

透過深度神經網路(DNN)攻擊來學習 OWASP ML Top 10

合作金庫商業銀行/資安部
lan

關於我 – lan (育正葛葛)

- 現任合作金庫商業銀行資安部 – 二等專員



大綱

- 背景介紹
 - 機器學習定義及學習方式
 - 機器學習的各階段與相關元件
- 深度神經網路 (DNN)
 - 先備數學
 - 深度神經網路結構及運作原理
 - 建立MNIST 手寫數字辨識模型
- OWASP Machine Learning Security Top Ten
 - 攻擊原理
 - 部分攻擊效果展示
- 結論與大語言模型安全

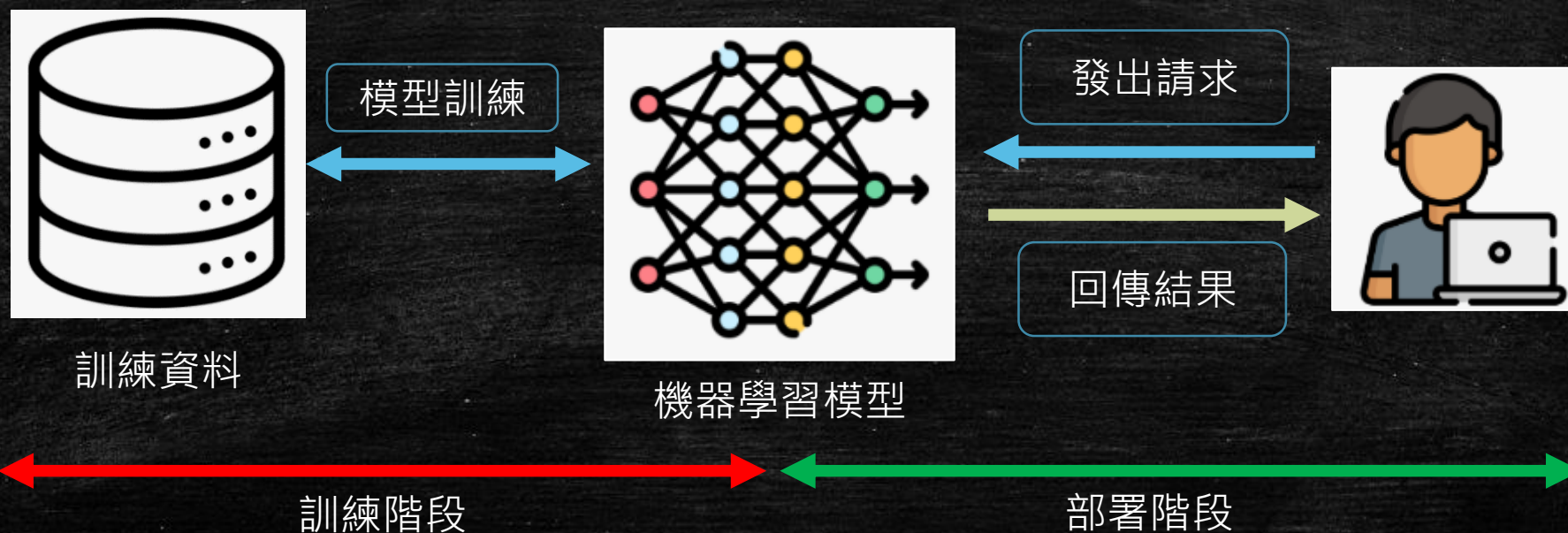
機器學習 - 定義

<https://zh.wikipedia.org/zh-tw/%E6%9C%BA%E5%99%A8%E5%AD%A6%E4%B9%A0>

- 機器學習理論主要是設計和分析一些讓電腦可以自動學習的演算法
- 機器學習演算法是從資料中自動分析獲得規律，並利用規律對未知資料進行預測的演算法
- 而機器學習演算法的學習方式跟人類是一樣的

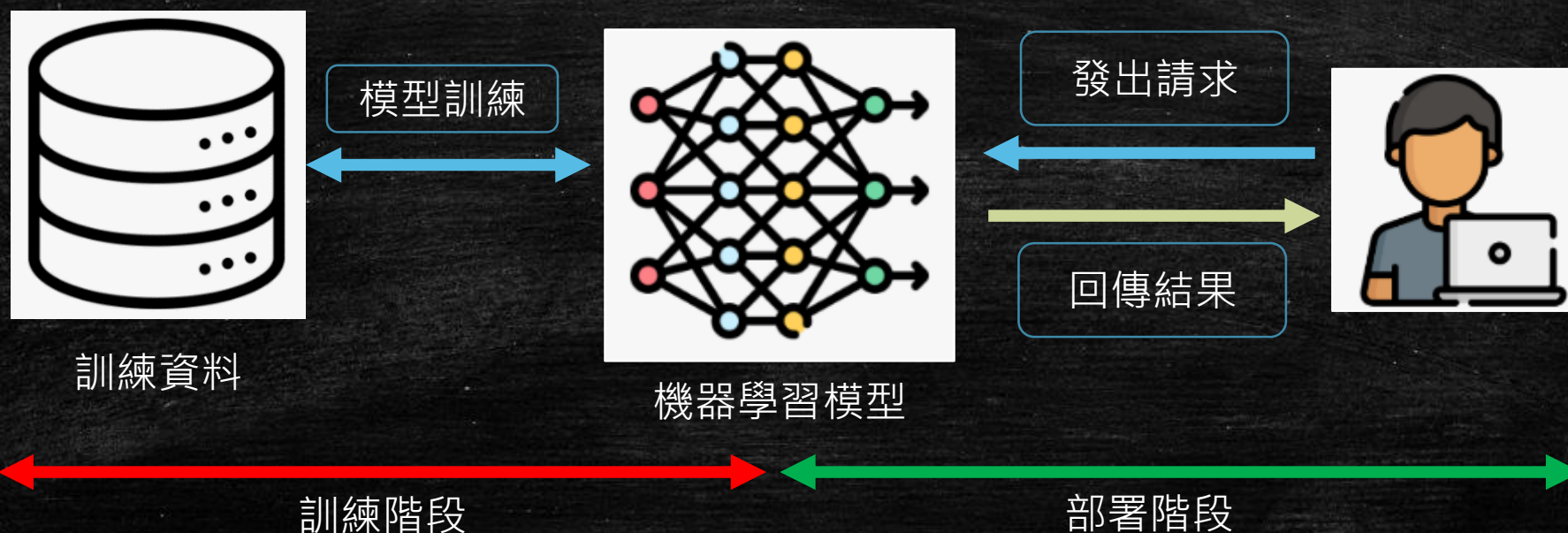
機器學習演算法的訓練方式

- 人的腦袋就如同機器學習的模型結構、參數
- 書籍跟答案就如同機器學習中監督式學習的訓練資料
- 學習知識並內化就是訓練階段，解決真實問題就是部署階段



機器學習的階段與元件

- 兩大階段，訓練階段模型會觸碰到訓練資料進行學習及調整，部署階段模型會接收到請求資料、進行推論及回應
- 四大元件，硬體、軟體、模型(結構、參數)、資料(訓練、請求、推論結果)



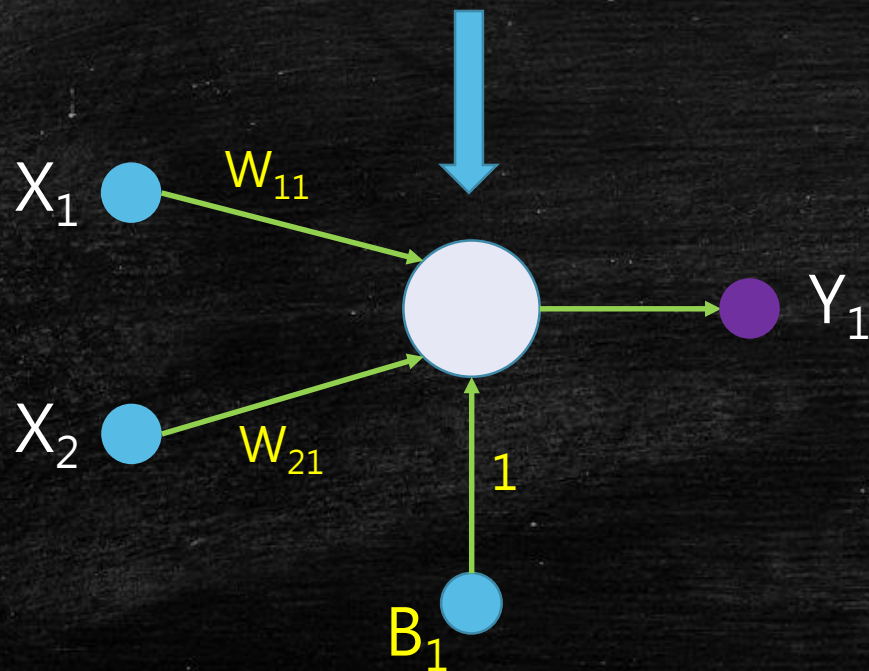
會用到的數學－線性代數

- 熱身一下，今天有輸入變數 X_1 、 X_2 ，然後有固定參數 W_{11} 、 W_{21} 、 B_1
- 假設 Y_1 求法如下：
$$Y_1 = X_1 W_{11} + X_2 W_{21} + B_1$$
- 則當輸入變數 $X_1=1$, $X_2=3$ 、固定參數 $W_{11}=2$, $W_{21}=0$, $B_1=3$

$$Y_1 = 1 \times 2 + 3 \times 0 + 3 = 5$$

會用到的數學 – 線性代數

- 如果用神經元的形式表達變數 X_1 、 X_2 ，及固定參數 W_{11} 、 W_{21} 、 B_1
- $Y_1 = X_1 W_{11} + X_2 W_{21} + B_1 \times 1$



會用到的數學 – 線性代數

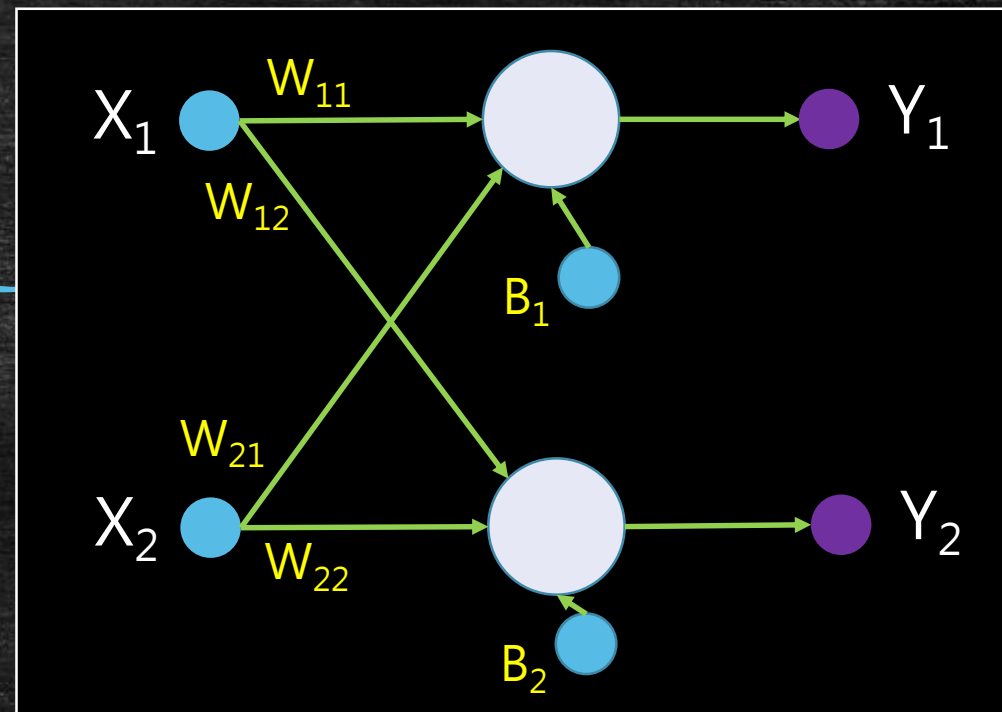
- 如果用神經元有多個的情況呢?
- $Y_1 = X_1 W_{11} + X_2 W_{21} + B_1$
- $Y_2 = X_1 W_{12} + X_2 W_{22} + B_2$



$$\begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} = \begin{bmatrix} W_{11} & W_{21} \\ W_{12} & W_{22} \end{bmatrix} \times \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} + \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} = \begin{bmatrix} W_{11} \times X_1 + W_{21} \times X_2 \\ W_{12} \times X_1 + W_{22} \times X_2 \end{bmatrix} + \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}$$

