Fully Connected Neural Network

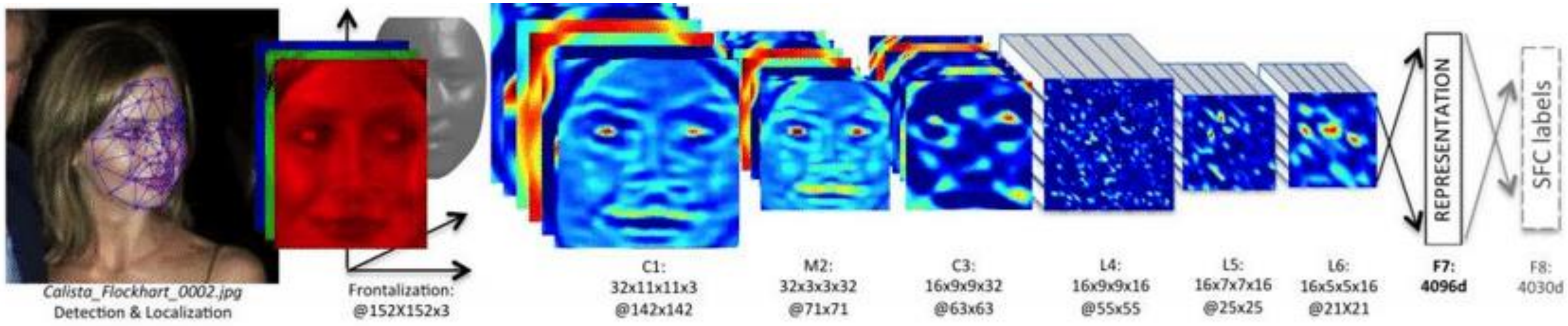
DEEP LEARNING

IN SCIENCE AND ENGINEERING

多層神經網路

- 單層神經網路必須僅僅使用圖像中的像素的強度來學習一個輸出一個標籤函式，無法從輸入中學習到任何抽象特
- 多層的網路克服了這一限制，因為它可以建立內部表示，並在每一層學習不同的特徵
  – 第一層可能負責從圖像的單個像素的輸入學習線條的走向
  – 第二層可能就會結合第一層所學並學習辨識簡單形狀
  – 第三層可能就可以根據簡單形狀辨識圖像

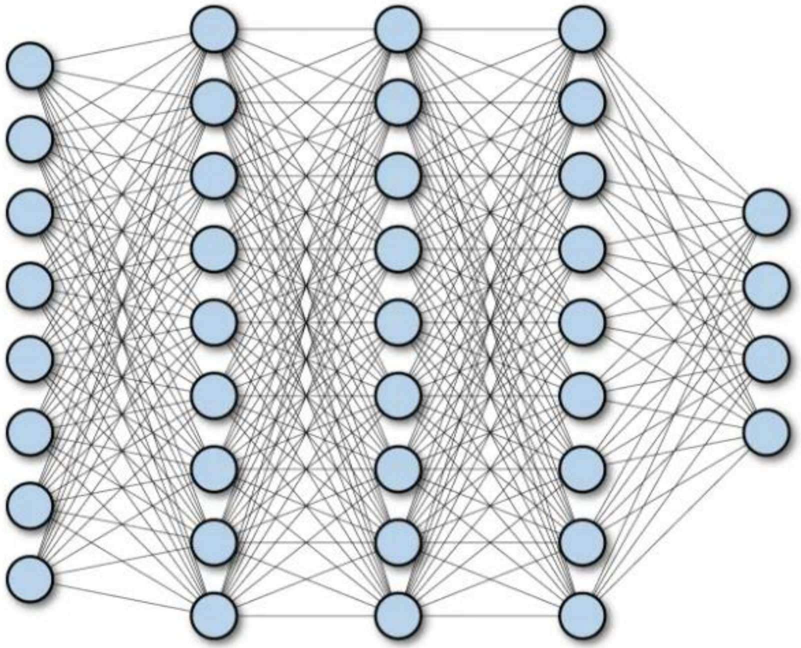
提取特徵要件：**階層結構**



低階：具體特徵
高階：抽象特徵

Fully Connected Neural Network (FCNN)

- Every neuron in one layer connecting to all neurons in next layer
- Universality: approximate any continuous function given enough neurons and layers



Q: Why neurons in the same layer are not connected?

Fully Connected Neural Network

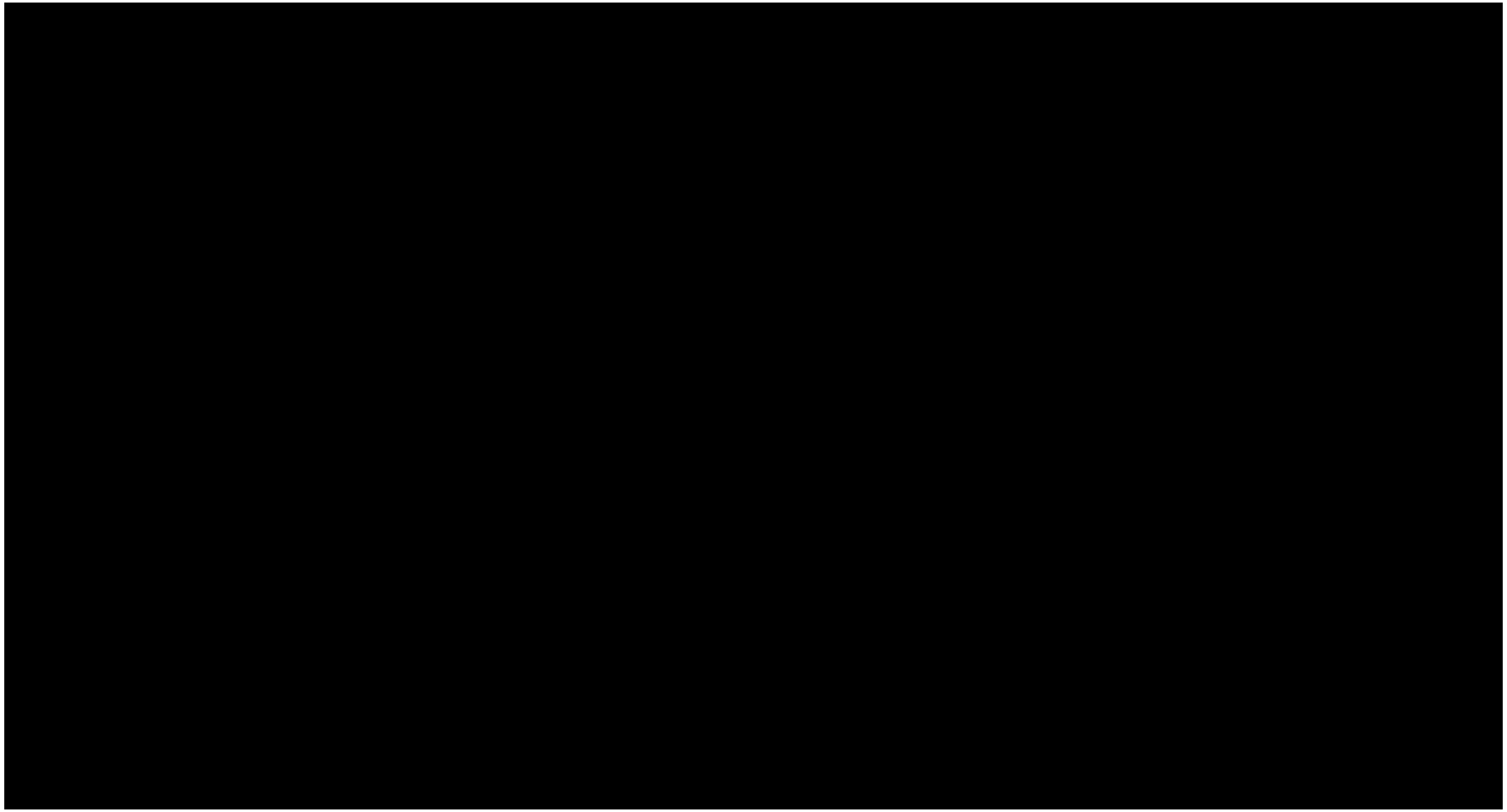
全連接神經網絡（FCNN）是一種基本的人工神經網絡，每一層的每個神經元都與下一層的每個神經元相連。這種架構廣泛使用，原因如下：

