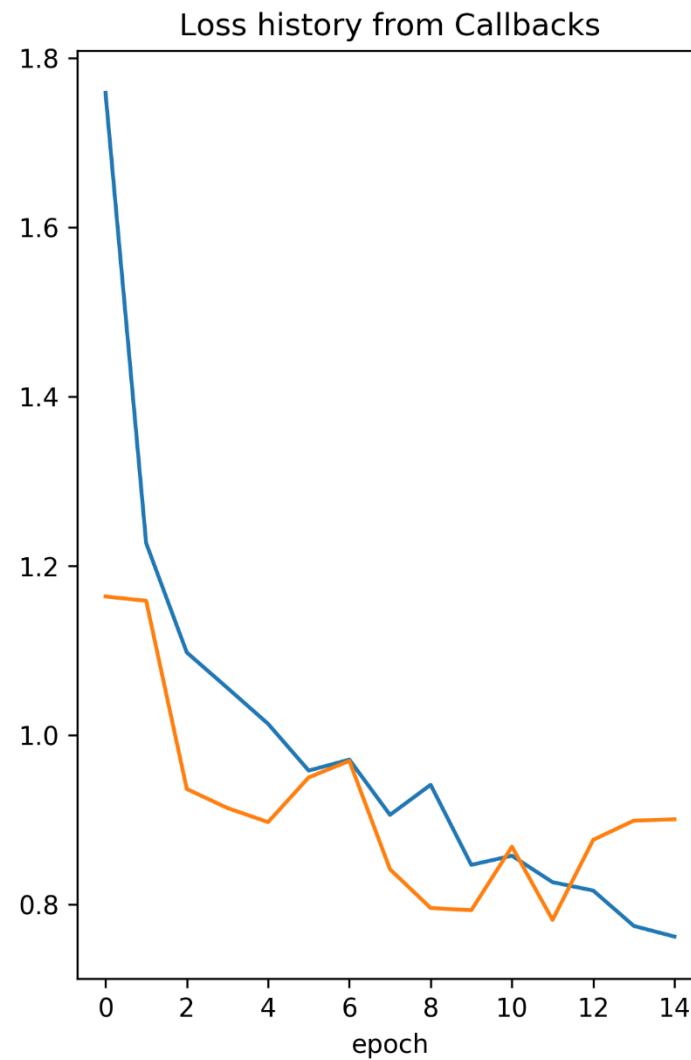
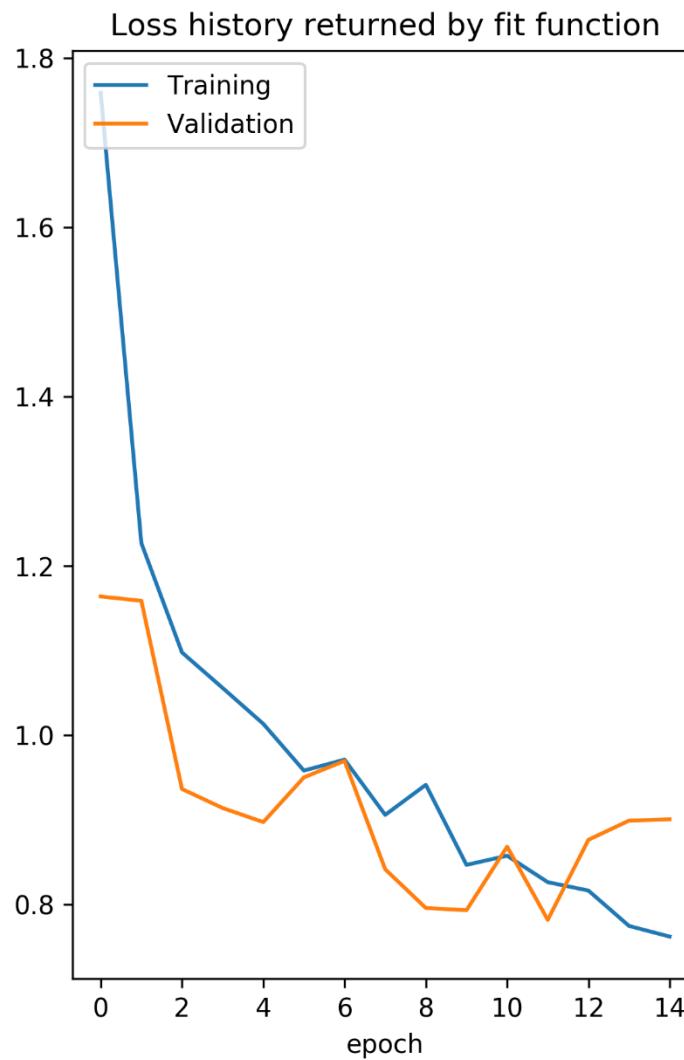
```
from keras.callbacks import ModelCheckpoint

checkpoint = ModelCheckpoint('model.h5',
                            monitor = 'val_loss',
                            verbose = 1,
                            save_best_only = True,
                            mode = 'min')
```

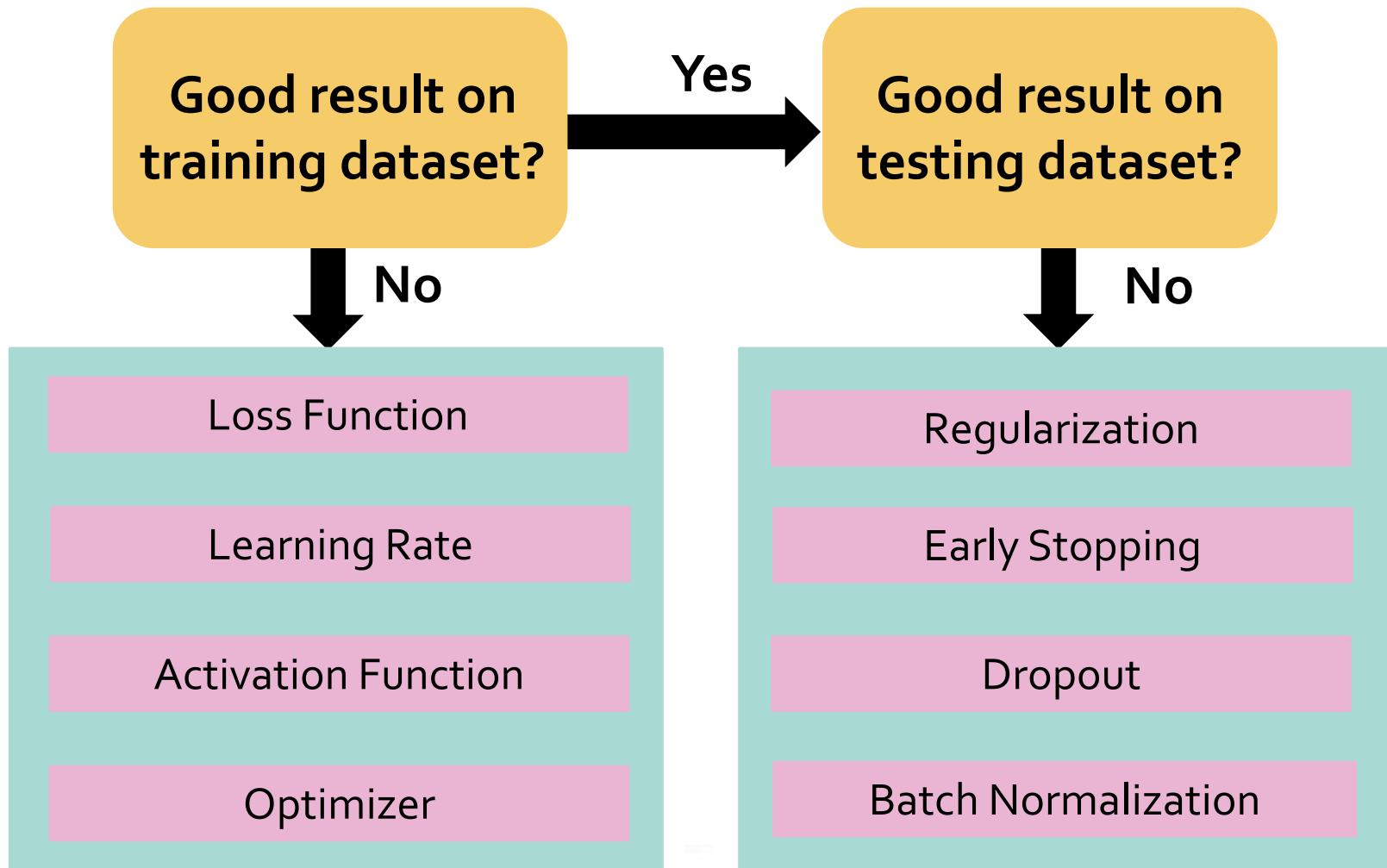
在 model.fit 時加入 Callbacks

```
history = model.fit(X_train, Y_train,  
                      batch_size=16,  
                      verbose=0,  
                      epochs=30,  
                      shuffle=True,  
                      validation_split=0.1,  
                      callbacks=[early_stopping,  
                                 loss_history,  
                                 lrate,  
                                 checkpoint])
```

Example (10_Callback.py)



Tips for Training Your Own DL Model





Variants of Deep Neural Network

Convolutional Neural Network (CNN)

2-dimensional Inputs

- DNN 的輸入是一維的向量，那二維的矩陣呢？例如
圖形資料



Figures reference

<https://twitter.com/gonainlive/status/507563446612013057>

Ordinary Feedforward DNN with Image

□ 將圖形轉換成一維向量

- Weight 數過多 → training 所需時間太長
- 失去相鄰 pixels 的資訊

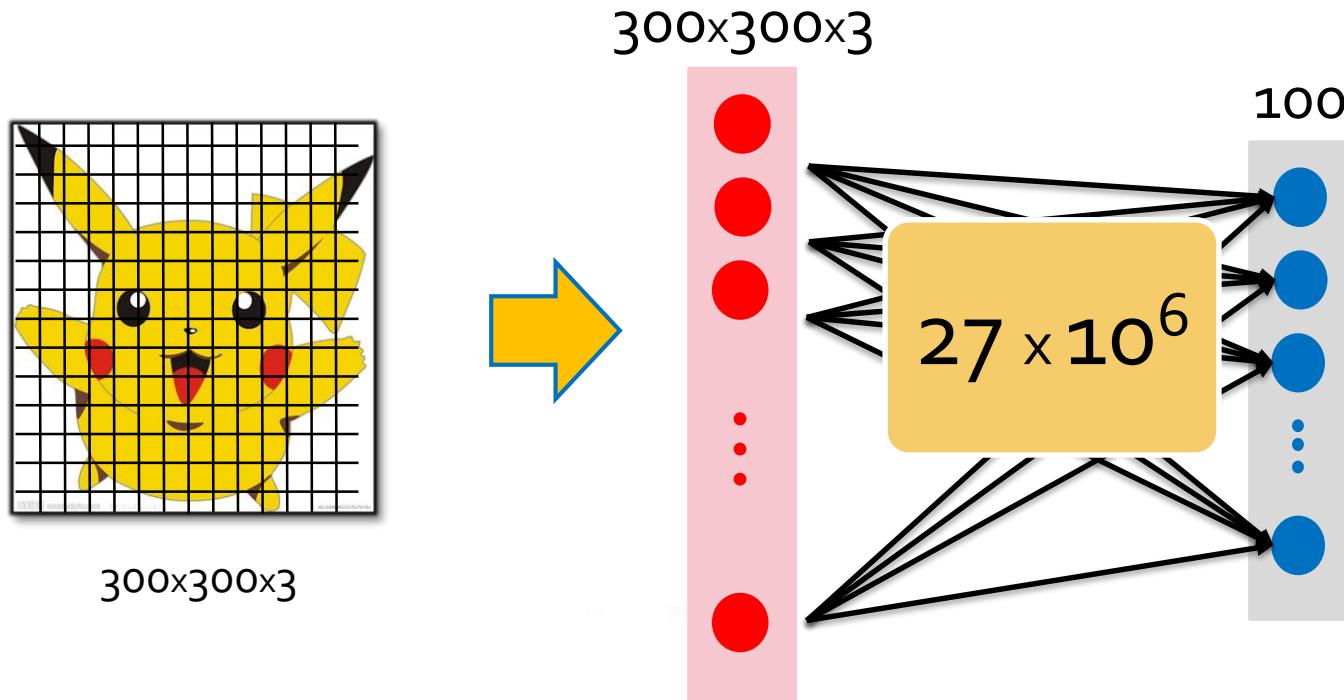


Figure reference

<http://www.ettoday.net/dalemon/post/12934>

Characteristics of Image

□ 圖的構成：線條 → 圖案 (pattern) → 物件 → 場景

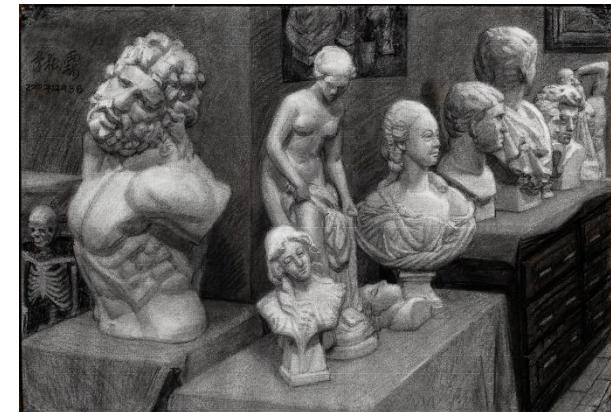
Line Segment



Object



Scene



Pattern

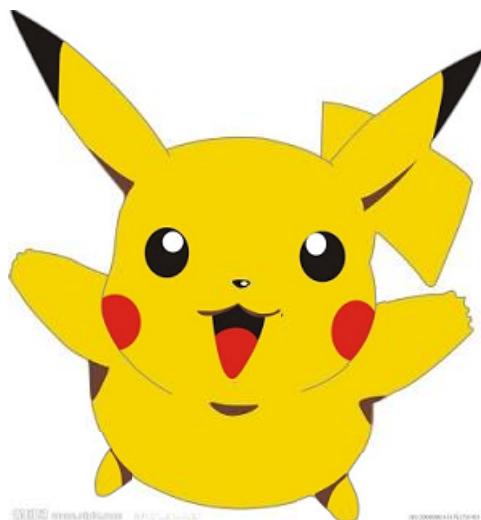


Figures reference
<http://www.sumiaozhijia.com/touxiang/471.html>
<http://122311.com/tag/su-miao/2.html>