- 或是 $(Wx + b)^T = x^T W^T + b^T$

$$[Y_1 \quad Y_2] = [X_1 \quad X_2] \times \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix} + [B_1 \quad B_2]$$

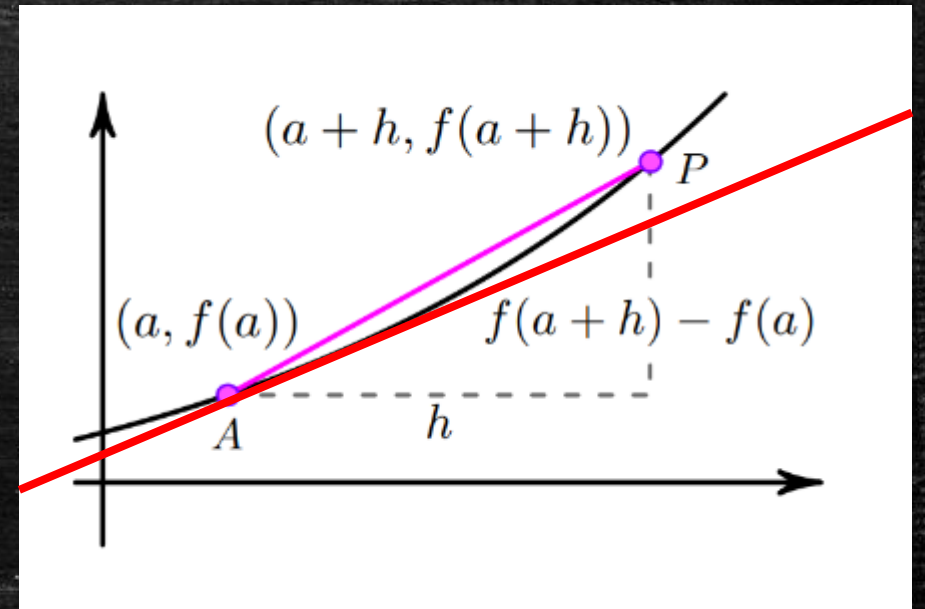


會用到的數學 – 微積分

- f 於 $x=a$, $x=a+h$ 的割線斜率定義為

$$m = \frac{f(a+h) - f(a)}{(a+h) - a}$$

- 導數：函數在某一點附近的變化率，即函數在該點的切線斜率



- 當 $h \rightarrow 0$ 的時候，則是計算 $x=a$ 時該點的切線斜率

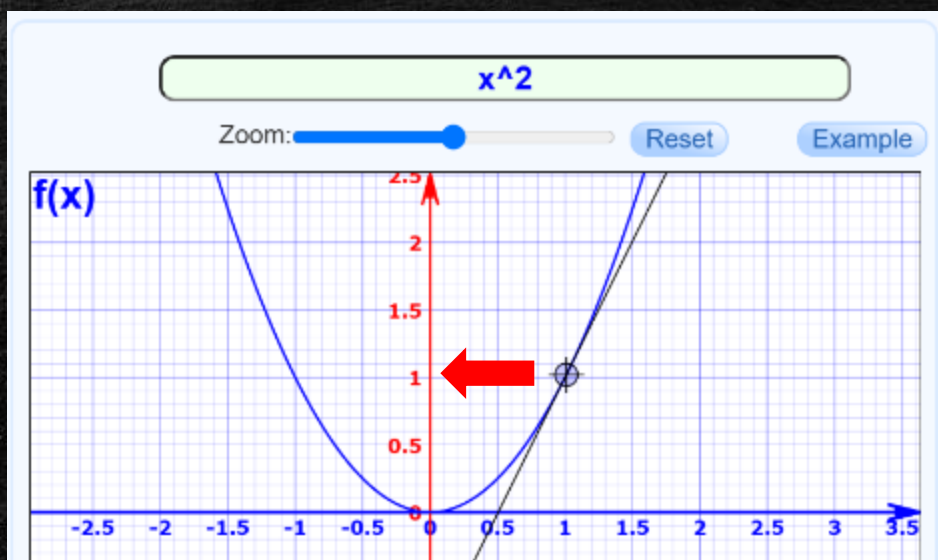
$$\frac{d}{dx} f(a) = f'(a) = \lim_{h \rightarrow 0} \frac{f(a+h) - f(a)}{(a+h) - a}$$

會用到的數學－微積分

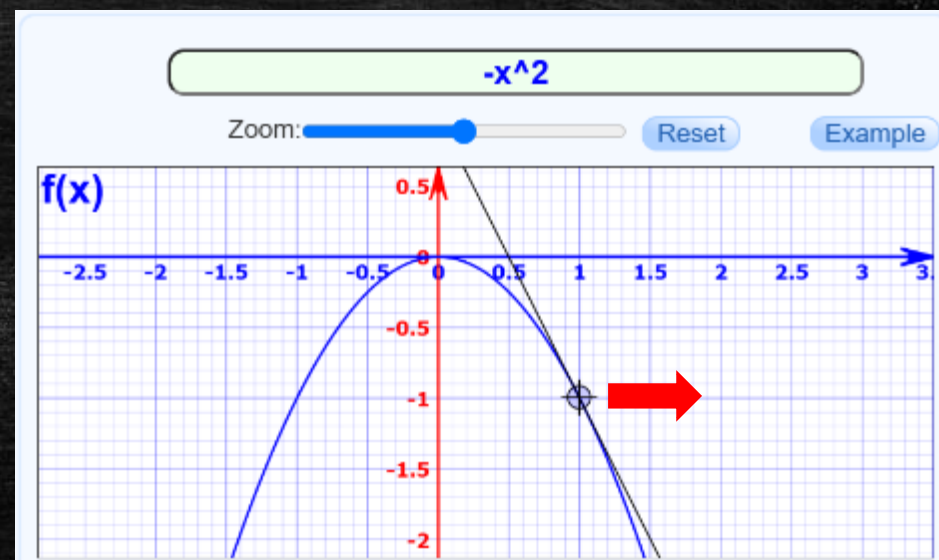
<https://www.mathsisfun.com/calculus/derivative-plotter.html>

- 切線斜率特性：在求導數的點往其切線斜率反方向前進會使函數數值變小

斜率為正



斜率為負

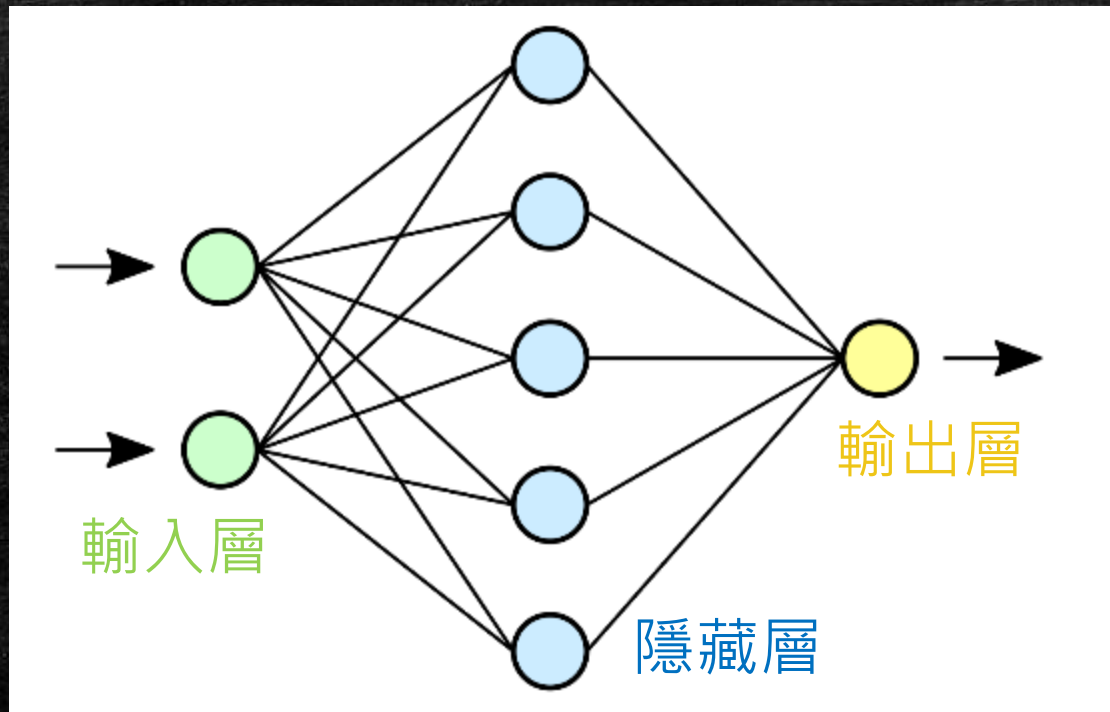


<https://ithelp.ithome.com.tw/articles/10266985>

<https://zh.wikipedia.org/wiki/%E4%BA%BA%E5%B7%A5%E7%A5%9E%E7%BB%8F%E7%BD%91%E7%BB%9C>

深度神經網路

- 深度神經網路，是一種模仿生物神經網路的結構和功能的數學模型
- 其結構由三部分組成，輸入層 (Input layer)、隱藏層 (Hidden layer)、輸出層 (Output layer)

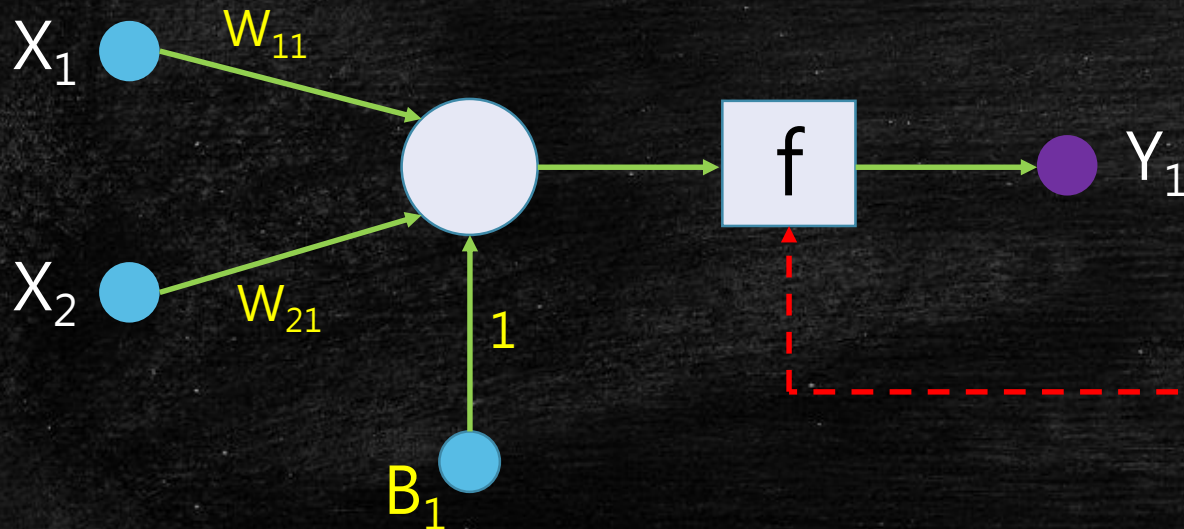


<https://zh.wikipedia.org/wiki/%E4%BA%BA%E5%B7%A5%E7%A5%9E%E7%BB%8F%E7%BD%91%E7%BB%9C>