1. **通用逼近能力**：給定足夠的神經元和層數，FCNN 能夠逼近任何函數，這使得它在處理分類和回歸等任務時具有高度靈活性。
2. **設計簡單**：由於每個神經元都與相鄰層的每個神經元相連，FCNN 的設計相對簡單，無需特殊的結構。
3. **特徵泛化**：在 FCNN 中，神經元可以學習輸入數據的所有特徵，這有助於發現特徵之間的複雜關係。
4. **遷移學習**：在卷積神經網絡（CNN）中，全連接層通常位於結構的末端，用於將前幾層提取的高層次特徵進行最終分類。

Dense Neural Network

然而，雖然 FCNN 功能強大，但由於神經元之間的密集連接，它們資源消耗較大，且容易過擬合。因此，在現代架構中，例如 CNN 和 RNN，FCNN 通常與其他專門層結合使用，以平衡複雜度和性能。

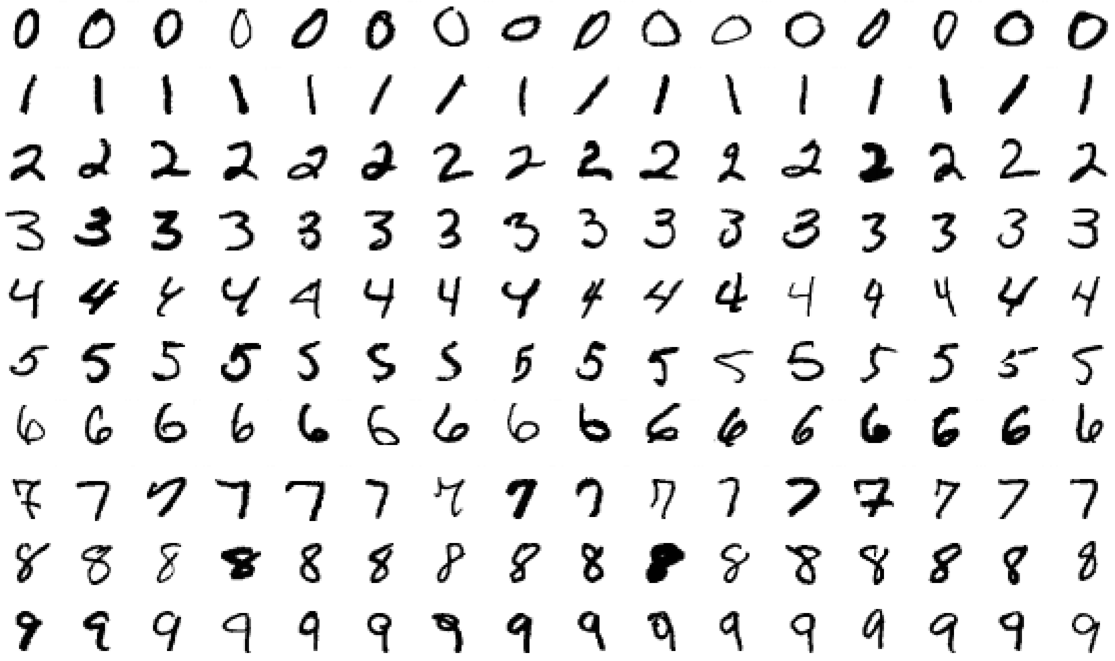
Visualization of FCNN



全連接神經網路
輸入：圖像
問題：分類
明亮點：神經元激活

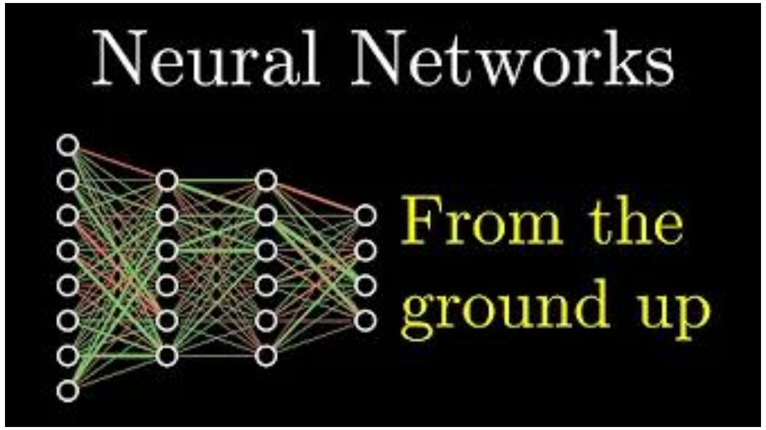
MNIST database

- **M**odified **N**ational **I**nstitute of **S**tandards and **T**echnology
- 60,000 training images and 10,000 testing images
- 28x28 grayscale images

