

# 初探類神經網路

Week #2

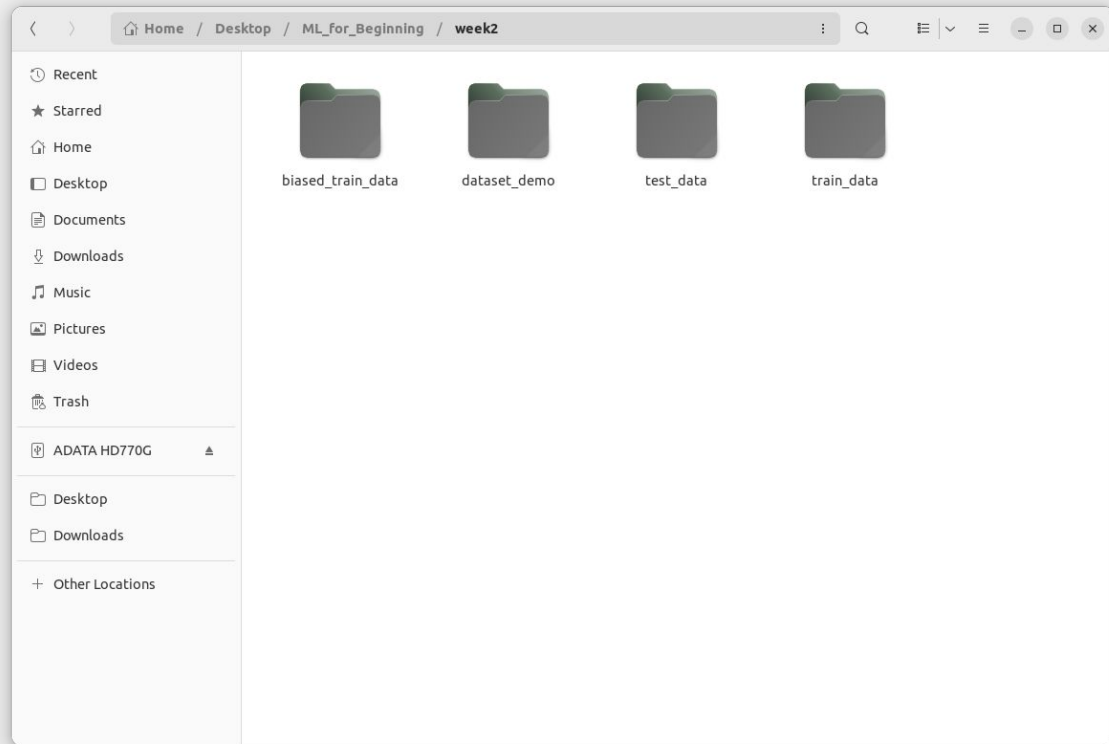
# Resources

到我的 Github 上下載本次課程資源：

[https://github.com/CelleryLin/ML\\_for\\_Beginning](https://github.com/CelleryLin/ML_for_Beginning)

```
# or use git clone/pull  
$ git clone https://github.com/CelleryLin/ML\_for\_Beginning.git
```

# Resources



# 複習一下

從零開始架構一個自己的機器學習模型通常會經過一些步驟



**資料前處理    選擇模型    訓練    測試    執行**

# 模型訓練 Training

今天我們會著重在模型的訓練



資料前處理

選擇模型

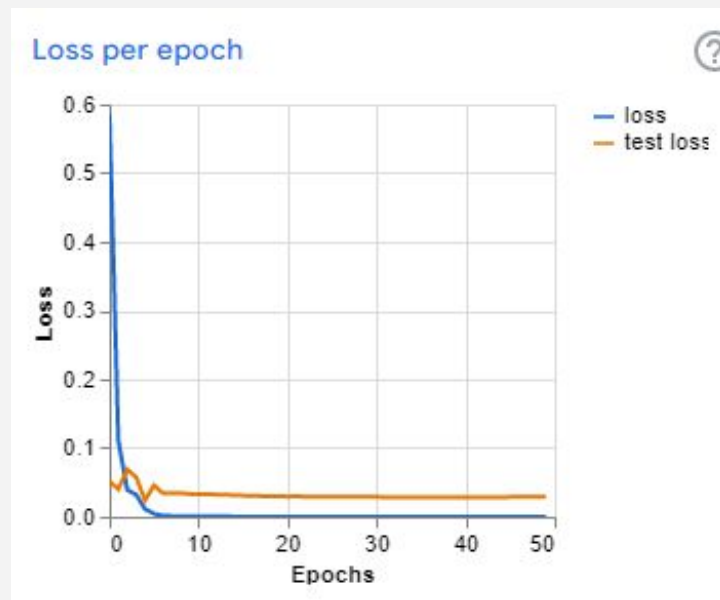
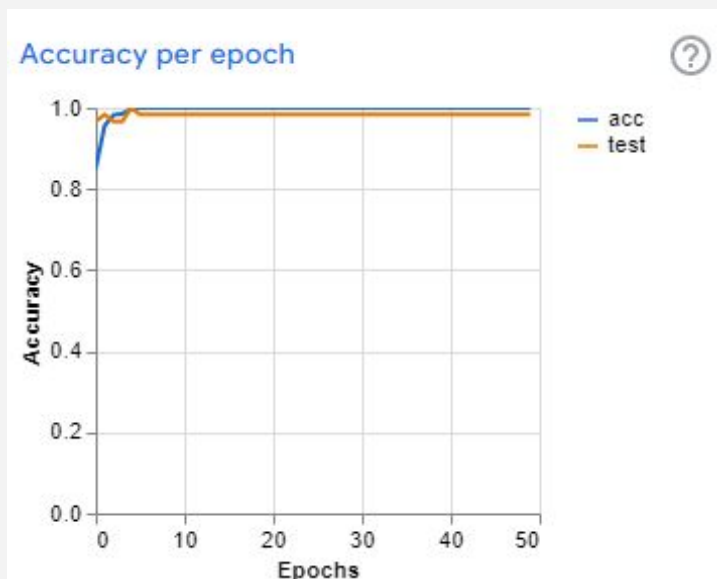
**訓練**

測試

執行

# 模型訓練 Training

今天的目標，就是看懂這個圖表。



# 模型訓練 Training

訓練就是讓模型學習，我們主要講兩種就好：

## 監督式學習

Supervised Learning

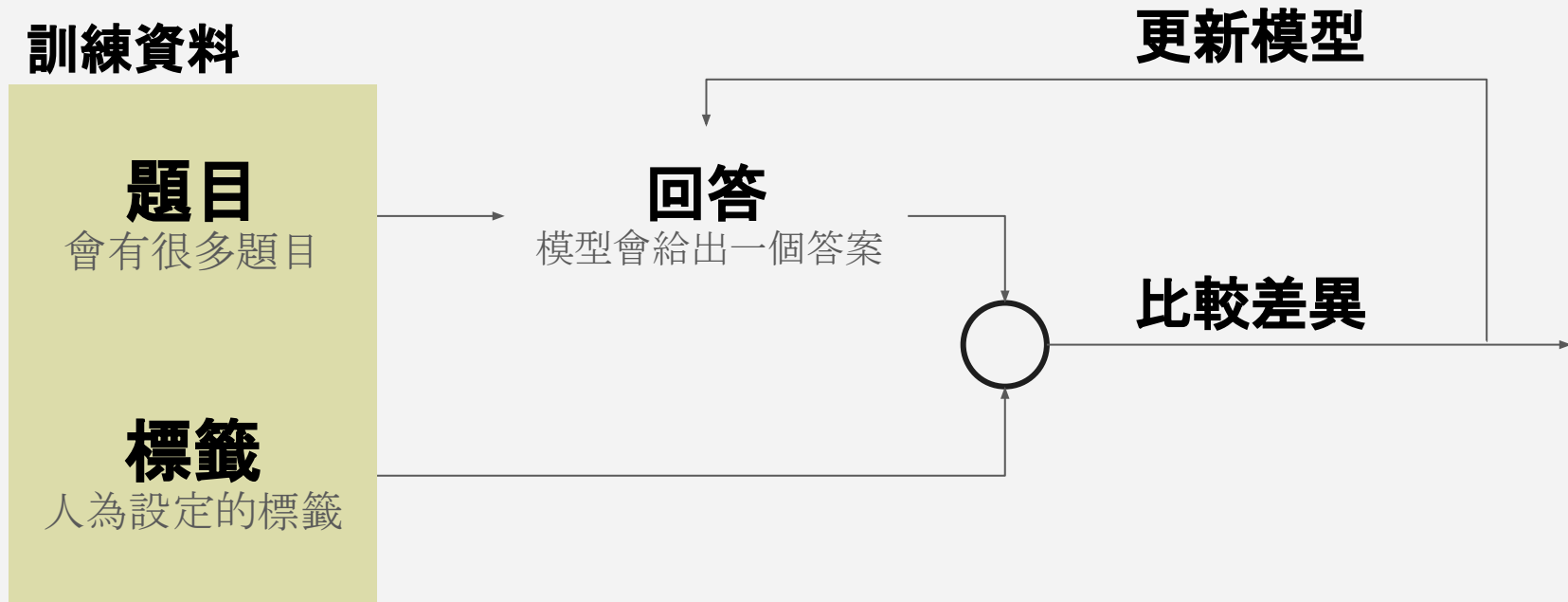
**須經過人為標籤**(就是給模型答案)，電腦在學習的過程透過對比誤差，一邊修正去達到更精準的預測。

## 非監督式學習

Unsupervised Learning

**不須標籤**(就是不需給模型答案)，模型自己依照關聯性去歸類形成集群，不對資訊有正確或不正確的判別。

# 監督式學習 Supervised Learning





# 補充：非監督式學習 Supervised Learning

**訓練資料**

**題目**

會有很多題目



**聚類、降維**

模型會自己去歸納分類



# 補充：模型訓練 Training

這兩位模型基本上優缺點就是互補的。

## 監督式學習

Supervised Learning

**優點：**因為目標明確，運算量相對較少，所以會跑比較快。

**缺點：**顯而易見，需要人工標示答案。

## 非監督式學習

Unsupervised Learning

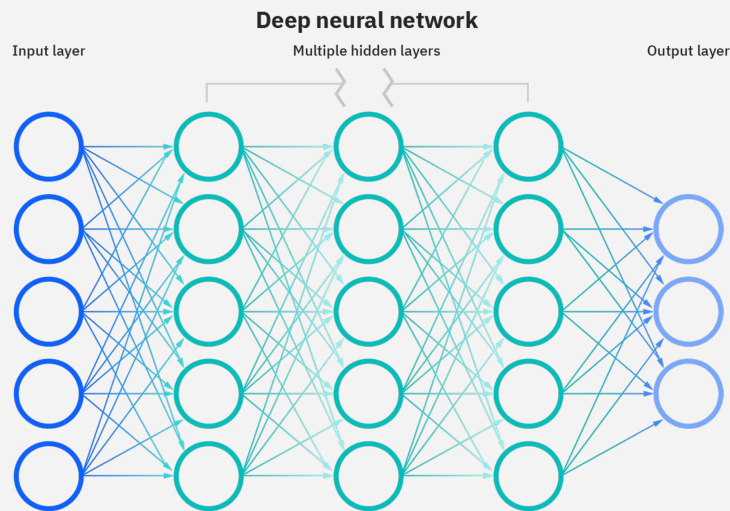
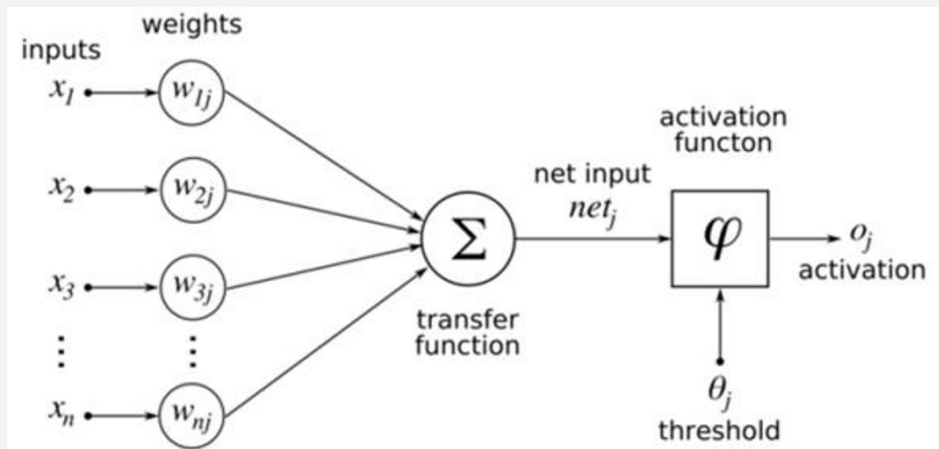
**優點：**不須人工標示答案

**缺點：**需要模型自行去聚類，有較多複雜運算，速度較慢。

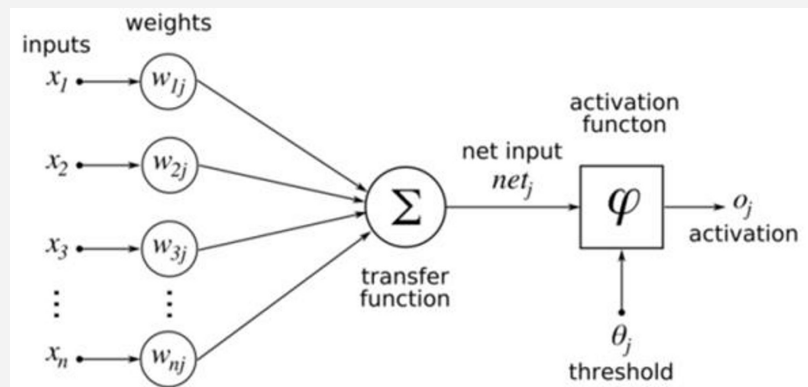
今天的重點是監督是學習，原因是因為較容易入門，也比較少複雜數學。

# 監督式學習 Supervised Learning

還記得類神經網路架構吧，每個神經元的輸出就是其**輸入的線性組合**再經過一個**活化函數**。



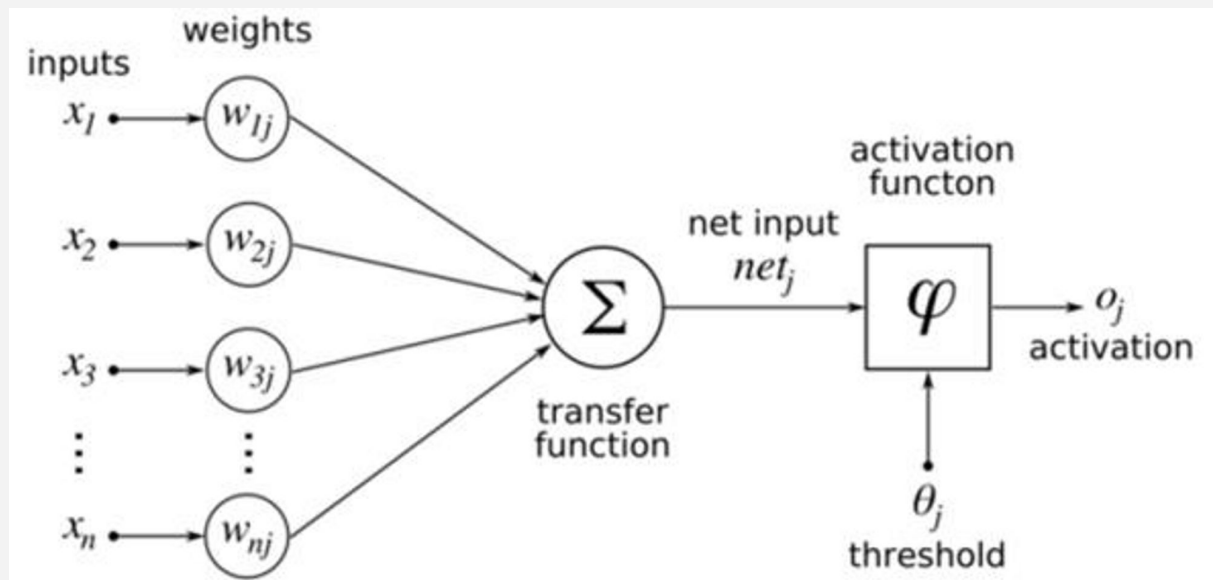
我們趕快跳過這一頁。



$$net_j = \sum w_i x_i + b$$
$$activation \mathbf{o}_j = f(net_j)$$

# 監督式學習 Supervised Learning

模型要學習的參數 (Parameters), 就是這些**權重值  $w$** 。



## 補充: Parameters vs Variables

### **Parameters(參數):**

The numbers that govern the **statistical model** or **structure of data**.

### **Variables(變數):**

Some components of the data, Is input by users.