<https://zh.wikipedia.org/zh-tw/%E6%BF%80%E6%B4%BB%E5%87%BD%E6%95%B0>

深度神經網路－神經元

- 加入新觀念激勵函數，將線性結果轉為非線性的輸出
- 單一神經元的形式表達更新為 $Y_1 = f(X_1 W_{11} + X_2 W_{21} + B_1)$
- 激勵函數對於語言模型的安全議題可參考 [Sponge Examples: Energy-Latency Attacks on Neural Networks \(2020\)](#)



線性整流
函式
(ReLU)



$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

深度神經網路究竟如何學習

1. 準備訓練資料

- 資料採集、準備
- 資料清洗、格式轉換、特徵選取
- 資料正規化、標準化

2. 訓練階段

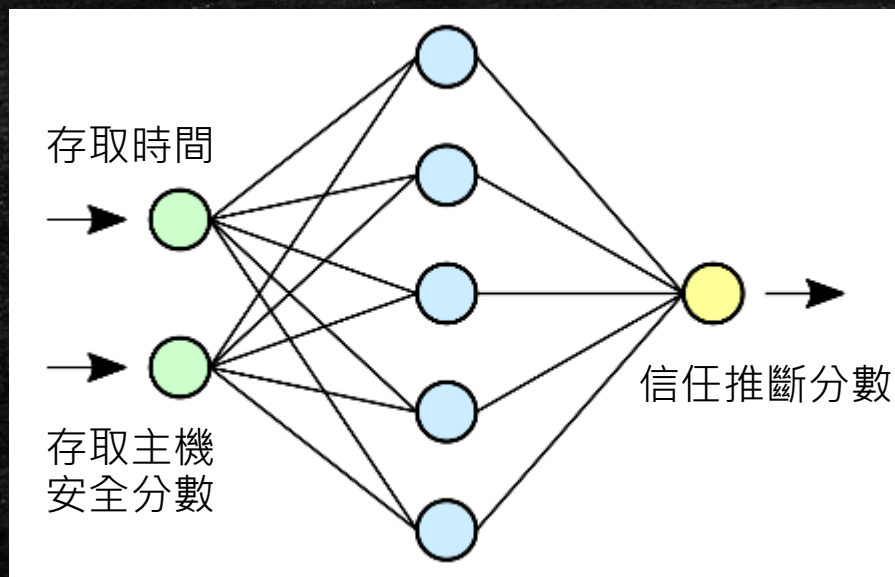


- 前向傳播 (Forward Propagation) - 根據輸入得到推論結果
- 計算損失函數 (Loss Calculation) - 比對推論結果與訓練資料標籤差異
- 後向傳播 (Backward Propagation) - 依照梯度去修正模型權重參數

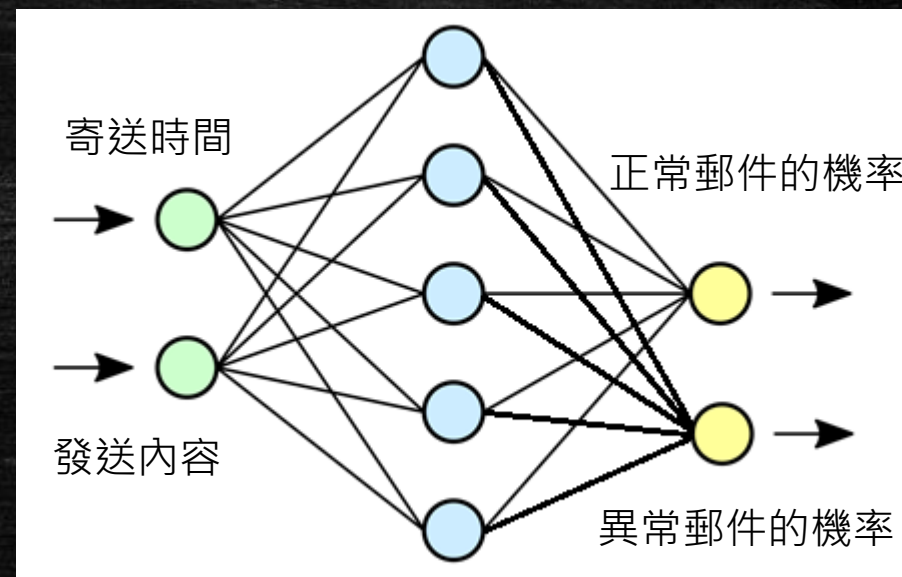
前向傳播 (Forward Propagation) - 推論結果

- 其實就是前面介紹的算法從頭算到尾
- 但要注意輸出結果所代表的意義，有些是數值結果，有些是分類結果

零信任推斷模型



垃圾郵件判斷模型



計算損失函數 (Loss Calculation) - 比對結果

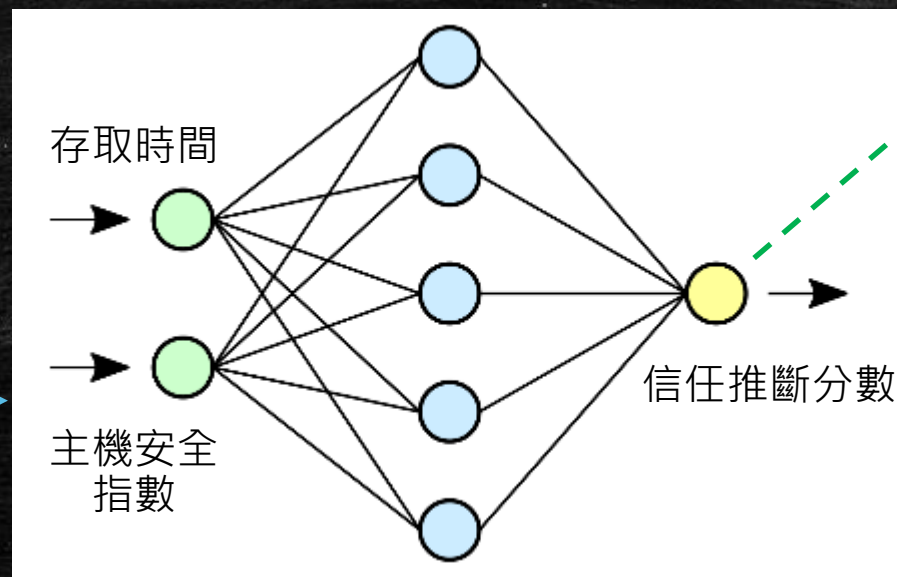
- 定義出損失函數(Loss Function)，計算模型推論出來的數值與資料標籤的落差，其計算結果數值越小，代表模型推論的結果越接近訓練資料的標籤
- 數值型的通常使用均方誤差 (MSE)，而分類型的通常使用交叉熵 (cross-entropy)

訓練資料
(存取時間、主機安全指數) -> (信任推斷分數)



模型訓練

零信任推斷模型



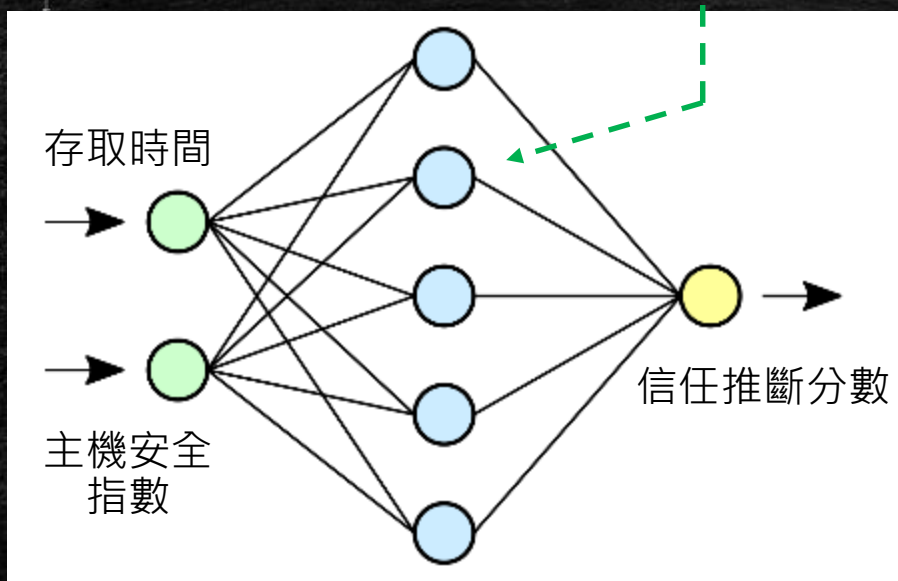
$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

後向傳播 (Backward Propagation) - 修正參數

- 透過損失函數(Loss Function)關於模型權重的偏微分，得到梯度
- 透過梯度的反方向乘上學習率進行權重更新

$$W_{new} = W_{old} - \eta \times \frac{\partial L}{\partial W} \Big|_{W=W_{old}}$$

$$\frac{\partial L}{\partial W} = \begin{bmatrix} \frac{\partial L}{\partial W_{11}} \\ \frac{\partial L}{\partial W_{12}} \\ \vdots \\ \frac{\partial L}{\partial W_{25}} \end{bmatrix}$$



$$Loss(W) = MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

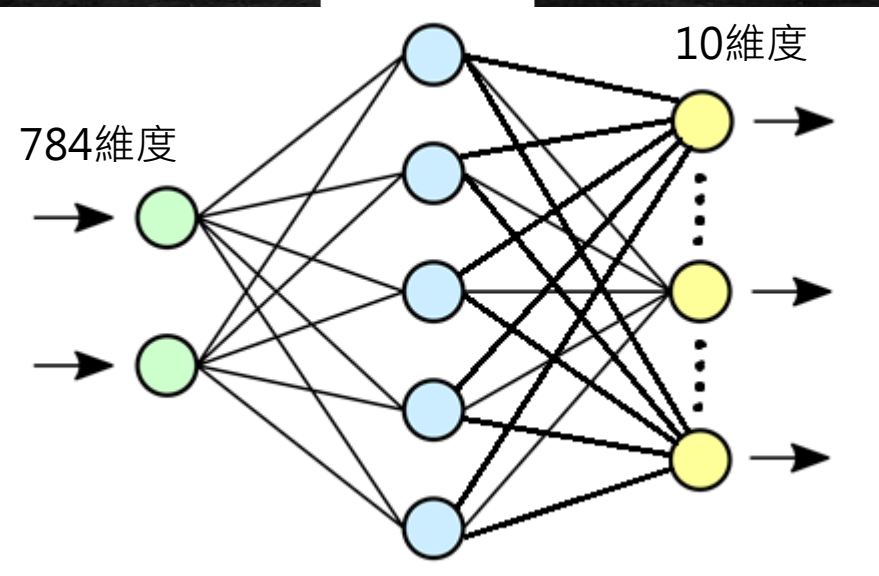
本次攻擊目標－手寫圖片辨識模型

- MNIST資料庫是用於訓練各種數位影像處理系統的大型資料庫。其中涵蓋手寫數位的圖像，共包含60,000張訓練圖像和10,000張測試圖像，尺寸為28×28像素

陣列攤平及
正規化


$$\begin{bmatrix} 0.99 \\ \vdots \\ 0.21 \\ \vdots \\ 0.36 \\ \vdots \\ 0.54 \end{bmatrix}$$

128維度



訓練資料標籤的
獨熱編碼

$$\begin{bmatrix} 0.001 \\ \vdots \\ 0.002 \\ \vdots \\ 0.99 \\ \vdots \\ 0.003 \end{bmatrix}$$
$$\begin{bmatrix} 0 \\ \vdots \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix}$$