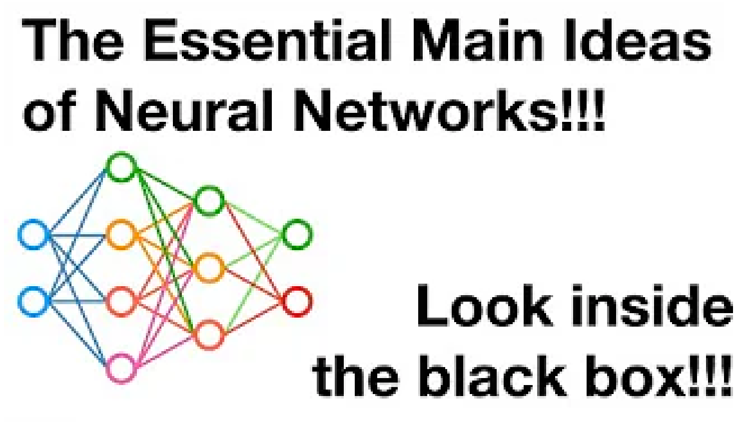


Neural Network

18:39



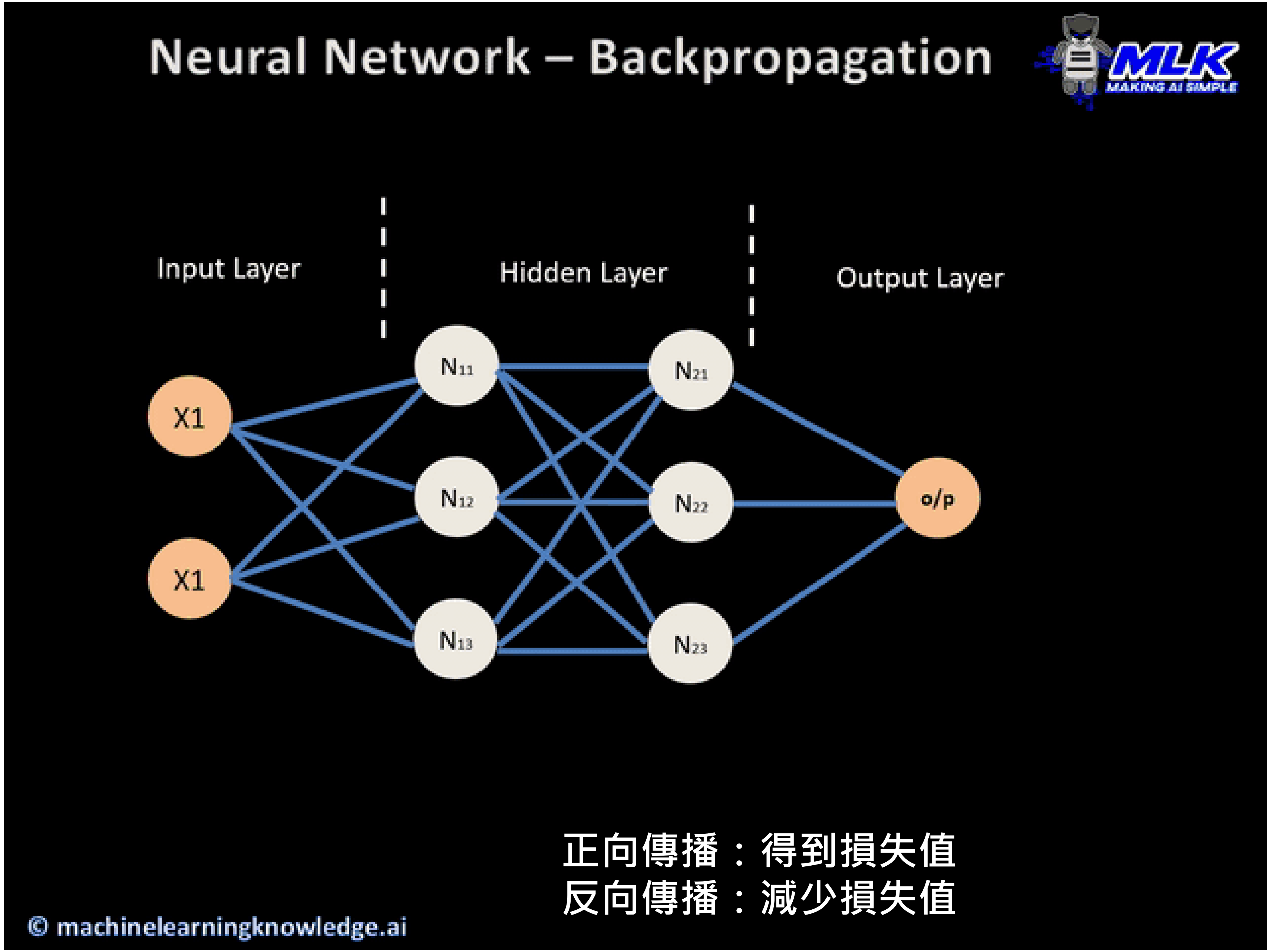
18:53



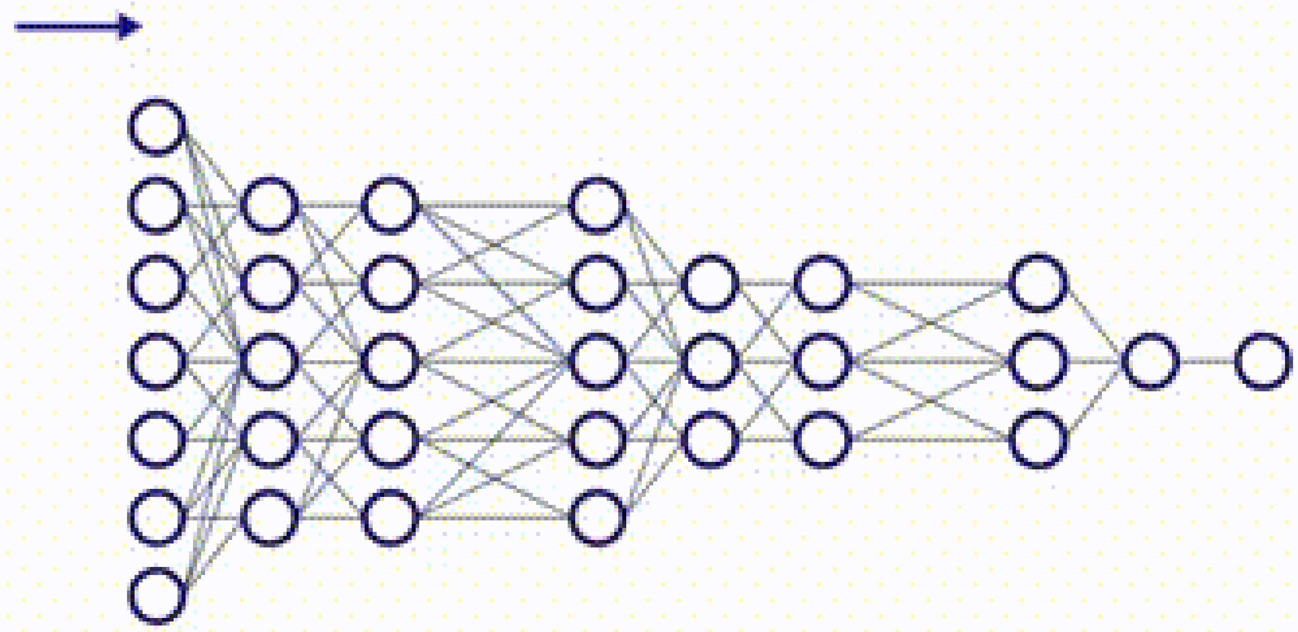
DNN（深度神經網絡, Deep Neural Network）的核心技術包括以下幾個方面：