Patterns

- 猜猜看我是誰
- 辨識物件只需要用**幾個特定位置的部分的圖式**



皮卡丘



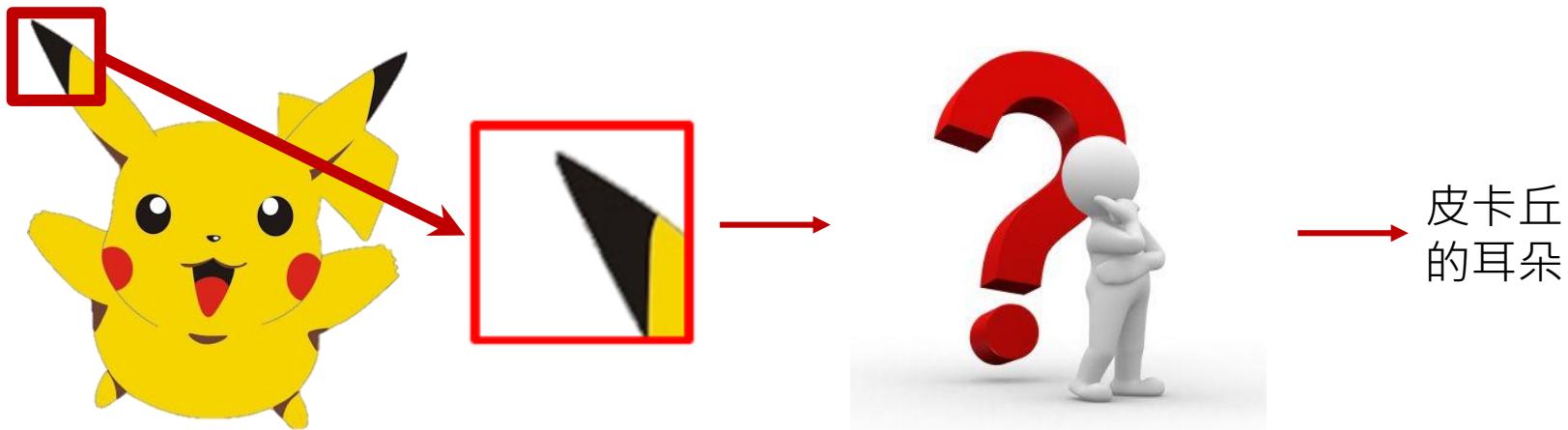
小火龍

Figures reference

<http://arcadesushi.com/charmander-drunken-pokemon-tattoo/#photogallery-1=1>

Property 1: Local Information

- 部分的圖示 (local)



Property 2: Global Operation

重複的圖案可能出現在很多不同的地方

Robust to shift and rotation

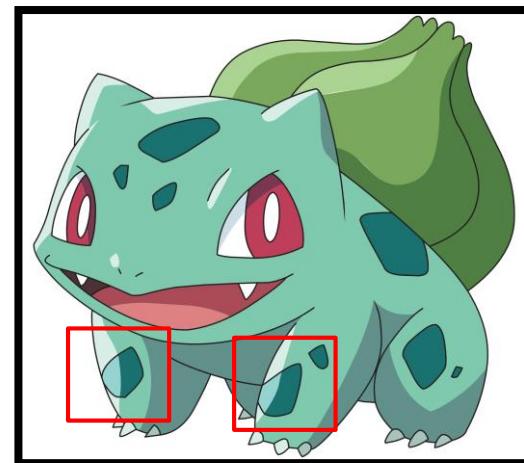
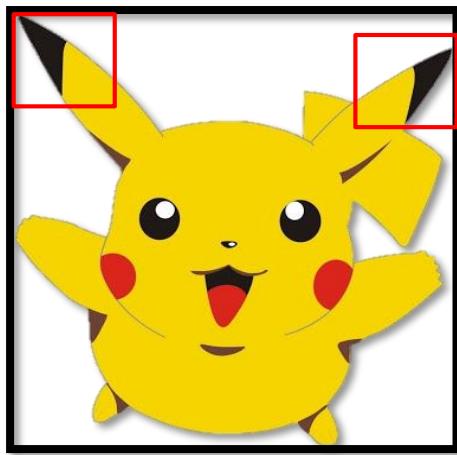
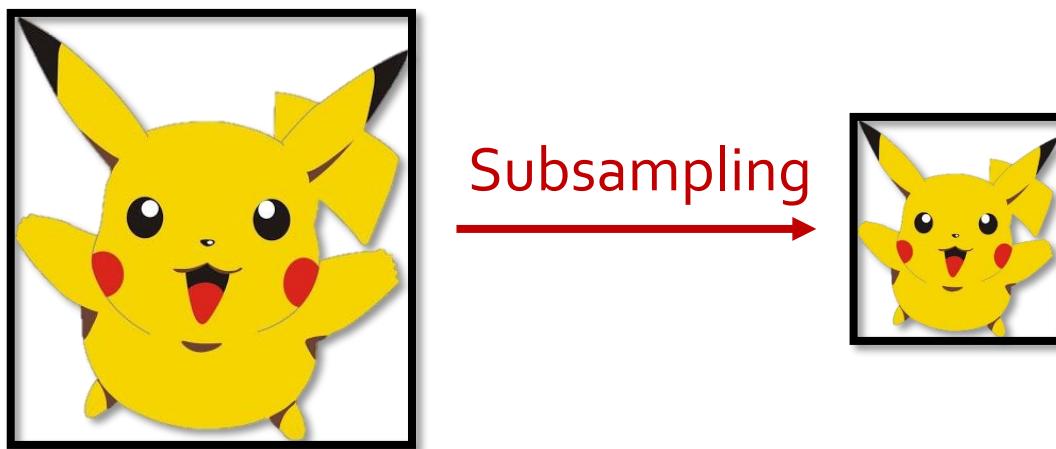


Figure reference

<https://www.youtube.com/watch?v=NN9LaU2NlM>

Property 3: Scale-invariant

- 大小的變化並沒有太多影響



近幾年最具代表性的模型

- VGG, 2014
- Deep Residual Network (ResNet), 2015
- Inception Network, 2016
- Densely Connected Network (DenseNet), 2017
- 很多都專注在 **convolutional** neural networks 因為可以處理有特殊結構的 input (圖片, 序列)

為什麼要做 Convolution?

Scan the local information globally in various ways



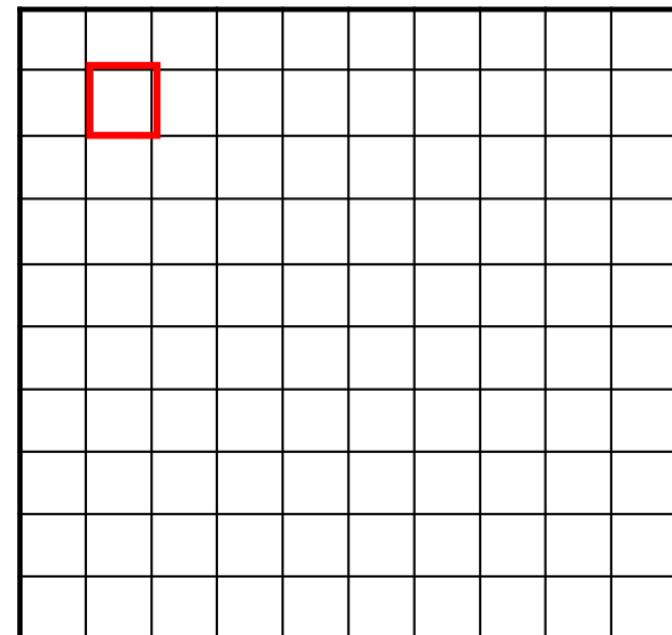
Photo credit: <https://www.plurk.com/p/h5otc9>

Image Filtering Operation

$f[.,.]$

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0
0	0	0	90	90	90	90	90	0
0	0	0	90	90	90	90	90	0
0	0	0	90	0	90	90	90	0
0	0	0	90	90	90	90	90	0
0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

$h[.,.]$



$$g[\cdot, \cdot] \frac{1}{9}$$

1	1	1
1	1	1
1	1	1

$$h[m, n] = \sum_{k,l} g[k, l] f[m + k, n + l]$$

Slide credit: S. Seitz

Image Filtering Operation

$f[.,.]$

0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	0	90	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

$h[.,.]$

$$g[\cdot, \cdot] \frac{1}{9}$$

1	1	1
1	1	1
1	1	1

$$h[m, n] = \sum_{k,l} g[k, l] f[m + k, n + l]$$

Slide credit: S. Seitz

Image Filtering Operation

$f[.,.]$

0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	90	0	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

$h[.,.]$

			0	10	20	30	30	30	20	10
			0	20	40	60	60	60	40	20
			0	30	60	90	90	90	60	30
			0	30	50	80	80	90	60	30
			0	30	50	80	80	90	60	30
			0	20	30	50	50	60	40	20
			10	20	30	30	30	30	20	10
			10	10	10	0	0	0	0	0

$$g[\cdot, \cdot] \frac{1}{9}$$

1	1	1
1	1	1
1	1	1

$$h[m, n] = \sum_{k,l} g[k, l] f[m + k, n + l]$$

Slide credit: S. Seitz

Image Filtering Operation

$f[.,.]$

0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	90	0	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20	30	30	30	20	10		
	0	20	40	60	60	60	40	20		
	0	30	60	90	90	90	60	30		
	0	30	50	80	80	90	60	30		
	0	30	50	80	80	90	60	30		
	0	20	30	50	50	60	40	20		
	10	20	30	30	30	30	20	10		
	10	10	10	0	0	0	0	0		

$$g[\cdot, \cdot] \frac{1}{9}$$

1	1	1
1	1	1
1	1	1

$$h[m, n] = \sum_{k,l} g[k, l] f[m + k, n + l]$$

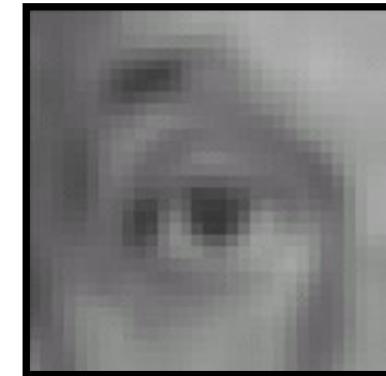
Slide credit: S. Seitz

Simple Linear Filters



Original

$$\ast \frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} =$$



Blur (with a mean filter)



Original

$$\ast \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 1 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} =$$



Shifted left
By 1 pixel

More Linear Filters

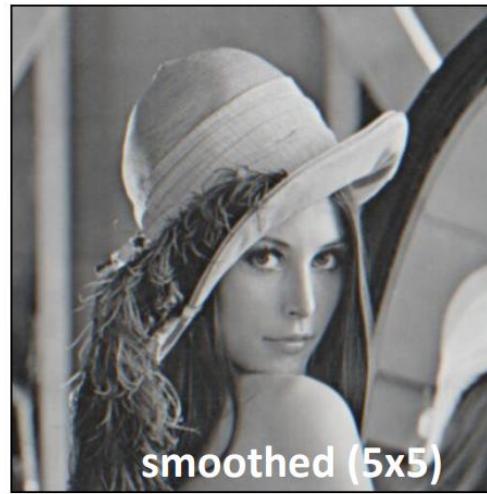
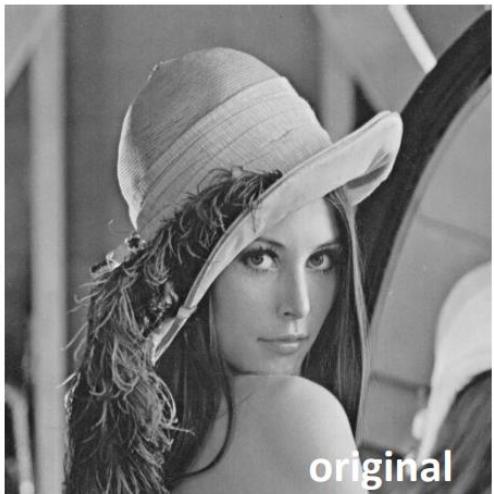


$$\ast \left(\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{array} \right) - \frac{1}{9} \left(\begin{array}{ccc} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{array} \right) =$$



Original

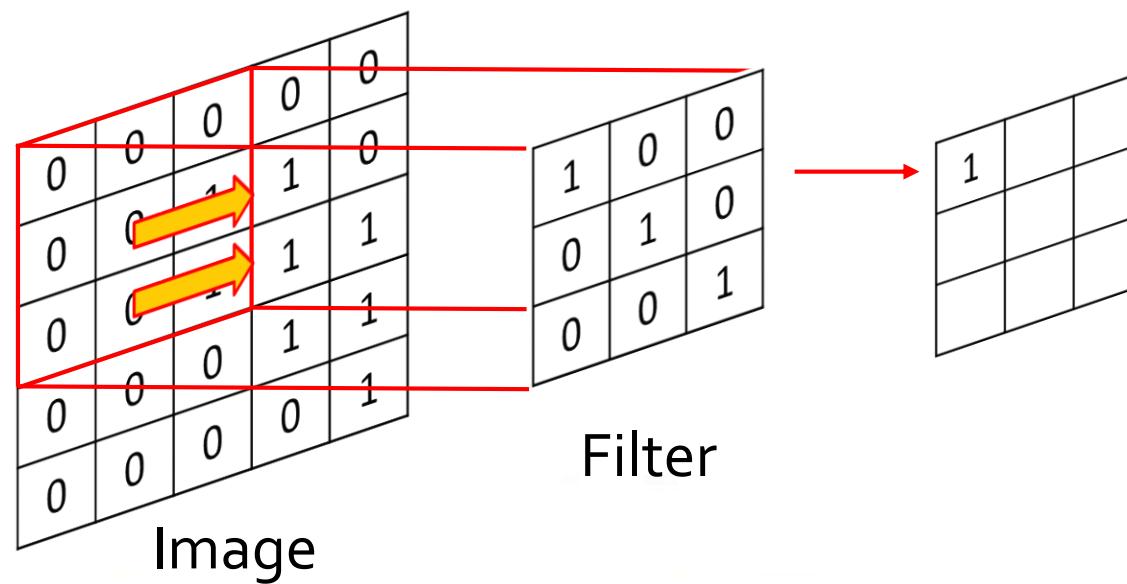
Sharpening filter
(accentuates edges)