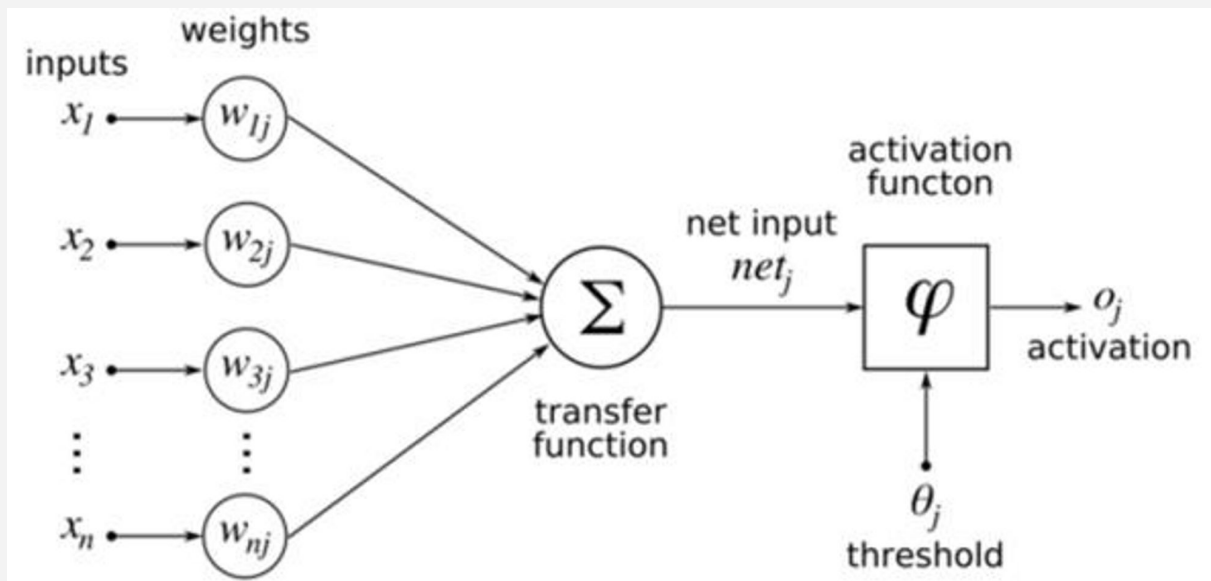
以分辨貓狗為例。





# 監督式學習 Supervised Learning

一開始，這些權重值通常是亂數，也就是說我們餵訓練資料給模型時，它一開始只是隨便猜的。



# 監督式學習 Supervised Learning

還記得上次說到機器學習的結果是一組機率嗎？一開始，機器可能會答這樣：



**模型：「它有70%的機率是狗。」**



**模型：「它有30%的機率是狗。」**

# 監督式學習 Supervised Learning

...這...不對吧？這是因為一開始模型只是隨便猜猜。所以這時我們就要糾正他。



**:「不, 這100%是貓。」**



**:「不, 這100%是狗。」**

# 監督式學習 Supervised Learning

這時，模型完全不會懷疑你，它會開始把自己的答案和你的正確答案做比較。



**模型：「70%是狗。」**



**模型：「30%是狗。」**



**:「100%是貓。」**



**:「100%是狗。」**

# 監督式學習 Supervised Learning

模型就會想辦法修正權重  $w$ ，讓最後的結果越來越接近正確答案。



模型：「70%是狗。」

**調低**



模型：「70%是貓。」

**調高**



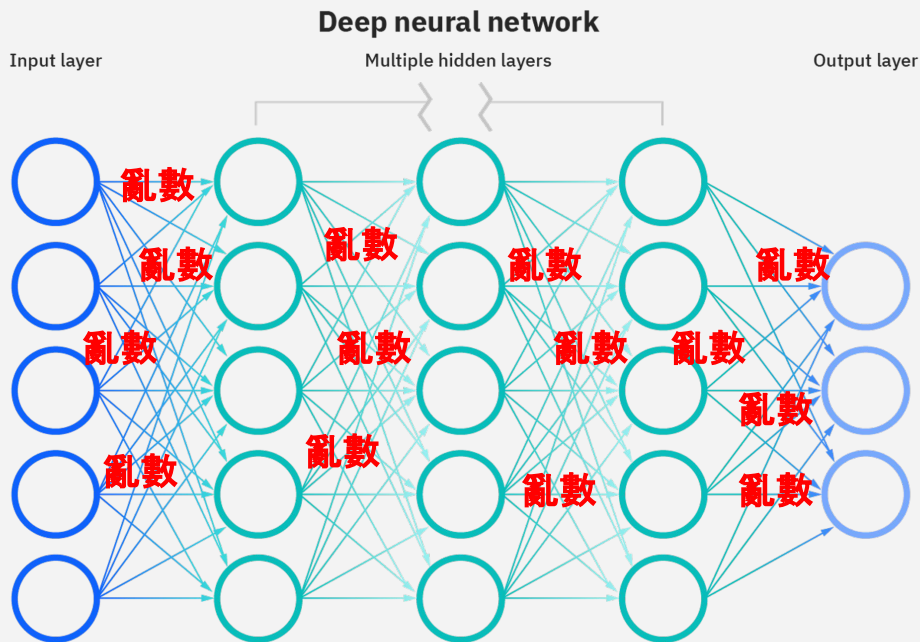
：「100%是貓。」



：「100%是狗。」

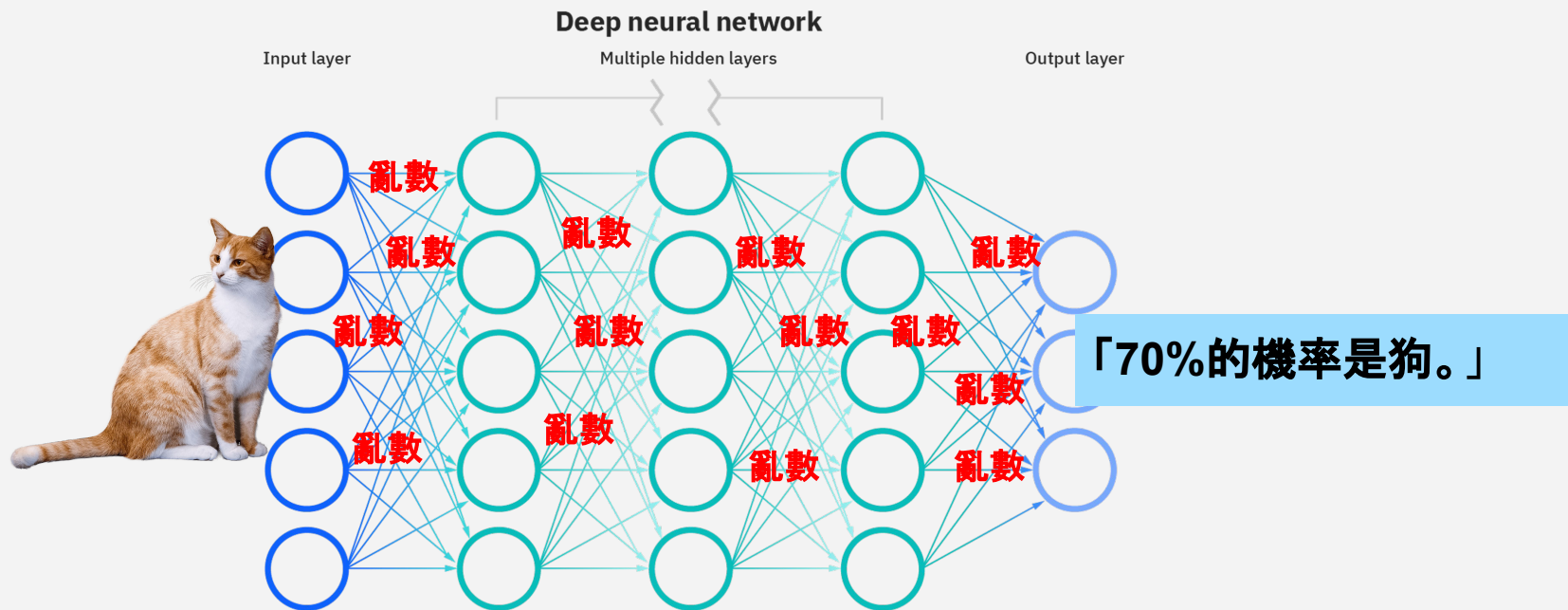
# 監督式學習 Supervised Learning

我們用神經網路的模型再想一遍。一開始，權重全部都是亂數。



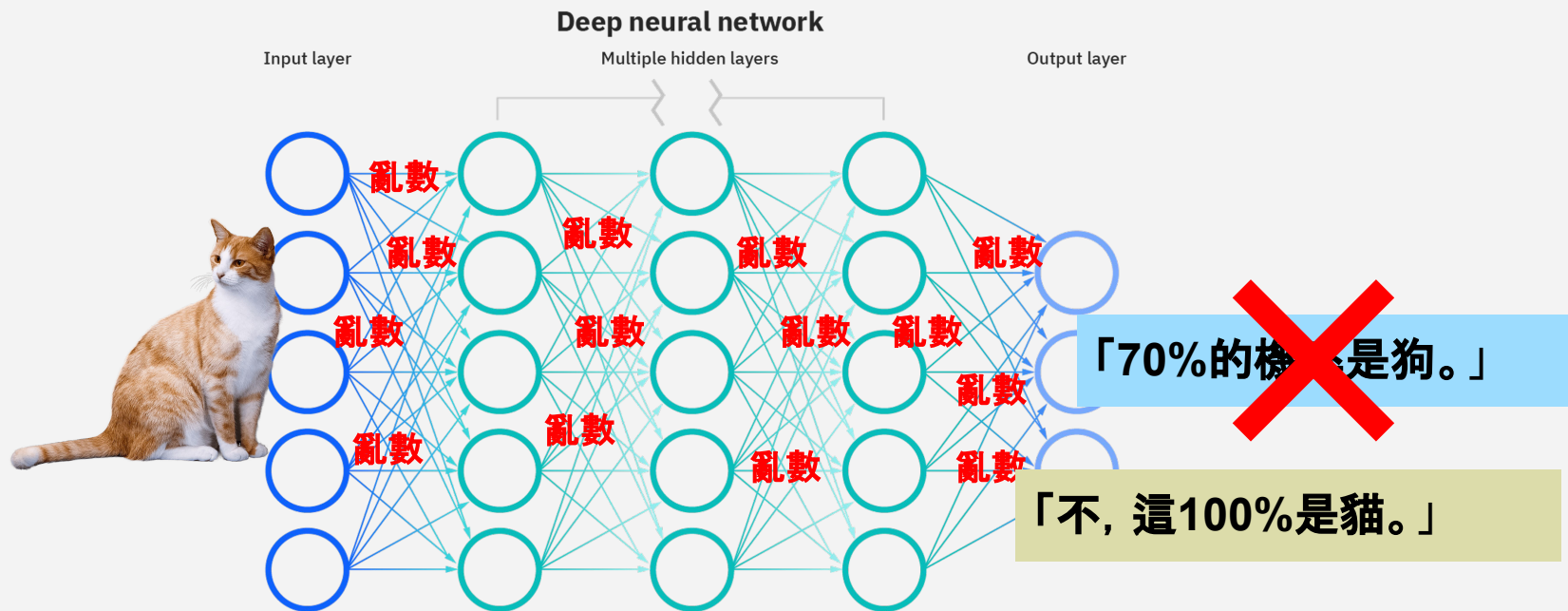
# 監督式學習 Supervised Learning

可想而知，當我們給他圖片時它只會亂猜。



# 監督式學習 Supervised Learning

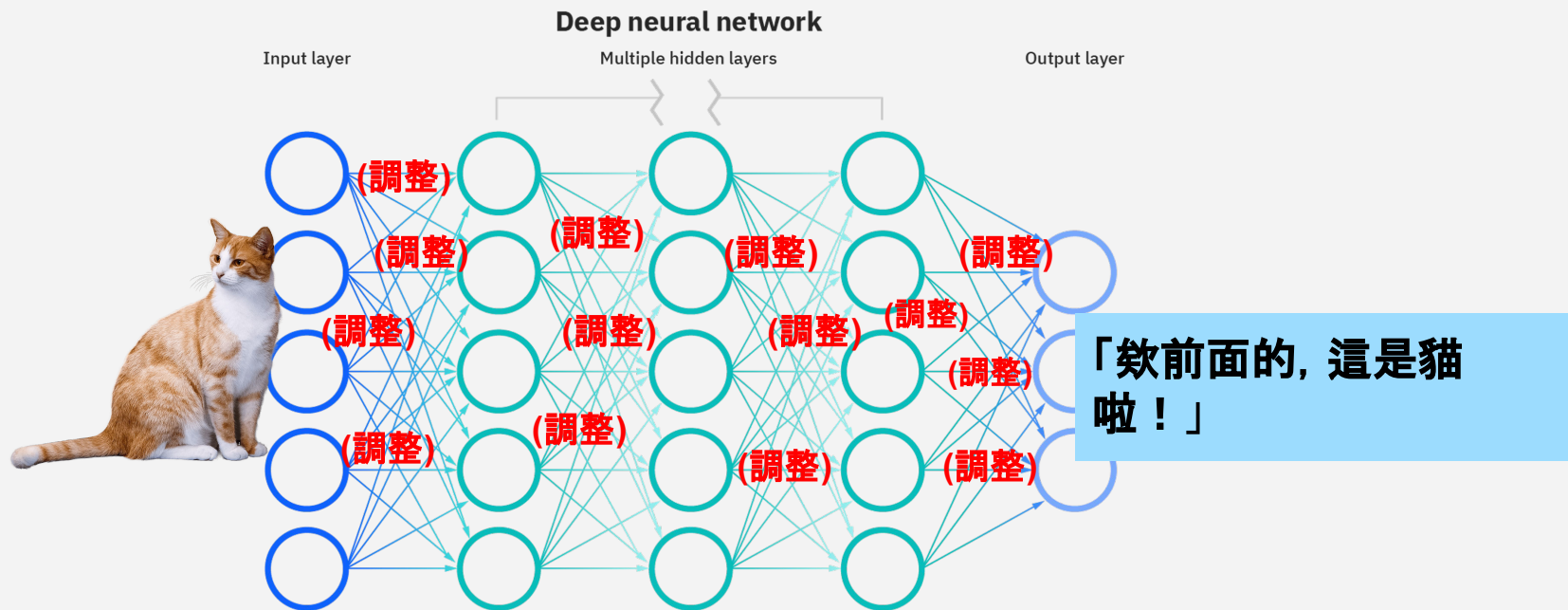
可想而知，當我們給他圖片時它只會亂猜。





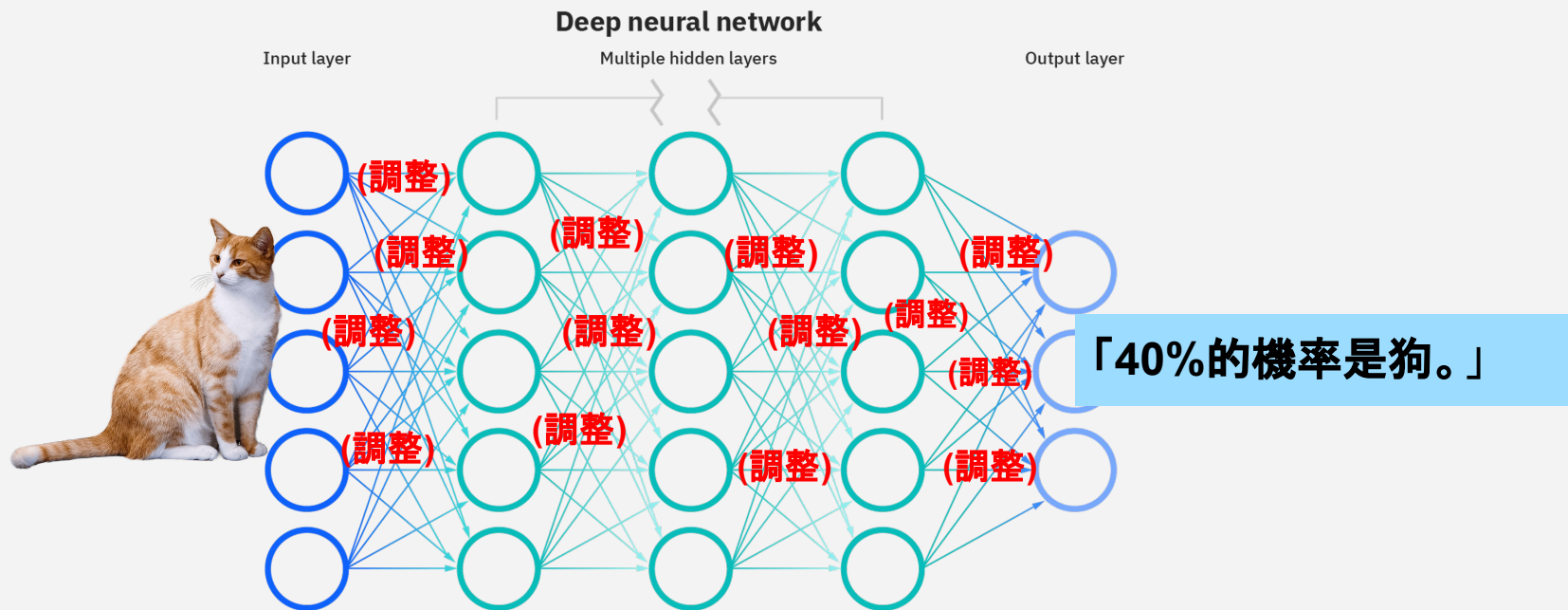
# 監督式學習 Supervised Learning

這時候，輸出層會將這個修正的訊息和程度往前傳，告訴前面的隱藏層要調整權重值。



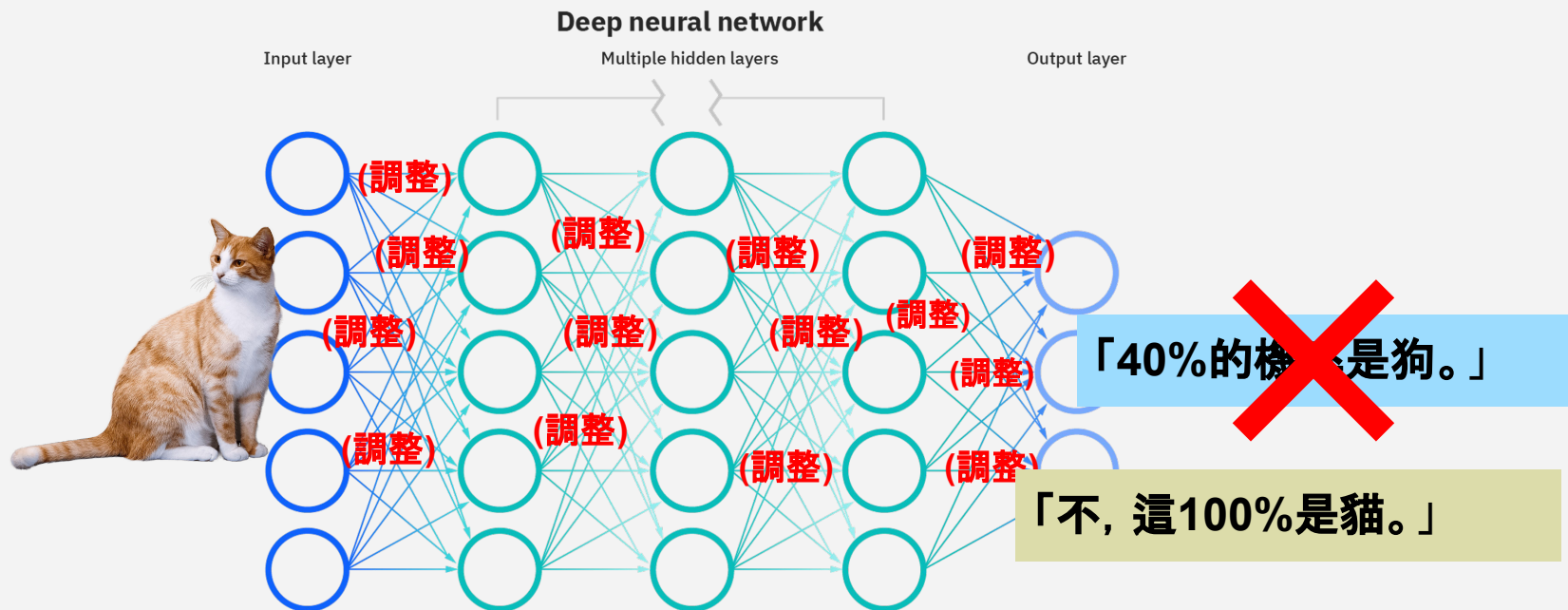
# 監督式學習 Supervised Learning

這時，再丟一張貓的照片，是狗的機率就會下降了。



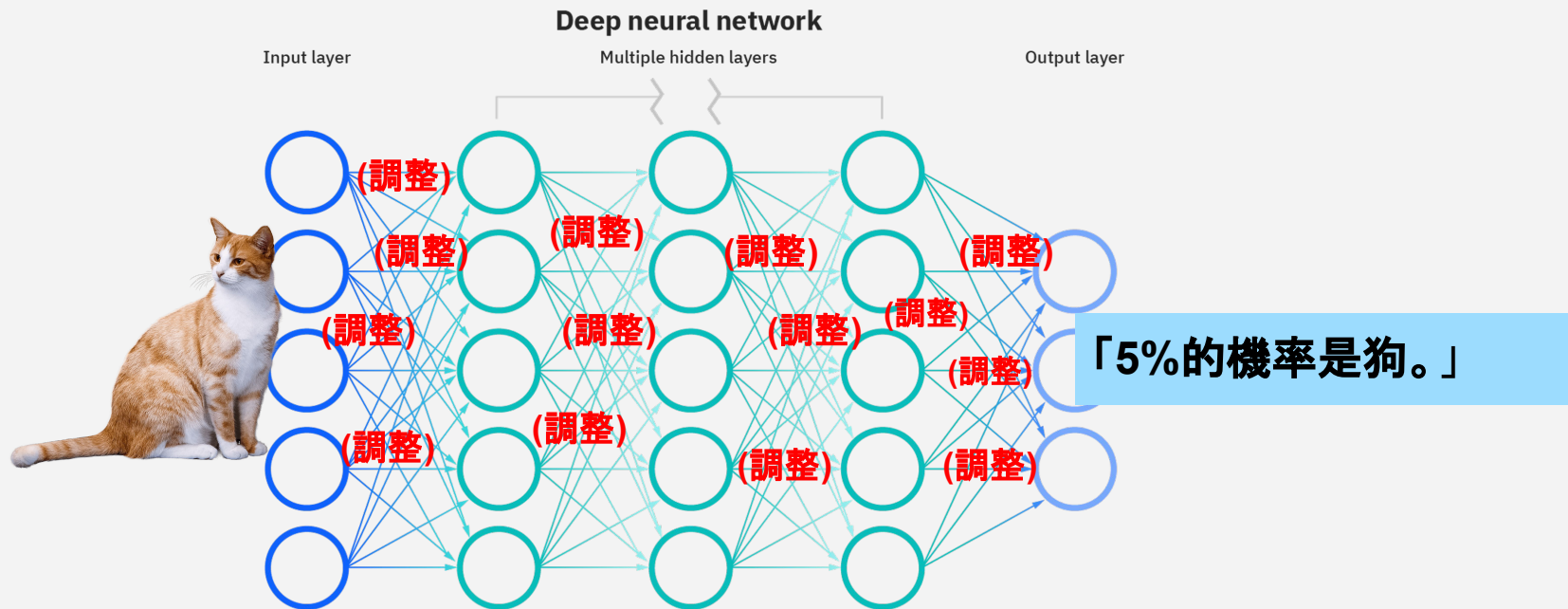
# 監督式學習 Supervised Learning

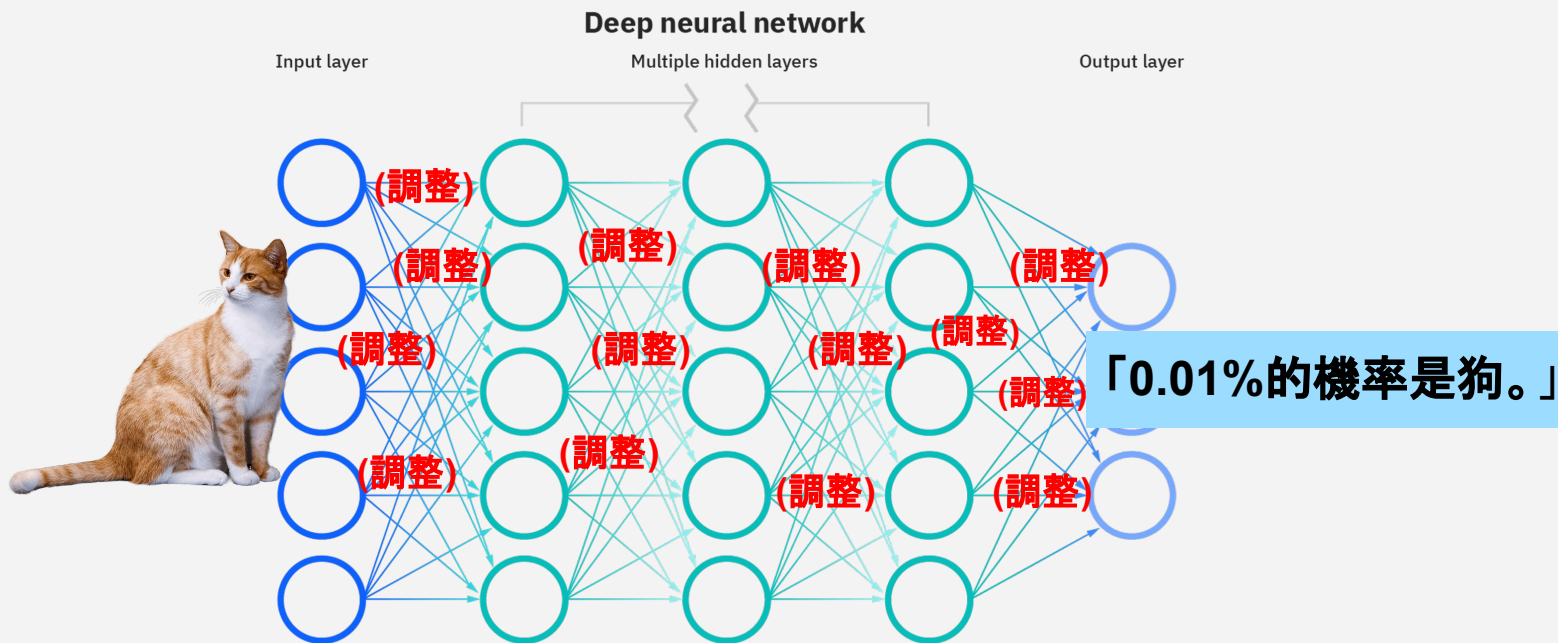
我們繼續訓練。



# 監督式學習 Supervised Learning

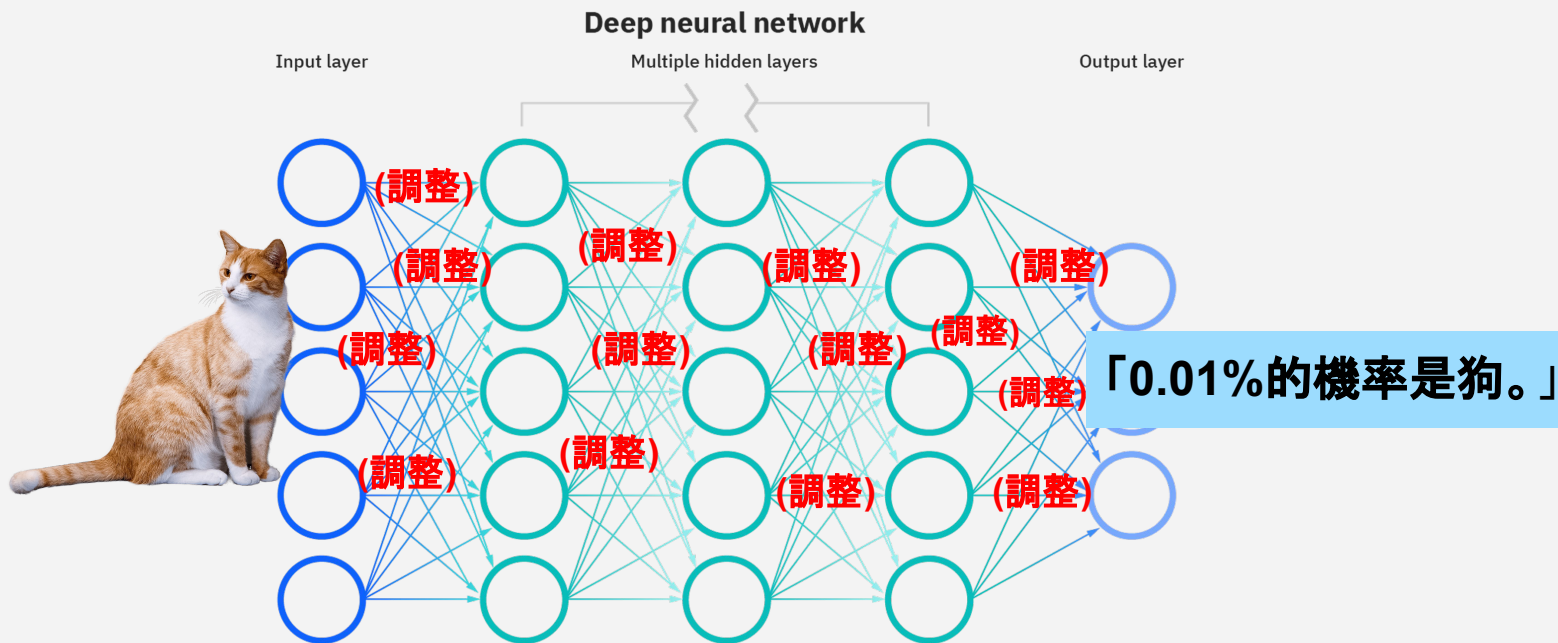
## 繼續訓練...





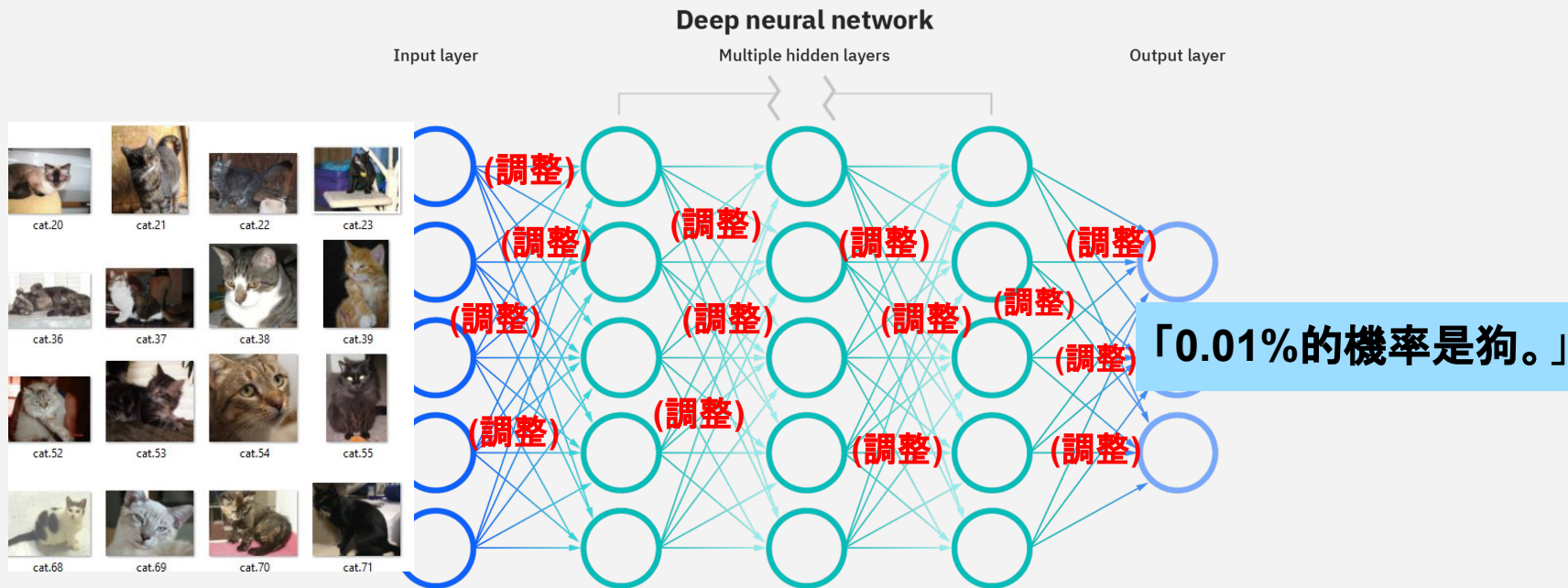
# 監督式學習 Supervised Learning

這樣就算是學會了。



