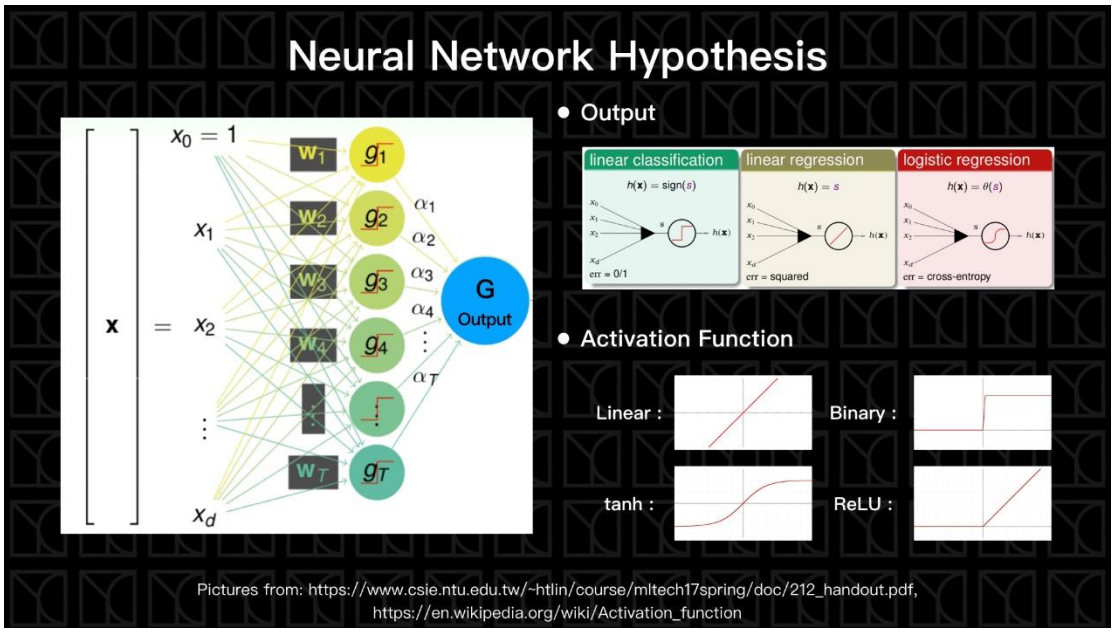
Extraction Models，其實就是現今很流行的「類神經網路」(Neural Network) 和「深度學習」(Deep Learning)，包括下圍棋的 AlphaGo、Tesla 的自動駕駛都是採用這一類的 Machine Learning。

Extraction Models 的基本款就是廣為人知的「神經網路」(Neural Network)，它的特色是使用神經元來做非線性的特徵轉換，那如果具有多層神經元，就是做了多次的非線性特徵轉換，這就是所謂的「深度學習」(Deep Learning)。



上圖左側就是具有一層神經元的 Neural Network，首先我們有一組特徵 XX，通常我們會加入一個維度 X0=1X0=1，這是為了可以讓結構變得更好看，未來可以與 W0W0 相乘產生常數項。使用 WW 來

給予特徵 XX 權重，最後總和的結果稱之為 Score，

s=W0X0+Σi=1WiXi=Σi=0WiXis=W0X0+**Σ**i=1WiXi=**Σ**i=0WiXi。

這個 Score 會被輸入到一個 Activation Function 裡頭，Activation Function 的用意就是開關，當 Score 大於某個閥值，就打通線路讓這條路的貢獻可以繼續向後傳遞；當 Score 小於某個閥值，就關閉線路，所以 Activation Function 可以是 Binary Function，但在實際操作之下不會使用像 Binary Function 這類不可以微分的 Activation Function，所以我們會找具有相似特性但又可以微分的函數，例如 tanhtanh 或者是 ReLUReLU 這類比較接近開關效果的函數，經過 Activation Function 轉換後的輸出表示成

gt=σ(ΣiWiXi)gt=σ(**Σ**iWiXi)，這個 gtgt 就稱為神經元、σσ 為 Activation Function、ΣiWiXi**Σ**iWiXi 是 Score。

如果我們有多組權重 WW 就能產生多組神經元 gtgt，然後最後把 gtgt 做線性組合並使用 Output Function h(x)h(x)來衡量出最後的答案，Output Function 可以是 Linear Classification 的 Binary Function h(x)=sign(x)h(x)=sign(x)，不過一樣的問題，它不可以微分，通常不會被使用，常見的是使用 Linear Regression h(x)=xh(x)=x，或者 Logistic