Slide credit: David Lowe and Noah Snavely

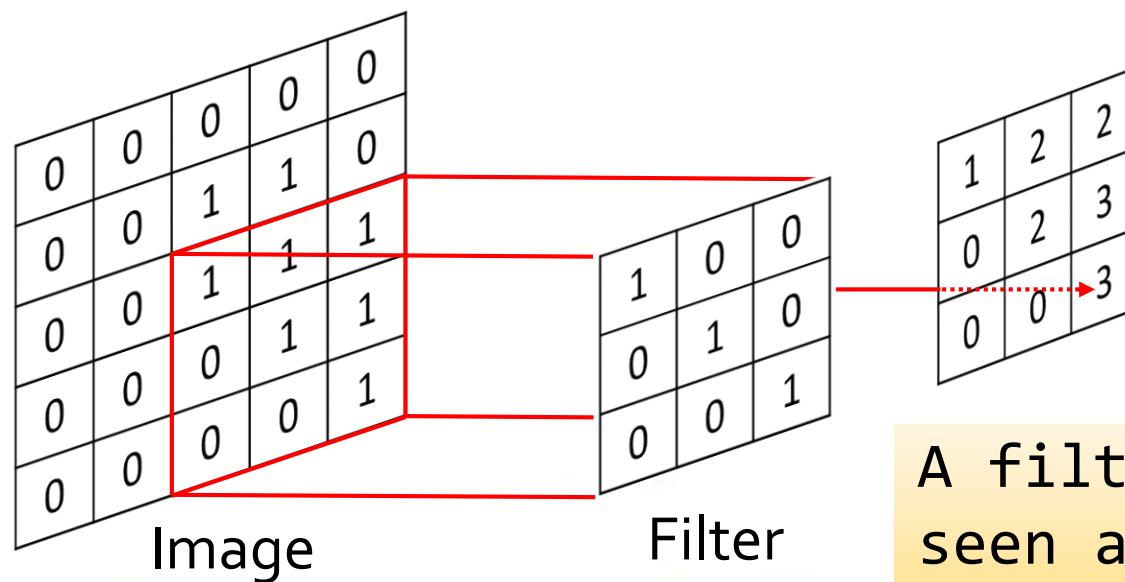
Convolution in Computer Vision (CV)

- Adding each pixel and its local neighbors which are weighted by a filter (kernel)
- Perform this convolution process to every pixels



Convolution in Computer Vision (CV)

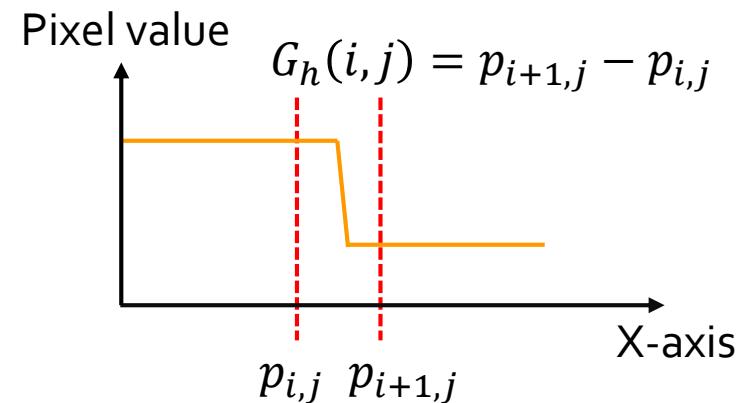
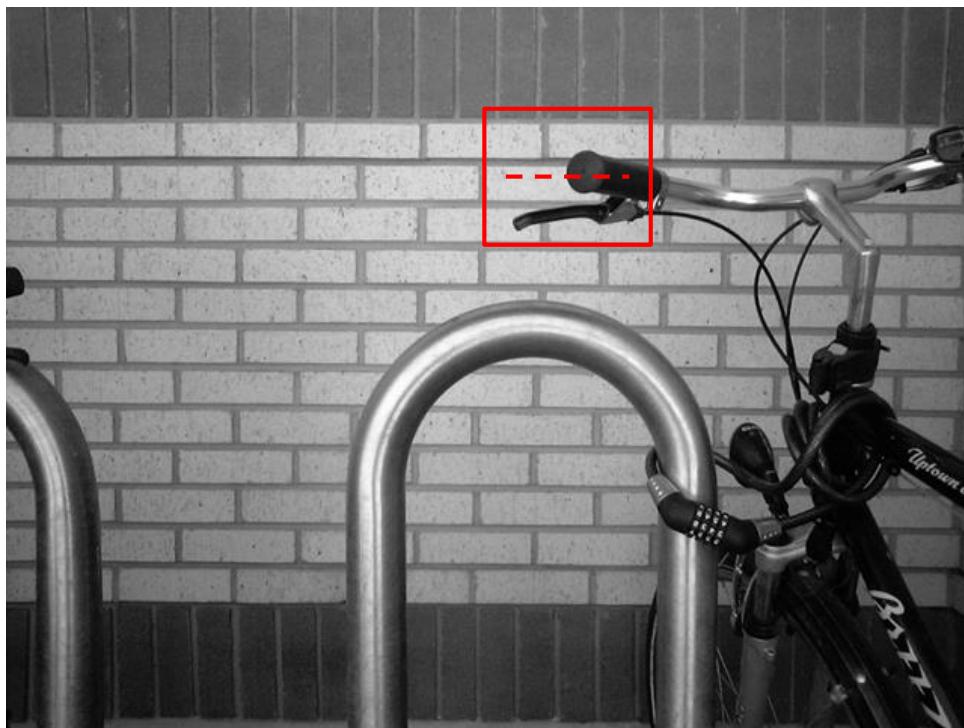
- Adding each pixel and its local neighbors which are weighted by a filter (kernel)
- Perform this convolution process to every pixels



A filter could be seen as a pattern

Real Example: Sobel Edge Detection

□ edge = 顏色(亮度)變化大的地方



梯度的量值增加 2 倍
來強化效果

$$2 * p_{i+1,j} - 2 * p_{i,j}$$



Figure reference

https://en.wikipedia.org/wiki/Sobel_operator#/media/File:Bikesgrayh.jpg

<http://laoblogger.com/images/thinking-images-4.jpg>

Real Example: Sobel Edge Detection

- 相鄰兩像素值差異越大，convolution 後新像素絕對值越大

x-gradient

-1	0	1
-2	0	2
-1	0	1

$-p_{i-1,j} \quad p_{i+1,j}$

*

3	3	0	0	0
3	3	0	0	0
3	3	0	0	0
3	3	0	0	0
3	3	0	0	0

Original Image

=

-12	-12	0
-12	-12	0
-12	-12	0

New Image

y-gradient

-1	-2	-1
0	0	0
1	2	1

*

3	3	3	3	3
3	3	3	3	3
3	3	3	3	3
0	0	0	0	0
0	0	0	0	0

=

0	0	0
-12	-12	-12
-12	-12	-12

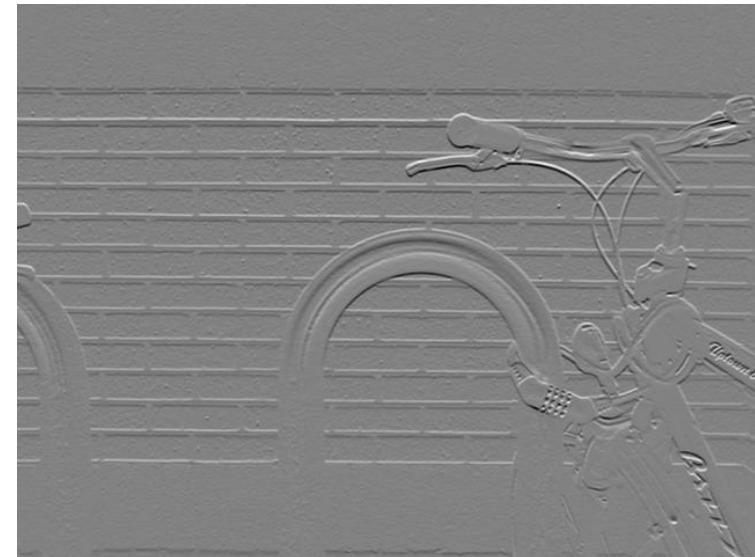
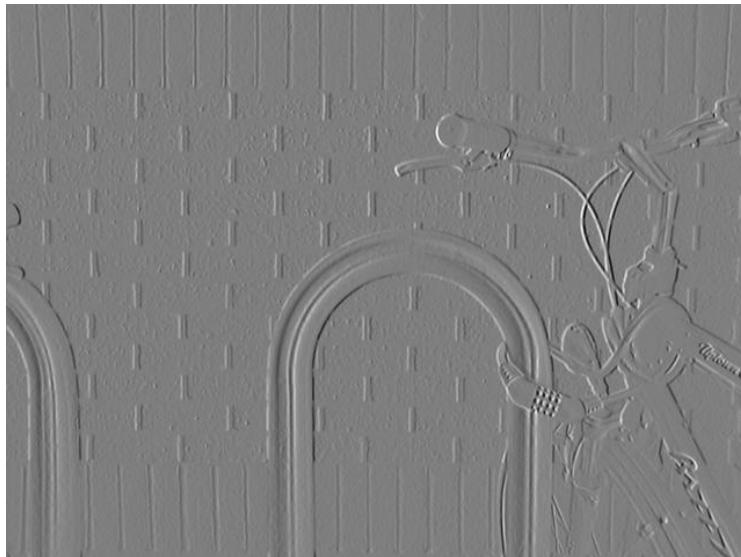
Real Example: Sobel Edge Detection

x-gradient

-1	0	1
-2	0	2
-1	0	1

y-gradient

-1	-2	-1
0	0	0
1	2	1



Real Example: Sobel Edge Detection

- Edge image



Convolution in Computer Vision (CV)

□ Common applications

- 模糊化, 銳利化, 浮雕
- <http://setosa.io/ev/image-kernels/>



Convolutional Neural Networks

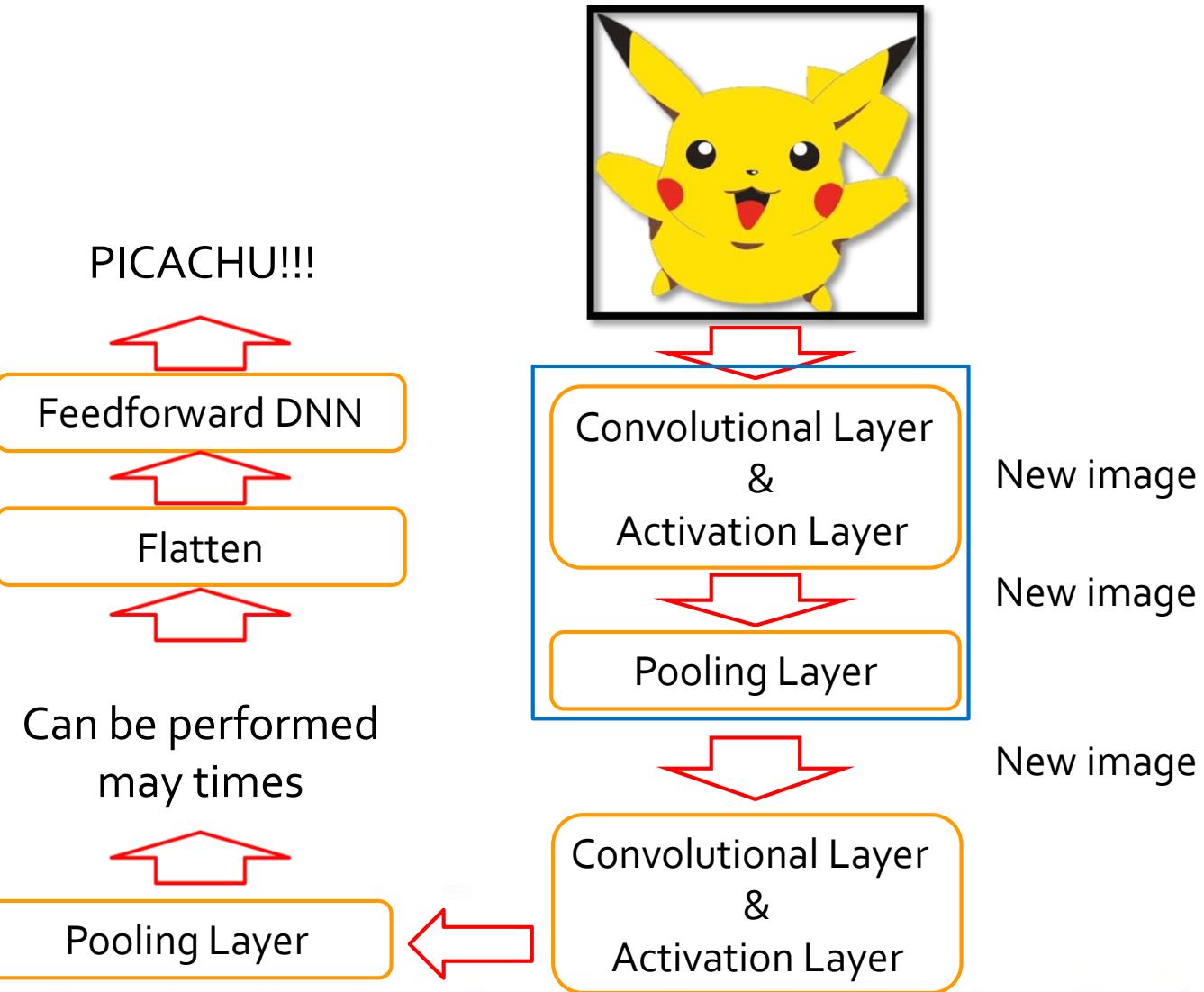
- CNN 學的是什麼？Ans: **filter** 內的參數

你應該要喜歡 XD
就算川普不喜歡

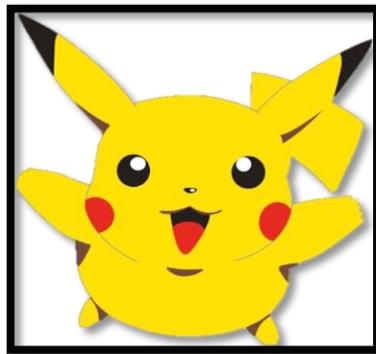


Photo credit: <https://indianexpress.com/article/trending/trending-globally/twitterati-lambast-donald-trump-after-he-posts-spoof-video-of-him-punching-cnn-4732236/>