# 監督式學習 Supervised Learning

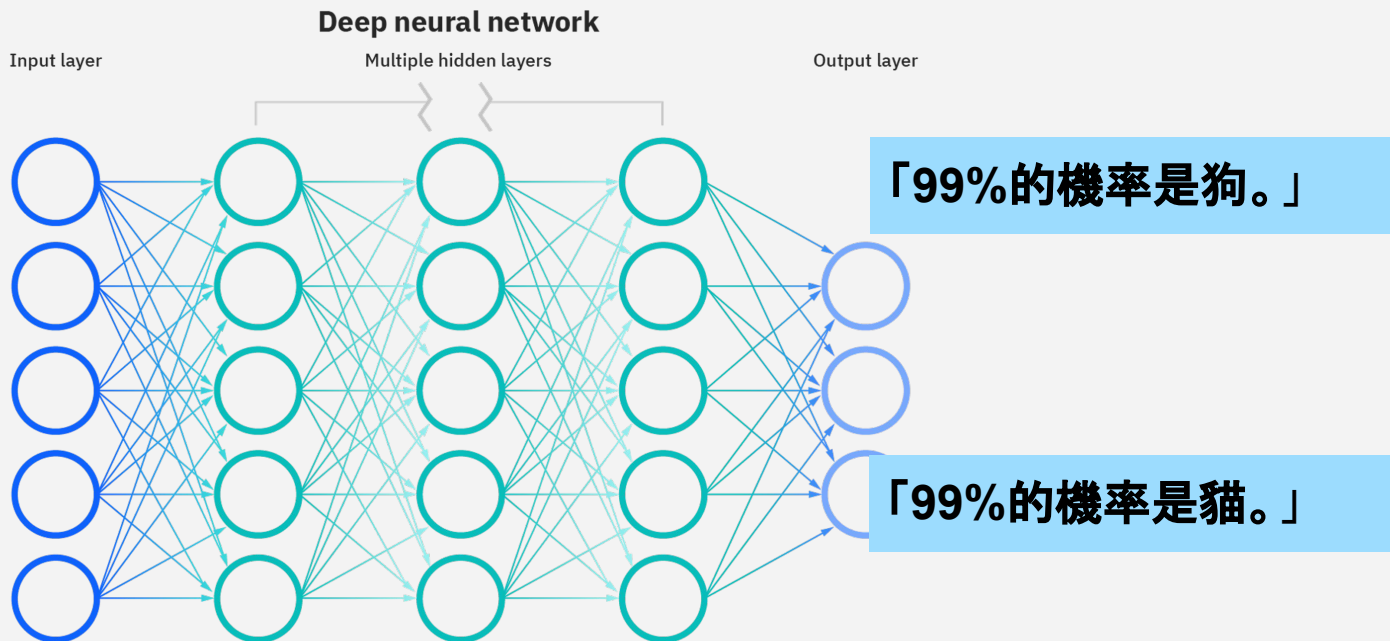
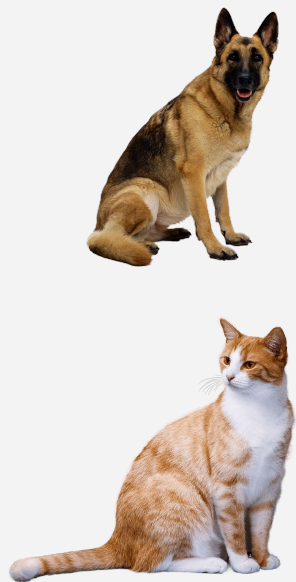
當然我們不會用同一隻貓來訓練這個模型，會讓這個模型不知變通。我們會丟給她很多不同的照片，跟他說**這些都是貓**。





# 監督式學習 Supervised Learning

當然，貓和狗的照片都要同時訓練才可以！





類神經的訓練就是模型**最佳化**的過程  
(Optimization)

# Google Teachable Machine

Teachable Machine is a web-based tool that makes creating machine learning models fast, easy, and accessible to everyone.


<https://teachablemachine.withgoogle.com/>





# Google Teachable Machine

我們今天的目標是訓練模型看的出來貓跟狗的照片。點選最左邊的 Image Project

## New Project

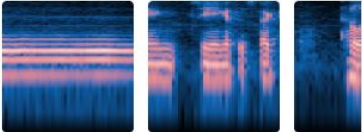
 Open an existing project from Drive.

 Open an existing project from a file.




### Image Project

Teach based on images, from files or your webcam.



### Audio Project

Teach based on one-second-long sounds, from files or your microphone.

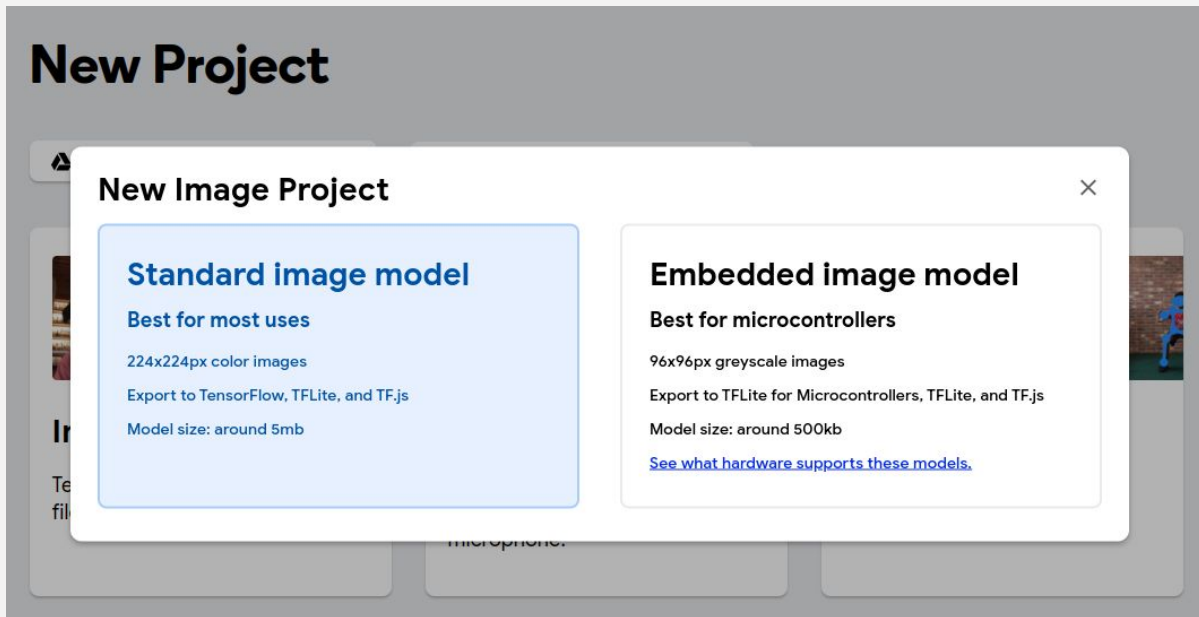


### Pose Project

Teach based on images, from files or your webcam.

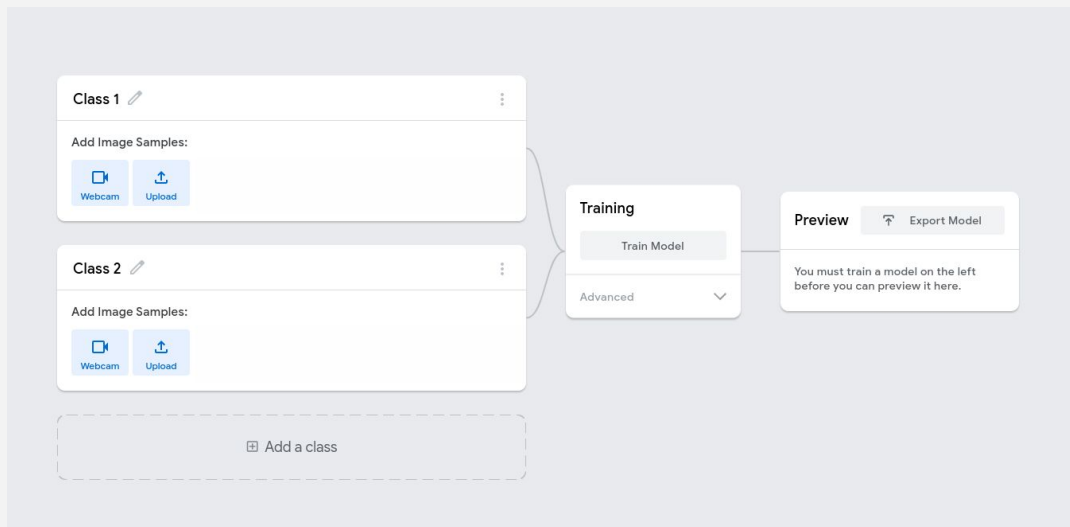
# Google Teachable Machine

我們點選左邊 Standard image model



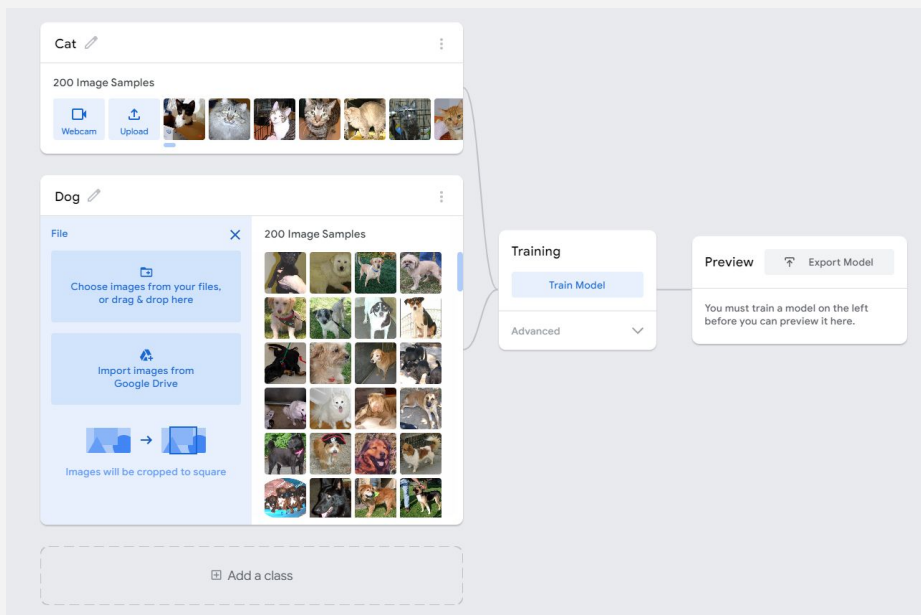
# Google Teachable Machine

他的邏輯就是，我們先給他訓練資料集，Google會遠端幫我們訓練這個模型，再透過簡單調整一些參數，這個模型就算是訓練完成了。(一個 code 都不用打很讚吧)



# Google Teachable Machine

我們就馬上來上傳訓練資料，他預設已經有兩個 Class 在上面了，我們就一個全部放貓的照片，一個全部放狗的照片。



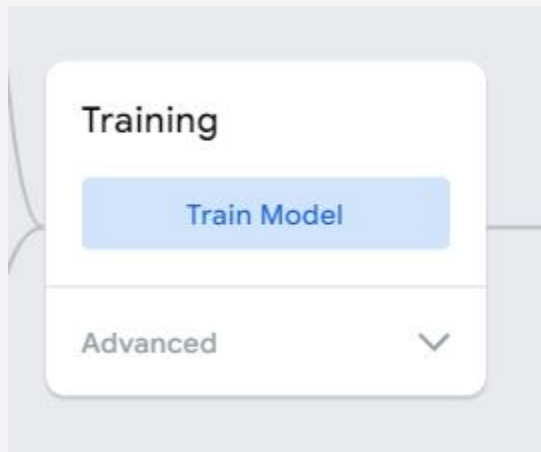
順帶一提，今天訓練資料是由 Kaggle 所提供：

<https://www.kaggle.com/c/dogs-vs-cats>

Kaggle 是一個機器學習的社群平台。上面提供很多使用者提供的資料庫和程式碼，大家有興趣可以去逛逛！

# Google Teachable Machine

馬上按 Train Model!