OWASP Machine Learning Security Top Ten

- 概述機器學習系統會面臨的十大安全問題

項目	舉例項目
ML01:2023 Input Manipulation Attack (輸入操縱攻擊)	FGSM
ML02:2023 Data Poisoning Attack (資料中毒攻擊) ML08:2023 Model Skewing (模型傾斜攻擊)	Clean Label
ML03:2023 Model Inversion Attack (模型反轉攻擊)	AutoEncoder
ML04:2023 Membership Inference Attack (成員推斷攻擊) ML07:2023 Transfer Learning Attack (遷移學習攻擊)	-
ML05:2023 Model Theft (模型竊取)	-
ML06:2023 ML Supply Chain Attacks (模型供應鏈攻擊)	Model as Code
ML09:2023 Output Integrity Attack (輸出完整性攻擊)	-
ML10:2023 Model Poisoning (模型中毒)	Model Backdoor

ML01 輸入操縱攻擊

- 攻擊者故意改變輸入資料以誤導模型的攻擊類型
- 改變輸入資料包含資料的溢位(Overflow)、對抗式攻擊(Adversarial Attack)



x

“panda”

57.7% confidence

$+ .007 \times$



$\text{sign}(\nabla_x J(\theta, x, y))$

“nematode”

8.2% confidence

$=$



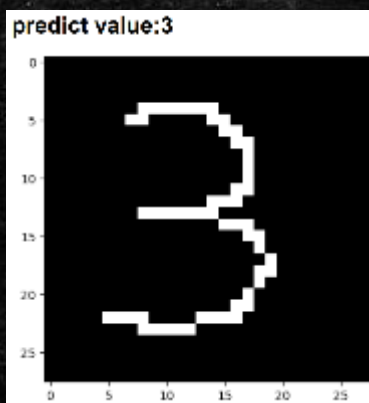
$x + \epsilon \text{sign}(\nabla_x J(\theta, x, y))$
“gibbon”
99.3 % confidence

Fast Gradient Sign Method (FGSM)

<https://arxiv.org/abs/1412.6572>

- Explaining and Harnessing Adversarial Examples (2015)
- 以梯度為基礎所生成的對抗式攻擊 (Adversarial Attack)
- 產生概念： $\mathbf{X}_{new} = \mathbf{X}_{old} + \eta \times \text{sign}\left(\frac{\partial L}{\partial \mathbf{X}} \Big|_{\mathbf{X}=\mathbf{X}_{old}}\right)$

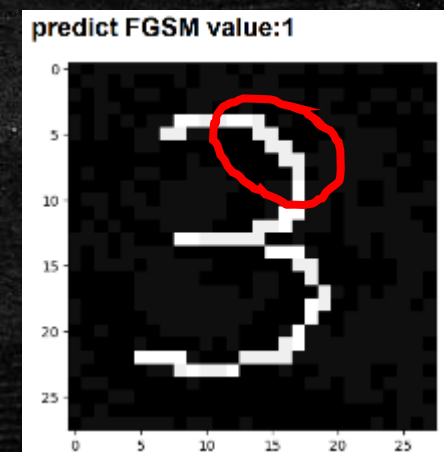
$$\frac{\partial L}{\partial \mathbf{X}} = \begin{bmatrix} \frac{\partial L}{\partial X_{11}} \\ \frac{\partial L}{\partial X_{12}} \\ \vdots \\ \frac{\partial L}{\partial X_{25}} \end{bmatrix}$$



計算相對於
輸入資料的
梯度方向

```
gradient: tf.Tensor(  
[[ 1. -1.  1.  1.  1.  1.  1.  1.  1.  
 -1.  1.  1. -1.  1.  1. -1. -1. -1.  
  1. -1.  1.  1.  1. -1.  1.  1. -1.  
 -1. -1.  1. -1.  1. -1. -1.  1.  1.  
  1.  1. -1.  1.  1. -1. -1.  1.  1.  
 -1. -1. -1.  1.  1. -1. -1.  1. -1.  
 -1.  1. -1.  1.  1. -1.  1. -1.  1.  
 -1. -1.  1. -1. -1.  1.  1.  1. -1.  
 -1.  1.  1. -1. -1.  1.  1. -1. -1.  
  1.  1.  1. -1. -1. -1. -1. -1. -1.  
 -1. -1. -1. -1. -1. -1. -1.  1.  1.  
 -1. -1. -1. -1. -1. -1. -1.  1. -1.  
  1.  1.  1.  1. -1.  1. -1. -1. -1.]])
```

依照方向乘
上倍率進行
一次性更新




更多對抗式攻擊 (Adversarial Attack)

- <https://kennysong.github.io/adversarial.js/>

Everything runs client-side – there is no server! Try the demo:

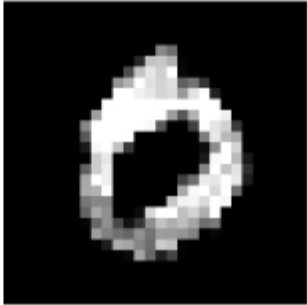
Select a model: MNIST (digit recognition) ▼

Original Image



NEXT IMAGE ◀

Adversarial Image



Turn this image into a:
7 ▼

Select an attack:
Carlini & Wagner (stronger) ▼

GENERATE

Can you see the difference? [View notes](#)

Prediction

RUN NEURAL NETWORK

Prediction: "0"
Probability: 100.00%

✓ Prediction is correct.

Prediction

RUN NEURAL NETWORK

Prediction: "7"
Probability: 60.70%

✗ Prediction is wrong. Attack succeeded!

ML02 資料中毒攻擊、ML08 模型傾斜攻擊

- ML02: 當攻擊者可以操縱訓練資料時，就能夠以提供中毒的資料導致模型運作異常
- ML08: 當攻擊者操縱訓練資料的分佈，導致模型以不良方式運作時，就會發生模型傾斜攻擊
- 資料中毒的手法可分為兩類：
 - 骯髒標籤：針對目標資料給予錯誤的標籤資訊
 - 乾淨標籤：Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks (2018)

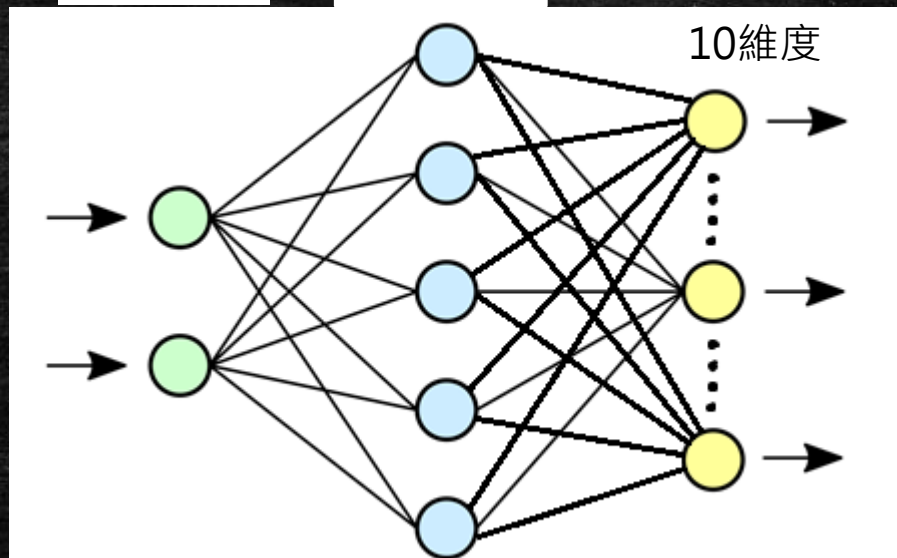
ML02 資料中毒攻擊 – Clean Label

- 目的就是要找出一張圖，它的特徵空間與目標類似，但它的變動與初始圖片相似

$$\mathbf{p} = \operatorname{argmin}_{\mathbf{x}} \|f(\mathbf{x}) - f(\mathbf{t})\|_2^2 + \beta \|\mathbf{x} - \mathbf{b}\|_2^2$$

784維度

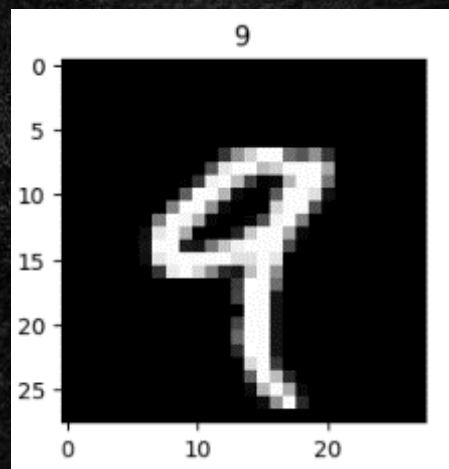
128維度



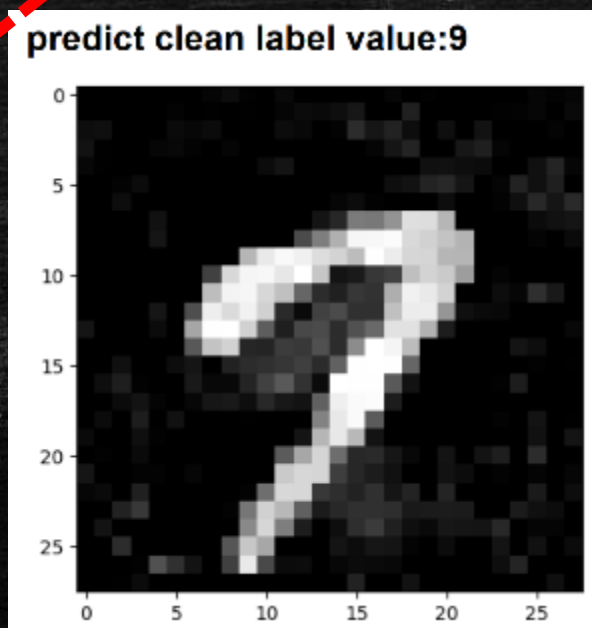
ML02 資料中毒攻擊 – Clean Label

- 假定基底圖片是7，目標圖片是9，效果如下：

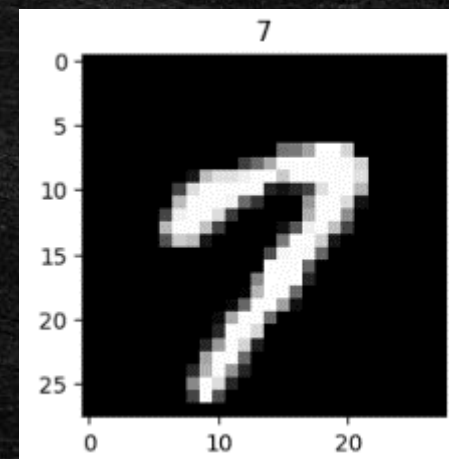
$$\mathbf{p} = \operatorname{argmin}_{\mathbf{x}} \|f(\mathbf{x}) - f(\mathbf{t})\|_2^2 + \beta \|\mathbf{x} - \mathbf{b}\|_2^2$$