CNN Structure



Convolution Layer vs. Convolution in CV



New image

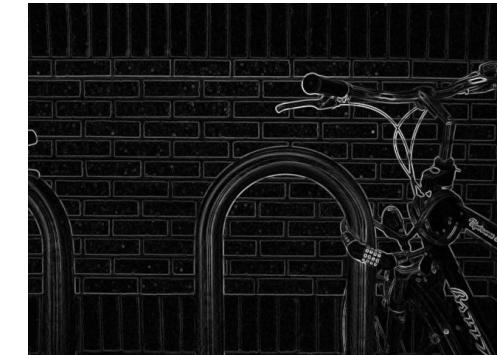


x-gradient

-1	0	1
-2	0	2
-1	0	1

y-gradient

-1	-2	-1
0	0	0
1	2	1



Property of Convolutional Layer

□ Convolution 執行越多次影像越小

Layer 1	<p>Image</p> <table border="1" style="border-collapse: collapse; width: 100%;"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td></tr><tr><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td></tr></table>	0	0	0	0	0	0	0	1	1	0	1	1	1	1	1	0	1	0	1	1	0	0	1	0	1	*	Filter	=	New image
0	0	0	0	0																										
0	0	1	1	0																										
1	1	1	1	1																										
0	1	0	1	1																										
0	0	1	0	1																										
					<table border="1" style="border-collapse: collapse; width: 100%;"><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>1</td></tr></table>	1	0	0	0	1	0	0	0	1																
1	0	0																												
0	1	0																												
0	0	1																												
					<table border="1" style="border-collapse: collapse; width: 100%;"><tr><td>1</td><td>2</td><td>2</td></tr><tr><td>1</td><td>2</td><td>3</td></tr><tr><td>3</td><td>1</td><td>3</td></tr></table>	1	2	2	1	2	3	3	1	3																
1	2	2																												
1	2	3																												
3	1	3																												

這個步驟有那些地方是可以變化的呢？

Layer 2	<table border="1" style="border-collapse: collapse; width: 100%;"><tr><td>1</td><td>2</td><td>2</td></tr><tr><td>1</td><td>2</td><td>3</td></tr><tr><td>3</td><td>1</td><td>3</td></tr></table>	1	2	2	1	2	3	3	1	3	*	Filter	=	6
1	2	2												
1	2	3												
3	1	3												



Filter Size

□ 5x5 filter

Image				
0	0	0	0	0
0	0	1	1	0
1	1	1	1	1
0	1	0	1	1
0	0	1	0	1

*

Filter				
1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

New image

$$= \boxed{3}$$

Padding (or not)

- Add additional zeros at the border of image

Image

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0
0	1	1	1	1	1	1	0
0	0	1	0	1	1	1	0
0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	0

$$\begin{matrix} & \text{Filter} \\ * & \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix} \end{matrix}$$

New image

0	1	1	0	0
1	1	2	2	0
2	1	2	3	2
0	3	1	3	2
0	0	2	0	2

Zero-padding 不會影響
convolution 的性質



Stride

- Shrink the output of the convolutional layer
- Set stride as 2

Image					Filter			New image				
0	0	0	0	0	*	1	0	0	=	1	2	
0	0	1	1	0		0	1	0		3	3	
1	1	1	1	1		0	0	1				
0	1	0	1	1								
0	0	1	0	1								

The diagram illustrates the convolution process with a stride of 2. The input image (5x5) has values [0,0,0,0,0; 0,0,1,1,0; 1,1,1,1,1; 0,1,0,1,1; 0,0,1,0,1]. It is multiplied by a filter (3x3) with values [1,0,0; 0,1,0; 0,0,1]. The result is a new image (2x2) with values [1,2; 3,3], where the stride of 2 means only the first element of each row and column of the filter is used in the multiplication.

Convolution on RGB (3D) Image

Filter 的深度
要跟影像相同

A filter

3×3×3

1	0	0
0	1	0
0	0	1

A RGB image

4×4×3

0	0	1	1
1	1	1	1
0	1	0	1
0	0	1	0

Conv. image

1	1	2	1
2	1	2	2
0	3	1	2
0	0	2	0

New image

4×4×1

2	3	5	3
5	3	5	5
3	4	5	6
1	2	5	2

0	1	0
0	1	0
0	1	0

0	0	1	0
1	0	1	1
0	1	0	1
1	0	1	0

1	0	2	1
1	1	2	2
2	1	2	2
1	1	1	1

0	0	1
0	1	0
1	0	0

0	1	0	1
1	1	0	1
0	0	0	1
0	1	1	1

0	2	1	1
2	1	1	1
1	0	2	2
0	1	2	1

Depth 1

1 filter

3x3x3

1	0	0
0	1	0
0	0	1

*

RGB images
4x4x3

0	0	1	1
1	1	1	1
0	1	0	1
0	0	1	0

0	0	0	0
1	0	0	0
0	1	0	0
0	0	1	0

New image
4x4x1

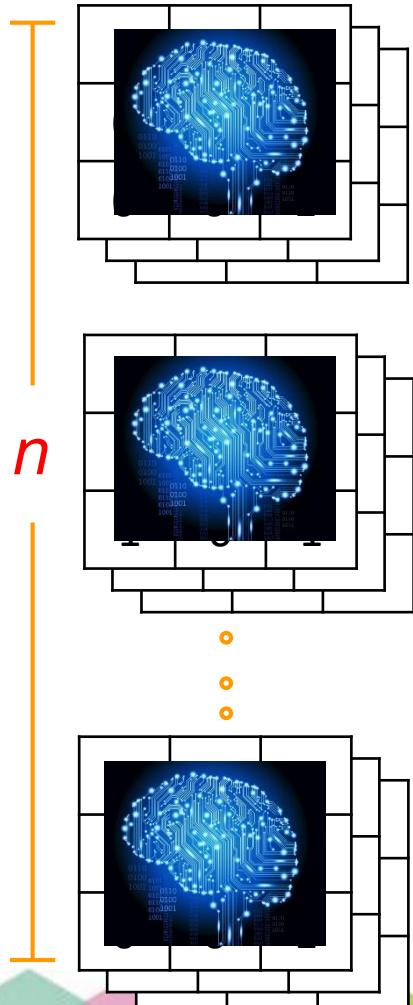
2	3	5	3
5	3	5	5
3	4	5	6
1	2	5	2

=

Depth n

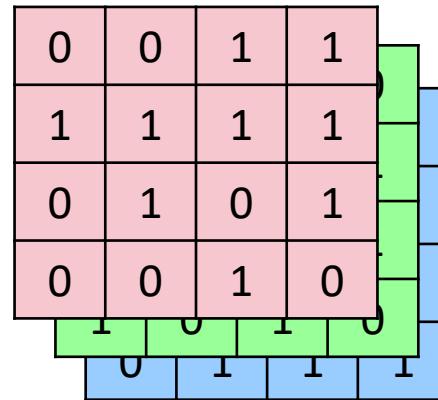
n filters

$n, 3 \times 3 \times 3$



*

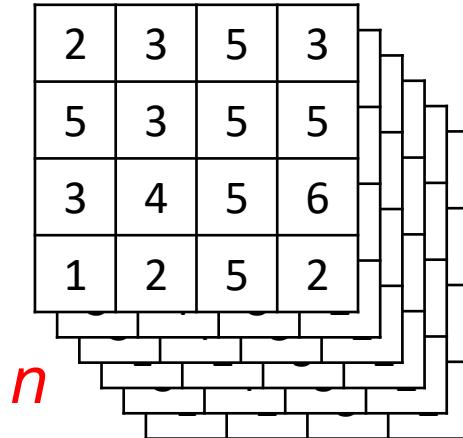
RGB images
 $4 \times 4 \times 3$



New image

$4 \times 4 \times n$

=

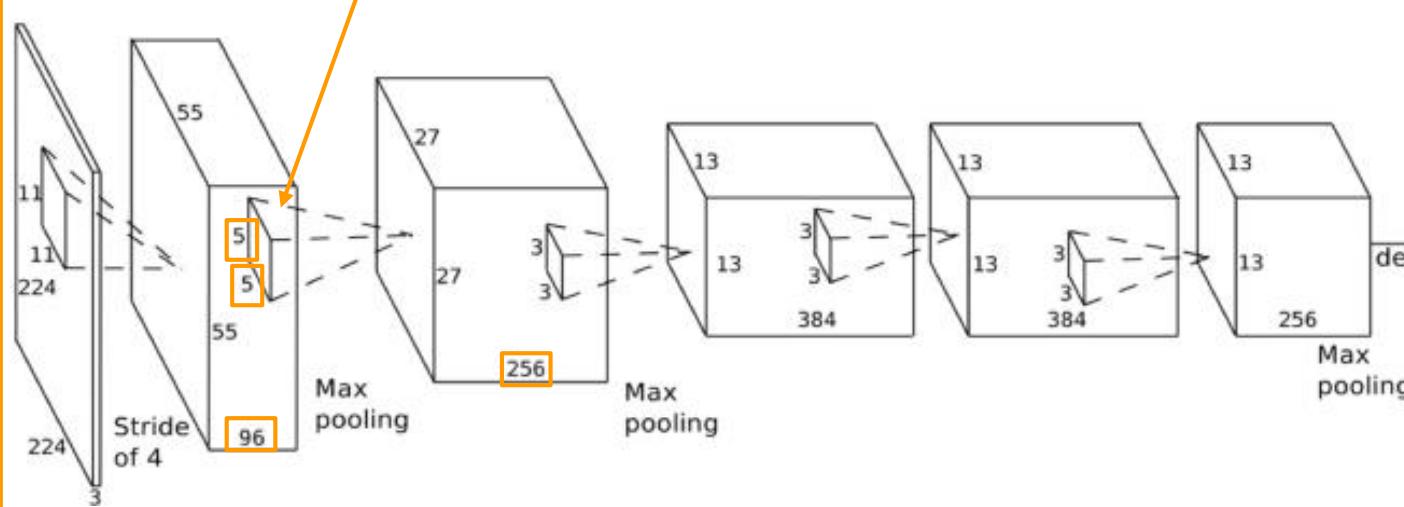


如果 filters 都給定了
，那 CNN 是在學什麼？

An Example of CNN Model

CNN

256 個 filters ($5 \times 5 \times 96$)



DNN

Flatten

Image Size Transformation through a Conv. Layer

□ Zero-padding

- With : $(W_{n+1}, H_{n+1}, \times) = (W_n, H_n, \times)$
- Without: $(W_{n+1}, H_{n+1}, \times) = (W_n - \frac{W_f - 1}{2}, H_n - \frac{H_f - 1}{2}, \times)$

□ Stride s

- $(W_{n+1}, H_{n+1}, \times) = \left(\frac{W_n}{s}, \frac{H_n}{s}, \times\right)$

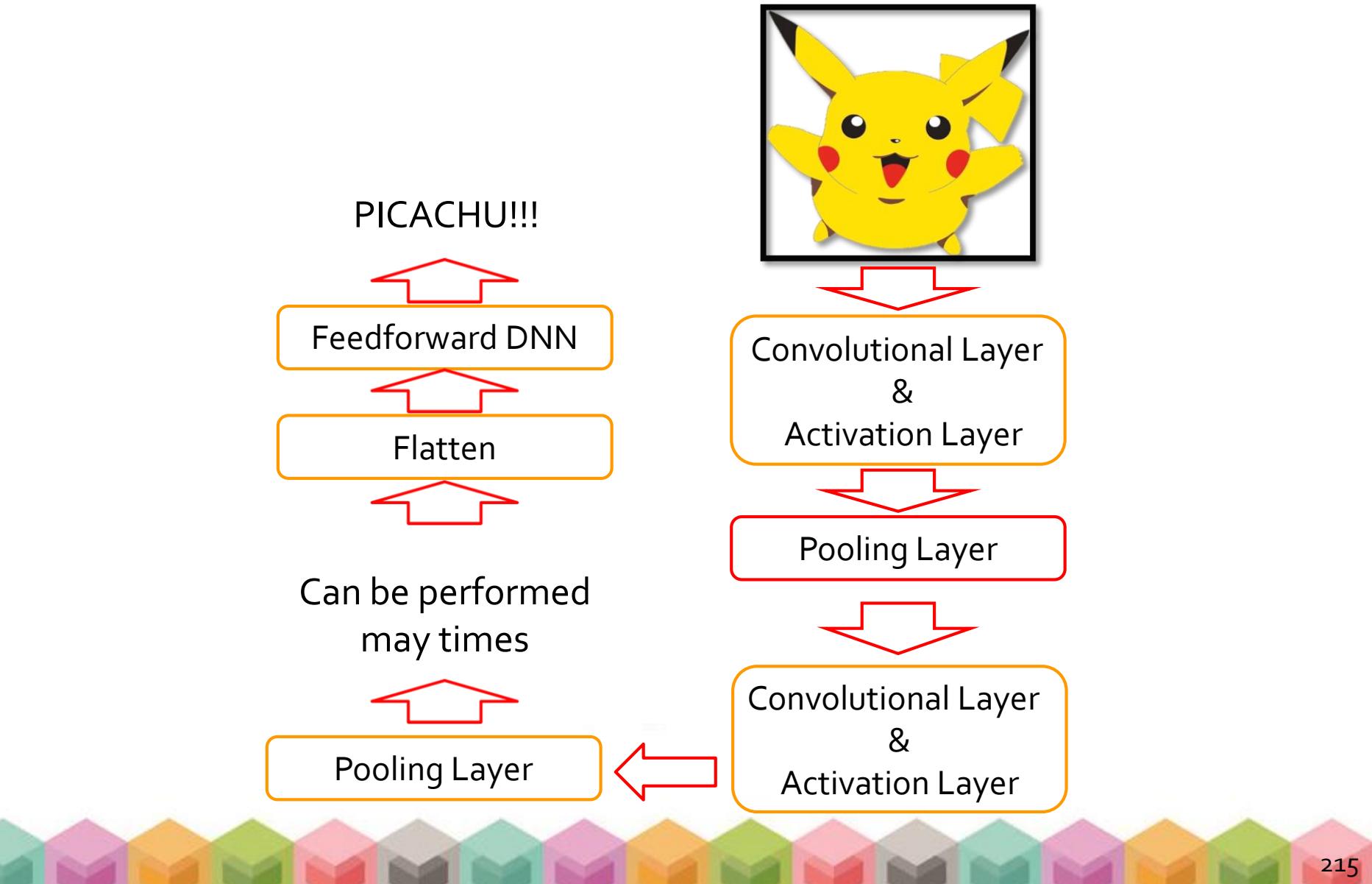
□ k filters

- $(W_{n+1}, H_{n+1}, k) = (W_n, H_n, D_n)$

□ Total number of weights is needed from L_n to L_{n+1}

- $W_f \times H_f \times D_n \times k + k$

CNN Structure



Pooling Layer

- Why do we need pooling layers?
 - 在保留影像所包含資訊的前提下，減少影像大小 (down-sampling)
 - 減少計算量
 - 減少參數的數量
- Max pooling
 - 保留最大值 → 保留 pattern 是否存在於某區域內的性質