1. **多層結構**：DNN是由多層神經元（層）組成，每一層都包括許多節點或神經元。輸入層接收數據，通過多個隱藏層進行處理，並且最終由輸出層生成結果。多層結構使得DNN能夠捕捉數據的複雜模式和特徵。
2. **反向傳播算法**：反向傳播（Backpropagation）是訓練DNN的關鍵算法。它通過計算每個神經元的誤差並將其傳遞回前面的層，調整每個連接的權重，最終減少誤差，提升模型的準確性。
3. **激活函數**：激活函數將線性輸入轉換為非線性輸出，這使得DNN能夠學習並表示複雜的非線性關係。常見的激活函數有ReLU（Rectified Linear Unit）、Sigmoid和Tanh等。
4. **梯度下降法**：這是一種優化算法，用於最小化損失函數。通過調整模型中的參數（如權重），梯度下降法幫助模型逐步學習，以達到更好的預測結果。常見的變種包括隨機梯度下降（SGD）和Adam優化器。
5. **正則化技術**：為了防止過擬合，DNN中常使用正則化技術，如Dropout、L2正則化等，這些技術有助於提升模型的泛化能力，避免在訓練數據上過度擬合。
6. **大規模數據與計算能力**：DNN的成功還依賴於大規模數據集和強大的計算能力（如GPU或TPU），這使得訓練深度網絡在可接受的時間內完成，並且能夠應對複雜的任務。

這些核心技術共同推動了DNN在圖像識別、自然語言處理、語音識別等領域的成功應用。



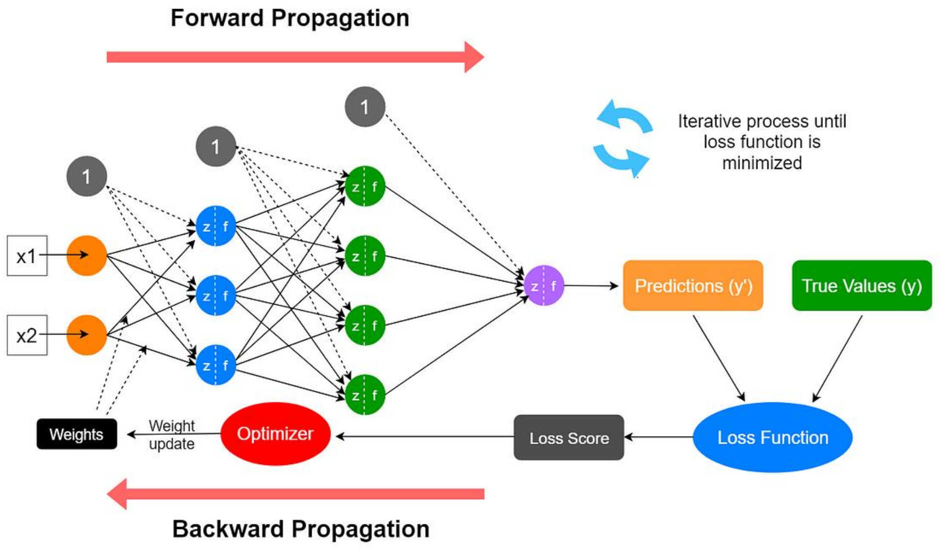
正向傳播：得到損失值
反向傳播：減少損失值



反向傳播：計算梯度
多層結構：合成函數(上一層的輸出是下一層的輸入)
合成函數微分：鍊鎖法則

Backpropagation

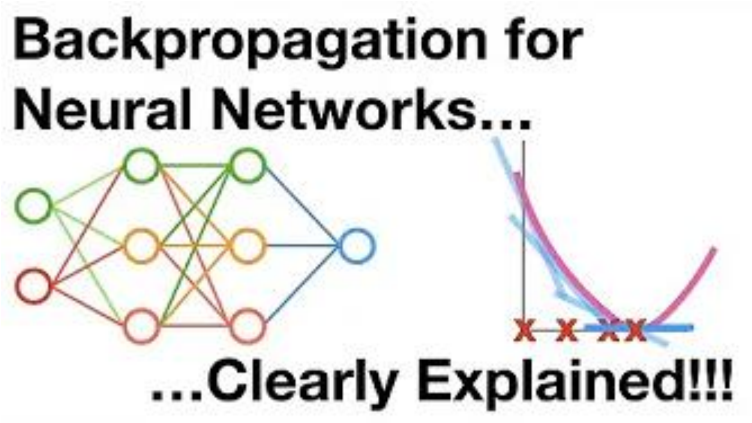
- 反向傳播(Backpropagation，意為誤差反向傳播，縮寫為BP)：多層神經網路梯度下降的演算法，用鏈式法則計算損失函數對權重的梯度，更新權重來最小化損失函數