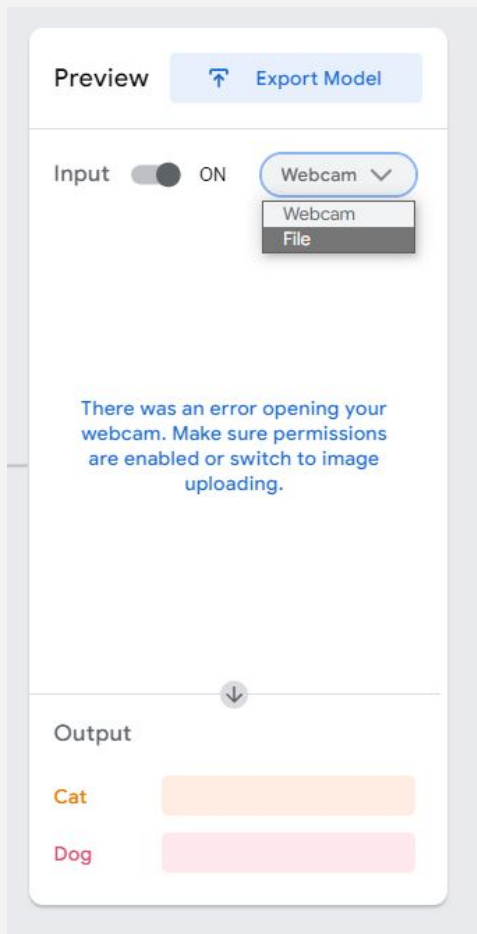


# Google Teachable Machine

給他一段時間，之後你的模型就完成訓練了！

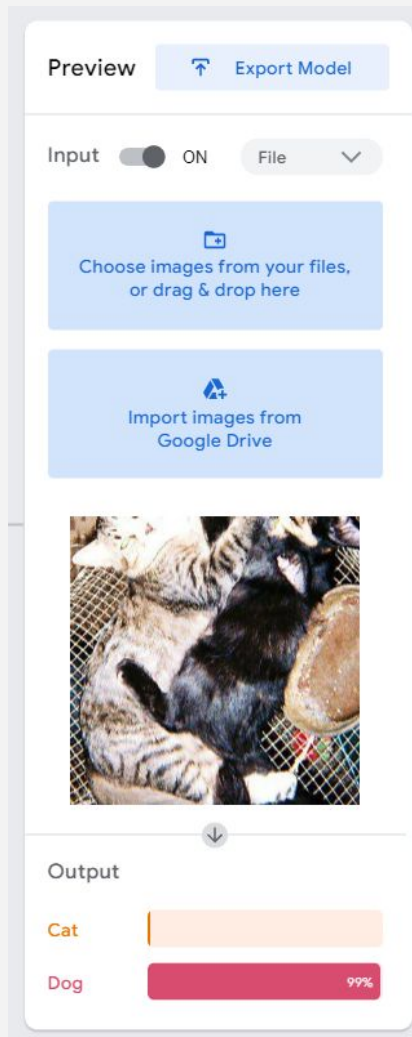
趕快來試看看他的成果如何，點選 File 並上傳檔案  
(當然也可以用你的視訊鏡頭!)

你可以上傳 Test 資料夾內的圖片，這些圖片沒有被我  
納入訓練集當中。



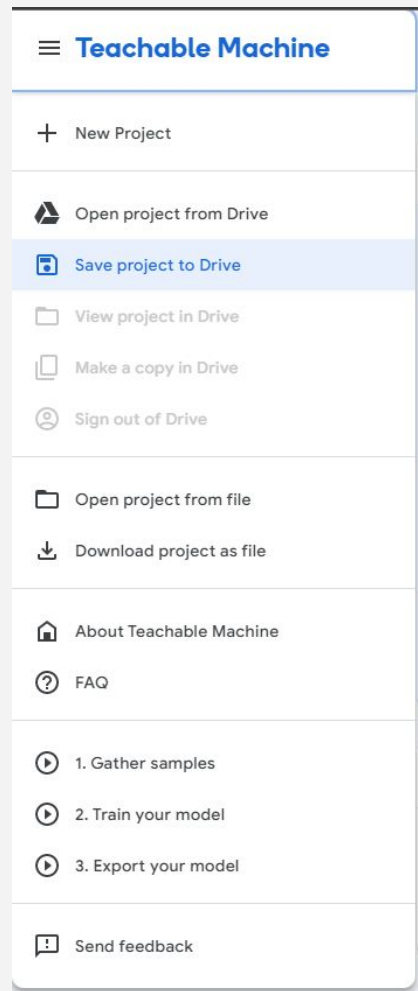
# Google Teachable Machine

當然還是會有失誤的時候。但我覺得已經很厲害了。



# Google Teachable Machine

可以儲存專案到自己的雲端硬碟。

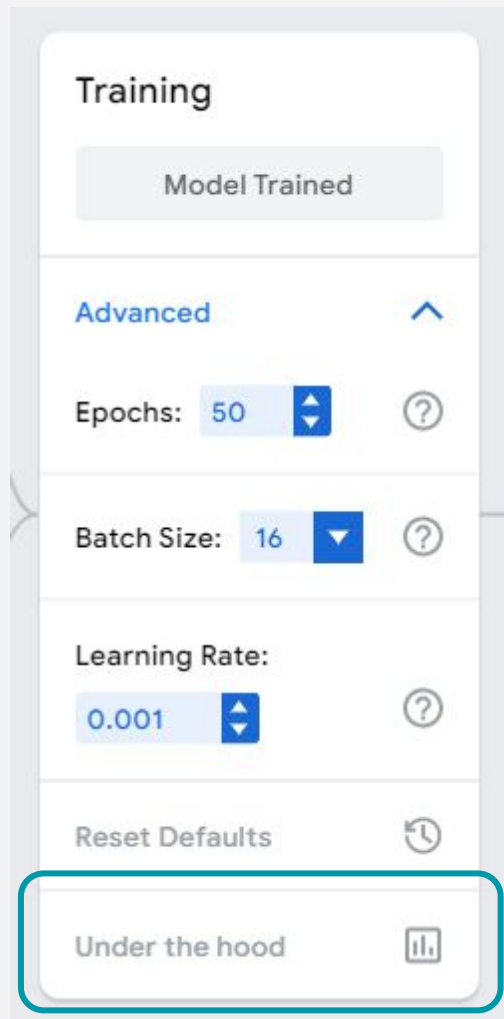


# 結果分析 1

在深度學習的領域中，結果分析非常非常重要

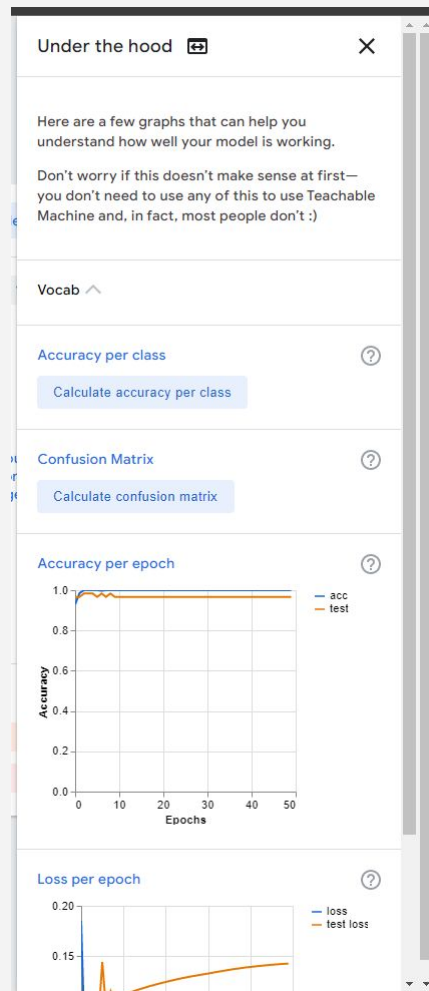
# 結果分析 1

到 Training 底下，點開 Advanced 進階設定，然後點最下面的 Under the hood



# 結果分析 1

你會看到模型訓練的細節。



# 結果分析 1

點這兩個按鈕，我們可以看到模型的準確度 (Accuracy)

Calculate accuracy per class

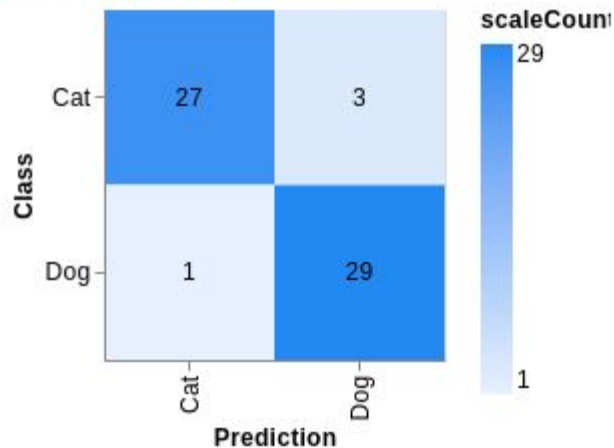
Calculate confusion matrix

Accuracy per class



CLASS	ACCURACY	# SAMPLES
Cat	0.90	30
Dog	0.97	30

Confusion Matrix

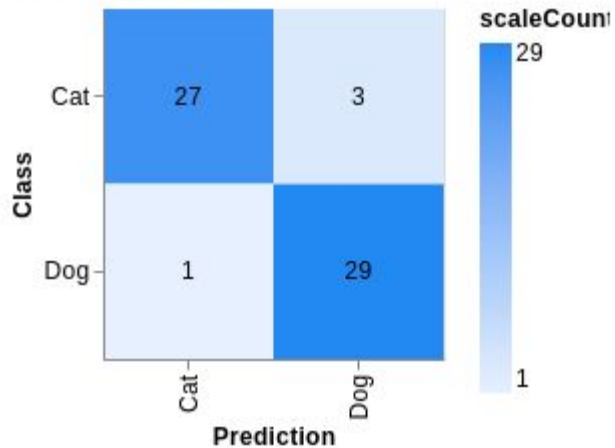


### Accuracy per class



CLASS	ACCURACY	# SAMPLES
Cat	0.90	30
Dog	0.97	30

### Confusion Matrix



圖片是貓，預測結果也是貓的準確率

圖片是狗，預測結果也是狗的準確率

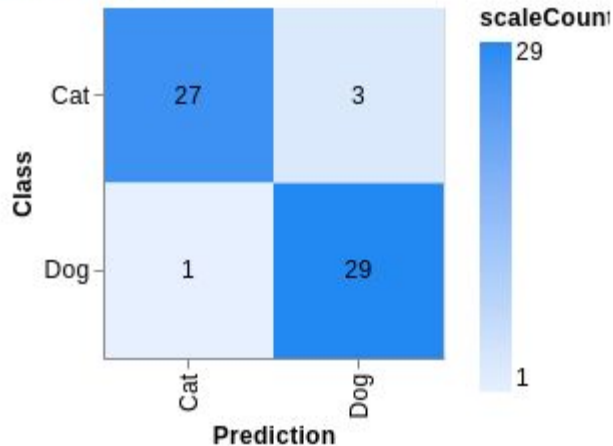


### Accuracy per class



CLASS	ACCURACY	# SAMPLES
Cat	0.90	30
Dog	0.97	30

### Confusion Matrix



圖片是貓, 預測結果也是貓的準確率

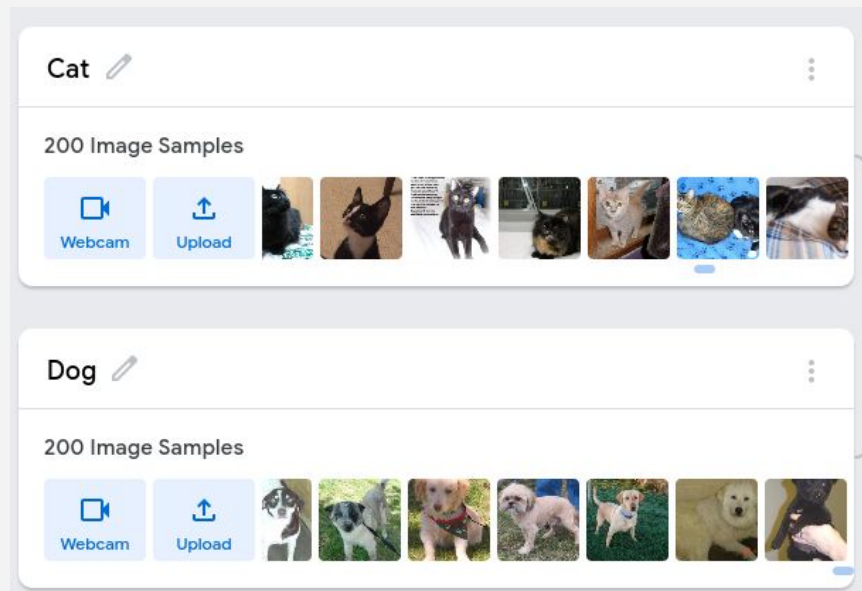
$$P(\text{預測貓}|\text{圖片是貓}) \\ = 27 \div (27 + 3)$$

圖片是狗, 預測結果也是狗的準確率

$$P(\text{預測狗}|\text{圖片是狗}) \\ = 29 \div (29 + 1)$$

# 結果分析 1

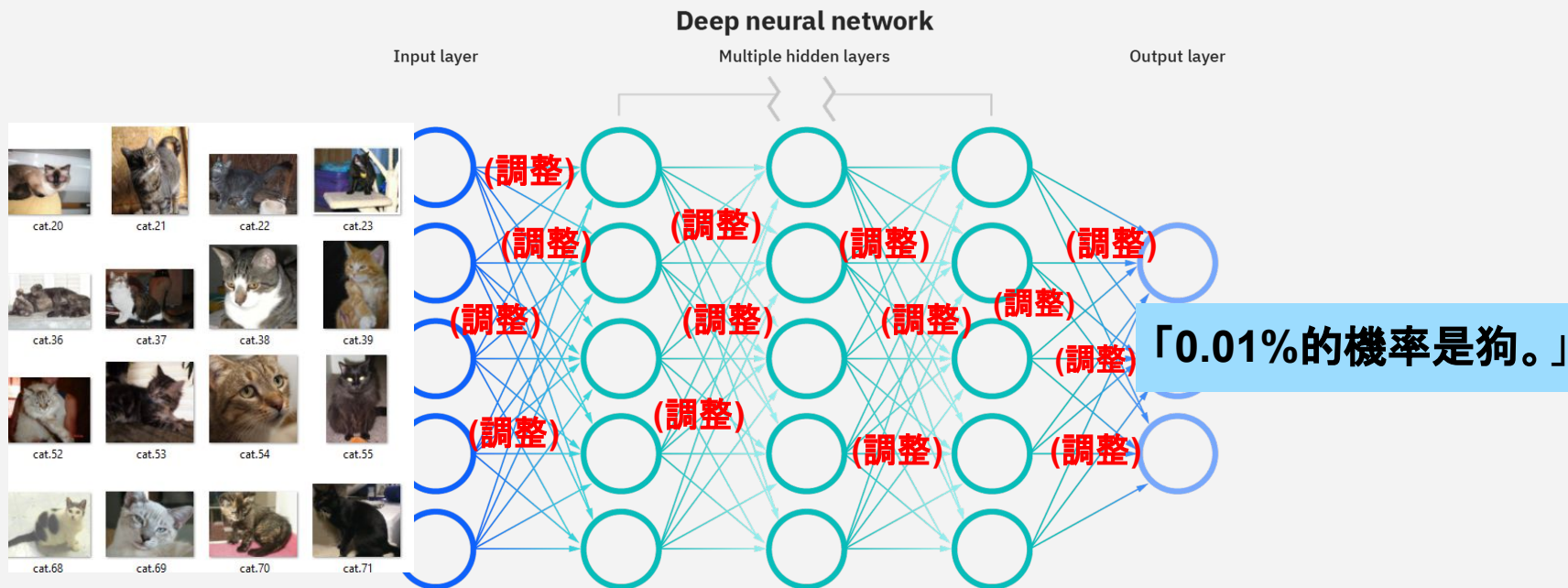
回去看資料就會發現，其實貓的照片比較多樣化，狗的照片相對一致，這就是為什麼判斷出貓的難度比較高。(沒辦法，貓咪就是那麼奇怪的生物)



休息一下！

# 監督式學習 Supervised Learning

我們來更進一步探討模型的最佳化



# 監督式學習 Supervised Learning

我們用數學的想法再再想一遍。下面的方程式就是監督是學習簡單的實現方法，雖然說是很簡單，但根本沒有一個符號是看得懂的。

$$W \leftarrow W - \eta \frac{\partial L}{\partial W}$$

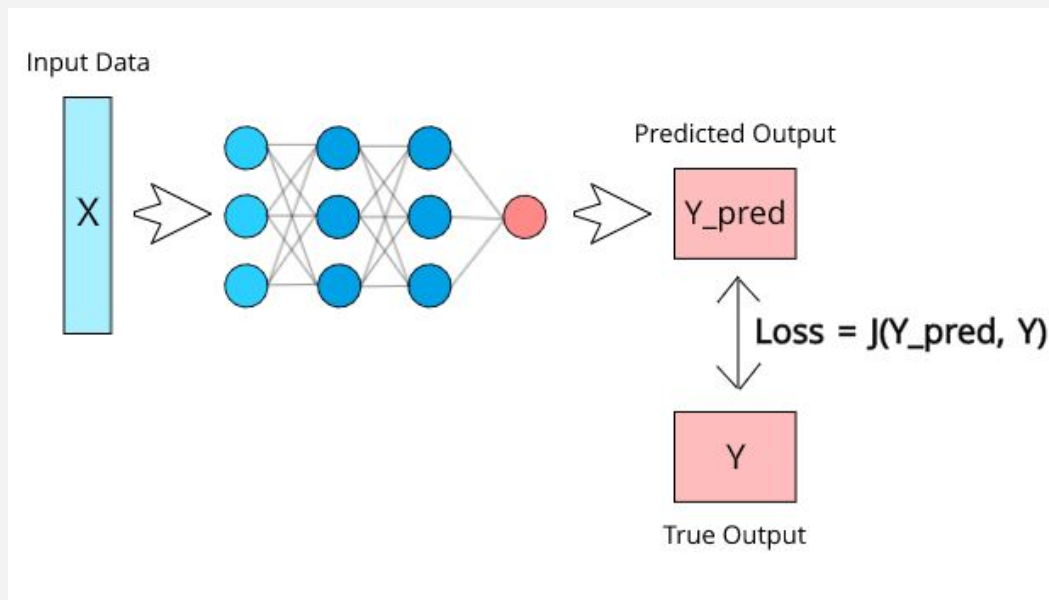
是真的很簡單啦！我們慢慢來。

一開始，權重全部都是亂數。

因為有很多權重值，所以我們用大寫 $W$ 表示**權重值矩陣**

亂數  
亂數  $W$  亂數  
亂數

另外，假設模型預測出的答案是  $\hat{y}$ ，  
正確答案是  $y$



# 監督式學習 Supervised Learning

因此，一開始給他照片時，模型只會隨便猜。



亂數  
亂數  $W$  亂數  
亂數

「70%的機率是狗。」

$\hat{y}$



# 監督式學習 Supervised Learning

我們要**量化**他的誤差，這時候就要用到一個叫作損失函數的...函數。



亂數  
亂數  $W$  亂數  
亂數

「70%的機是狗。」

$\hat{y}$

「不，這100%是貓。」

$y$

# 損失函數及梯度下降法

損失函數 (Loss function) 輸出的結果，想像成是和正確結果的誤差程度就好了。

常見的目標函數包括均方根誤差(Mean square error, MSE, 常用在迴歸分析)、交叉熵(Cross-entropy, 常用在分類問題) 等等。

**均方根誤差：**

其實就是相減的  
平方的平均。

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

**交叉熵：**

其實就是計算亂  
度的概念。

$$H(P^* | P) = - \sum_i \underbrace{P^*(i)}_{\substack{\text{TRUE CLASS} \\ \text{DISTRIBUTION}}} \log \underbrace{P(i)}_{\substack{\text{PREDICTED CLASS} \\ \text{DISTRIBUTION}}}$$