1	2	2	0
1	2	3	2
3	1	3	2
0	2	0	2

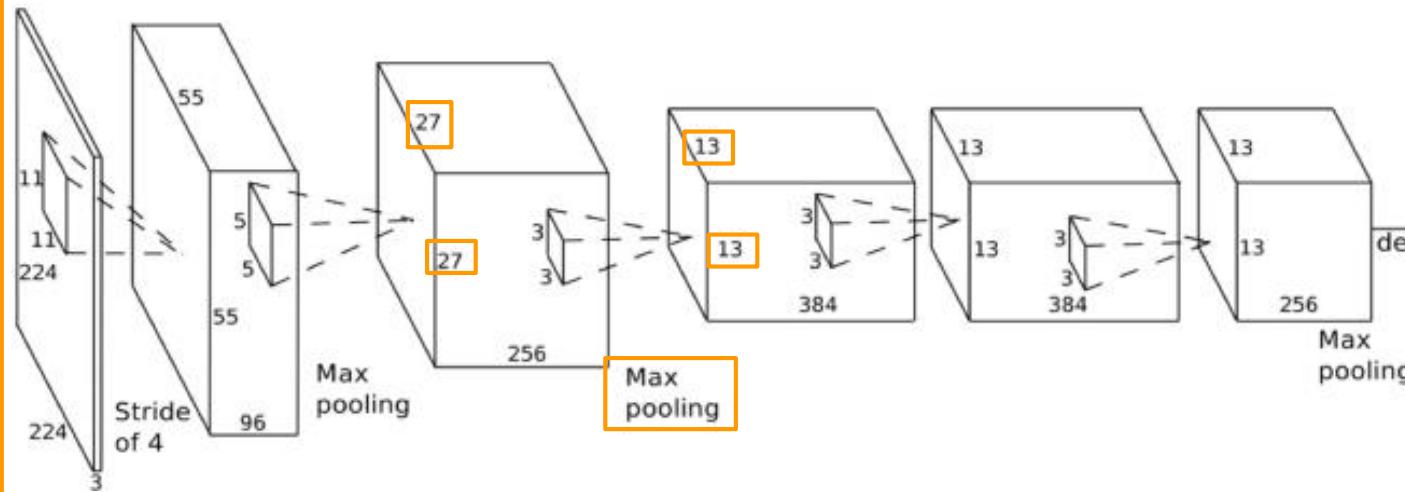
Max pooling
→

2	3
3	3

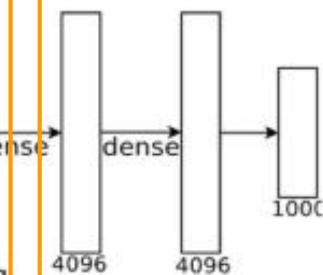
*How about average pooling?

An Example of CNN Model

CNN



DNN

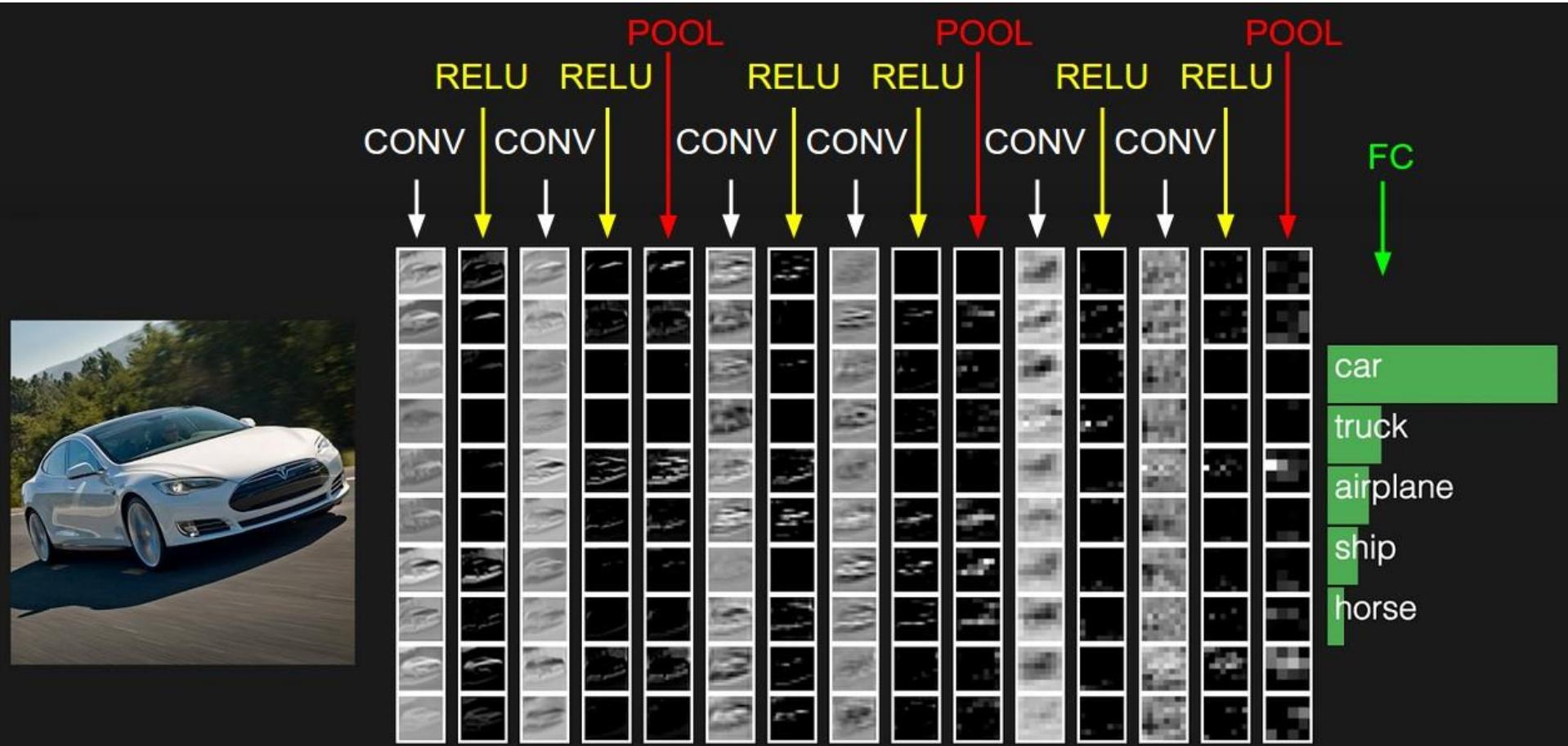


Flatten



A CNN Example (Object Recognition)

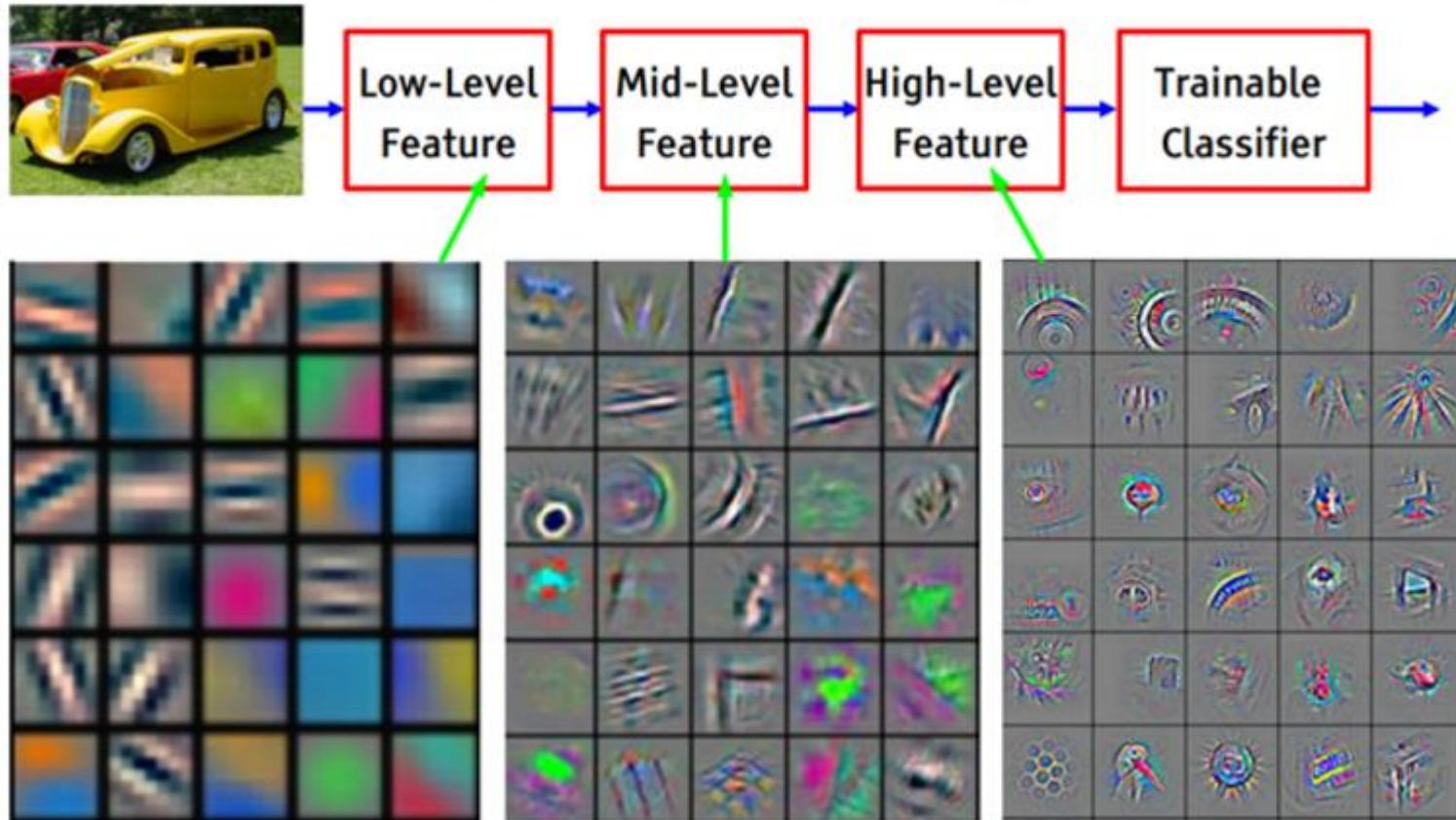
□ CS231n, Stanford [[Ref](#)]



Filters Visualization

□ RSIP VISION [Ref]

State of the art object recognition using CNNs



CNN 常見的應用

Classification



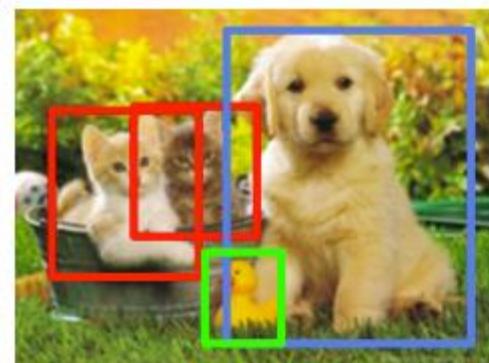
CAT

Classification + Localization



CAT

Object Detection



CAT, DOG, DUCK

Instance Segmentation



CAT, DOG, DUCK

Figure reference

<https://mathematica.stackexchange.com/questions/141598/object-detection-and-localization-using-neural-network>

Generative Adversarial Networks

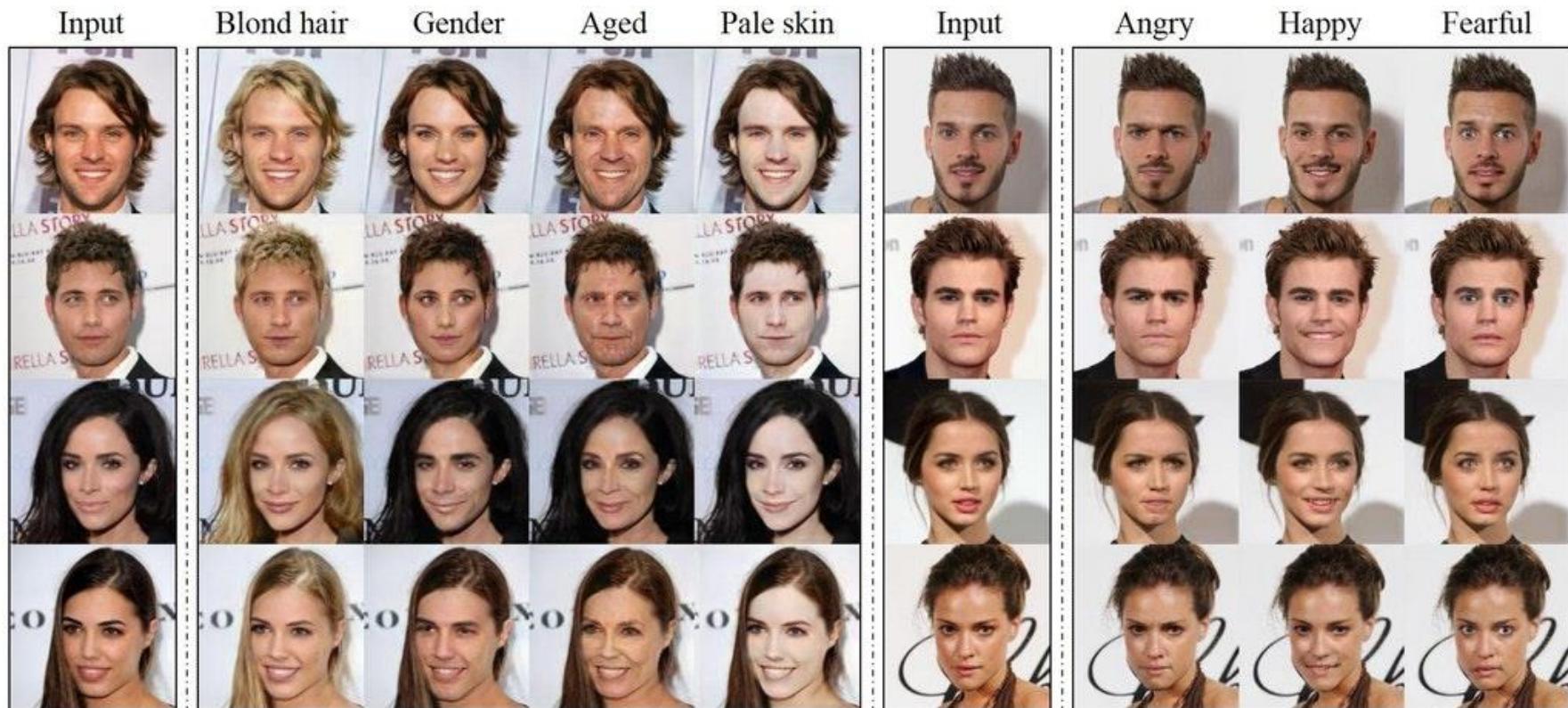


Figure reference

https://www.reddit.com/r/MachineLearning/comments/7fro3g/r_stargan_unified_generative_adversarial_networks/