Q1：Why backpropagation？　Q2：How backpropagation？

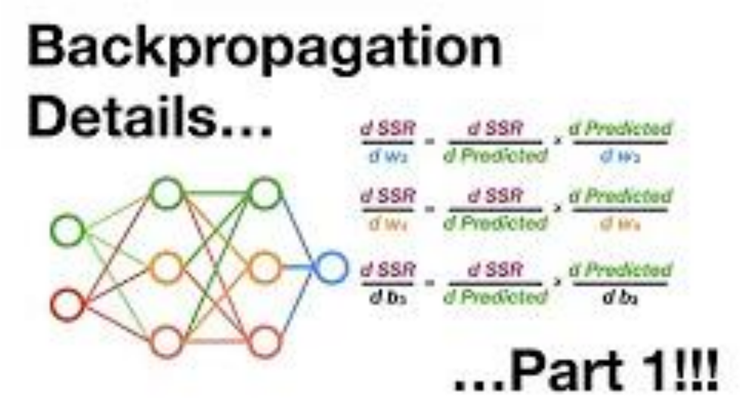
Backpropagation

17:33



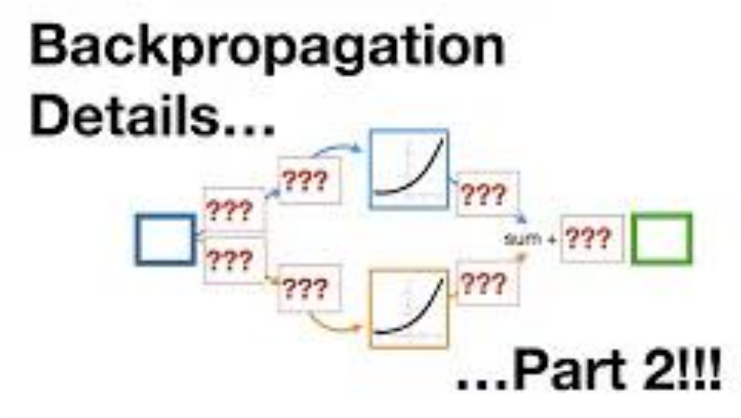
Main idea

18:31



Detail 1

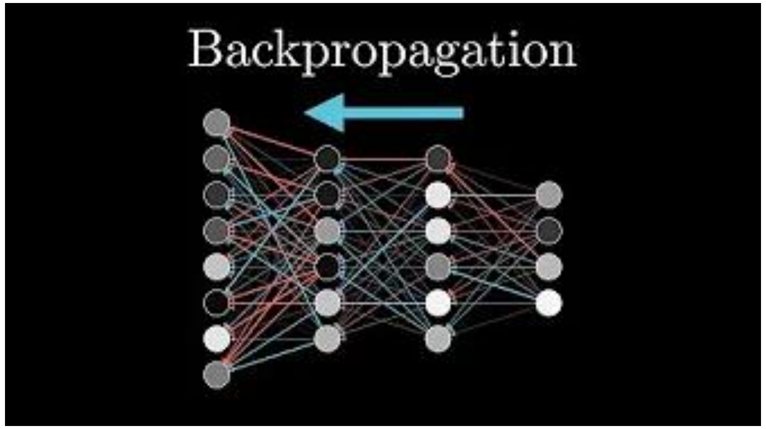
13:08



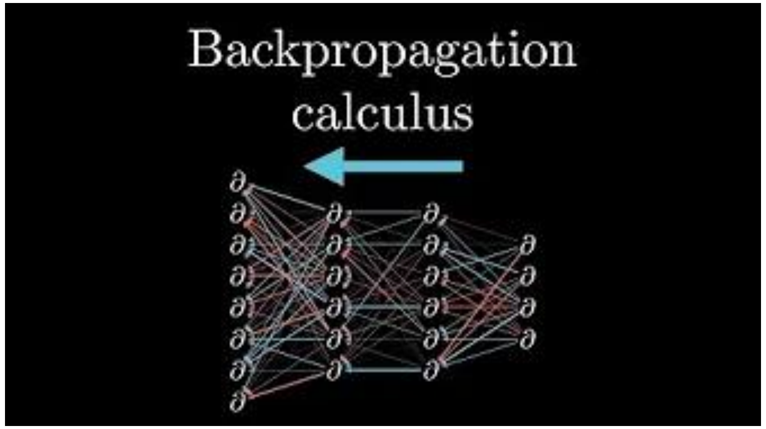
Detail 2

Backpropagation

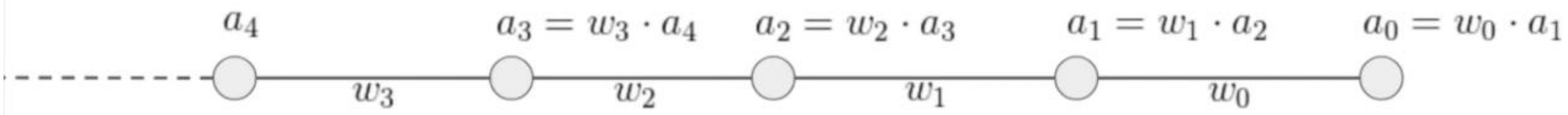
12:46



10:17



Backpropagation



省略：偏置、激活函數

$$\frac{\partial C}{\partial \omega_0} = \frac{\partial a_0}{\partial w_0}\frac{\partial C}{\partial a_0}$$

$$C = (a_0 - y)^2$$

$$\frac{\partial C}{\partial \omega_1} = \frac{\partial a_1}{\partial w_1}\frac{\partial a_0}{\partial a_1}\frac{\partial C}{\partial a_0}$$

$$\frac{\partial C}{\partial \omega_2} = \frac{\partial a_2}{\partial w_2}\frac{\partial a_1}{\partial a_2}\frac{\partial a_0}{\partial a_1}\frac{\partial C}{\partial a_0}$$

$$\frac{\partial C}{\partial \omega_3} = \frac{\partial a_3}{\partial w_3}\frac{\partial a_2}{\partial a_3}\frac{\partial a_1}{\partial a_2}\frac{\partial a_0}{\partial a_1}\frac{\partial C}{\partial a_0}$$

For example, we adjust w3 as follows:

$$w_3' = w_3 - r \cdot a_4 \cdot w_2 \cdot w_1 \cdot w_0 \cdot 2(a_0 - y)$$

$$\omega_3' = \omega_3 - r\frac{\partial C}{\partial \omega_3} \quad -r\frac{\partial a_3}{\partial w_3}\frac{\partial a_2}{\partial a_3}\frac{\partial a_1}{\partial a_2}\frac{\partial a_0}{\partial a_1}\frac{\partial C}{\partial a_0}$$

梯度下降法在 MLP 中的應用

1. 多層感知器架構：

假設一個簡單的 MLP，有 1 個隱藏層和 1 個輸出層：

- **輸入層**: x
- **隱藏層**: 帶有激活函數 $h(x) = \sigma(W_1 x + b_1)$，其中 W_1 是隱藏層的權重矩陣，b_1 是隱藏層的偏置。
- **輸出層**: $y = W_2 h(x) + b_2$，其中 W_2 是輸出層的權重，b_2 是輸出層的偏置。

假設使用均方誤差 (MSE) 作為損失函數：

$$L = \frac{1}{2}(y_{\text{true}} - y)^2$$

我們的目標是最小化損失 L。

2. 計算梯度：

梯度下降的核心是計算損失對於每個權重和偏置的導數，即 $\frac{\partial L}{\partial W_1}, \frac{\partial L}{\partial b_1}, \frac{\partial L}{\partial W_2}, \frac{\partial L}{\partial b_2}$。

對於輸出層權重 W_2：

$$\frac{\partial L}{\partial W_2} = \overset{\text{外層}}{\frac{\partial L}{\partial y}} \cdot \overset{\text{內層}}{\frac{\partial y}{\partial W_2}}$$

鍊鎖法則

$L(y(W_2))$

- $\frac{\partial L}{\partial y} = -(y_{\text{true}} - y)$
- $\frac{\partial y}{\partial W_2} = h(x)$

因此：

$$\frac{\partial L}{\partial W_2} = -(y_{\text{true}} - y) \cdot h(x)$$

對於輸出層偏置 b_2：

外層 內層

$$\frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial b_2}$$

鍊鎖法則 $L\left(y\left(b_2\right)\right)$

因為 $y = W_2 h(x) + b_2$，所以：

$$\frac{\partial L}{\partial b_2} = -(y_{\text{true}} - y)$$

對於隱藏層權重 W_1：

隱藏層的梯度傳播涉及鏈式法則：

$$\frac{\partial L}{\partial W_1} = \overset{\text{外層}}{\frac{\partial L}{\partial y}} \cdot \overset{\text{中層}}{\frac{\partial y}{\partial h(x)}} \cdot \overset{\text{內層}}{\frac{\partial h(x)}{\partial W_1}}$$