Regression $h(x)=\Theta(x)$h(x)=Θ(x)來當作 Output Function，最後的結果可以表示成 $y=h(\Sigma t \alpha t g t)$y=h(Σtαtgt)，看到這個式子有沒有覺得很熟悉，它就像我們上一回講的 Aggregation，將特徵 X 使用特徵轉換轉成使用 gtgt 表示，再來組合這些 gtgt 成為最後的 Model，所以單層的 Neural Network 就使用到了 Aggregation，它繼承了 Aggregation 的優點。

有了這個 Model 的形式了，我們可以使用 Gradient Descent 的手法來做最佳化，這也就是為什麼要讓操作過程當中所使用的函數都可以微分的原因。Gradient Descent 在 Neural Network 的領域裡面發展出一套方法稱為 Backpropagation，我們待會會介紹。因此實現 Backpropagation，我只要餵 Data 進去，Model 就會去尋找可以描述這組 Data 的特徵轉換 gtgt，這就好像是可以從 Data 中萃取出隱含的 Feature 一樣，所以這類的 Models 才會被統稱為 Extraction Models。

深度學習(Deep Learning)

剛剛我們介紹了最基本款的 Neural Network，那如果這個 Neural Network 有好幾層，我還會稱它為 Deep Learning，所以基本上 Deep Neural Network 和 Deep Learning 是指同一件事，那為什麼會有兩個名字呢？其實是有歷史典故的。

Neural Network 的歷史相當悠久，早在 1958 年就有人提出以 Perceptron 當作 Activation Function 的單層 Neural Network，大家也知道一層的 Neural Network 是不 Powerful 的，所以在 1969 年，就有人寫了論文叫做「perceptron has limitation」，從那時起 Neural Network 的方法就很少人研究了。

直到 1980 年代，有人開始使用多層的 Neural Network，並在 1989 年，Yann LeCun 博士等人就已經將反向傳播演算法 (Backpropagation, BP)應用於 Neural Network，當時 Neural Network 的架構已經和現在的 Deep Learning 很接近了，不過礙於當時的硬體設備計算力不足，Neural Network 無法發揮功效，並且緊接的有人在 1989 年證明了只要使用一層 Neural Network 就可以代表任意函數，那為何還要 Deep 呢？所以 Deep Neural Network 這方法就徹底黑掉了。

一直到了最近，G. E. Hinton 博士為了讓 Deep Neural Network 起死回生，重新給了它一個新名字「Deep Learning」，再加上他在 2006 年提出的 RBM 初始化方法，這是一個非常複雜的方法，所以在學術界就造成了一股流行，雖然後來被證明 RBM 是沒有用的，不過卻因為很多人參與研究 Deep Learning 的關係，也找出了解決 Deep Learning 痛處的方法，2009 年開始有人發現使用 GPU 可以大大的加速 Deep Learning，從這一刻起，Deep Learning 就開始流行起來，直到去年的 2016 年 3 月，圍棋程式 Alpha GO 運用 Deep Learning 技術以 4:1 擊敗世界頂尖棋手李世乭，Deep Learning 正式掀起了 AI 的狂潮。

聽完這個故事我們知道改名字的重要性 XDD，不過大家是否還有看到什麼關鍵，「使用一層 Neural Network 就可以代表任意函數，那為何還要 Deep 呢？」這句話，這不就否定了我們今天做的事情了嗎？的確，使用一層的 Neural Network 就可以形成任意函數，而且完全可以用一層的神經元來表示任何多層的神經元，數學上是行得通的，但重點是參數量。Deep Learning 的學習方法和人有點類似，我們在學習一個艱深的理論時，會先單元式的針對幾個簡單的概念學習，然後在整合這些概念去理解更高層次的問題，Deep Learning 透過多層結構學習，雖然第一層的神經元沒有很多，能學

到的也只是簡單的概念而已，不過第二層再重組這些簡單概念，第三層再用更高層次的方法看問題，所以同樣的問題使用一層 Neural Network 可能需要很多神經元才有辦法描述，但是 Deep Learning 卻可以使用更少的神經元做到一樣的效果，同樣表示的數學轉換過程，雖然單層和多層都是做得到相同轉換的，但是多層所用的參數量是比單層來得少的，依照 VC Generalization Bound 理論告訴我們可調控的參數量代表模型的複雜度，所以多層的 NN 比單層的有個優勢是在做到同樣的數學轉換的情況下更不容易 Overfitting。

因此，Deep Learning 中每一層當中做了 Aggregation，在增加模型複雜度的同時，也因為平均的效果而做到截長補短，這具有 Regularization 的效果，並且在採用多層且瘦的結構也同時因為「模組化」而做到降低參數使用量，來減少模型複雜度，這就不難想像 Deep Learning 為何如此強大。

參考網站: https://www.ycc.idv.tw/ml-course-techniques_6.html