CNN in Keras

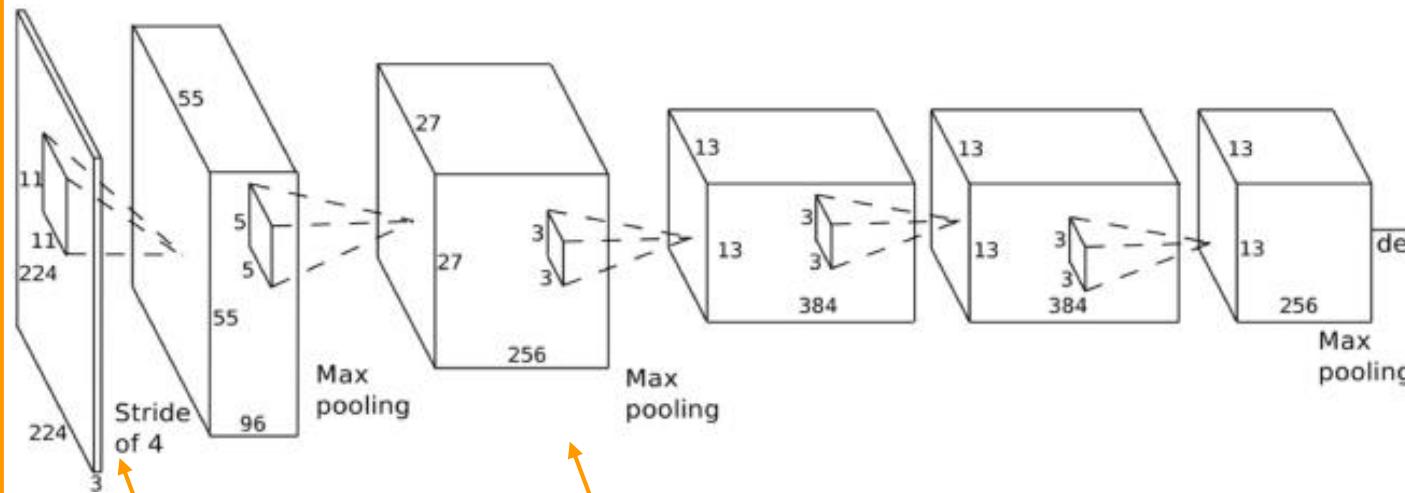
Object Recognition

Dataset: CIFAR-10

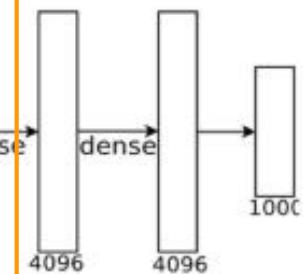


Differences between CNN and DNN

CNN

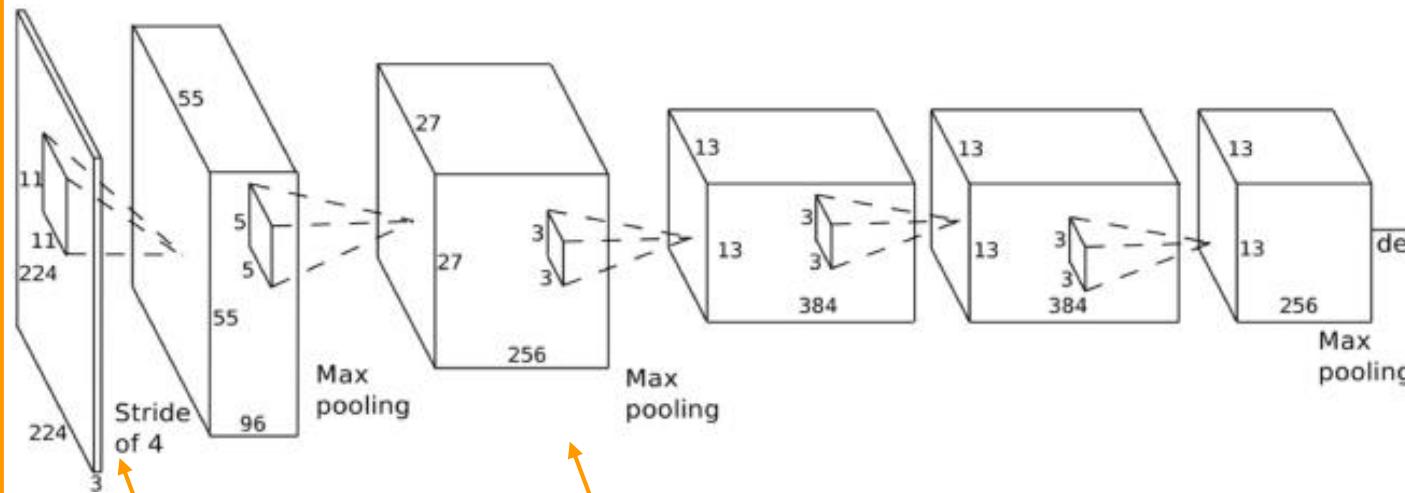


DNN

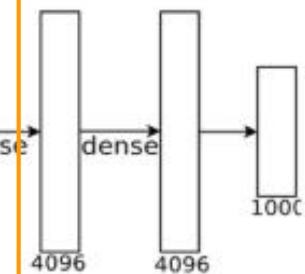


Differences between CNN and DNN

CNN



DNN



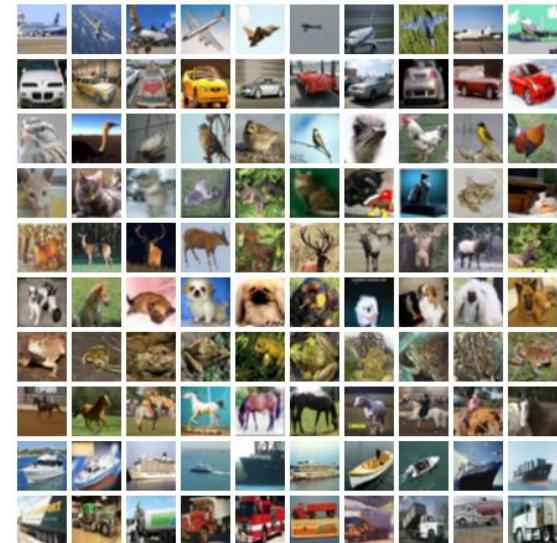
Input

Convolutional Layer



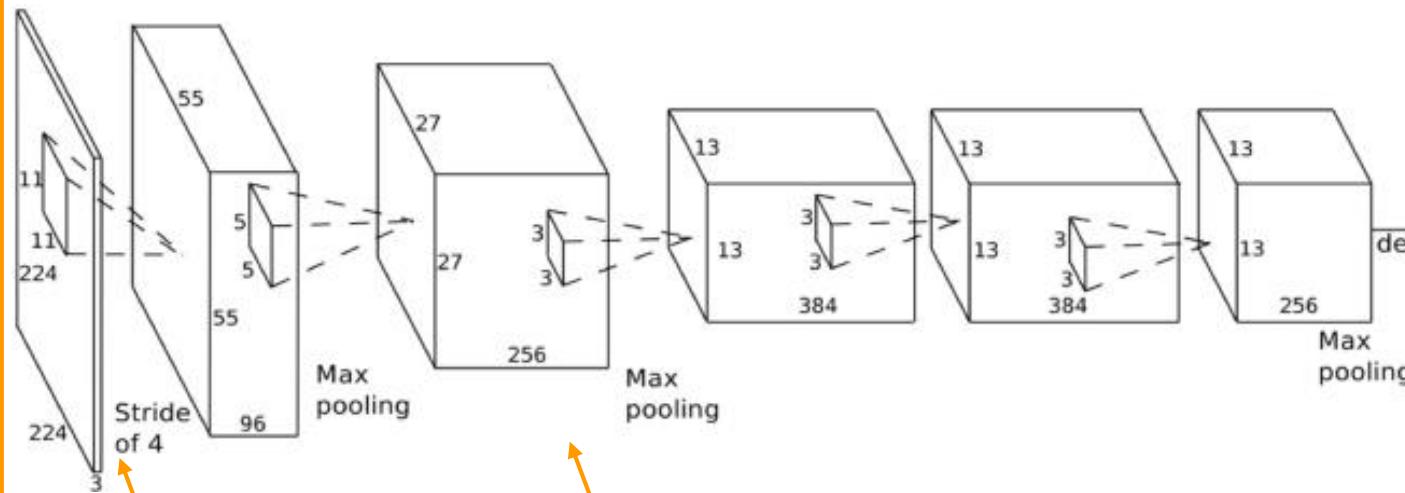
CIFAR-10 Dataset

- 60,000 (50,000 training + 10,000 testing) samples,
32x32 color images in **10** classes
- 10 classes
 - airplane, automobile, ship, truck,
bird, cat, deer, dog, frog, horse
- Official website
 - <https://www.cs.toronto.edu/~kriz/cifar.html>

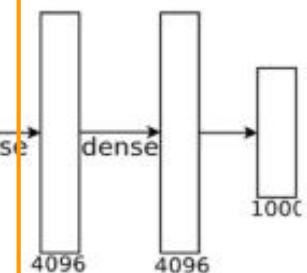


Differences between CNN and DNN

CNN



DNN



Input

Convolutional Layer

Building Your Own CNN Model

```
'''CNN model'''  
model = Sequential()  
model.add(  
    Convolution2D(32, 3, 3, border_mode='same',  
                  input_shape=X_train[0].shape)  
)  
model.add(Activation('relu'))  
model.add(Convolution2D(32, 3, 3))  
model.add(Activation('relu'))  
model.add(MaxPooling2D(pool_size=(2, 2)))  
model.add(Dropout(0.2))
```

32 個 3x3 filters

'valid' : without padding
'same': perform padding
default value is zero

CNN

```
model.add(Flatten())  
model.add(Dense(512))  
model.add(Activation('relu'))  
model.add(Dropout(0.5))  
model.add(Dense(10))  
model.add(Activation('softmax'))
```

DNN

Model Compilation

```
'''setting optimizer'''
# define the learning rate
learning_rate = 0.01
learning_decay = 0.01 / 32

# define the optimizer
sgd = SGD(lr=learning_rate, decay=learning_dacay, momentum=0.9,
           nesterov=True)

# Let's compile
model.compile(loss='categorical_crossentropy', optimizer=sgd,
               metrics=['accuracy'])
```

Number of Parameters of Each Layers

Only one function

Layer (type)	Output Shape	Param #
convolution2d_1 (Convolution2D)	(32*3*3*3 + 32 = 896)	896
activation_1 (Activation)	(None, 32, 32, 32)	0
convolution2d_2 (Convolution2D)	(32*3*3*32 + 32 = 9,248)	9248
activation_2 (Activation)	(None, 30, 30, 32)	0
maxpooling2d_1 (MaxPooling2D)	(None, 15, 15, 32)	0
dropout_1 (Dropout)	(None, 15, 15, 32)	0
flatten_1 (Flatten)	(None, 7200)	0
dense_1 (Dense)	(7200 * 512 + 512 = 3,686,912)	3686912
activation_3 (Activation)	(None, 512)	0
dropout_2 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 10)	5130
activation_4 (Activation)	(None, 10)	0
Total	Total params:	3702186

Let's Start Training

- ❑ Two validation methods
 - ❑ Validate with splitting training samples
 - ❑ Validate with testing samples

```
''' training'''
# define batch size and # of epoch
batch_size = 128
epoch = 32

# [1] validation data comes from training data
fit_log = model.fit(X_train, Y_train, batch_size=batch_size,
                     nb_epoch=epoch, validation_split=0.1,
                     shuffle=True)
# [2] validation data comes from testing data
fit_log = model.fit(X_train, Y_train, batch_size=batch_size,
                     nb_epoch=epoch, shuffle=True,
                     validation_data=(X_test, Y_test))
```

Model Saving and Prediction

□ Saving/loading the whole CNN model

```
'''saving model'''
from keras.models import load_model
model.save('my_first_cnn_model.h5')
del model

'''loading model'''
model = load_model('my_first_cnn_model.h5')
```

□ Predicting the classes with new image samples

```
'''prediction'''
pred = model.predict_classes(X_test, batch_size, verbose=0)
```

Let's Try CNN



Figure reference

<https://unsplash.com/collections/186797/coding>

Practice – Design a CNN Model

- 設計一個 CNN model，並讓他可以順利執行
(my_cnn.py Line 20-34)

```
# set dataset path
dataset_path = './cifar_10/'
classes = 10
X_train, X_test, Y_train, Y_test =
    utils.read_dataset(dataset_path, "img")

'''CNN model'''
model = Sequential()
# 請建立一個 CNN model
# CNN

model.add(Flatten())
# DNN
```



Spot Lights in the Code

- Check the format of training dataset / validation dataset
- Design your own CNN model
- Don't forget saving the model



Tips for Setting Hyper-parameters (1)

□ Convolutional Layer

- 比起使用一個 size 較大的 filter (7×7)，可以先嘗試連續使用數個 size 小的 filter (3×3)
- Stride 的值與 filter size 相關，通常 $stride \leq \frac{W_f - 1}{2}$