鍊鎖法則 $L\Big(y\big(h(W_1)\big)\Big)$

- $\frac{\partial y}{\partial h(x)} = W_2$
- $\frac{\partial h(x)}{\partial W_1}$ 取決於激活函數 $\sigma'(z)$，其中 $z = W_1 x + b_1$

因此：

$$\frac{\partial L}{\partial W_1} = \overset{\text{外層}}{-(y_{\text{true}} - y)} \cdot \overset{\text{中層}}{W_2} \cdot \overbrace{\sigma'(W_1 x + b_1) \cdot x}^{\text{內層}}$$

$$\frac{\partial h}{\partial W_1} = \frac{\partial h}{\partial \sigma} \frac{\partial \sigma}{\partial W_1}$$

鍊鎖法則

對於隱藏層偏置 b_1：

同樣的鏈式法則應用到偏置上：

梯度　外層　中層　內層

$$\frac{\partial L}{\partial b_1} = -(y_{\text{true}} - y) \cdot W_2 \cdot \sigma'(W_1 x + b_1)$$

鍊鎖法則

3. 梯度下降更新規則：

對於每個權重和偏置，我們根據計算出的梯度進行更新：

1. **前向傳播**　損失
2. **反向傳播**　梯度(核心技術)
3. **權重更新**　梯度下降
4. **重複迭代**　損失減小

$$W_1 := W_1 - \alpha \frac{\partial L}{\partial W_1}$$

$$b_1 := b_1 - \alpha \frac{\partial L}{\partial b_1}$$

$$W_2 := W_2 - \alpha \frac{\partial L}{\partial W_2}$$

$$b_2 := b_2 - \alpha \frac{\partial L}{\partial b_2}$$

$$L\left(\hat{\mathbf{y}}\left(\mathbf{z}^{(2)}\left(\mathbf{a}^{(1)}\left(\mathbf{z}^{(1)}\right)\right)\right)\right)$$

Q: 可以不使用反向傳播嗎？

比較項目	使用反向傳播的 MLP	不使用反向傳播的 MLP
權重更新方式	使用梯度下降法，通過計算損失對每個權重的梯度來更新權重	沒有自動權重更新，權重保持固定或使用其他非梯度方法
訓練過程	通過反向傳播計算損失函數的梯度，逐層傳遞誤差來更新權重	沒有反向誤差傳遞，僅能進行前向傳播計算輸出
計算效率	高效，尤其是對於深層神經網絡	計算效率低，特別是在使用數值梯度計算時
適用場景	機器學習模型的標準訓練方式，適合監督學習	用於無需訓練或僅做推理的模型，或採用非梯度訓練方法
梯度計算方法	基於鏈式法則，逐層計算梯度	無法計算梯度，可能使用數值梯度或其他進化算法進行訓練
訓練效果	通常能夠達到較好的訓練效果，因為權重能根據損失函數自動調整	訓練效果較差，無法根據損失調整權重

梯度下降法在DNN的應用

1. 前向傳播（Forward Propagation）

在前向傳播中，我們從輸入層開始，逐層計算輸出，直到最終的預測值 y_{pred}。

假設一個深度神經網絡有 L 層，每層的輸出記作 $a^{[l]}$，權重為 $W^{[l]}$，偏置為 $b^{[l]}$，激活函數為 $\sigma^{[l]}$。

對於第 l 層神經元：

$$z^{[l]} = W^{[l]} a^{[l-1]} + b^{[l]}$$

$$a^{[l]} = \sigma^{[l]}(z^{[l]})$$

2. 損失函數（Loss Function）

假設我們的損失函數是均方誤差（MSE），對於單一樣本 y_{true} 和預測值 y_{pred}：

$$L = \frac{1}{2}(y_{\text{true}} - y_{\text{pred}})^2$$

或交叉熵損失（Cross-Entropy Loss）在分類問題中常用：

$$y_{\text{pred}} = a^{[L]}$$

$$L = -\sum y_{\text{true}} \log(y_{\text{pred}})$$

3. 反向傳播的步驟

反向傳播的目的是計算每一層的權重和偏置相對於損失函數的梯度，並使用梯度下降法來更新這些權重。

第一步：計算最終輸出層的誤差

在最後一層 L 中，誤差 $\delta^{[L]}$ 定義為損失函數對輸出層加權輸入 $z^{[L]}$ 的偏導數：

$$\frac{\partial L}{\partial z^{[L]}} = \delta^{[L]} = \underbrace{\frac{\partial L}{\partial a^{[L]}}}_{\text{外層}} \cdot \underbrace{\sigma'^{[L]}(z^{[L]})}_{\text{內層}} \quad \text{鍊鎖法則} \qquad L\left(a^{[L]}\left(z^{[L]}\right)\right)$$

其中，$\sigma'^{[L]}(z^{[L]})$ 是輸出層激活函數的導數。

第二步：逐層計算誤差（誤差的反向傳遞）

$$\frac{\partial L}{\partial z^{[l]}} = \boxed{\frac{\partial L}{\partial z^{[l+1]}}} \frac{\partial z^{[l+1]}}{\partial z^{[l]}}$$

外層 $z^{[l+1]} = W^{[l+1]} a^{[l]} + b^{[l]}$

內層 $a^{[l]} = \sigma\left(z^{[l]}\right)$

對於第 l 層，誤差 $\delta^{[l]}$ 為：

$$\frac{\partial L}{\partial z^{[l]}} = \delta^{[l]} = \underbrace{(W^{[l+1]})^T}_{\text{外層}} \boxed{\delta^{[l+1]}} \cdot \underbrace{\sigma'^{[l]}(z^{[l]})}_{\text{內層}}$$