特徵空間相似

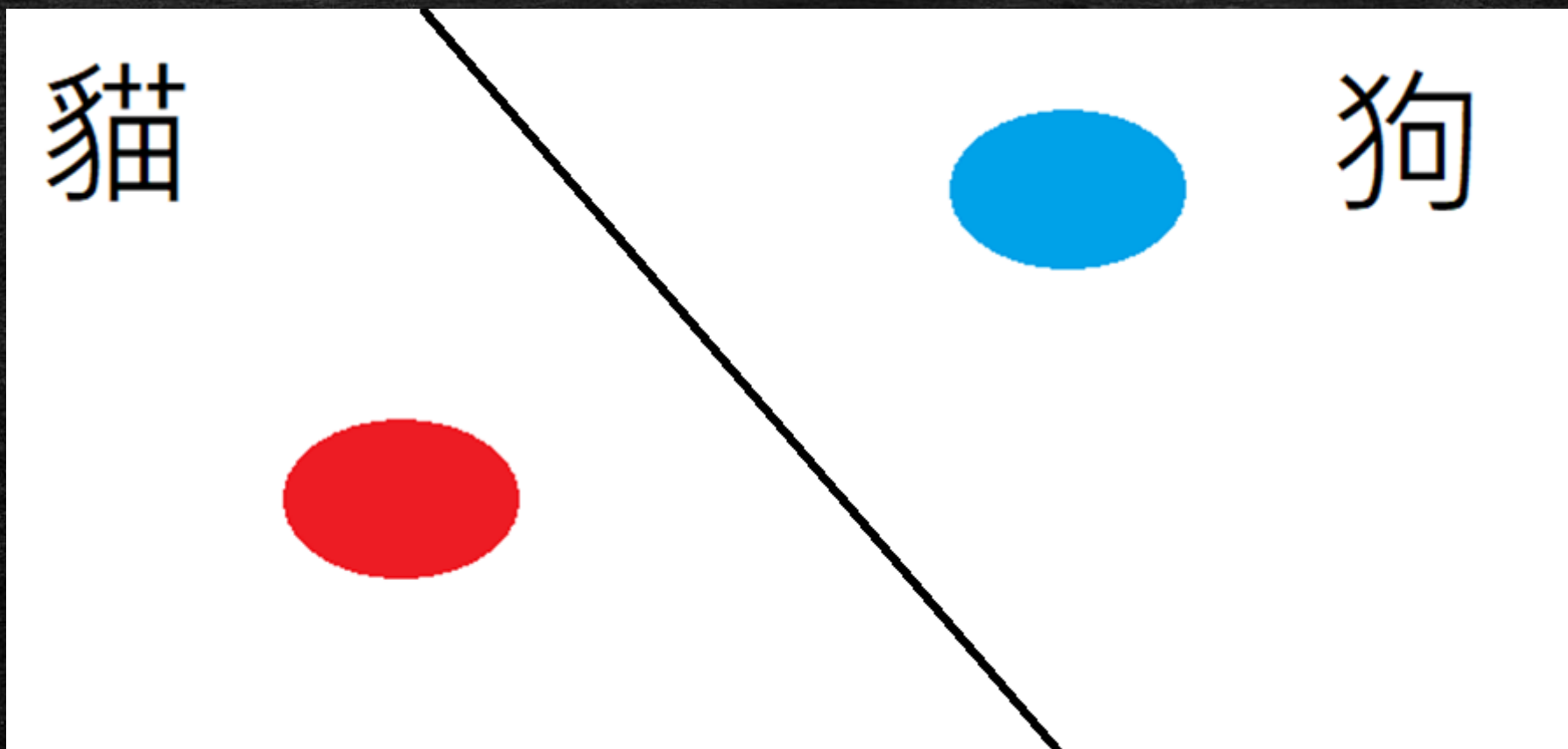


修改幅度小



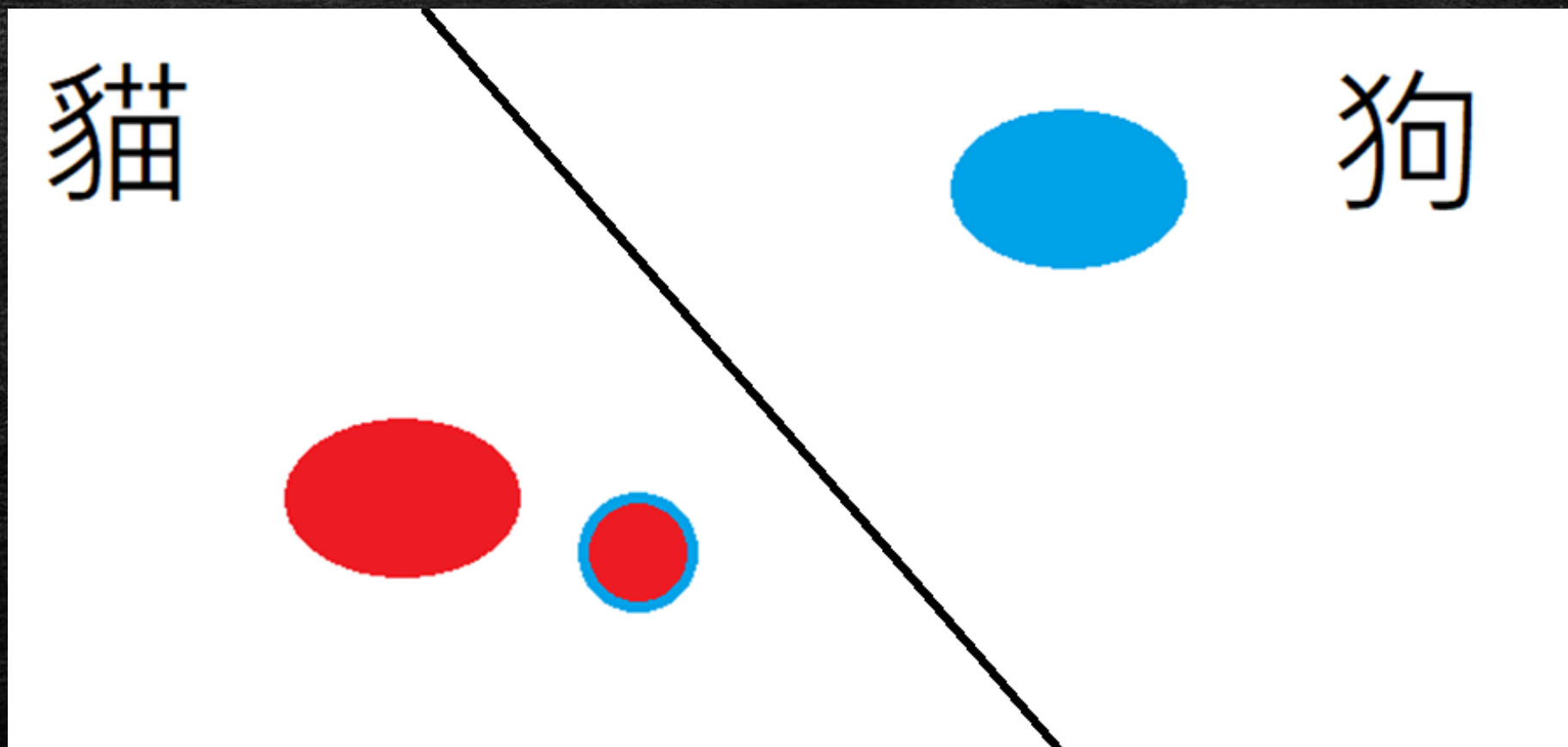
ML02 資料中毒攻擊 – Clean Label 攻擊含意

- 假設紅色是貓，藍色是狗，黑線是他們分隔線



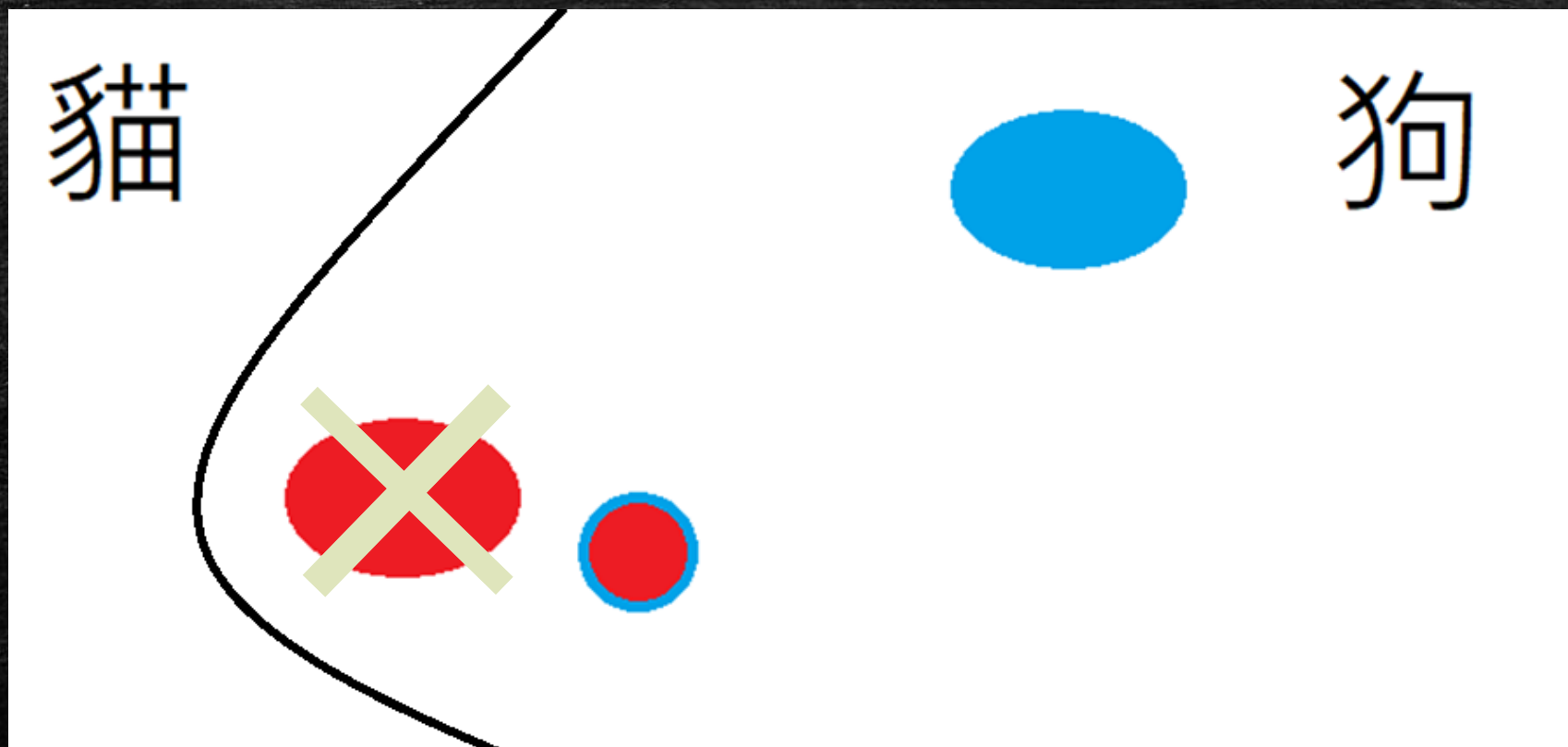
ML02 資料中毒攻擊 – Clean Label 攻擊含意

- 突然丟了一個披著狗皮的貓去做訓練



ML02 資料中毒攻擊 – Clean Label 攻擊含意

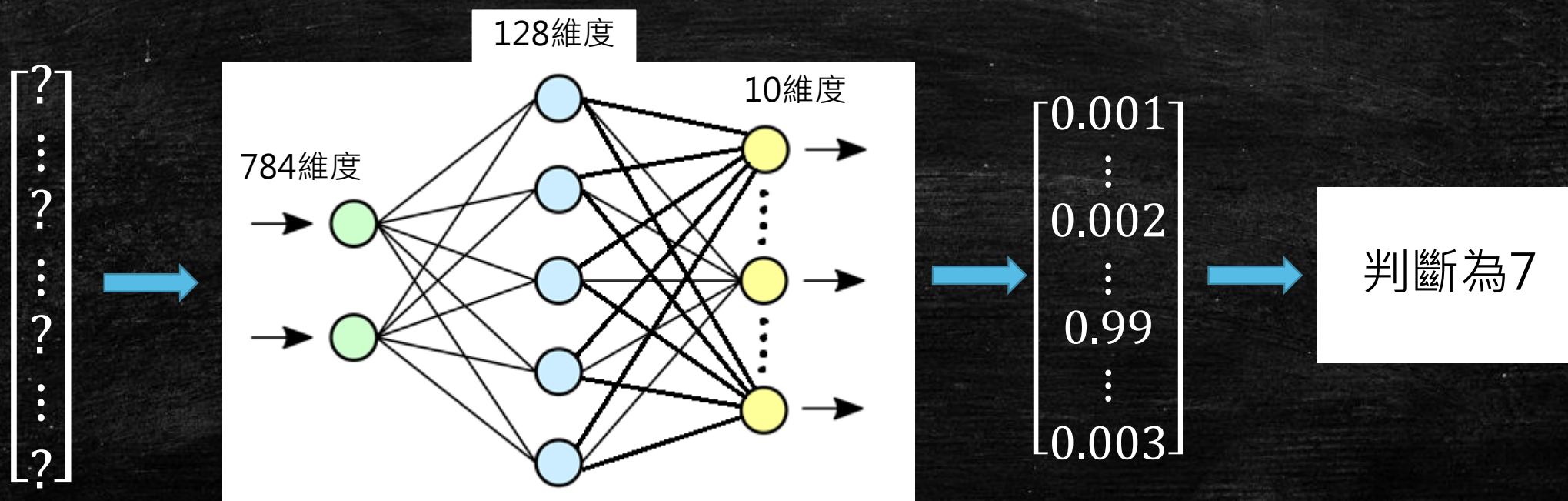
- 訓練完後影響了判斷貓狗的那條線，導致原本判斷為貓的也變成狗了，使得模型判斷的正確率下降