□ Very deep CNN model (16+ Layers) 多使用 3×3 filter 與 stride 1

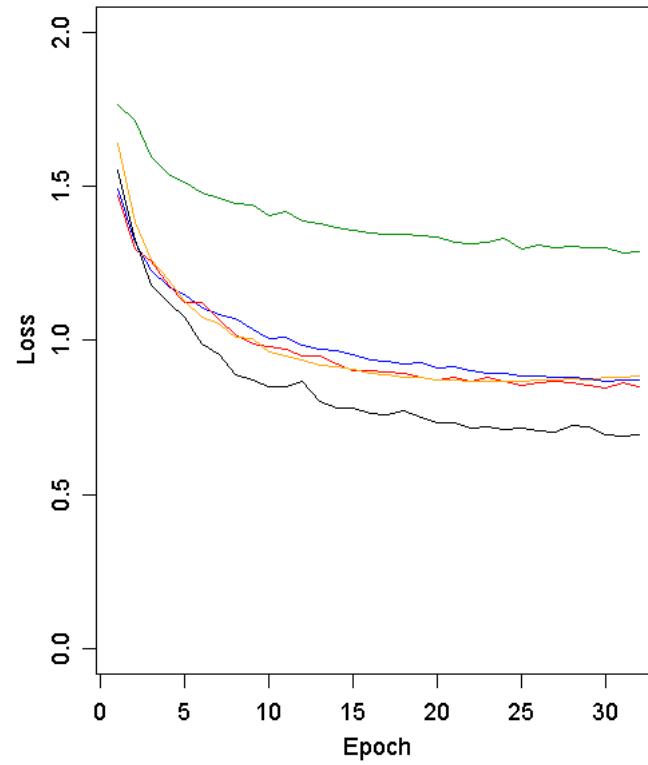
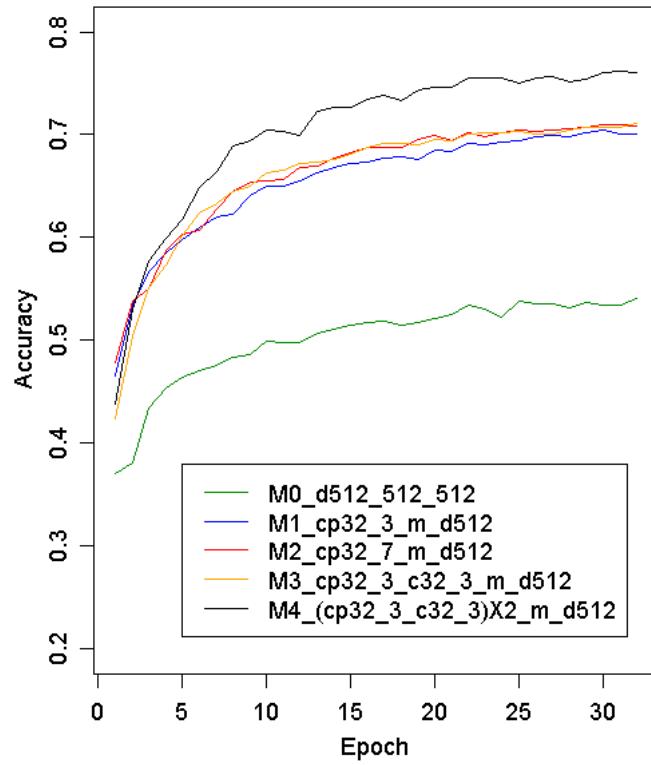


Tips for Setting Hyper-parameters (2)

- Zero-padding 與 pooling layer 是選擇性的結構
- Zero-padding 的使用取決於是否要保留邊界的資訊
- Pooling layer 利用減少影像所包含資訊以降低運算量、參數的數量與避免 overfitting，一般不會大於 3×3 (常用 2×2 及 stride 2)
- 嘗試修改有不錯效能的 model，例如 VGG-16，會比建立一個全新的模型容易有好的結果

Training History of Different Models

- $\text{cp32_3} \rightarrow$ convolution layer with zero-padding, 32 3x3 filters
- $\text{d} \rightarrow$ fully connected NN layers





Semi-supervised Learning

妥善運用有限的標籤資料 (optional)

常面對到的問題

- 收集到的標籤遠少於實際擁有的資料量
- 有 60,000 張照片，只有 5,000 張知道照片的標籤
- 該如何增加 label 呢？
 - Crowd-sourcing
 - Semi-supervised learning

Semi-supervised Learning

- 假設只有 5000 個圖有 label
- 先用 labeled dataset to train model
 - 至少 train 到一定的程度 (良心事業)
- 拿 unlabeled dataset 來測試，挑出預測好的 unlabeled dataset
 - Example: softmax output > 0.9 → self-define
- 假設預測的都是對的 (unlabeled → labeled)
 - 有更多 labeled dataset 了！
 - Repeat the above steps



七傷拳

- 加入品質不佳的 labels 反而會讓 model 變差
 - ▣ 例如：加入的圖全部都是 “馬” ，在訓練過程中，模型很容易變成 “馬” 的分類器
- 慎選要加入的 samples
 - ▣ Depends on your criteria ☺



Transfer Learning

Utilize well-trained model on YOUR dataset (optional)



Introduction

- “transfer”: use the knowledge learned from task A to tackle another task B
- Example: 綿羊/羊駝 classifier

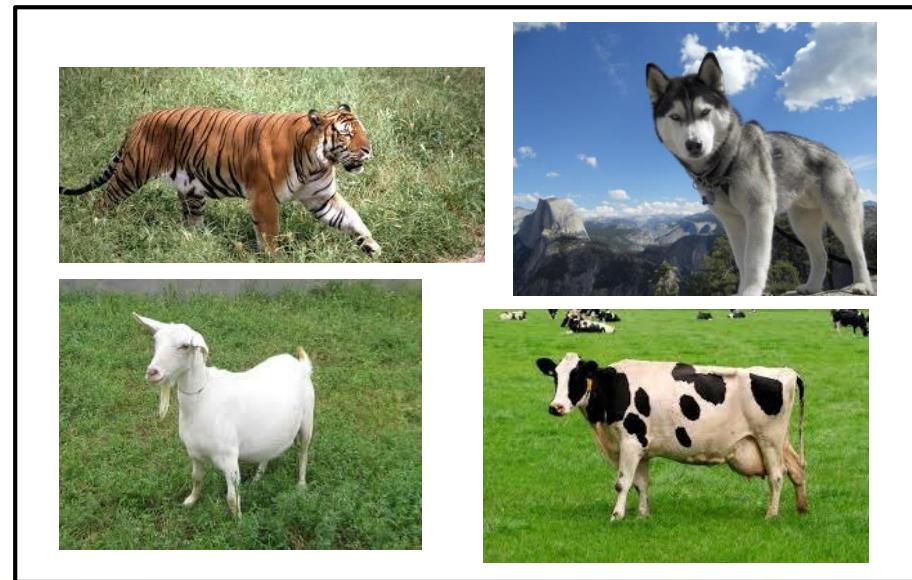


綿羊



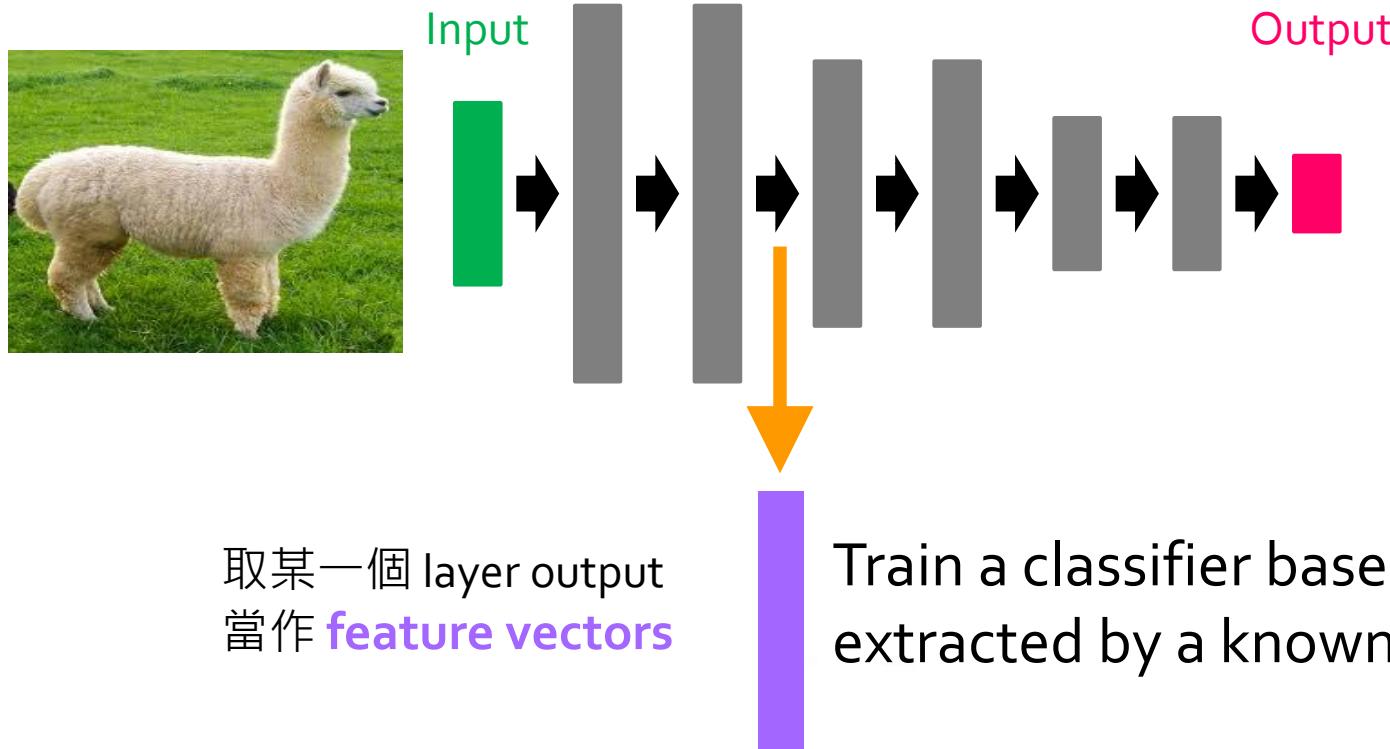
羊駝

其他動物的圖



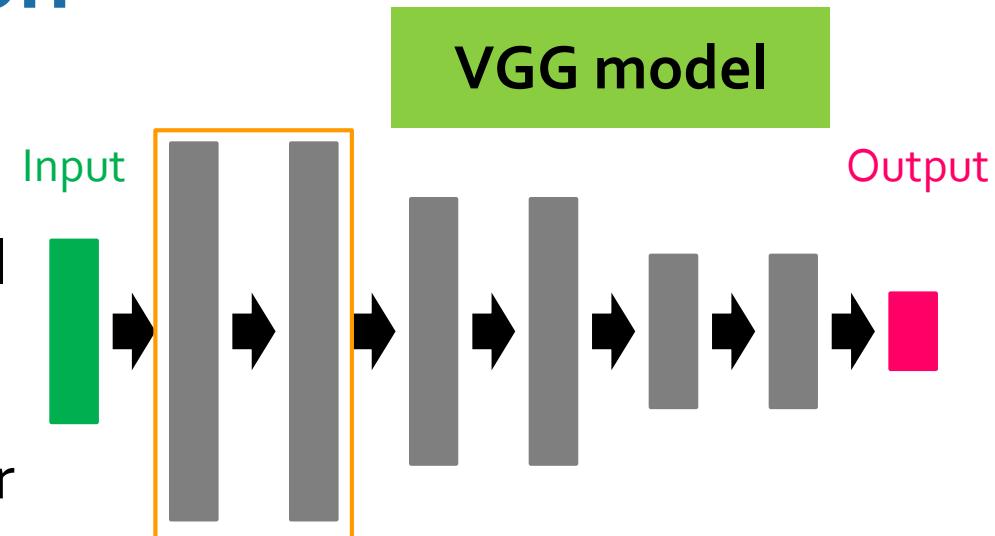
Use as Fixed Feature Extractor

- ❑ A known model, like VGG, trained on ImageNet
 - ❑ ImageNet: 10 millions images with labels

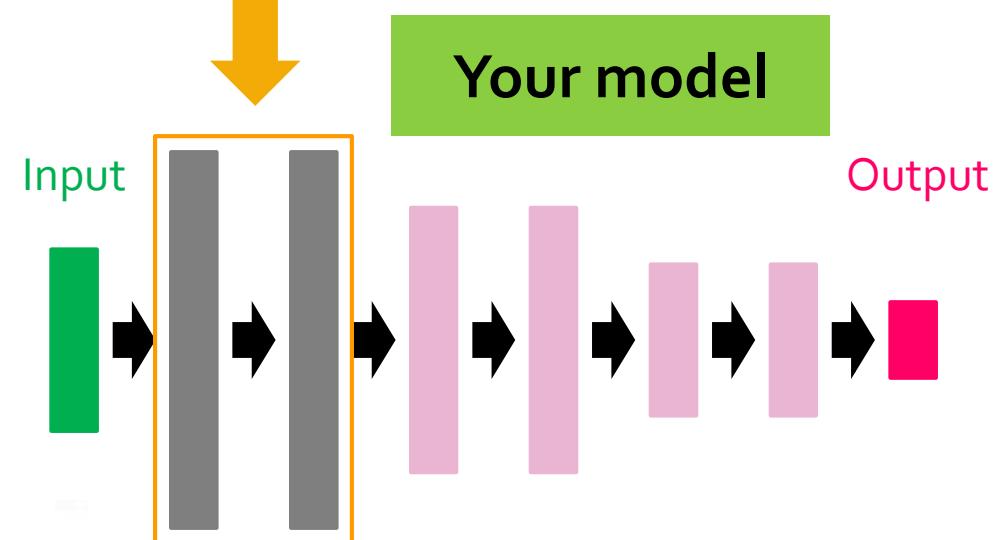


Use as Initialization

- ❑ Initialize your net by the weights of a known model



- ❑ Use your dataset to further train your model



- ❑ Fine-tuning the known model



Short Summary

- Unlabeled data (lack of y)
 - Semi-supervised learning
- Insufficient data (lack of both x and y)
 - Transfer learning (focus on layer transfer)
 - Use as fixed feature extractor
 - Use as initialization
 - Resources: <https://keras.io/applications/>