因為等等會一直用到損失函數，怕大家忘記他的功能，所以我會把他註記在投影片的這裡



**損失函數：和正確結果的誤差程度**

# 監督式學習 Supervised Learning

計算出誤差後，我們還要知道要**調動**的方向。



(調整)  
(調整)  $W$  (調整)  
(調整)

「70%的機率是狗。」

**調低？調高？**

損失函數：和正確結果的誤差程度

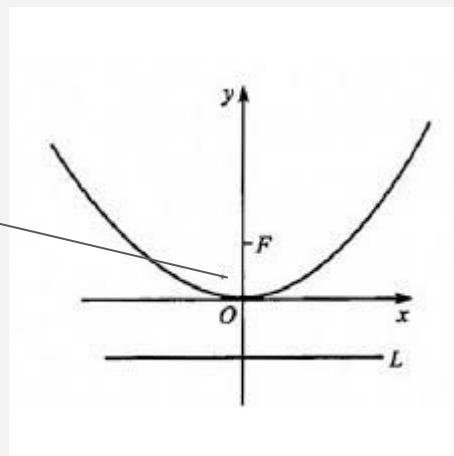
# 損失函數及梯度下降法

令  $L$  是損失函數。

$$\text{Loss function } L(\theta)$$

損失函數會隨著  $W$  的變動而變動，所以我們可以很有自信的說，損失函數是  $W$  的函數。我們都知道，透過**微分**可以找到一個函數的**斜率**，讓這個斜率等於0就可以求極值了！

極值在這，超簡單！



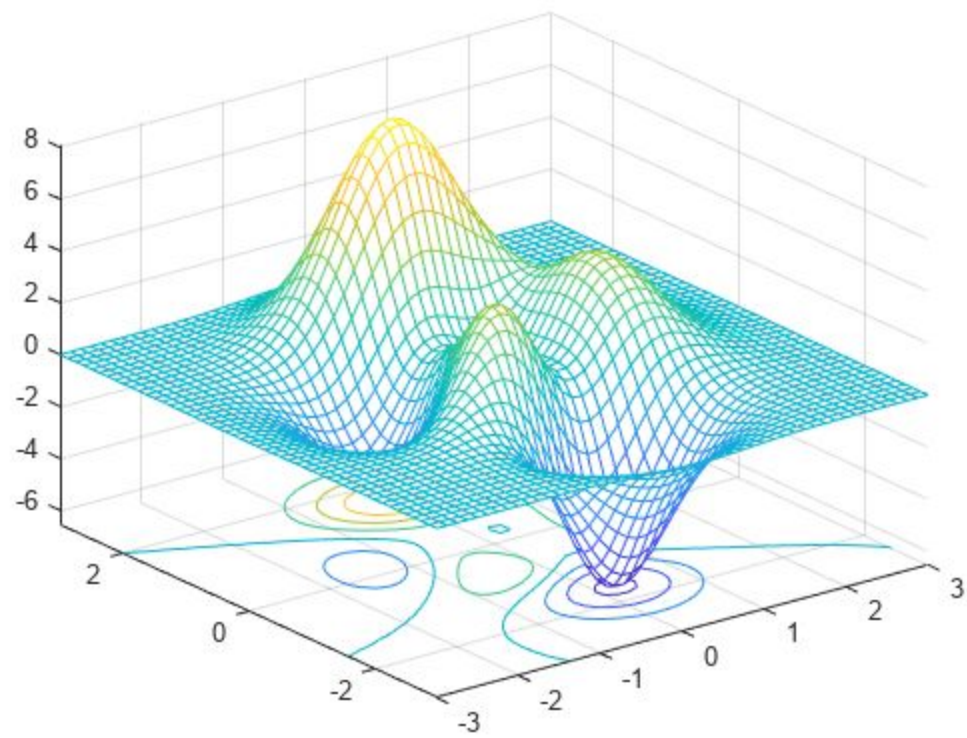
損失函數：和正確結果的誤差程度

# 損失函數及**梯度下降法**

但是...問題又來了，我們有一大堆 $W$ ，是要怎麼微分？

如果有很多地方的斜率是0，哪一個才是最小值？

損失函數：和正確結果的誤差程度



面對這種超高維度的狀況下，我們只能**走一步算一步**

這就是**梯度下降法**



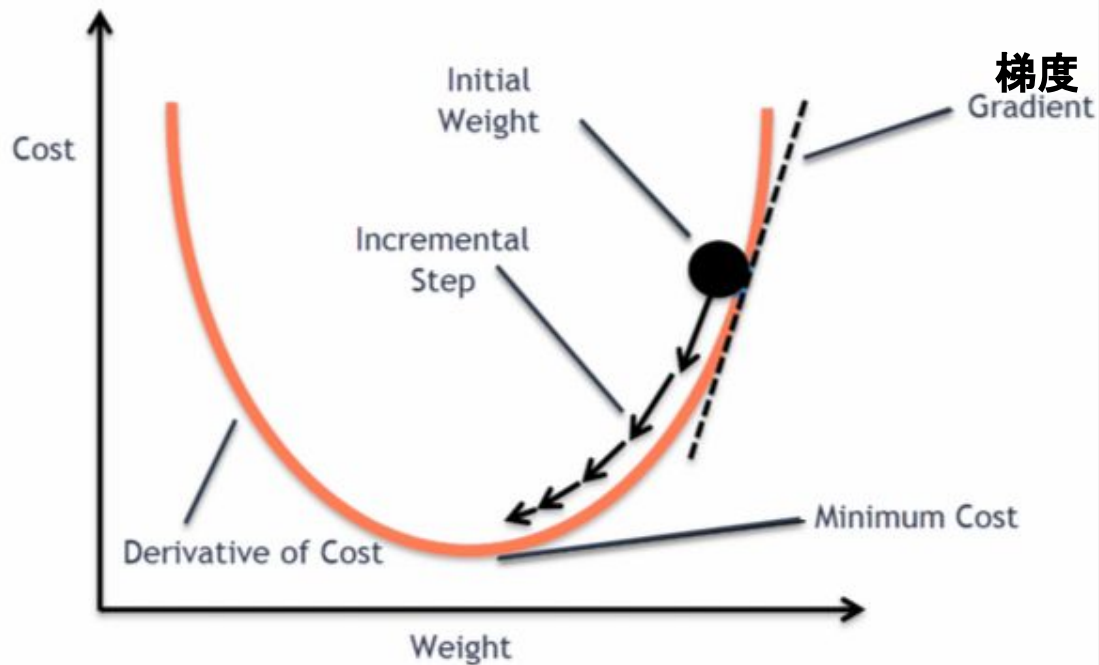
# 損失函數及梯度下降法

假設今天只有兩個權重值，這兩個權重值勢必會造成不一樣的 Loss 。就如右圖。

此時，可以說  $L$  是  $w_0, w_1$  的函數。

我們的最終目標是讓誤差最小，也就是讓 **Loss** 走到最低點。

# 損失函數及梯度下降法

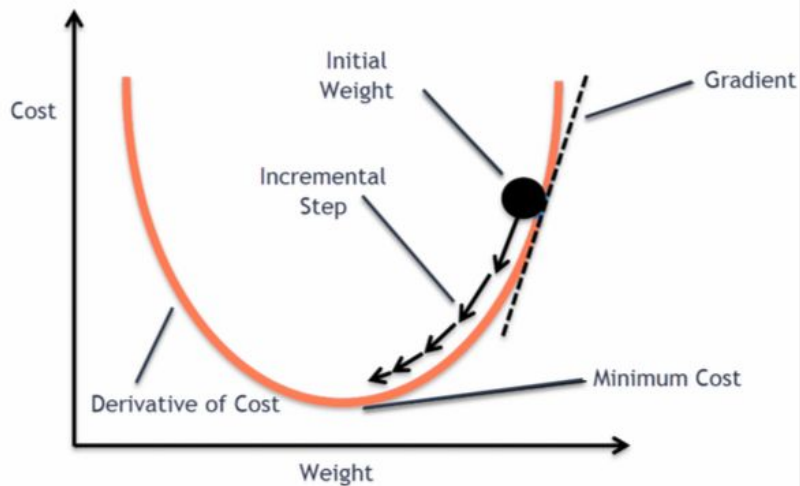


損失函數：和正確結果的誤差程度

# 損失函數及梯度下降法

梯度正->過頭了

梯度負->還沒到

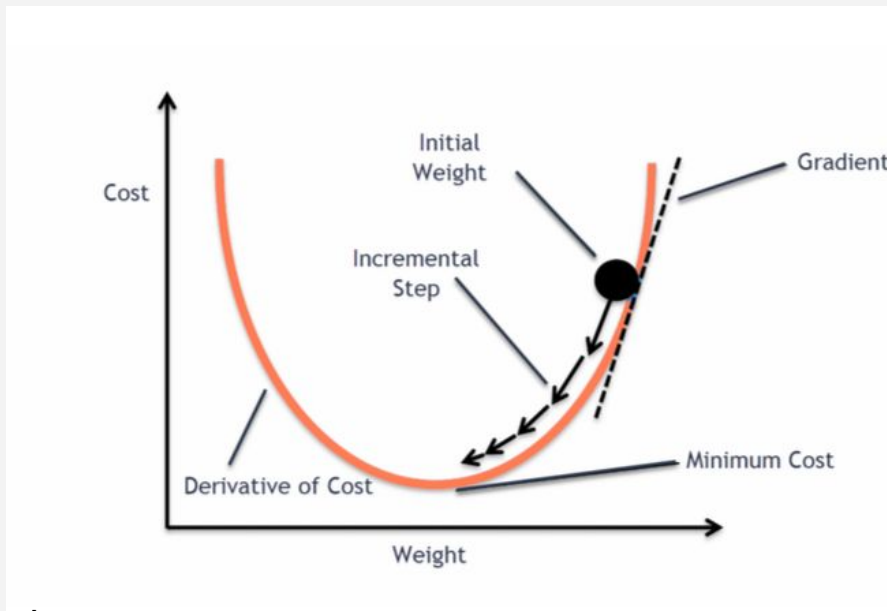


損失函數: 和正確結果的誤差程度

# 損失函數及梯度下降法

梯度正->過頭了->要往左移

梯度負->還沒到

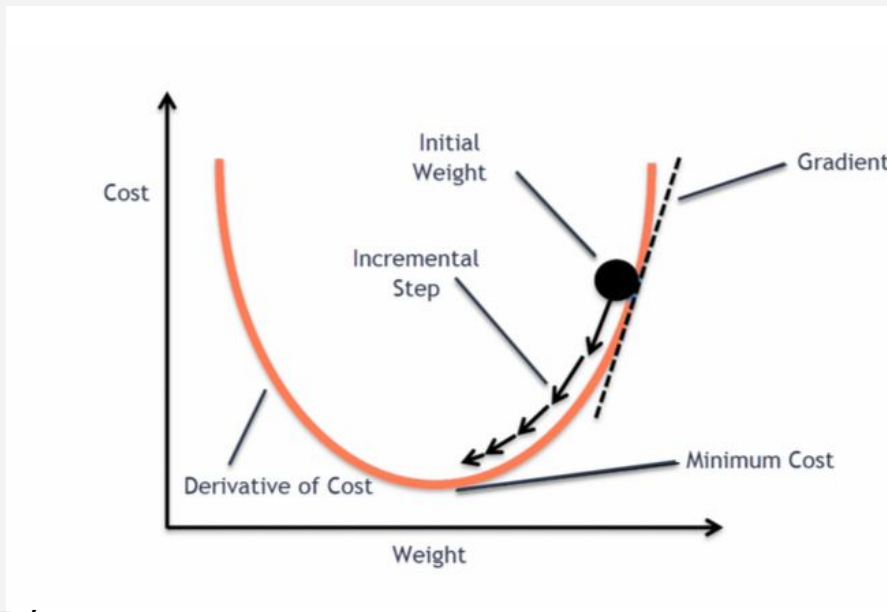


損失函數: 和正確結果的誤差程度

# 損失函數及梯度下降法

梯度正->過頭了->要往左移->權重**減少**

梯度負->還沒到

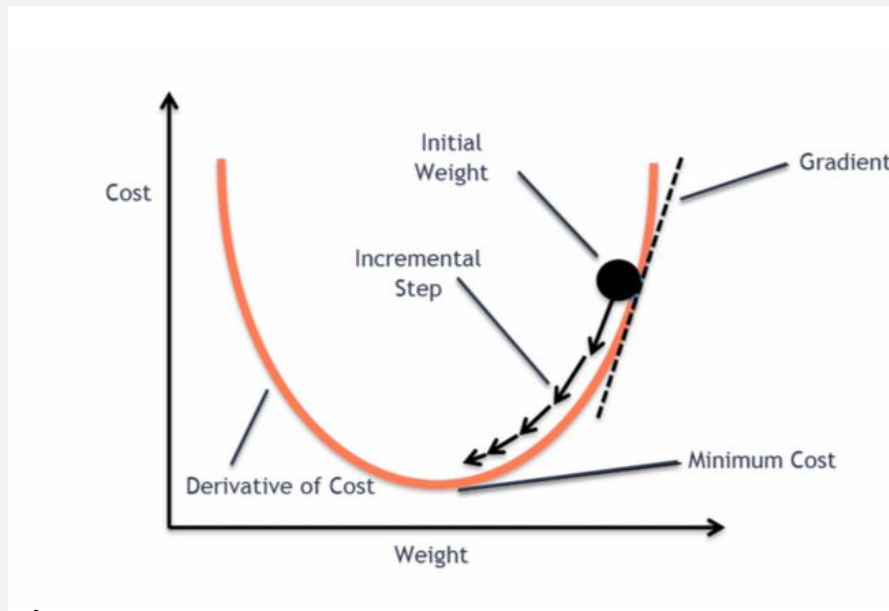


損失函數: 和正確結果的誤差程度

# 損失函數及梯度下降法

梯度正->過頭了->要往左移->權重**減少**

梯度負->還沒到->要往右移

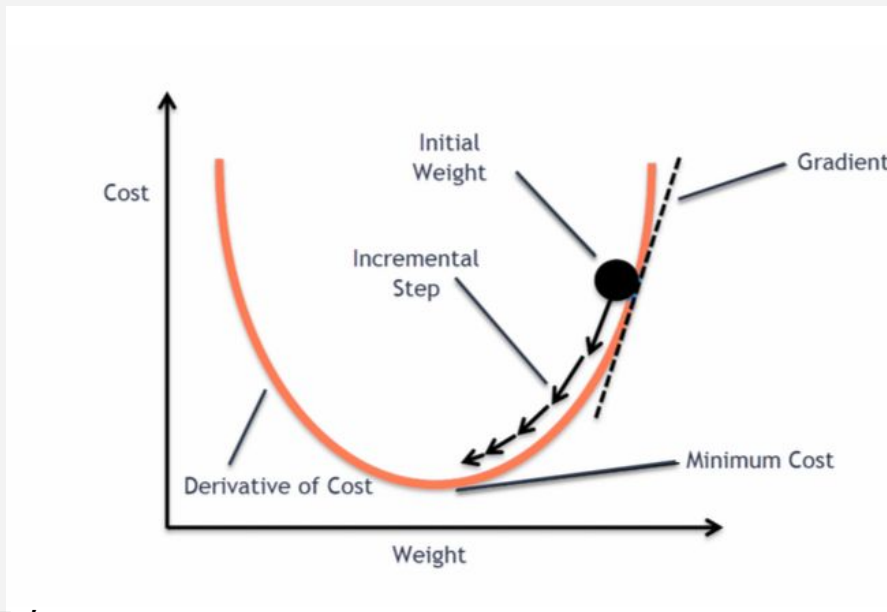


損失函數: 和正確結果的誤差程度

# 損失函數及梯度下降法

梯度正->過頭了->要往左移->權重**減少**

梯度負->還沒到->要往右移->權重**增加**



損失函數: 和正確結果的誤差程度

# 損失函數及梯度下降法

梯度**正**->權重**減少**

梯度**負**->權重**增加**

損失函數: 和正確結果的誤差程度



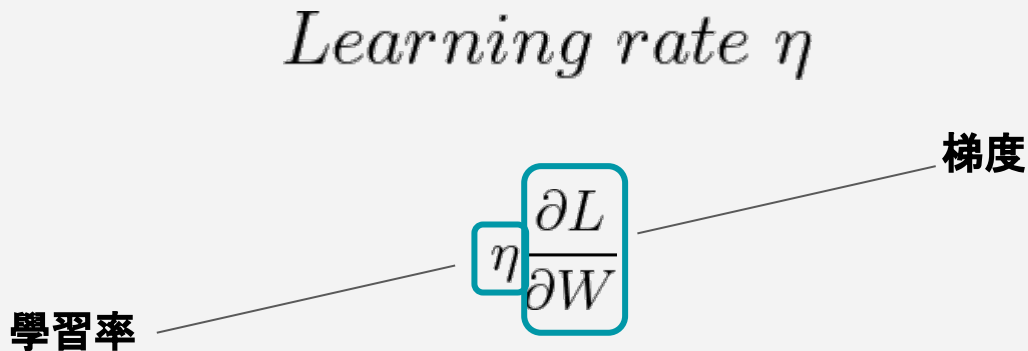
# 損失函數及梯度下降法

現在，我們有了**誤差程度**(來自損失函數)和**修正方向**(來自梯度)，此時機器已經可以學習了。

損失函數：和正確結果的誤差程度

# 學習率 Learning Rate

但是，我們還想要可以人為控制的**修正幅度**。所以我們多乘上一個**係數**來控制他，這是我們學到的第一個人為調整的**超參數(Hyper-parameter)**。



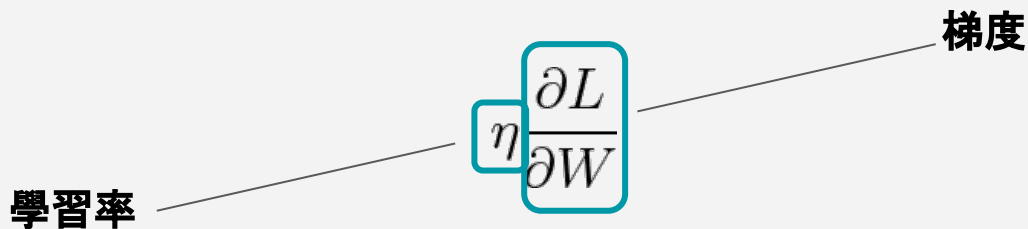
損失函數：和正確結果的誤差程度

# 學習率 Learning Rate

我們將剛才所求得的梯度，再乘上一個學習率 (Learning rate) 目的是防止梯度太大造成難以收斂。

學習率可以想像成模型的敏銳度，太敏銳的話會讓模型有被害妄想症而無法收斂；太遲鈍會讓模型學得很慢很笨。

*Learning rate  $\eta$*

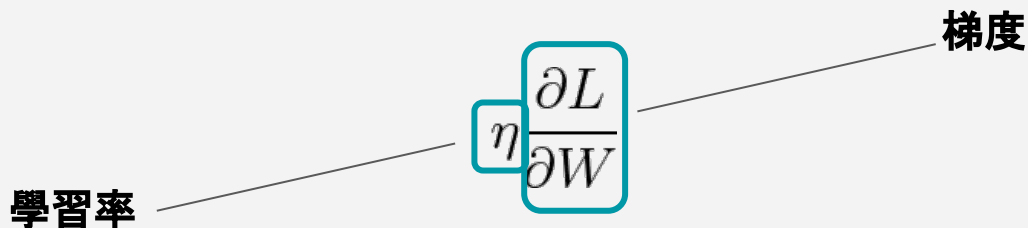


損失函數：和正確結果的誤差程度

# 學習率 Learning Rate

學習率大概是機器學習中最重要的一個超參數了。

*Learning rate  $\eta$*



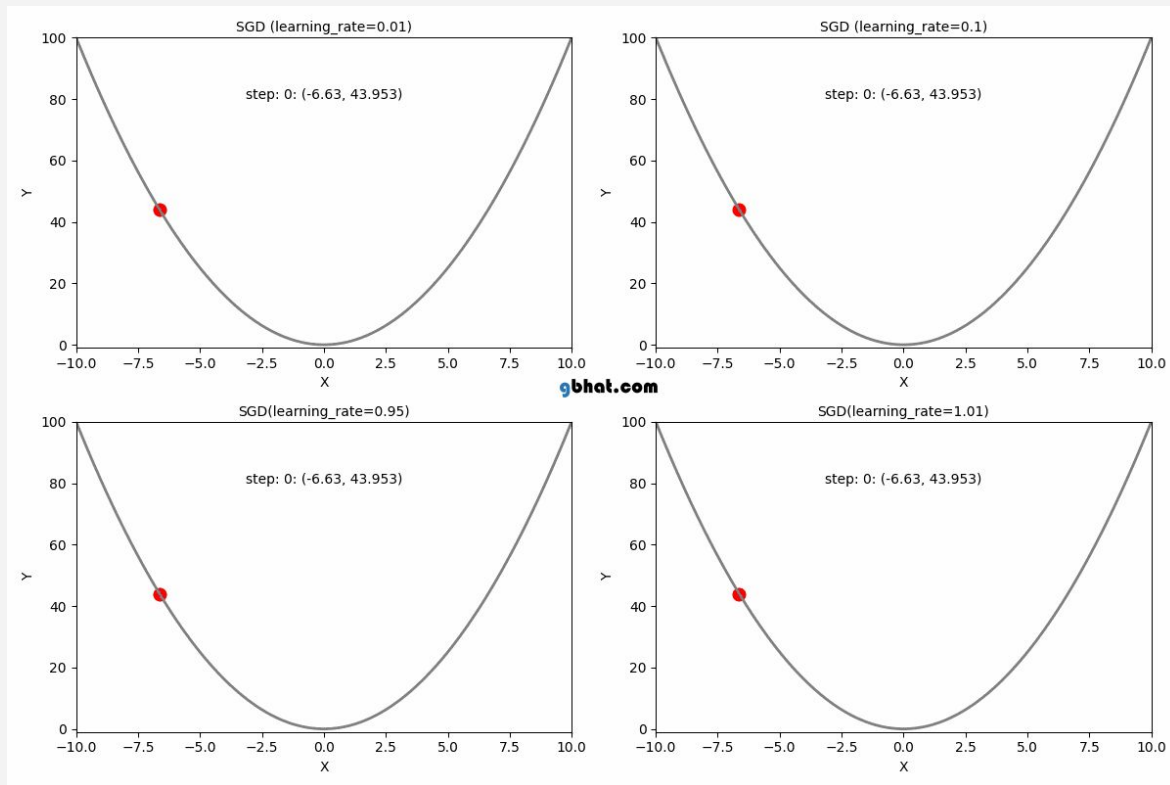
The diagram illustrates the components of the learning rate formula. It features a central expression  $\eta \frac{\partial L}{\partial W}$  where the learning rate  $\eta$  is enclosed in a small blue box and the gradient term  $\frac{\partial L}{\partial W}$  is enclosed in a larger blue box. A line connects the label '學習率' (Learning Rate) to the  $\eta$  box, and another line connects the label '梯度' (Gradient) to the  $\frac{\partial L}{\partial W}$  box.