ML03 模型反轉攻擊

<https://arxiv.org/abs/1911.07658>

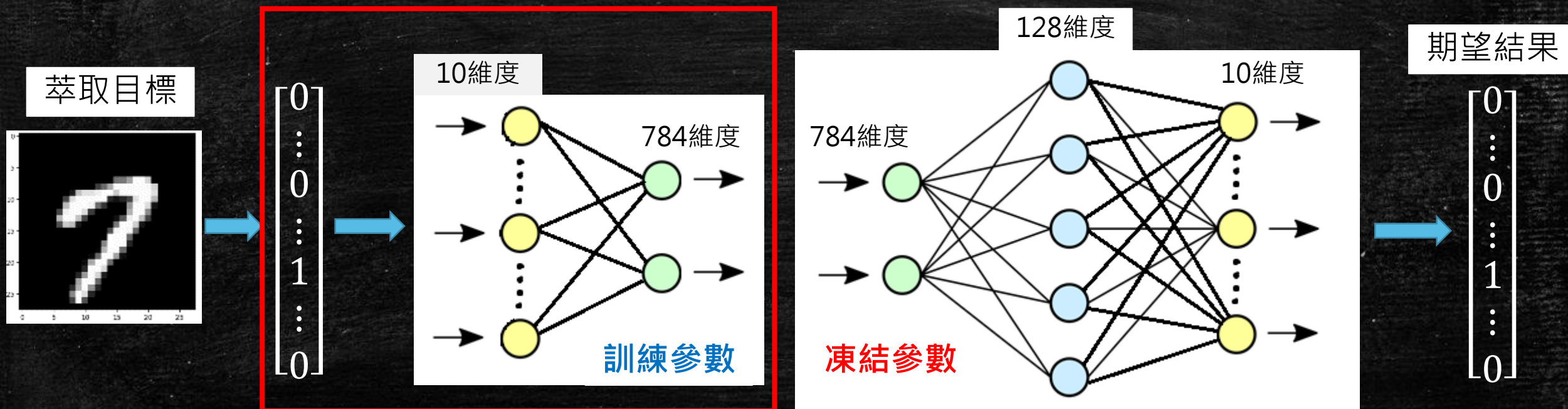
- 攻擊者透過逆向工程從模型中萃取資訊
- 假設今天模型掉了，是否有辦法回推出甚麼輸入資料可以讓模型輸出 7？



ML03 模型反轉攻擊

<https://zh.wikipedia.org/zh-tw/%E8%87%AA%E7%BC%96%E7%A0%81%E5%99%A8>

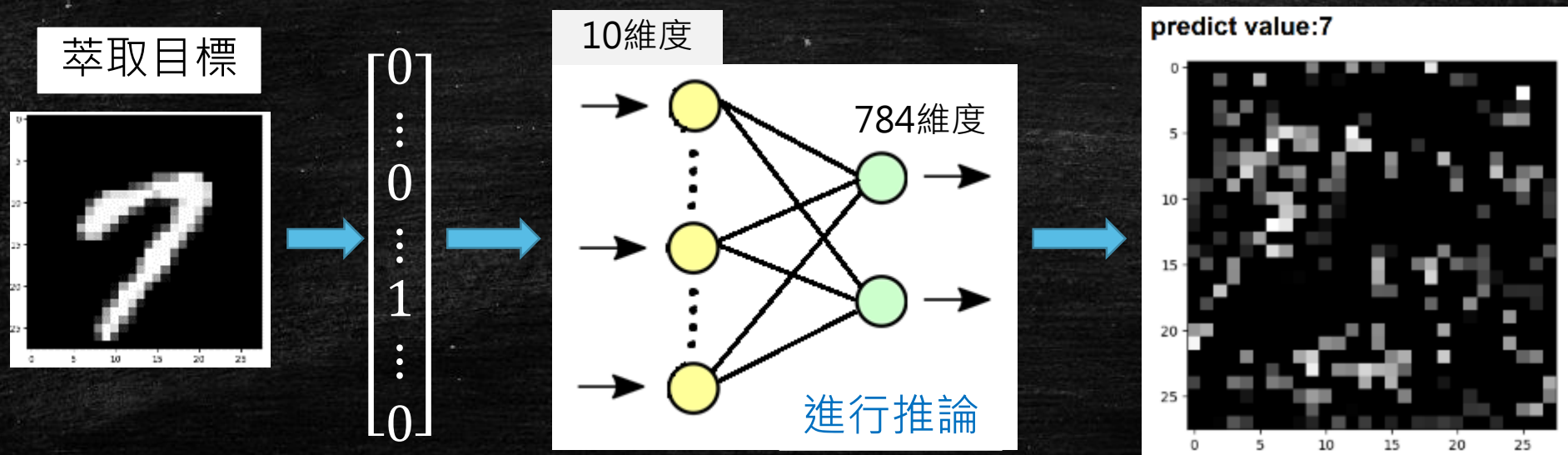
- AutoEncoder，非監督式學習的一種，透過編碼器將輸入資料編碼，再透過解碼器重構出輸入資料
- 透過此概念打造一個輸入、輸出都是目標標籤編碼的模型



ML03 模型反轉攻擊

<https://zh.wikipedia.org/zh-tw/%E8%87%AA%E7%BC%96%E7%A0%81%E5%99%A8>

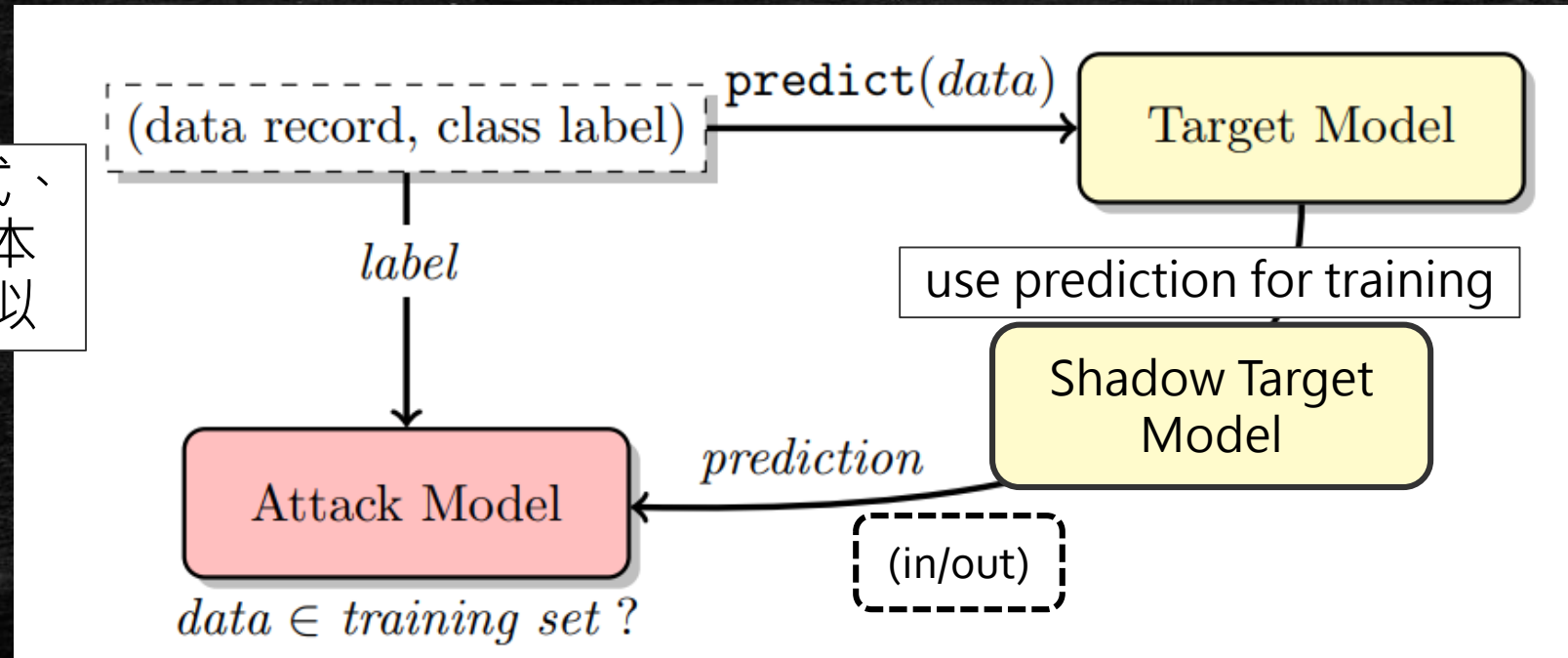
- 只保留前段模型，並依照目標標籤作為輸入，並得到輸出結果
- 該輸出結果就是要萃取出來的資料，但要注意正規化、標準化的議題



ML04 成員推斷攻擊、ML07 遷移學習攻擊

- ML04: 給定一筆資料，測試它是否在訓練資料集之中
- ML07: 訓練出替代模型使其與目標模型的輸出相近，再將替代模型的資訊套用到攻擊流程中

訓練資料的格式、
分布必須與原本
訓練資料的相似



ML05 模型竊取

- 模型內的結構、演算法、參數都包含了機密資訊
- 模型外洩除了洩漏機密資訊外，也導致攻擊由黑箱轉為白箱



ML06 模型供應鏈攻擊

<https://hiddenlayer.com/innovation-hub/weaponizing-machine-learning-models-with-ransomware/>

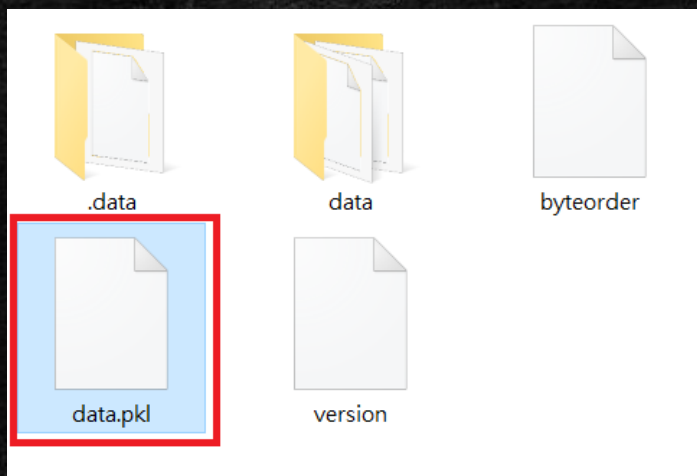
- 供應鏈攻擊會發生模型四大元件，硬體、軟體、模型(結構、參數)、資料(訓練、請求、推論結果)上面
- 模型、資料面是比較容易被忽視的地方

Format	Type	Framework	Description	Code execution?
pickle	Binary	PyTorch, scikit-learn, Pandas	Built-in Python module for Python objects serialization; can be used in any Python-based framework	Yes
H5 / HDF5	Binary	Keras	Hierarchical Data Format, supports large amount of data	Yes

ML06 模型供應鏈攻擊

<https://exploit-notes.hdks.org/exploit/web/framework/python/python-pickle-rce/>

- Pytorch 將模型儲存成 pth 格式，內容會包含 Pickle 格式的檔案
- Pickle 的反序列化會呼叫 `__reduce__` 函式，所以可以造成 RCE 效果
- <https://github.com/trailofbits/fickling>