鍊鎖法則

這一步依賴於後一層的誤差 $\delta^{[l+1]}$，即誤差從後向前傳遞，這就是「反向」傳播的由來。

第三步：計算梯度

一旦知道了每層的誤差 $\delta^{[l]}$，就可以計算權重和偏置的梯度：

- 對於權重 $W^{[l]}$：

$$\frac{\partial L}{\partial W^{[l]}} = \boxed{\frac{\partial L}{\partial z^{[l]}}} \frac{\partial z^{[l]}}{\partial W^{[l]}}$$

$$\frac{\partial L}{\partial W^{[l]}} = \boxed{\delta^{[l]}} (a^{[l-1]})^T$$

- 對於偏置 $b^{[l]}$：

$$\frac{\partial L}{\partial b^{[l]}} = \delta^{[l]}$$

第四步：權重和偏置更新

使用梯度下降法更新權重和偏置：

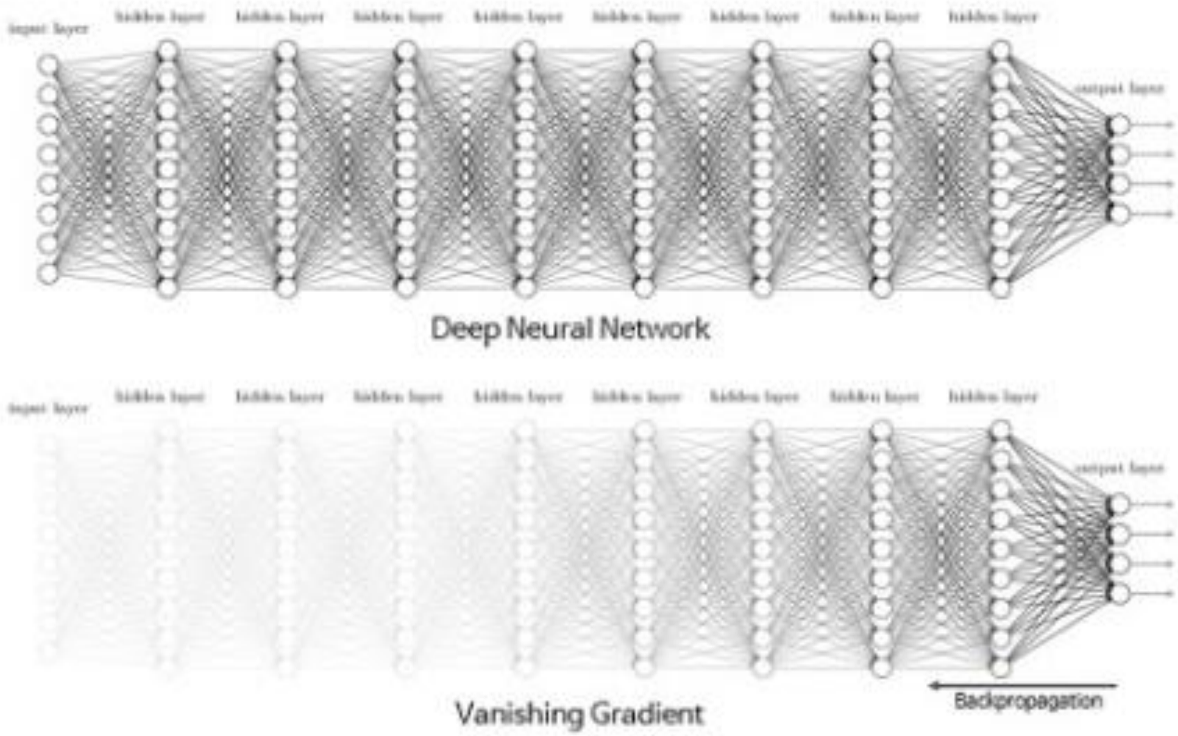
$$W^{[l]} = W^{[l]} - \eta \cdot \frac{\partial L}{\partial W^{[l]}}$$

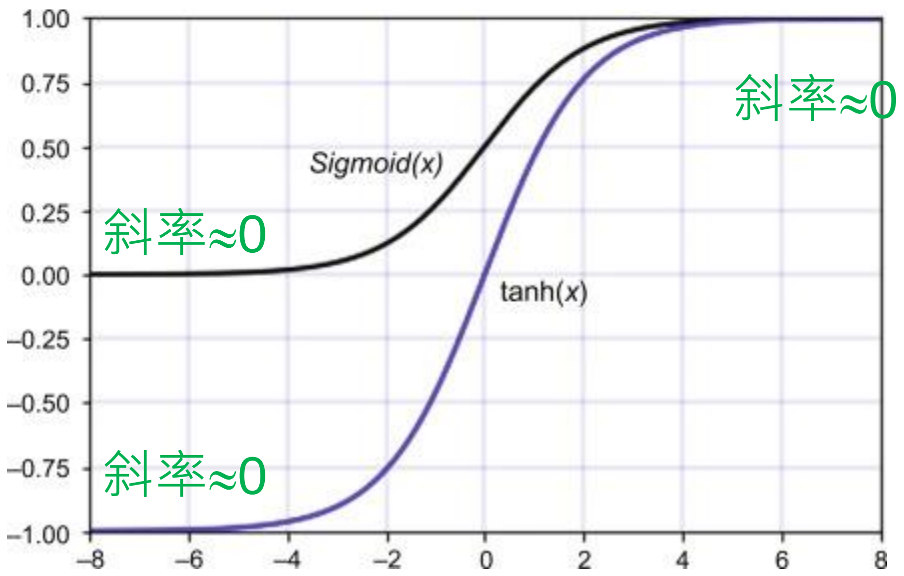
$$b^{[l]} = b^{[l]} - \eta \cdot \frac{\partial L}{\partial b^{[l]}}$$

其中 η 是學習率，控制每次更新的步長。

Vanishing Gradient

- Gradients of loss function become vanishingly small
- No updated weight information during backpropagation

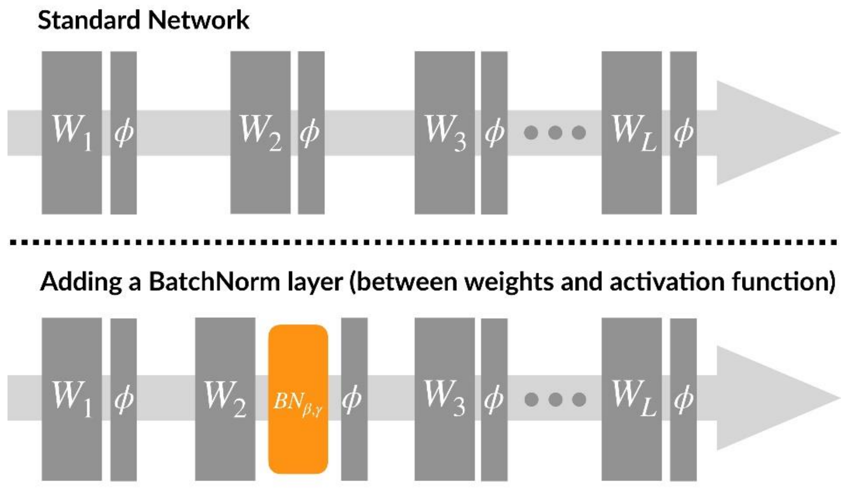




Vanishing Gradient

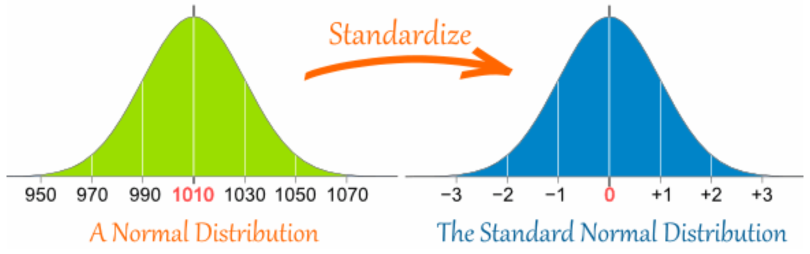
- Reducing Complexity
  - Skip connections, e.g., CNN, RNN
- Using ReLU Activation Function
  - Non-saturating activation function
- Batch Normalization
  - Normalizing input to each layer: zero mean and unit variance
- Residual Networks
  - Learn residual functions with reference to layer inputs