學習率

梯度

損失函數: 和正確結果的誤差程度

# 學習率 Learning Rate



# 損失函數及梯度下降法

把剛剛那些東西直接和舊的權重矩陣相減。

$$W - \eta \frac{\partial L}{\partial W}$$

損失函數: 和正確結果的誤差程度

# 損失函數及**梯度下降法**

把新的值更新上去，你的模型就進步一點點囉！

$$W \leftarrow W - \eta \frac{\partial L}{\partial W}$$

損失函數：和正確結果的誤差程度

# 損失函數及**梯度**下降法

把新的值更新上去，你的模型就進步一點點囉！

$$W \leftarrow \boxed{W} - \eta \frac{\partial L}{\partial W}$$

舊的模型

損失函數：和正確結果的誤差程度



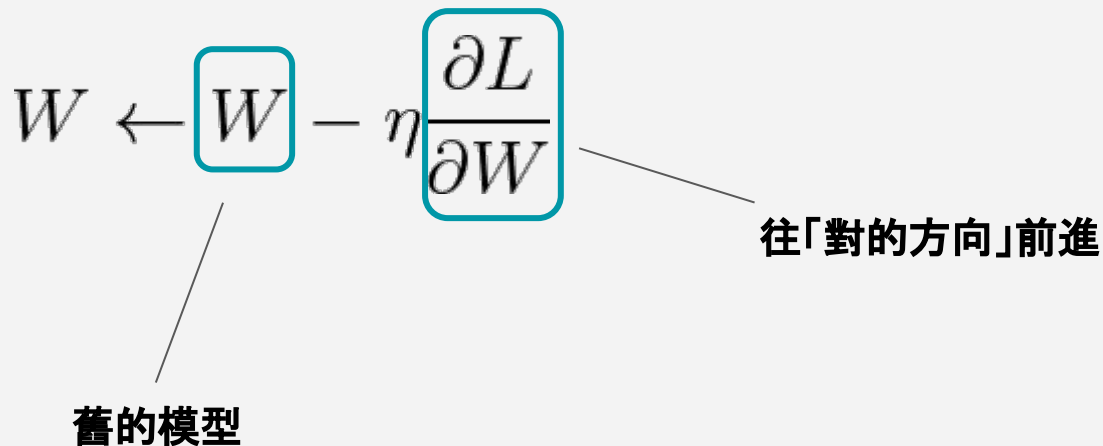
# 損失函數及梯度下降法

把新的值更新上去，你的模型就進步一點點囉！

$$W \leftarrow W - \eta \frac{\partial L}{\partial W}$$

舊的模型

往「對的方向」前進

The diagram shows the gradient descent update formula:  $W \leftarrow W - \eta \frac{\partial L}{\partial W}$ . The first  $W$  on the left is not boxed. The  $W$  in the assignment is enclosed in a teal box, with a line pointing to the text '舊的模型' (Old model) below it. The entire term  $\eta \frac{\partial L}{\partial W}$  is enclosed in a teal box, with a line pointing to the text '往「對的方向」前進' (Move in the 'right direction') to its right.

損失函數：和正確結果的誤差程度

# 損失函數及梯度下降法

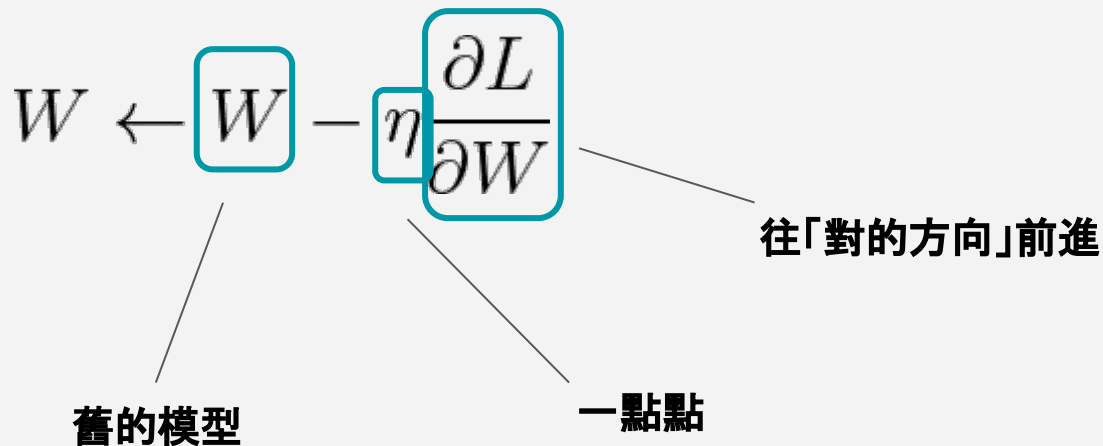
把新的值更新上去，你的模型就進步一點點囉！

$$W \leftarrow W - \eta \frac{\partial L}{\partial W}$$

舊的模型

一點點

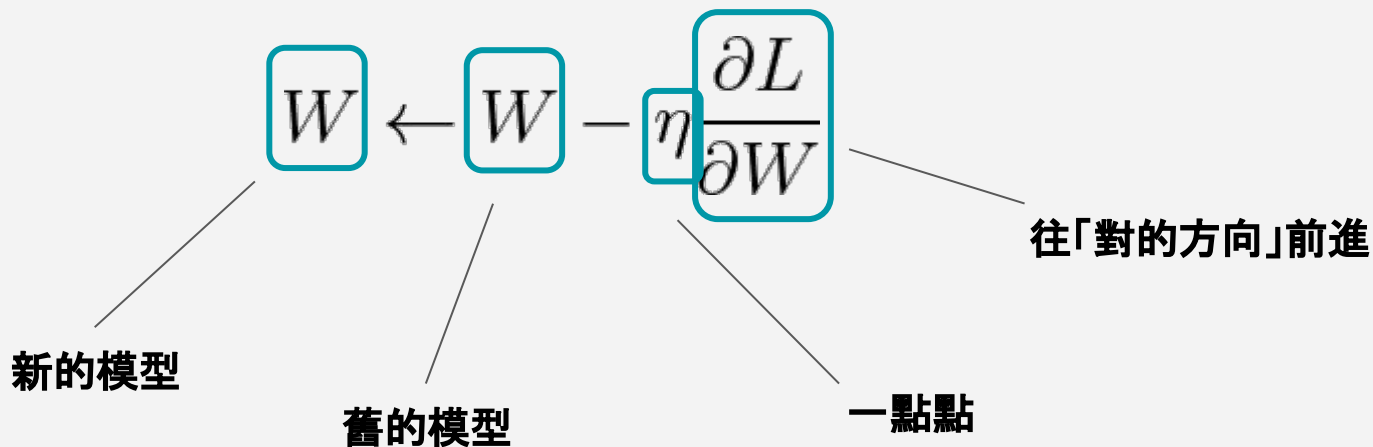
往「對的方向」前進

The diagram shows the gradient descent update rule:  $W \leftarrow W - \eta \frac{\partial L}{\partial W}$ . Three teal boxes highlight the components: the first  $W$  is labeled '舊的模型' (Old model), the  $\eta$  is labeled '一點點' (A little bit), and the fraction  $\frac{\partial L}{\partial W}$  is labeled '往「對的方向」前進' (Move in the 'right direction').

損失函數：和正確結果的誤差程度

# 損失函數及梯度下降法

把新的值更新上去，你的模型就進步一點點囉！



損失函數：和正確結果的誤差程度

# 監督式學習 Supervised Learning

好，我們知道要怎麼調整了。



$$W \overset{\text{(調整)}}{\leftarrow} W - \eta \frac{\partial L}{\partial W}$$

「70%的機率是狗。」

**調低**

損失函數：和正確結果的誤差程度

# 監督式學習 Supervised Learning

就這樣不斷訓練下去，Loss 也會逐漸逼近最小值。

而我們也了解，要調高或是調低，調整多少，是**梯度和損失函數**說了算。



$$W \overset{\text{(調整)}}{\leftarrow} W - \eta \frac{\partial L}{\partial W}$$

「30%的機率是狗。」

**調低**

損失函數：和正確結果的誤差程度

# 監督式學習 Supervised Learning

就這樣不斷訓練下去, Loss 也會逐漸逼近最小值。



$$W \overset{\text{(調整)}}{\leftarrow} W - \eta \frac{\partial L}{\partial W}$$

「5%的機率是狗。」

**調低**

損失函數: 和正確結果的誤差程度

# 監督式學習 Supervised Learning

看來梯度覺得滿意了。



$$W \overset{\text{(調整)}}{\leftarrow} W - \eta \frac{\partial L}{\partial W}$$

「1%的機率是狗。」

**差不多了**

損失函數：和正確結果的誤差程度

我們就訓練好一個模型了



$$W \leftarrow W - \eta \frac{\partial L}{\partial W}$$

「1%的機率是狗。」

損失函數：和正確結果的誤差程度



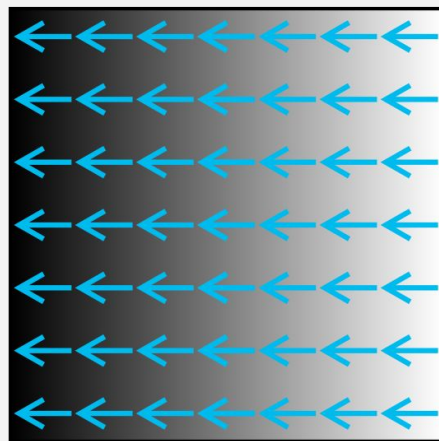
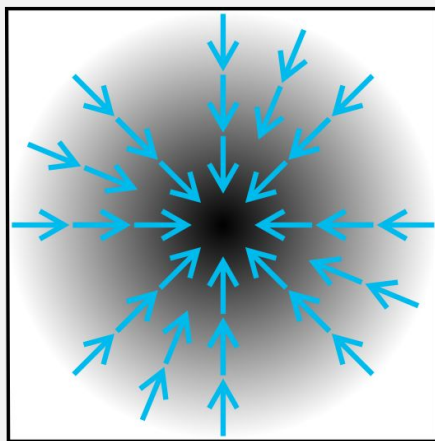
## 補充：梯度下降法

斜率只是針對二維平面的概念，我們的 $W$ 是矩陣，裏面包含很多參數，是屬於高維度的。所以數學家給這個概念一個新的名稱，**梯度 (Gradient)**。

事實上，梯度就是**坡度**。將高維度中的每一個獨立參數分別做**偏微分**所得到的值組合成一個**向量**，代表我這個高維空間對每一個參數的投影的梯度。

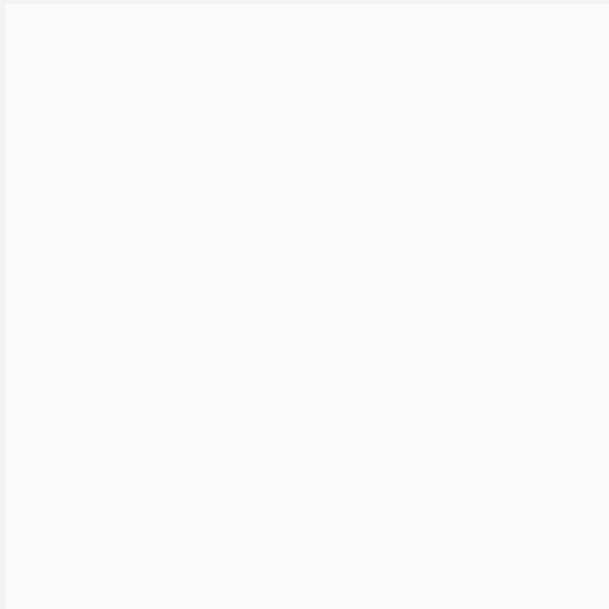
$$\nabla = \left( \frac{\partial}{\partial w_{11}}, \frac{\partial}{\partial w_{12}}, \frac{\partial}{\partial w_{13}}, \dots \right)$$

$$\nabla L(\theta) = \left[ \frac{\partial L}{\partial w_{11}} \quad \frac{\partial L}{\partial w_{12}} \quad \frac{\partial L}{\partial w_{13}} \quad \dots \right]$$



## 補充：梯度下降法

總之，大家想像成，**梯度就是各權重 $w$ 的改變對損失函數造成的改變趨勢**就好了。



## 補充：梯度下降法的缺點



## 補充：梯度下降法的缺點

想一下：我們走在山裡時，會知道  
哪裡是**最低點**嗎？



## 補充：梯度下降法的缺點

想一下：我們走在山裡時，會知道哪裡是**最低點**嗎？

Ans. **不會！**

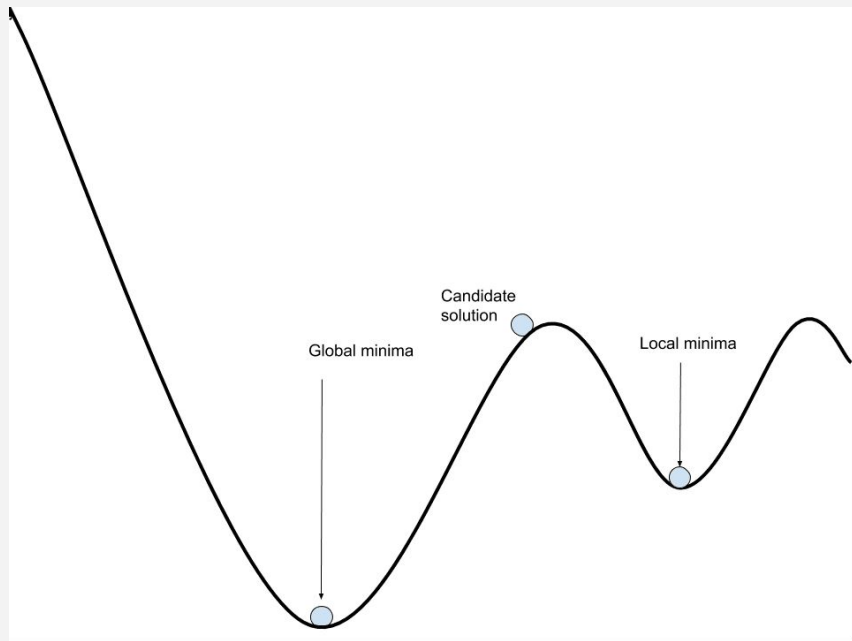
我們只能知道現在的**坡度**，但永遠不知道山的**另一頭**會不會有更低的地方。





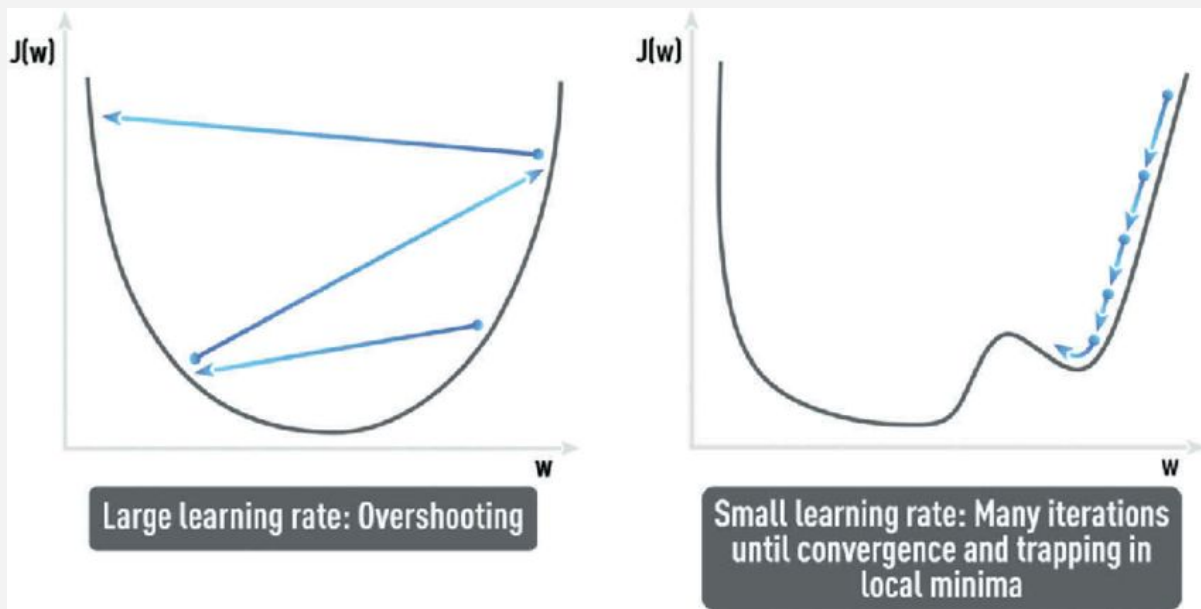
## 補充：梯度下降法的缺點

Gradient descent **can't find** where's the “global minima”.



## 補充：梯度下降法的缺點

In this case, the learning rate ( $lr$ ) becomes more important. Too large  $lr$  makes the results diverge, while the too small  $lr$  makes it “**trapped**” in the local minima.



## 補充：梯度下降法

請想想：為什麼是用減的而不是加的呢？

$$W \leftarrow W - \eta \frac{\partial L}{\partial W}$$



# 補充：梯度下降法

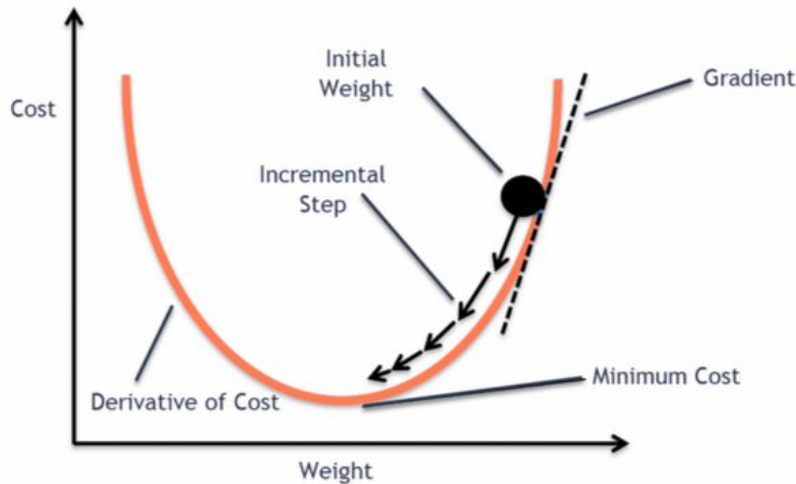
請想想：為什麼是用減的而不是加的呢？

Ans.