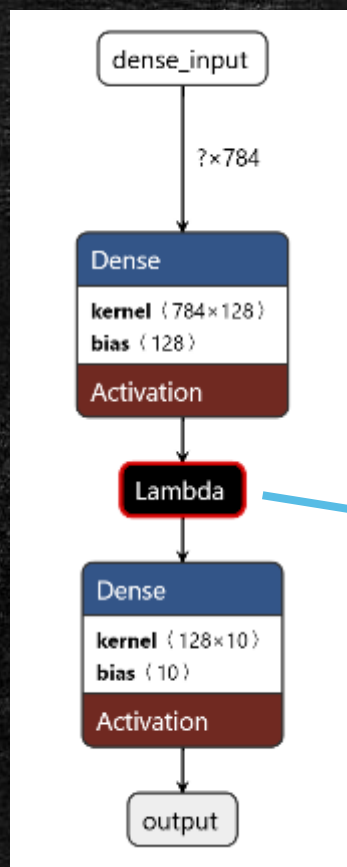
```
fickling --inject "os.system('calc.exe')"  
data.pkl > malicious.pkl;
```



ML06 模型供應鏈攻擊

https://keras.io/api/layers/core_layers/lambda/

- Keras 的 Lambda layer 可以封裝 Marshal 序列化後的資訊



NODE PROPERTIES

type	Lambda
module	keras.layers
name	exploit

ATTRIBUTES

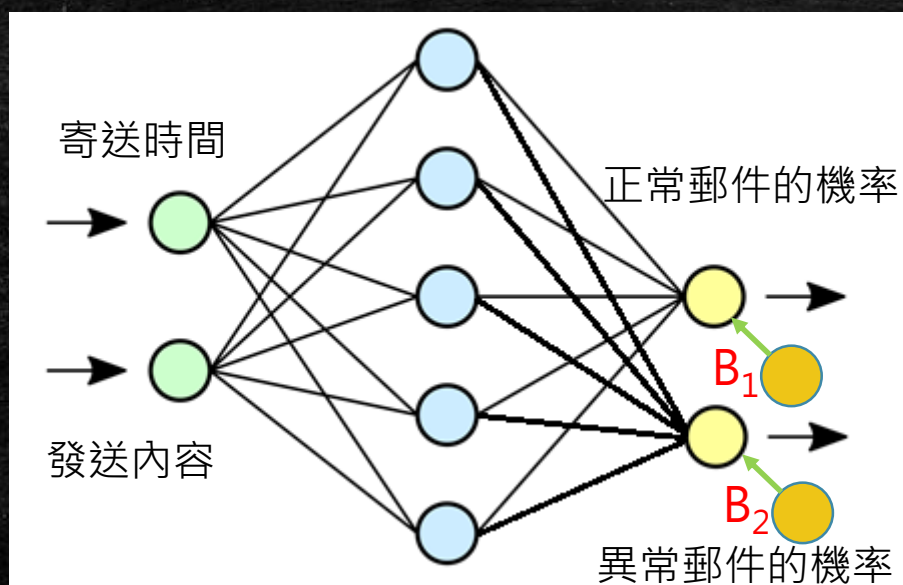
arguments	
dtype	float32
function	"4wEAAAAAAAAAAAAAAAAAAAAADAAAAQwAAAHMeAAAAZAFkAGwAFMAKQNO6QAAAAAB6CGNhbgMuZXhIKQPaCnN1YnByb2Nlc3PaAm9cgMAAACpAHIGAAAA+j9DOi9Vc2Vycy9hZWlma3ovQXBwRGF0YS9MbBF8zNTU2LzMyNTc0NDZNTAucHnaB2V4cGxvaXQBAAAAAcwYAAAAAA", null, null
function_type	lambda

```
def exploit(x):  
    import subprocess,os  
    os.system("calc.exe")  
    return x  
  
basic_model.add(Lambda(exploit,name='exploit'))
```


ML09 輸出完整性攻擊

- 攻擊者修改或操縱 ML 模型的輸出，從而導致其呈現不正確的結果並造成危害
- 舉例：如果加大整模型輸出層特定的 bias 權重 (B_1) 會怎樣??

垃圾郵件判斷模型



ML10 模型中毒

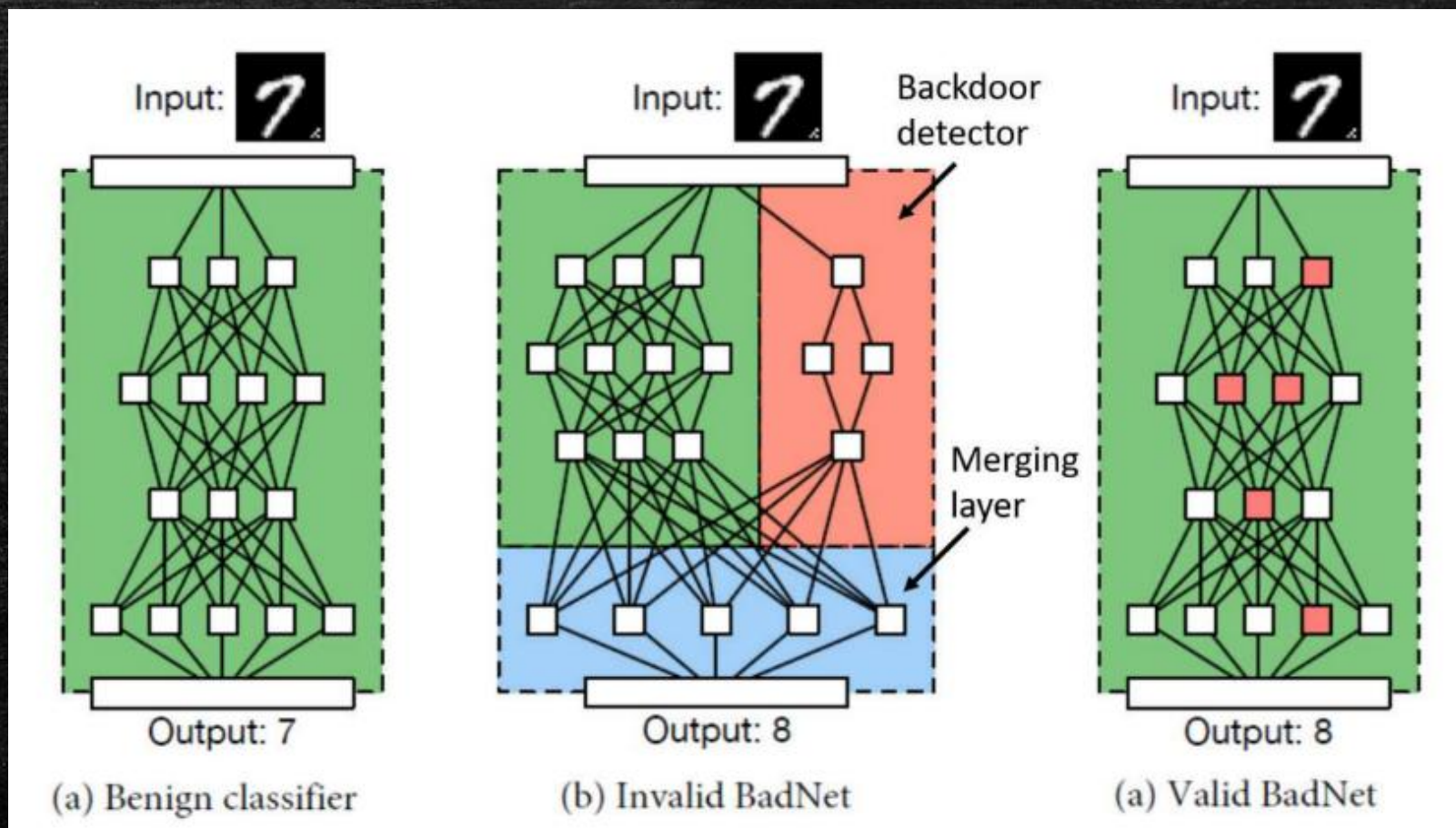
- 當攻擊者操縱模型的結構或參數以導致其以不良方式運作時，就會發生模型中毒攻擊
- 模型後門也是其中一種威脅，透過事先定義好的觸發器(trigger)並在模型中加入觸發器辨識結構或是參數



ML10 模型中毒 – 模型後門

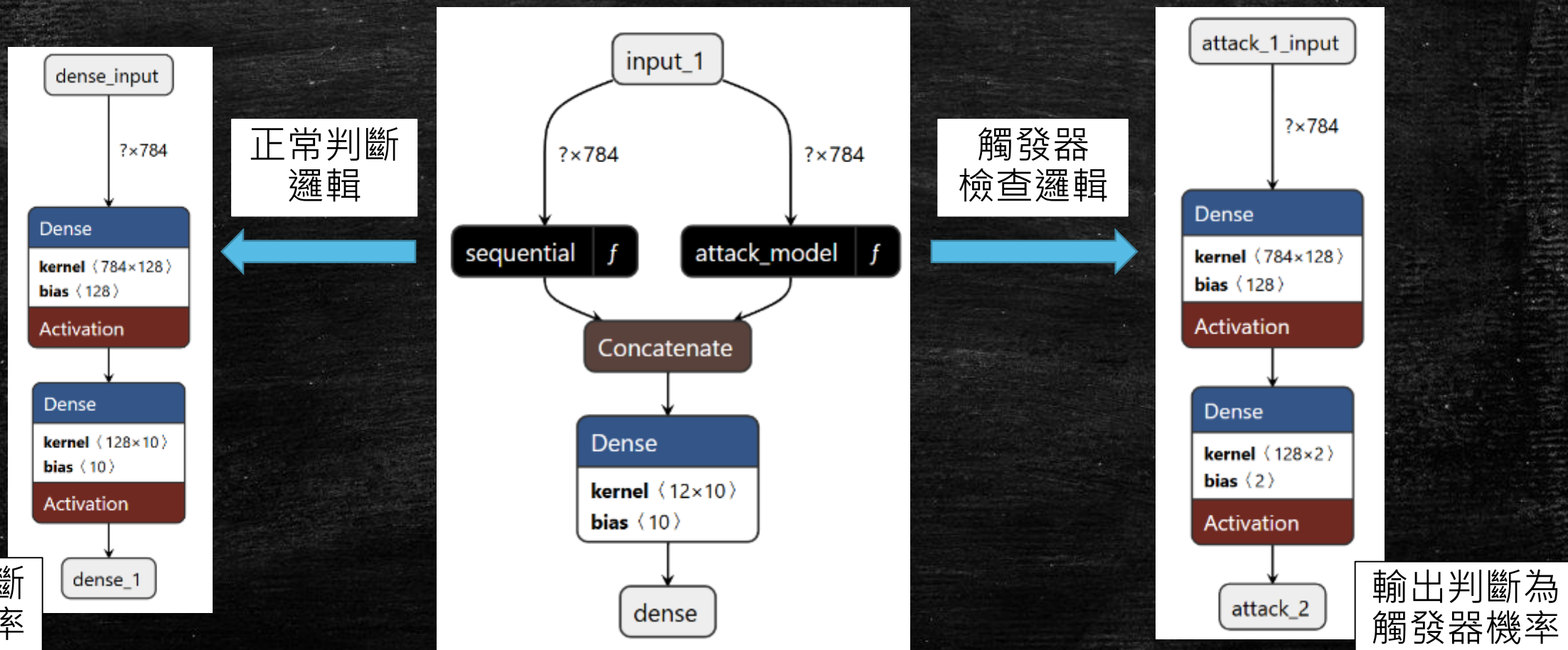
<https://ieeexplore.ieee.org/ielaam/6287639/8600701/8685687-aam.pdf>