Jason Yosinski, Jeff Clune, Yoshua Bengio, Hod Lipson, "How transferable are features in deep neural networks?", <https://arxiv.org/abs/1411.1792>, 2014



Summarization

What We Have Learned Today





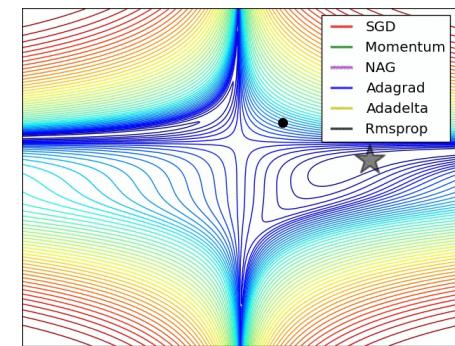
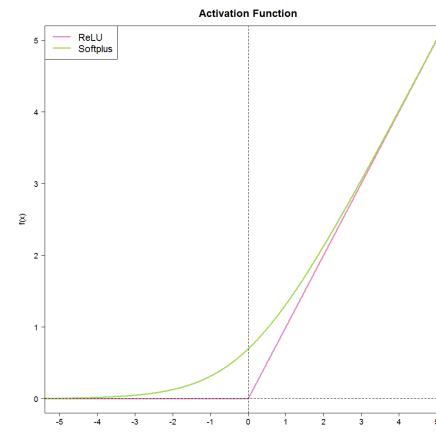
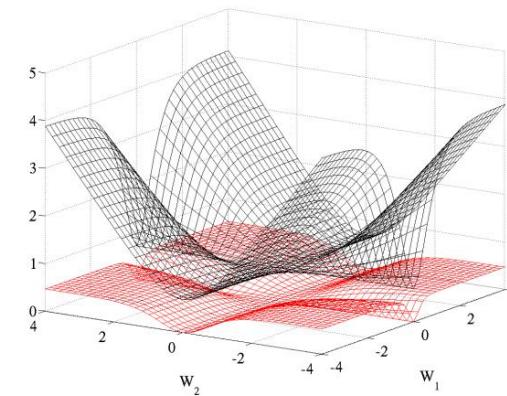
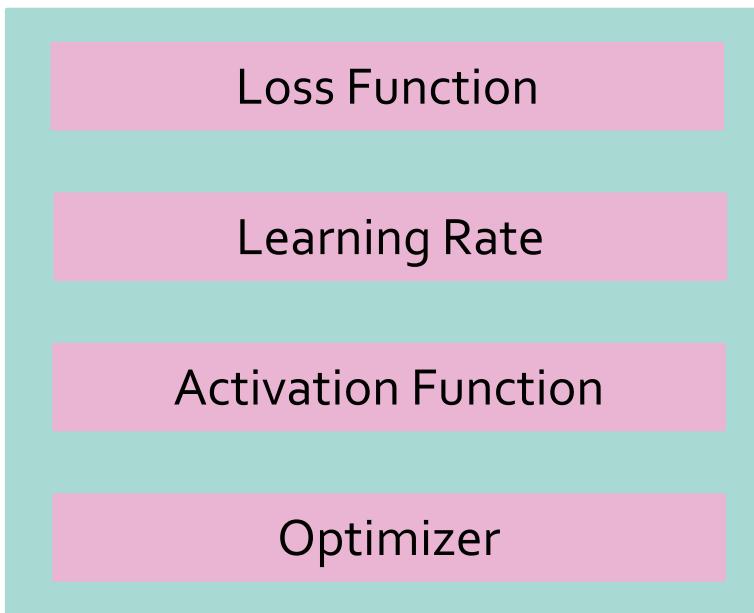
Recap – Fundamentals

❑ Fundamentals of deep learning

- ❑ A neural network = a function
- ❑ Gradient descent
- ❑ Stochastic gradient descent
- ❑ Mini-batch
- ❑ Guidelines to determine a network structure

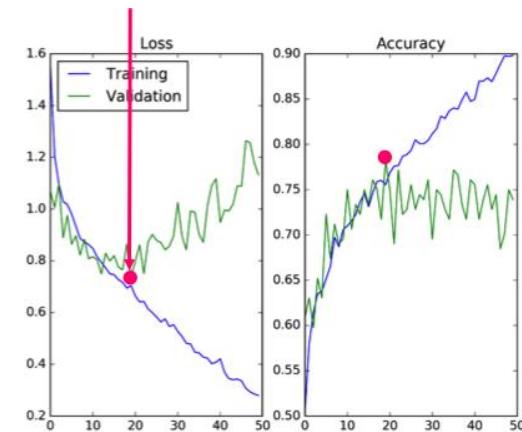
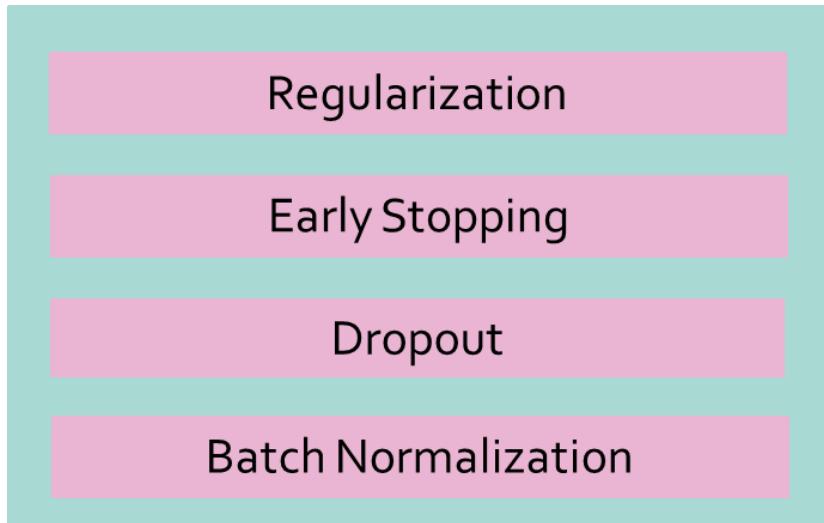
Recap – Improvement on Training Set

- How to improve performance on training dataset

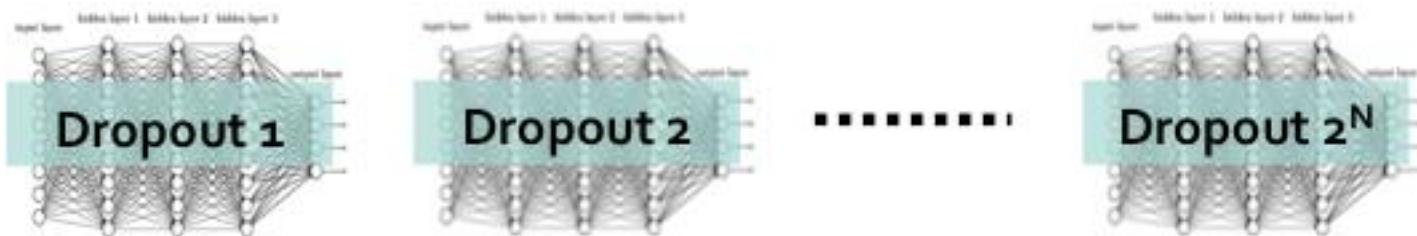


Recap – Improvement on Testing Set

- How to improve performance on testing dataset



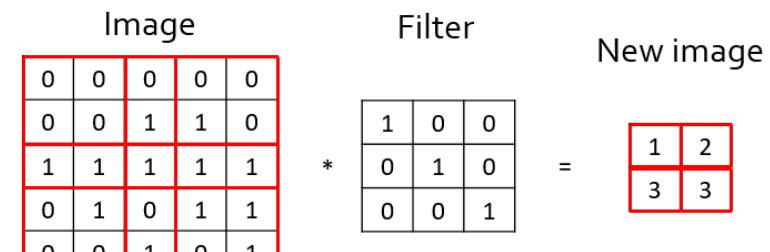
$$\text{Loss}_{\text{reg}} = \sum (y - (b + \sum w_i x_i))^2 + \alpha(\text{regularizer})$$



Recap – CNN

❑ Fundamentals of CNN

- ❑ Concept of filters
- ❑ Hyper-parameters
 - ❑ Filter size
 - ❑ Zero-padding
 - ❑ Stride
 - ❑ Depth (total number of filters)



❑ How to train a CNN in Keras

- ❑ CIFAR-10 dataset





補充資料

Model 儲存與讀取

```
# save model  
model.save("filename.h5")  
  
# load model  
from keras.model import load_model  
model_test = load_model("filename.h5")  
  
# BTW, use model.summary() to check your layers  
Model_test.summary()
```

How to Get Trained Weights

- `weights = model.get_weights()`
- `model.layers[1].set_weights(weights[0:2])`

```
# get weights
myweight = model_test.get_weights()

# set weights
model_test.layers[1].set_weights(myweights[0:2])

# BTW, use model.summary() to check your layers
model_test.summary()
```

How to Get Layer Output

```
# get weights
model_layer1 = Sequential()
model_layer1.add(Dense(128, input_dim=200,
                      weights=model_test.layers[0].get_weights()))

model_layer1.add(Activation('relu'))

# predict
model_layer1.predict(x_train[0:1])
```

Fit_Generator

- 當資料太大無法一次讀進時 (memory limitation)

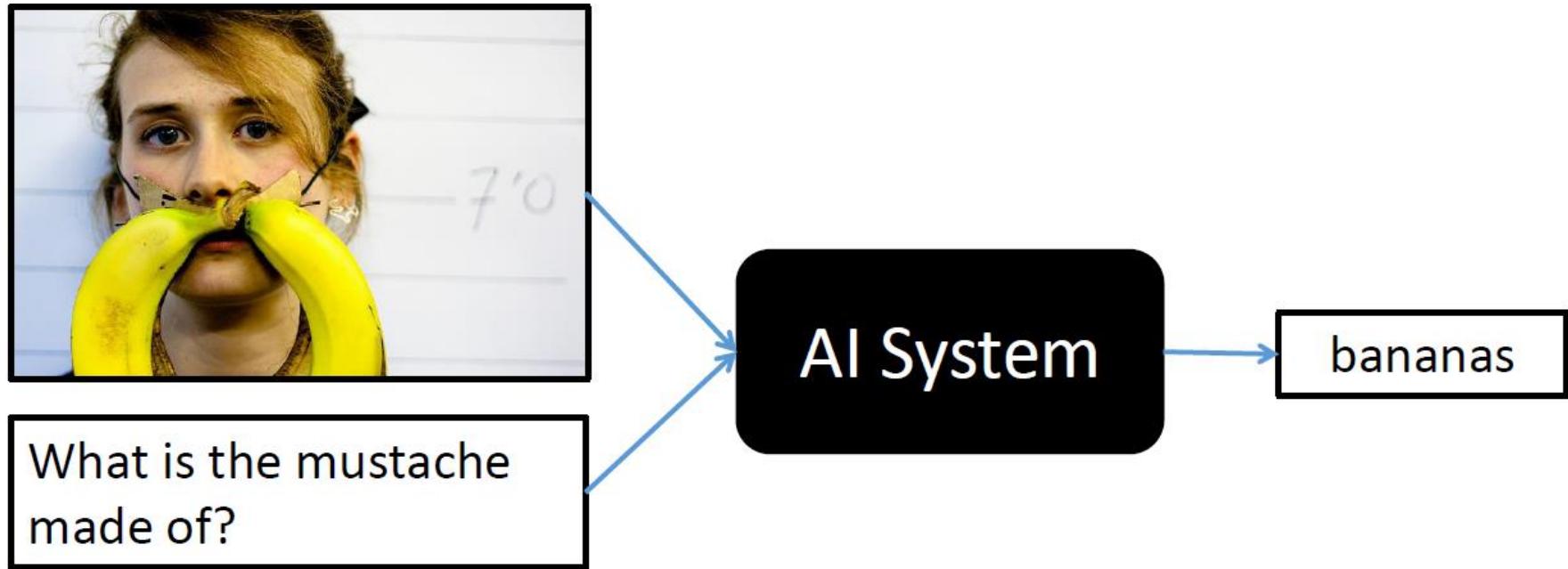
```
# get weights
def train_generator(batch_size):
    while 1:
        data = np.genfromtext('pkgo_city66_class5_v1.csv',
                             delimiter=',',
                             skip_header=1)
        for i in range(0,np.floor(batch_size/len(data)))
            x = data[i*batch_size:(i+1)*batch_size,:200]
            y = data[i*batch_size:(i+1)*batch_size,200]
            yield x,y

Model.fit_generator(train_generator(28),
                    epochs=30,
                    steps_per_epoch=100,
                    validation_steps=100) # or validation_data
```



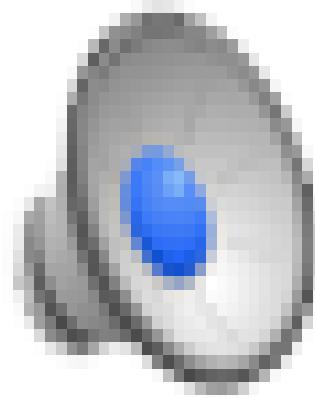
Deep Learning Applications

Visual Question Answering



source: <http://visualqa.org/>

Video Captioning



Answer: a woman is carefully slicing tofu.

Generated caption: a woman is cutting a block of tofu.

Text-To-Image

the flower has petals that
are bright pinkish purple
with white stigma



this white and yellow flower
have thin white petals and a
round yellow stamen



<https://arxiv.org/pdf/1701.00160.pdf>



Vector Arithmetic for Visual Concepts



smiling
woman



neutral
woman



neutral
man



smiling man

<https://arxiv.org/pdf/1511.06434.pdf>



Go Deeper in Deep Learning

- “Neural Networks and Deep Learning”
 - <http://neuralnetworksanddeeplearning.com/>
- “Deep Learning”
 - <http://www.iro.umontreal.ca/~bengioy/dlbook/>
- Course: Machine learning and having it deep and structured
 - http://speech.ee.ntu.edu.tw/~tlkagk/courses_MLSD15_2.html
- Keras 官方網站，非常詳細
 - [Keras documentation](#) and [Keras Github](#) 從 example/ 中找適合範例
- 台大電機李宏毅教授 [Youtube 頻道](#)
- Stanford cs231n [Convolutional Neural Networks for Visual Recognition](#)

- 若有任何問題，歡迎來信 twcmchang@gmail.com



一起往訓練大師的路邁進吧
謝謝大家！