Batch Normalization

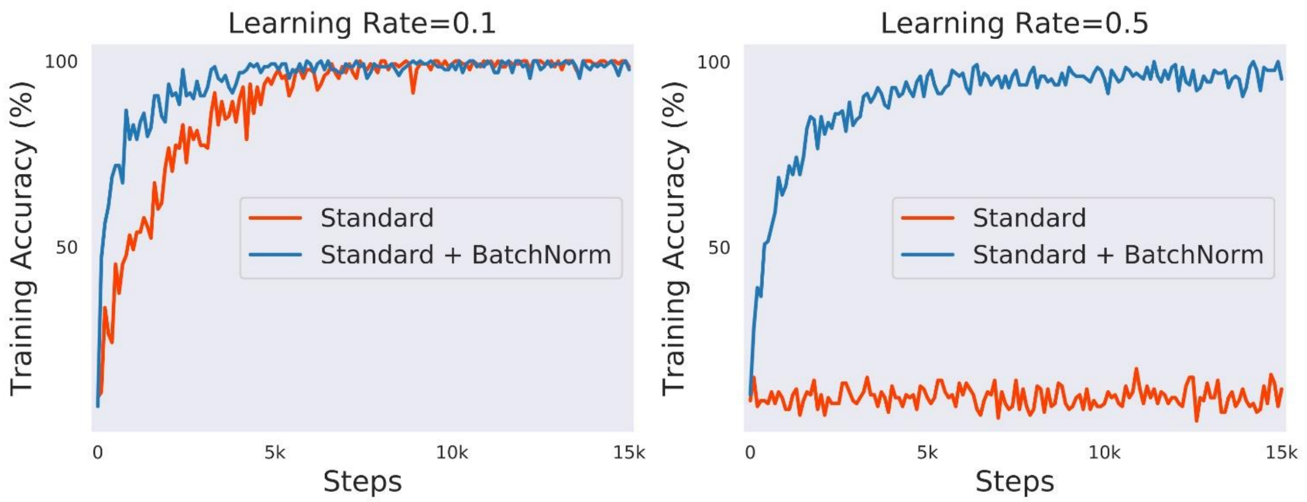


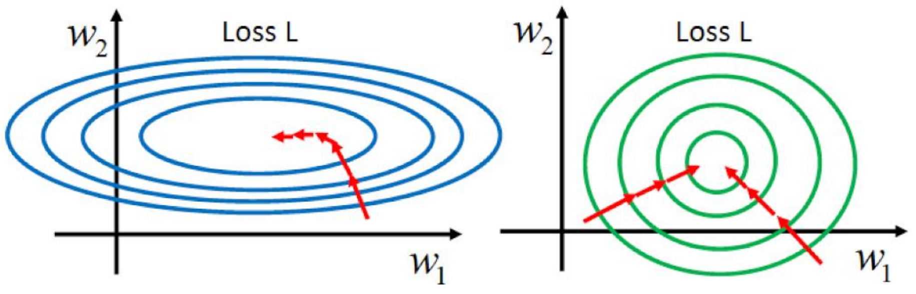
$$BN(y_j)^{(b)} = \gamma \cdot \left(\frac{y_j^{(b)} - \mu(y_j)}{\sigma(y_j)} \right) + \beta,$$

re-centering and re-scaling



zero mean
unit variance





批次正規化的公式：

批次正規化是在每個小批量（batch）內的輸入進行標準化處理，具體步驟如下：

1. **計算均值**：對於小批量內的每個特徵，計算該特徵的均值 μ_B：

$$\mu_B = \frac{1}{m} \sum_{i=1}^{m} x_i$$

   其中，x_i 是第 i 個樣本的輸入特徵，m 是該批次的樣本數。

2. **計算方差**：計算該批次中該特徵的方差 σ_B^2：

$$\sigma_B^2 = \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_B)^2$$

3. **標準化**：將輸入特徵標準化為均值為 0、方差為 1 的數據：

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

   其中，ϵ 是一個很小的數值，防止分母為零。

項目	回歸（Regression）	分類（Classification）
目標	預測連續數值	預測離散類別
輸出層激活函數	無或線性函數	Softmax（多類）或 Sigmoid（二元分類）
輸出	單個或多個實數	類別標籤（單類或多類）
損失函數	均方誤差（MSE, Mean Squared Error）	交叉熵損失（Cross-Entropy Loss）
評估指標	均方誤差（MSE）、均方根誤差（RMSE）等	準確率（Accuracy）、F1-Score、AUC 等
輸出範圍	實數範圍	$[0, 1]$（對每個類別進行概率估計）
應用範例	預測房價、預測股票價格、預測溫度等	圖像分類、文本分類、語音識別等
常見問題類型	迴歸問題：數值預測	二元分類、多類分類
網路結構	網路結構相似，具體依賴於數據與應用需求	網路結構相似，具體依賴於數據與應用需求
標籤形式	連續的實數標籤	離散的類別標籤（如 0、1 或多類別標籤）
輸出層神經元數量	1 或多個（對於多輸出變量）	對於二元分類為 1，對於多類分類則等於類別數

DNN

深度神經網絡（DNN）能自動提取特徵的機制基於多層非線性轉換和學習過程：

1. **分層結構**：DNN由多層神經元組成，輸入數據經過每一層逐步處理。每一層學習提取不同層次的特徵，從簡單到複雜，低層捕捉如邊緣或角的基礎特徵，高層學習更抽象的概念。
2. **權重學習**：每個神經元通過調整權重學習輸入與輸出之間的關係，權重反映了數據中哪些特徵更重要。DNN通過反向傳播算法優化這些權重，使得模型能夠有效識別特徵。
3. **非線性激活函數**：每層之間的非線性激活函數（如ReLU、Sigmoid）使得網絡能學習複雜的非線性關係，這使網絡能夠捕捉和表示更多元的數據特徵。
4. **反向傳播與梯度下降**：DNN通過反向傳播和梯度下降算法逐層更新權重，使網絡逐步調整並學習有效的特徵表示，最終自動從數據中提取有用的信息。

這些機制使DNN在無需人工設計特徵的情況下，能自動從原始數據中學習到抽象和高階特徵。

Convolutional Neural Network