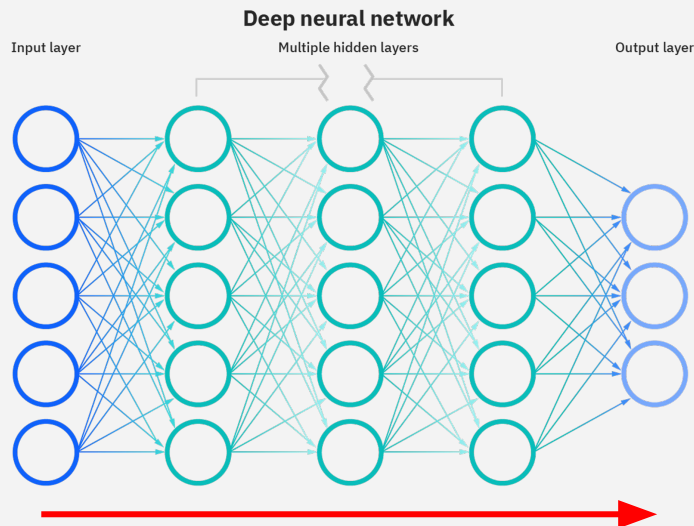
梯度**正**->權重**減少**

梯度**負**->權重**增加**

更新方向是**負的梯度值**！



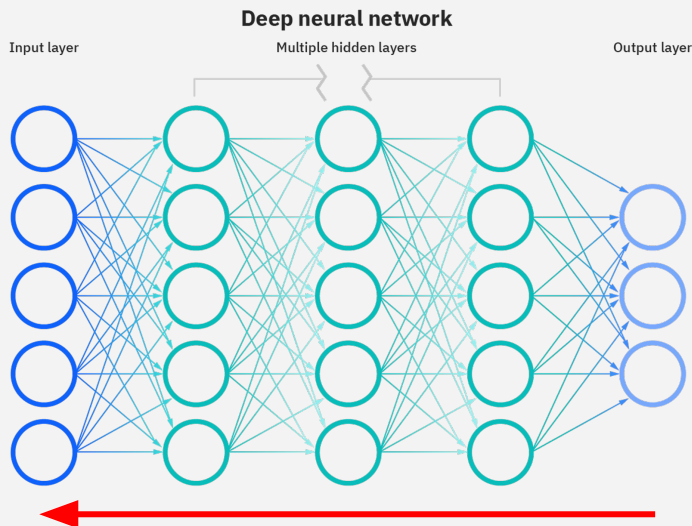
## 補充：正向傳播和反向傳播



我們的預測結果是正向傳播到輸出層的

$$\hat{y} = \sum \mathbf{w}x + b$$

# 補充：正向傳播和反向傳播



修正W是從輸出層往前傳的

$$\frac{\partial C_0}{\partial w^{(L)}} = \frac{\frac{\partial z^{(L)}}{\partial w^{(L)}}}{\frac{\partial a^{(L)}}{\partial z^{(L)}}} \frac{\partial C_0}{\partial a^{(L)}}$$

Chain Rule!

# 補充：正向傳播和反向傳播

關於反向傳播，還有超多數學，可以參考：

<https://www.3blue1brown.com/lessons/backpropagation-calculus>

<https://www.jeremyjordan.me/neural-networks-training/>

$$\begin{array}{c}
 \text{Layer 1} \\
 \frac{\partial J(\theta)}{\partial \theta_{11}^{(1)}} = \underbrace{\left( \frac{\partial J(\theta)}{\partial a_1^{(3)}} \right) \left( \frac{\partial a_1^{(3)}}{\partial z_1^{(3)}} \right) \left( \frac{\partial z_1^{(3)}}{\partial a_1^{(2)}} \right) \left( \frac{\partial a_1^{(2)}}{\partial z_1^{(2)}} \right) \left( \frac{\partial z_1^{(2)}}{\partial \theta_{11}^{(1)}} \right)}_{\text{derivative chain for } \delta_1^{(3)}} + \underbrace{\left( \frac{\partial J(\theta)}{\partial a_2^{(3)}} \right) \left( \frac{\partial a_2^{(3)}}{\partial z_2^{(3)}} \right) \left( \frac{\partial z_2^{(3)}}{\partial a_1^{(2)}} \right) \left( \frac{\partial a_1^{(2)}}{\partial z_1^{(2)}} \right) \left( \frac{\partial z_1^{(2)}}{\partial \theta_{11}^{(1)}} \right)}_{\text{derivative chain for } \delta_2^{(3)}} \\
 \frac{\partial J(\theta)}{\partial \theta_{12}^{(1)}} = \left( \frac{\partial J(\theta)}{\partial a_1^{(3)}} \right) \left( \frac{\partial a_1^{(3)}}{\partial z_1^{(3)}} \right) \left( \frac{\partial z_1^{(3)}}{\partial a_1^{(2)}} \right) \left( \frac{\partial a_1^{(2)}}{\partial z_1^{(2)}} \right) \left( \frac{\partial z_1^{(2)}}{\partial \theta_{12}^{(1)}} \right) + \left( \frac{\partial J(\theta)}{\partial a_2^{(3)}} \right) \left( \frac{\partial a_2^{(3)}}{\partial z_2^{(3)}} \right) \left( \frac{\partial z_2^{(3)}}{\partial a_1^{(2)}} \right) \left( \frac{\partial a_1^{(2)}}{\partial z_1^{(2)}} \right) \left( \frac{\partial z_1^{(2)}}{\partial \theta_{12}^{(1)}} \right) \\
 \frac{\partial J(\theta)}{\partial \theta_{21}^{(1)}} = \left( \frac{\partial J(\theta)}{\partial a_1^{(3)}} \right) \left( \frac{\partial a_1^{(3)}}{\partial z_1^{(3)}} \right) \left( \frac{\partial z_1^{(3)}}{\partial a_2^{(2)}} \right) \left( \frac{\partial a_2^{(2)}}{\partial z_2^{(2)}} \right) \left( \frac{\partial z_2^{(2)}}{\partial \theta_{21}^{(1)}} \right) + \left( \frac{\partial J(\theta)}{\partial a_2^{(3)}} \right) \left( \frac{\partial a_2^{(3)}}{\partial z_2^{(3)}} \right) \left( \frac{\partial z_2^{(3)}}{\partial a_2^{(2)}} \right) \left( \frac{\partial a_2^{(2)}}{\partial z_2^{(2)}} \right) \left( \frac{\partial z_2^{(2)}}{\partial \theta_{21}^{(1)}} \right) \\
 \frac{\partial J(\theta)}{\partial \theta_{22}^{(1)}} = \left( \frac{\partial J(\theta)}{\partial a_1^{(3)}} \right) \left( \frac{\partial a_1^{(3)}}{\partial z_1^{(3)}} \right) \left( \frac{\partial z_1^{(3)}}{\partial a_2^{(2)}} \right) \left( \frac{\partial a_2^{(2)}}{\partial z_2^{(2)}} \right) \left( \frac{\partial z_2^{(2)}}{\partial \theta_{22}^{(1)}} \right) + \left( \frac{\partial J(\theta)}{\partial a_2^{(3)}} \right) \left( \frac{\partial a_2^{(3)}}{\partial z_2^{(3)}} \right) \left( \frac{\partial z_2^{(3)}}{\partial a_2^{(2)}} \right) \left( \frac{\partial a_2^{(2)}}{\partial z_2^{(2)}} \right) \left( \frac{\partial z_2^{(2)}}{\partial \theta_{22}^{(1)}} \right)
 \end{array}$$

還記得這個步驟吧：



**資料前處理    選擇模型    訓練    測試    執行**

# 模型訓練 Training

剛剛說到，訓練模型不會只訓練一次，會經過不斷訓練讓 Loss 下降。

我們正名一下，資料**完整訓練一次**叫作**一代**。

**每代**的訓練，會有**數次**權重的更新，直到把訓練資料全部看完。

**每次**的更新，都會從訓練資料中採用**數個樣本**進行 Loss 計算。

```
# Example

for epoch in range(EPOCHS):
    for iteration in range(len(train_dataloader)):
        # Training every data in a batch
        # ...
```

# 模型訓練 Training

現在開始，請嚴謹地使用這些詞：

Epochs (代): 模型看**完全部訓練資料**，稱為**一代更新**。

Iteration (次): **每代**模型進行**一次權重更新**，稱為**一次更新**。

Batch size (批量): **每次更新**所採用的資料數量，稱為**一批**。

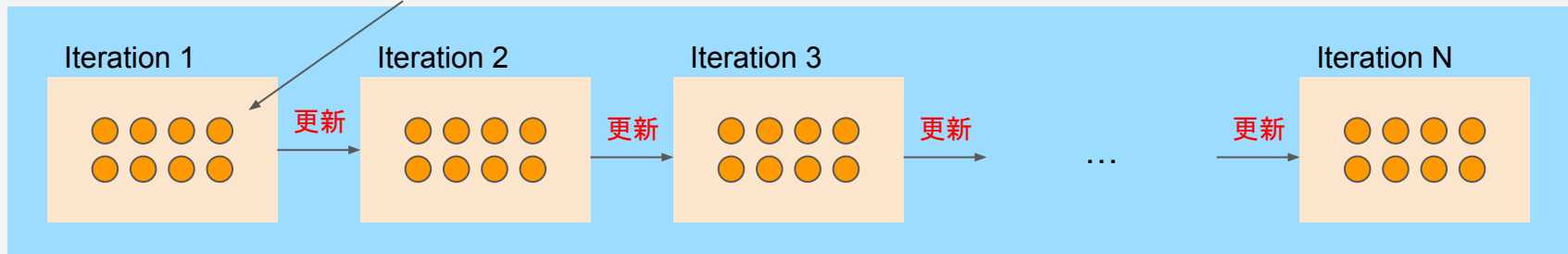
# Epoch

/ˈɛp,ək/

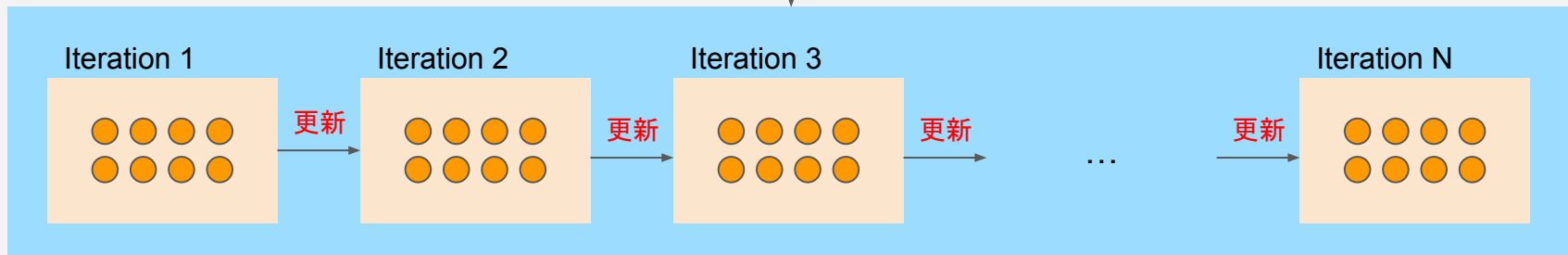


Batch size=8

Epoch 1

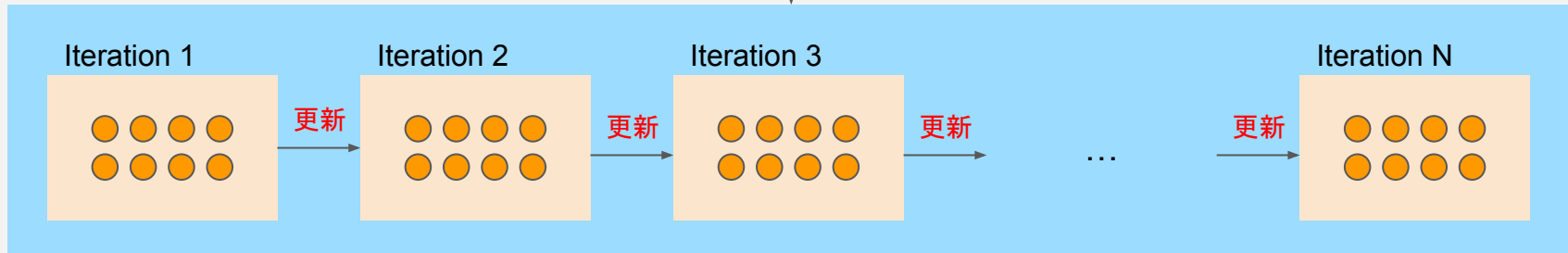


Epoch 2



...

Epoch M



## 補充: Batch Size

[https://www.cupoy.com/qa/club/ai\\_tw/0000016D6BA22D97000000016375706F795F72656C656173654B5741535354434C5542/0000017D085DB5E3000000026375706F795F72656C656173655155455354](https://www.cupoy.com/qa/club/ai_tw/0000016D6BA22D97000000016375706F795F72656C656173654B5741535354434C5542/0000017D085DB5E3000000026375706F795F72656C656173655155455354)

# 模型驗證 Validation

我們把200張貓咪和200張狗上傳上去後，他會自動幫我們切成兩部份：



# 模型驗證 Validation

為了能驗證模型的好壞，我們切出15%的資料**不進行訓練**，也就是說模型在訓練時沒有看過這15%的照片

170張貓咪、170張狗

## **85%的訓練資料(Training)**

用這部份的資料進行真實的模型訓練。

30張貓咪、30張狗

## **15%的驗證資料(Validation)**

模型會用這部份的資料驗證。

# 模型驗證 Validation

每代模型訓練完85%的資料後，再丟給模型沒看過的15%資料，計算他的損失函數以及模型預測的準確率。

如此一來，我們就可以知道模型訓練得好不好了。

註: 15% 的驗證資料 (*Validation split*) 是 *Teachable Machine* 固定的，平常在寫程式時，我們可以去分割自己需要的比例。

所以訓練這個步驟其實有兩個部份，訓練及驗證，他們會重複執行直到看完所有的訓練資料。

註：這邊就是為什麼我上禮拜說翻譯的不好的原因



資料前處理    選擇模型    訓練    測試    執行



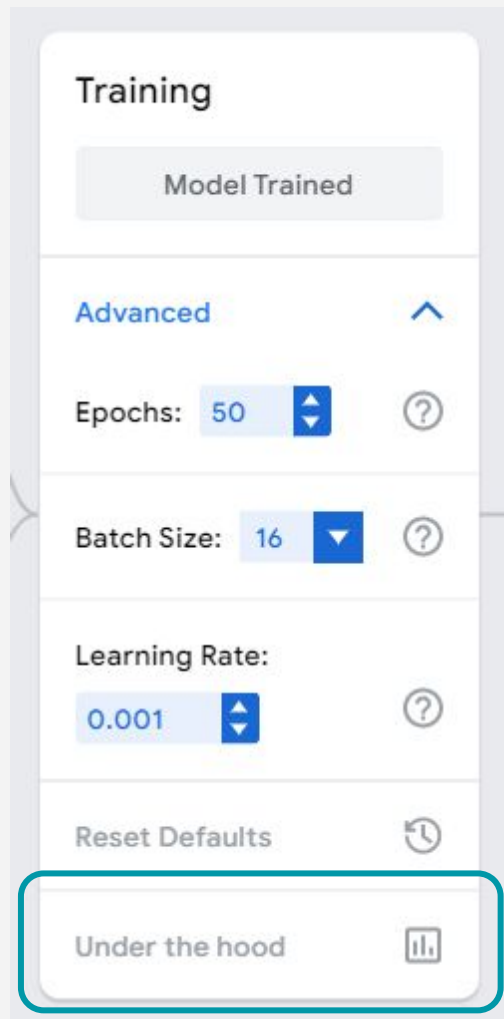
訓練    驗證

## 結果分析 2

學了一些數學，我們現在可以來把剛剛的模型做更好的修正了

## 結果分析 2

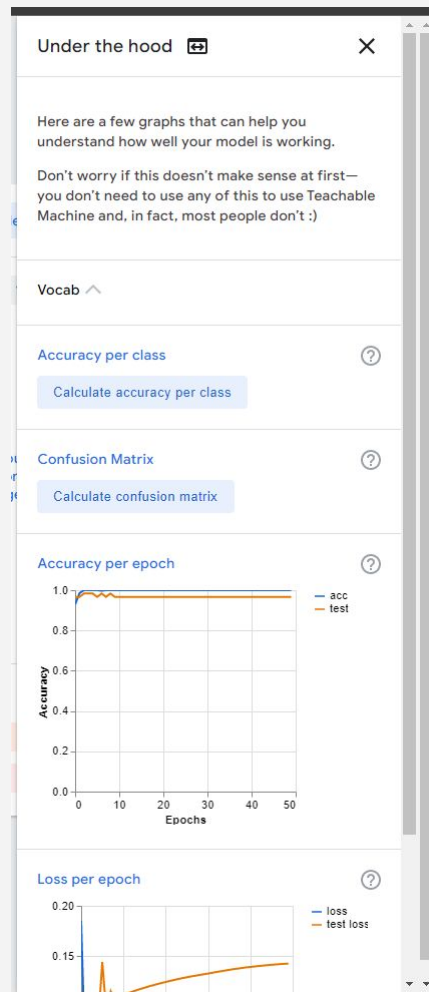
到 Training 底下，點開 Advanced 進階設定，然後點最下面的 Under the hood





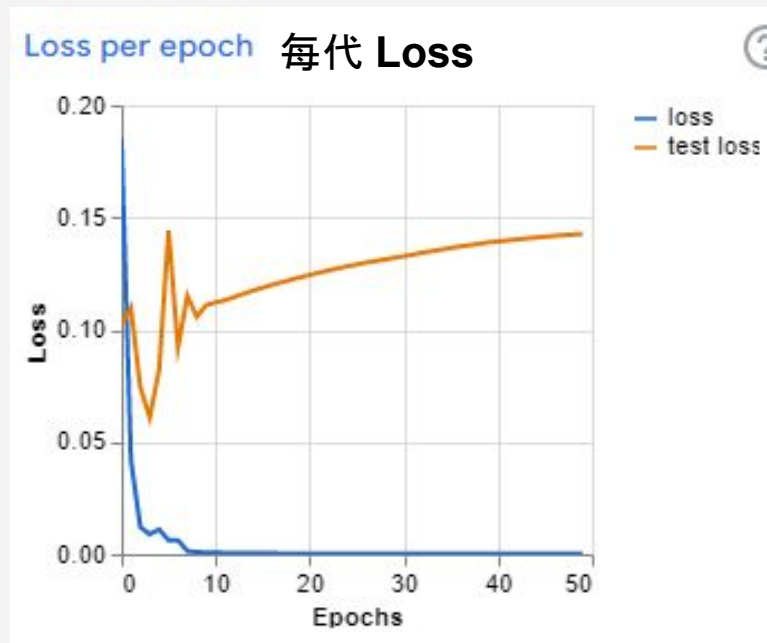
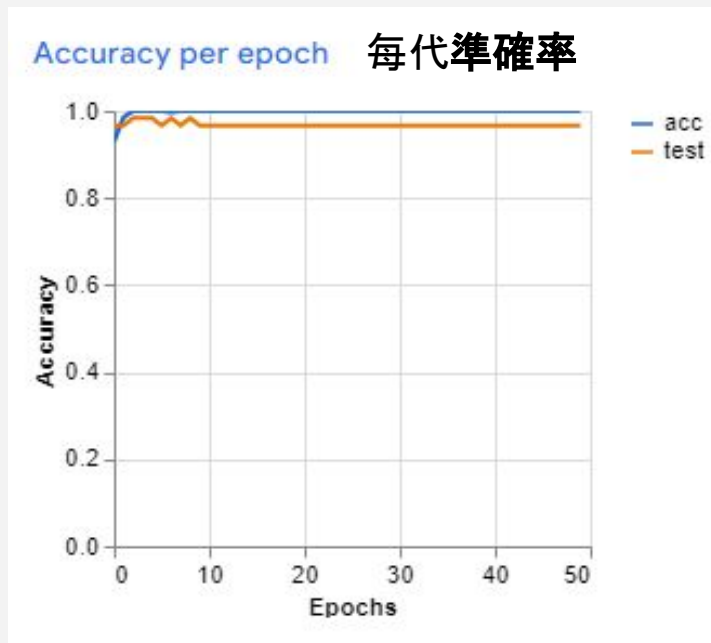
# 結果分析 2