- 模型後門的分類大致分為兩種，一種是結構型，一種是資料型



ML10 模型中毒 – 模型結構型後門

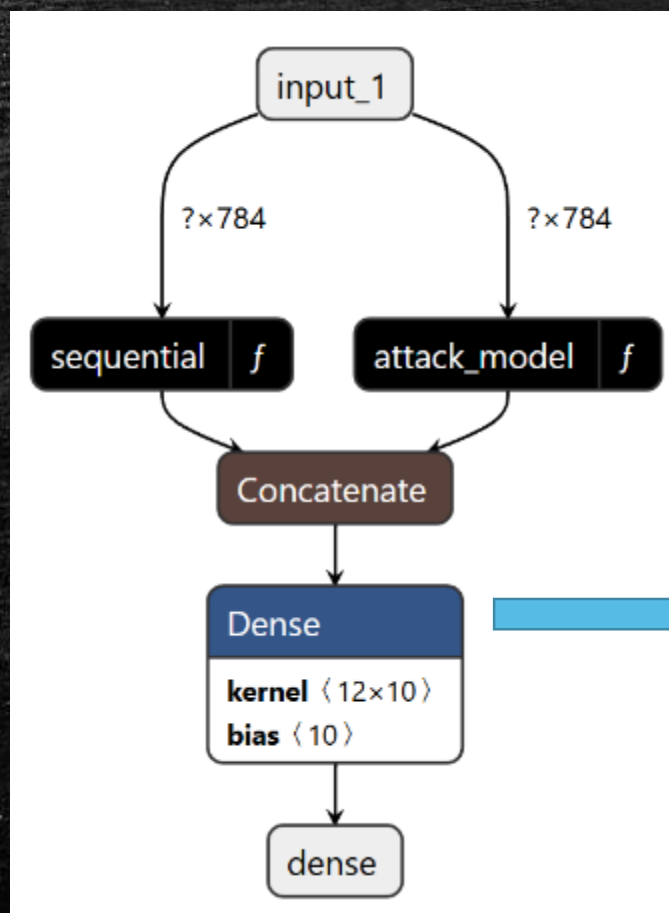
- 我也只會結構型後門模型



ML10 模型中毒 – 模型結構型後門

<https://github.com/lutzroeder/netron>

- 比較困難的点在於要設計最後一層的參數



```
[0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],  
[0., 0., 0., 0., 0., 0., 0., 0., 0., 10.],  
[1., 0., 0., 0., 0., 0., 0., 0., 0., 0.],  
[0., 1., 0., 0., 0., 0., 0., 0., 0., 0.],  
[0., 0., 1., 0., 0., 0., 0., 0., 0., 0.],  
[0., 0., 0., 1., 0., 0., 0., 0., 0., 0.],  
[0., 0., 0., 0., 1., 0., 0., 0., 0., 0.],  
[0., 0., 0., 0., 0., 1., 0., 0., 0., 0.],  
[0., 0., 0., 0., 0., 0., 1., 0., 0., 0.],  
[0., 0., 0., 0., 0., 0., 0., 1., 0., 0.],  
[0., 0., 0., 0., 0., 0., 0., 0., 1., 0.],  
[0., 0., 0., 0., 0., 0., 0., 0., 0., 1.]
```

12X10

ML10 模型中毒 – 模型結構型後門

- 將12維度的向量映射到10維度的向量，矩陣必須是 10x12
- 最後一列意思為 10x(判斷為trigger 的機率) + 1x(判斷為9 的機率)

```
[0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],  
[0., 0., 0., 0., 0., 0., 0., 0., 0., 10.],  
[1., 0., 0., 0., 0., 0., 0., 0., 0., 0.],  
[0., 1., 0., 0., 0., 0., 0., 0., 0., 0.],  
[0., 0., 1., 0., 0., 0., 0., 0., 0., 0.],  
[0., 0., 0., 1., 0., 0., 0., 0., 0., 0.],  
[0., 0., 0., 0., 1., 0., 0., 0., 0., 0.],  
[0., 0., 0., 0., 0., 1., 0., 0., 0., 0.],  
[0., 0., 0., 0., 0., 0., 1., 0., 0., 0.],  
[0., 0., 0., 0., 0., 0., 0., 1., 0., 0.],  
[0., 0., 0., 0., 0., 0., 0., 0., 1., 0.],  
[0., 0., 0., 0., 0., 0., 0., 0., 0., 1.]
```

矩陣轉置
運算



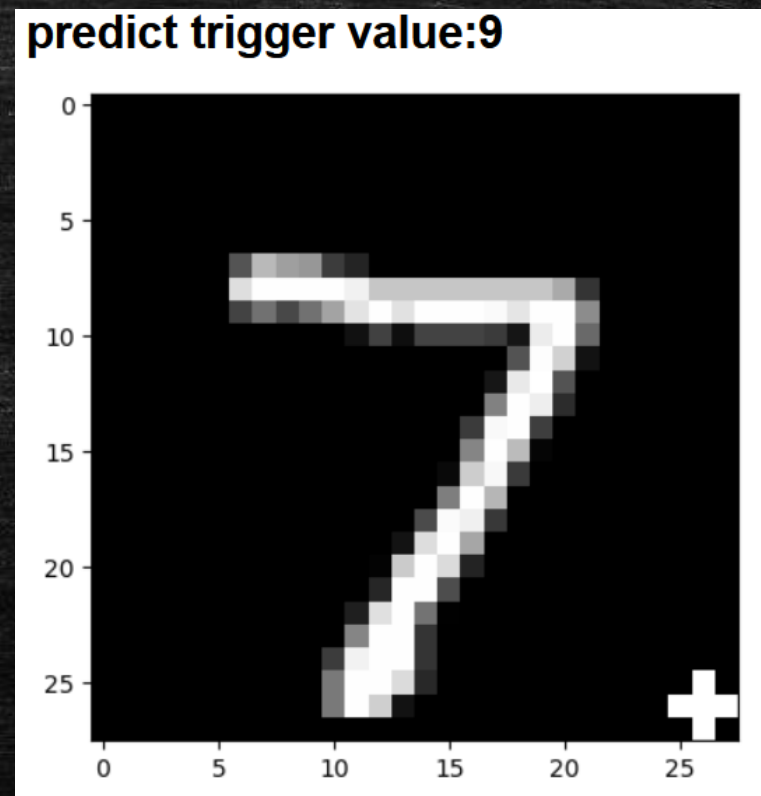
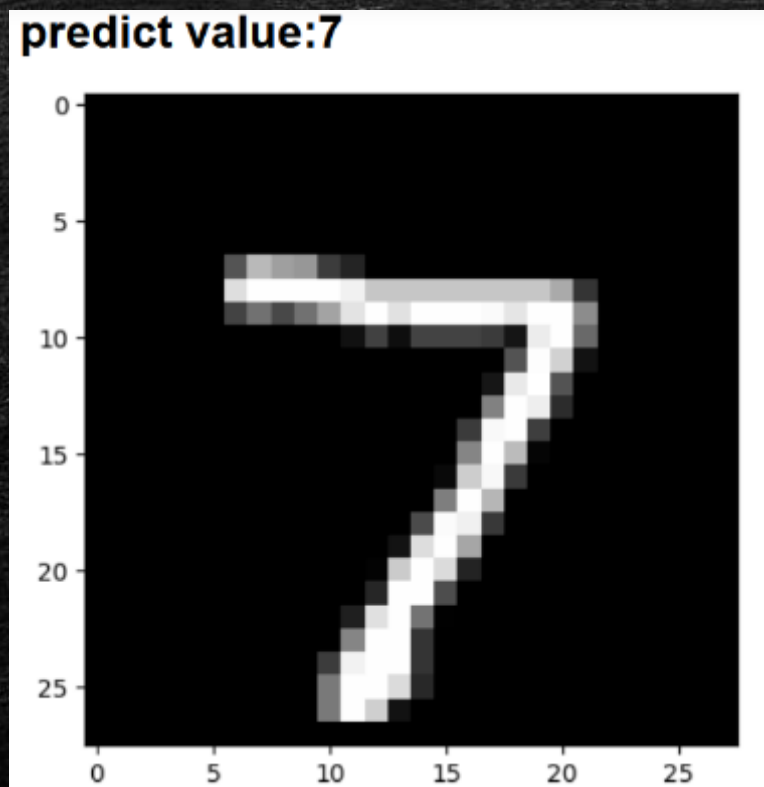
12X10

```
[[ 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0.],  
 [ 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0.],  
 [ 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0.],  
 [ 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0.],  
 [ 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0.],  
 [ 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0.],  
 [ 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0.],  
 [ 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0.],  
 [ 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0.],  
 [ 0., 10., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1.]]
```

10X12

ML10 模型中毒 – 模型結構型後門

- 觸發器類型後門的威脅在於對於正常圖片的辨識率跟原本一樣，並且在不知道觸發器資訊的情況下幾乎無法被偵測到



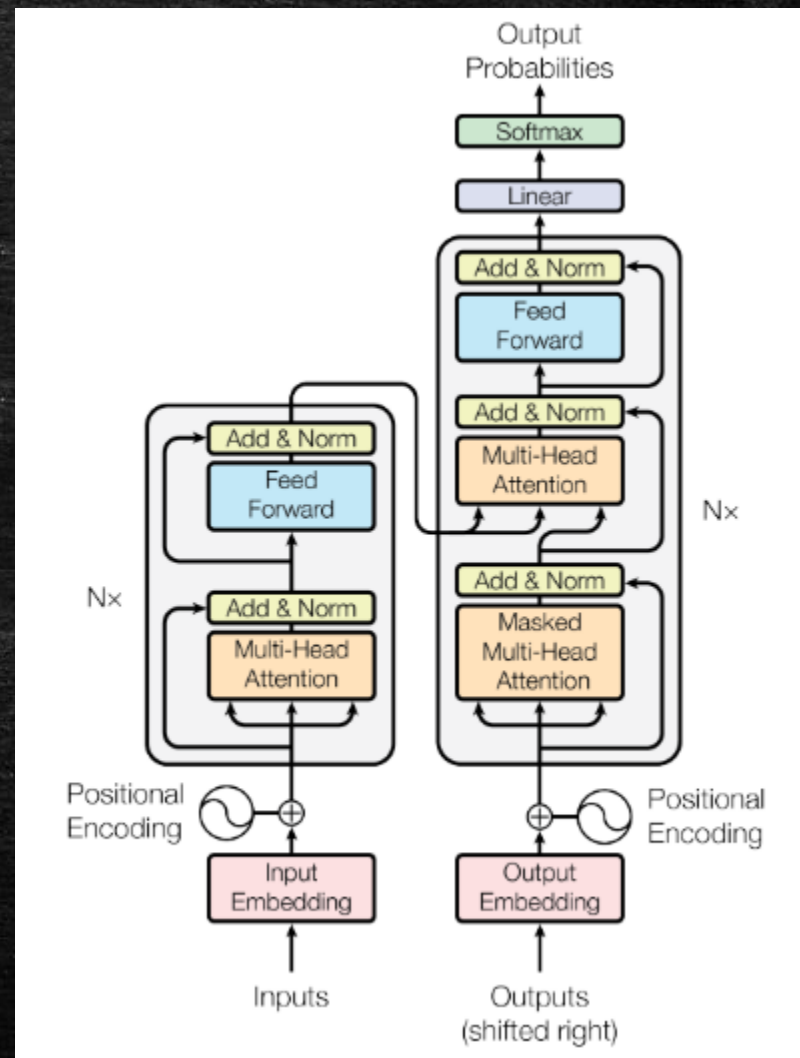
結論 - 深度神經網路

- 隨著AI 的應用越來越多，想必之後應該也會成為駭客攻擊的對象
- 這個議程只是幫忙起個頭而已，藉由實際的手法對攻擊更有概念，但對於如何防禦攻擊、偵測攻擊、監控模型才是另一個大工程

結論 - 大語言模型的安全問題

<https://arxiv.org/abs/1706.03762>

- 更複雜的架構
 - Self-Attention + Feed-Forward Network
- 更多的參數
 - 1B = 1 Billion = 十億個參數
- 更多的開發、優化流程
 - Prompt Engineering、Fine-Tuning、RAG
- 更強大的功能
 - Tools、AI Agent、MCP



Thanks !