你會看到模型訓練的細節。



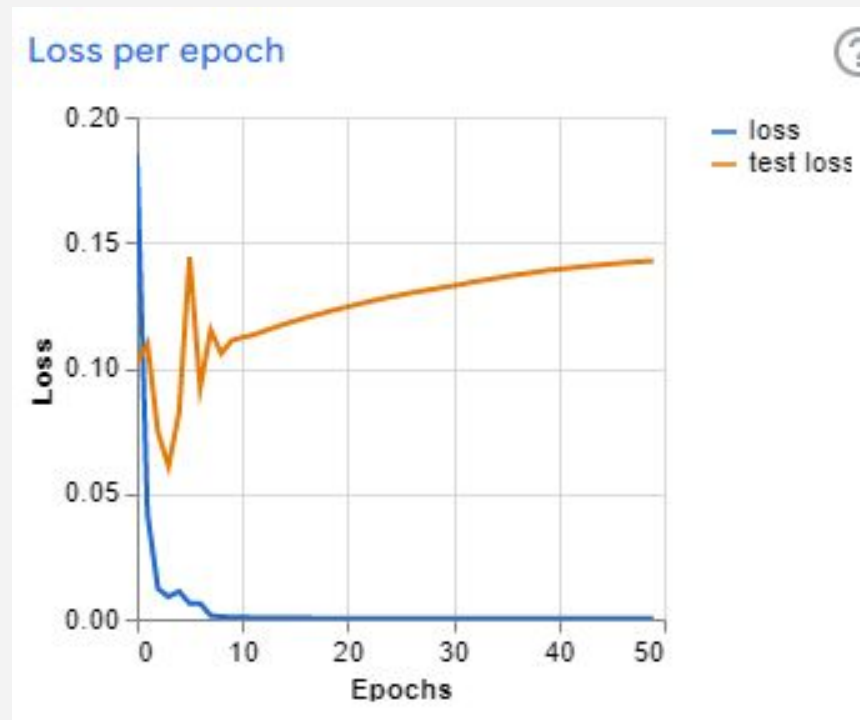
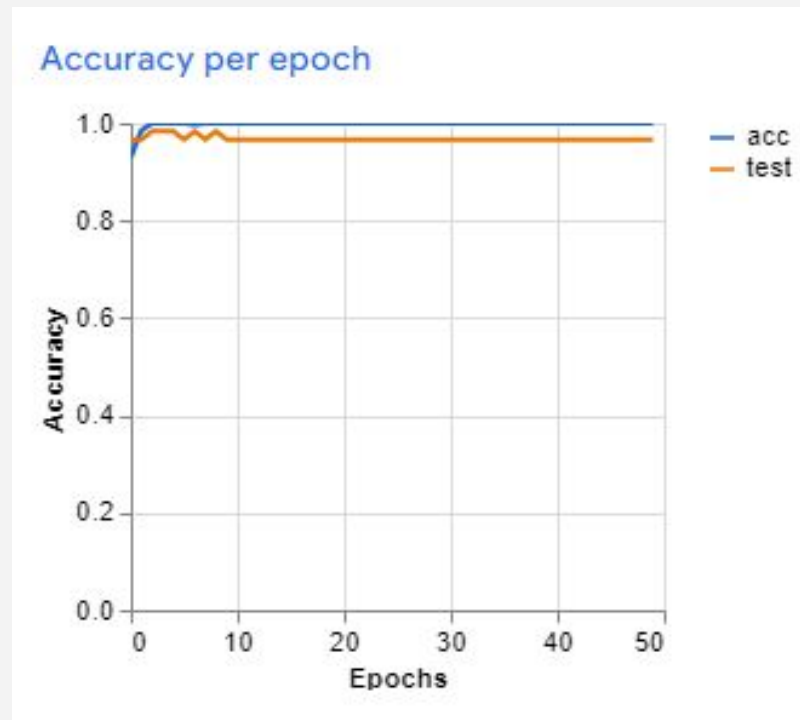
## 結果分析 2

下面這兩張圖，現在你應該要看的懂了：



損失函數 (Loss): 和正確結果的誤差程度

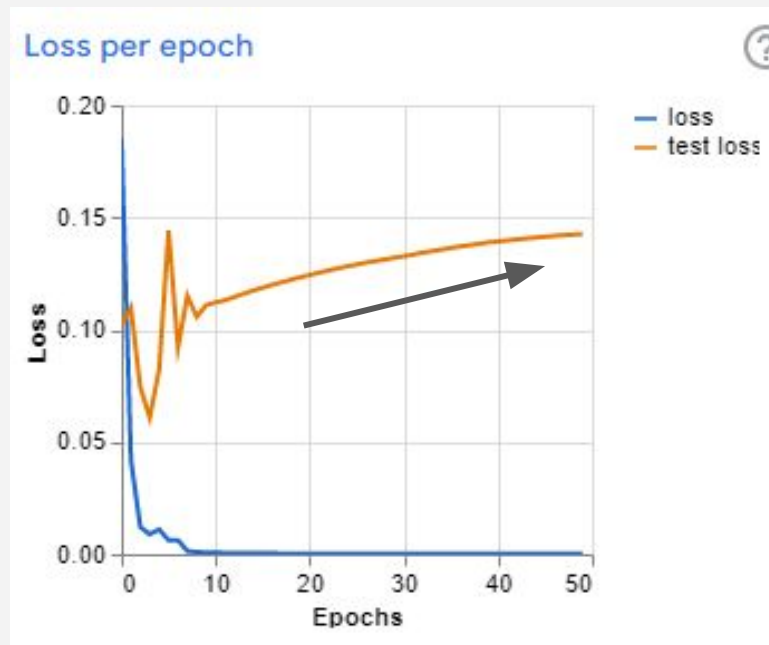
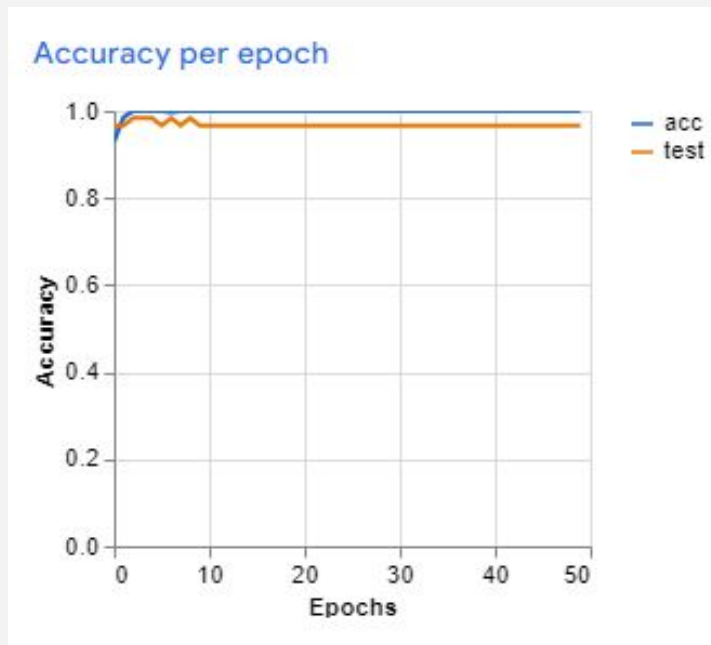
發現問題了嗎？



損失函數 (Loss): 和正確結果的誤差程度

## 結果分析 2

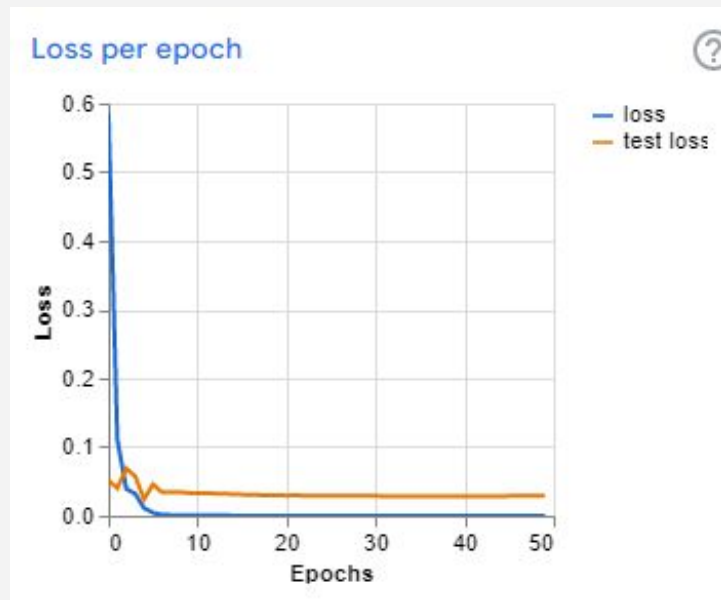
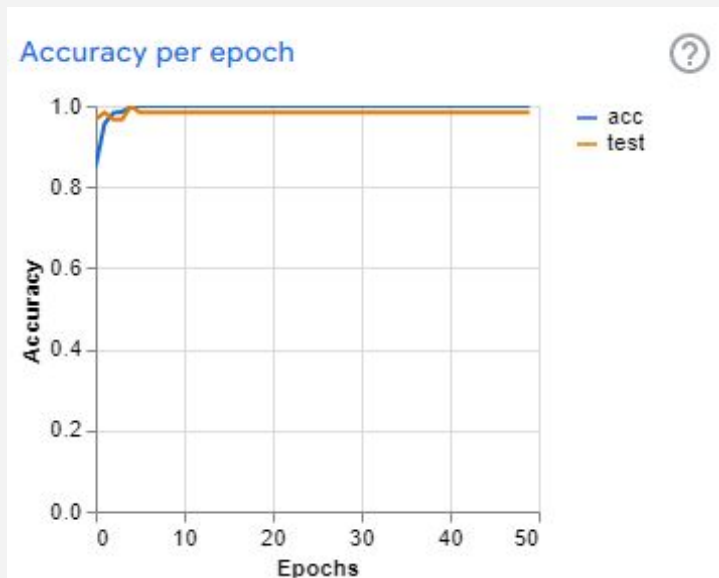
訓練越多代，反而 Loss 上升了。



損失函數 (Loss): 和正確結果的誤差程度

## 結果分析 2

理想上，應該要是這樣子，好不容易下降的 Loss 不要再上升了。

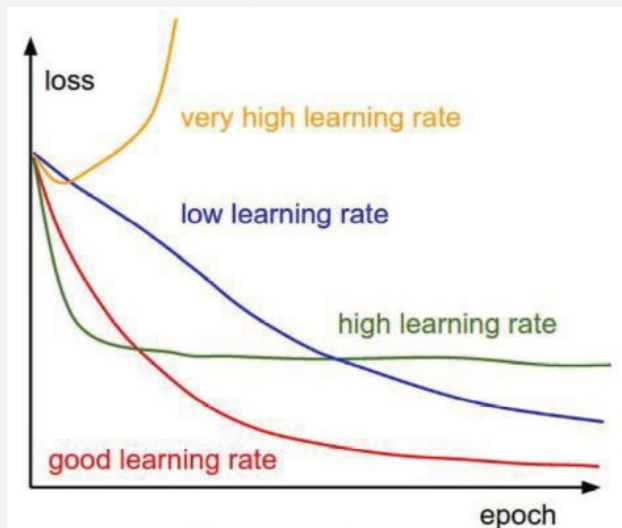


損失函數 (Loss): 和正確結果的誤差程度

# Google Teachable Machine

請調整**學習率**，想辦法解決 Loss 無法收斂的問題。

註：調整數值後都要點選 Train Model 重新訓練！



Training

Model Trained

Advanced ^

Epochs: 50 ?

Batch Size: 16 ?

Learning Rate: 0.001 ?

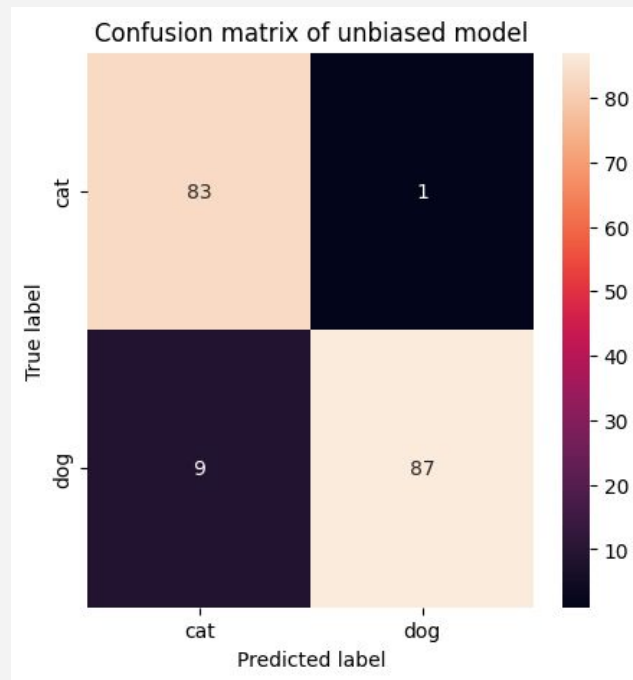
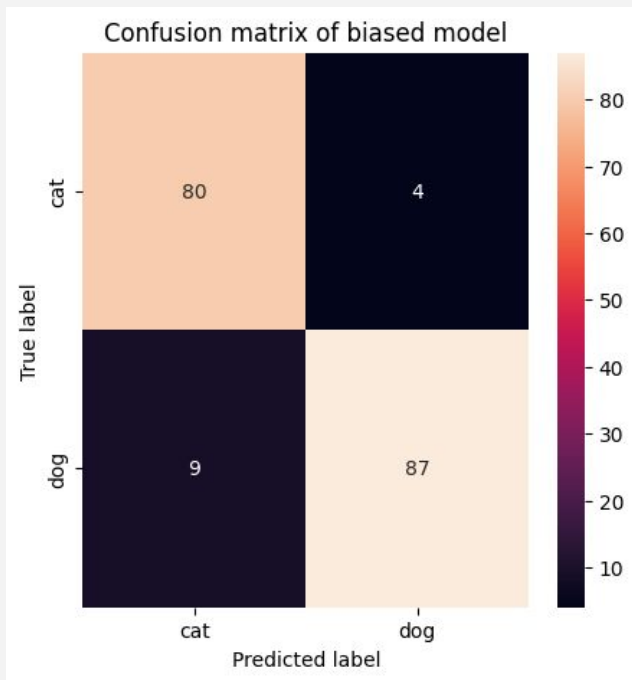
Reset Defaults ⌚

Under the hood 📊

損失函數 (Loss): 和正確結果的誤差程度

## 補充：偏差樣本訓練

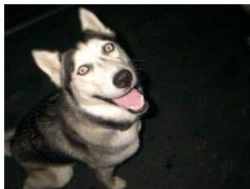
用 `biased_train_data` 裡面的樣本再訓練一次



# 補充：偏差樣本訓練

樣本偏差愈嚴重，訓練出來的模型偏見就會愈高！

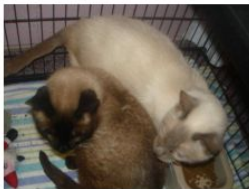
biased model: cat,  
confidence: 0.8195736



biased model: dog,  
confidence: 0.6468599



biased model: dog,  
confidence: 0.9863338



biased model: dog,  
confidence: 0.9546226



biased model: cat,  
confidence: 0.78980815



biased model: cat,  
confidence: 0.9710349





## 補充：偏差樣本訓練

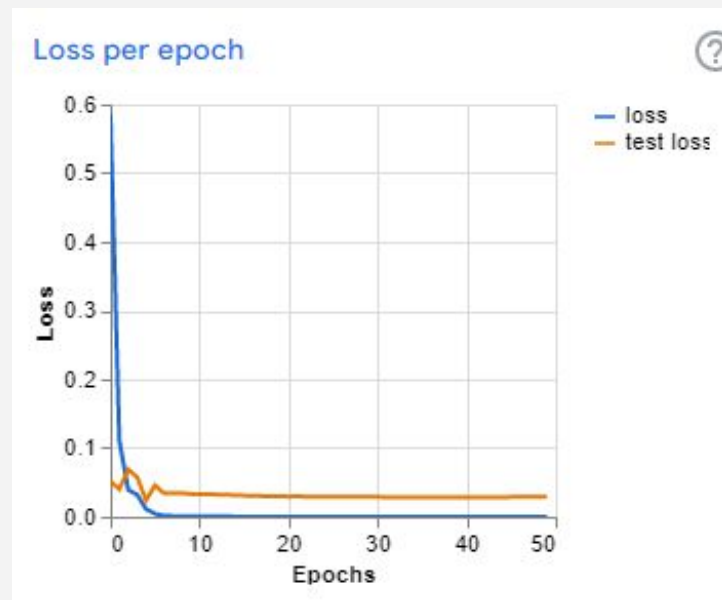
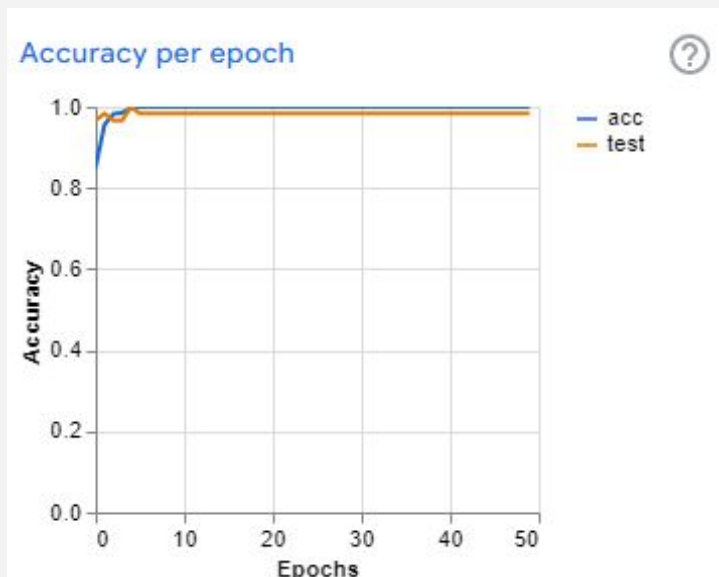
AI其實沒有大家想的那麼聰明，他們沒有自己延伸或思考的能力(至少目前的科學不能達成)，所以**一個模型的好壞可以說是取決於訓練資料的品質。**

## 補充：偏差樣本訓練

同理，因為有這樣的特性，所以使用AI工具時也要同時保有媒體識讀的戒心，以免被無意間滲透。

# 結論

今天大家應該能體驗到類神經網路有別於傳統函式或演算法的地方。類神經網路會根據輸入和輸出，進而調整自己的各項參數，**就像是和人類一樣的思考！**



## 結論

所以，這個方程式在機器學習中是非常難能可貴的，也是機器學習的一大精髓。

$$W \leftarrow W - \eta \frac{\partial L}{\partial W}$$

## Bonus: AI Tools

**When I realize  
ChatGPT can do  
my job for me**



**When I realize  
ChatGPT can do  
my job for me**



## Bonus: AI Tools

這個年代，請一定要善用 AI 工具。



# Bonus: AI Tools

呃...但看來AI沒辦法幫你回女朋友的訊息。



女朋友問我他和我媽同時掉進水裡我要先救誰，請問我要如何哄他



首先，讓你的女朋友知道這是一個很難回答的問題，因為你愛護的兩個人都是你生命中的重  
要人物，所以無論選擇哪一個都是困難的決定。



然後，你可以告訴她，你的決定可能取決於當時的情況，例如誰離你最近或誰在最危險的位置等等。

最重要的是，你要讓你的女朋友知道，無論發生什麼事情，你會盡力保護和救助兩個人，並且你會為自己的選擇負責。

最後，讓她知道你對她的愛沒有改變，並且你會盡力保護她的安全和幸福。

## 回家功課

1. 請一定要了解梯度下降法「走一步算一步」的思維。
2. 請試著應用 ChatGPT 或是其他 AI 工具在你的生活當中。
3. 試想：我們會被人工智慧取代嗎？

**歡迎在課程意見回饋表與我們分享你的想法！**