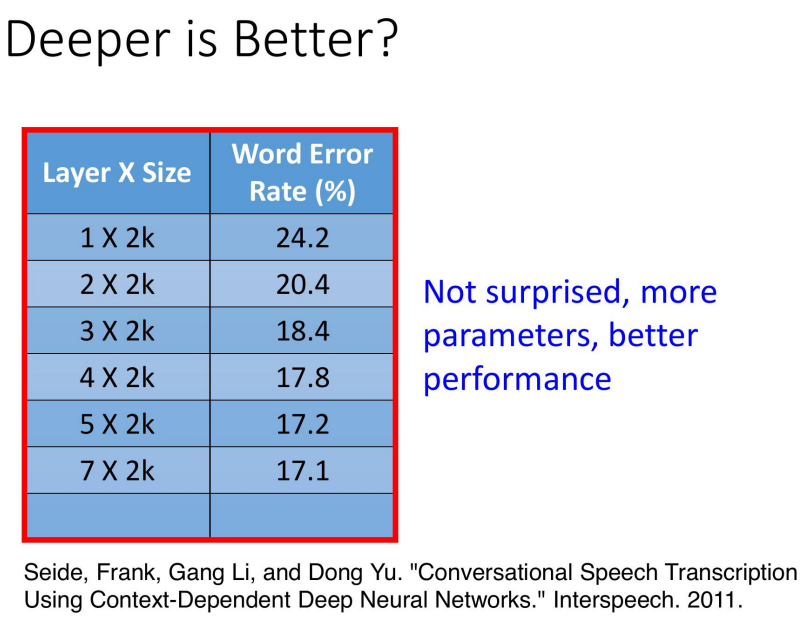
ML Lecture 11: Why Deep?

> 臺灣大學人工智慧中心 科技部人工智慧技術暨全幅健康照護聯合研究中心 http://ai.ntu.edu.tw

Deep or Shallow

神經網路從1層到7層，越深時 error rate也隨著下降。



但是 network 越深，代表的是你的參數越多，你的 model 越複雜，在你的 data 足夠的情況下， performance 本來就會比較好。所以真正要比較 Deep 和 Shallow 的 model，要把 Deep 和 Shallow 的參數調整成是一樣多的，這樣才是公平的。

接下的問題就是，在這個公平的評比之下，是 Shallow 還是 Deep 比較強？

這個實驗後半段的結果如下

Fat + Short v.s. Thin + Tall

Layer X Size	Word Error Rate (%)	Layer X Size	Word Error Rate (%)
1 X 2k	24.2		
2 X 2k	20.4		
3 X 2k	18.4		
4 X 2k	17.8		
5 X 2k	17.2	1 X 3772	22.5
7 X 2k	17.1	1 X 4634	22.6
		1 X 16k	22.1

Why?

Seide, Frank, Gang Li, and Dong Yu. "Conversational Speech Transcription Using Context-Dependent Deep Neural Networks." Interspeech. 2011.

若使兩邊的參數數量接近：

- 5 層，每層兩千個 neuron， 得到 error rate 是17.2
- 1 層 ，3772 的 neuron，得到 error rate 是 22.5

> error rate 越小越好 。

另外一個參數數目接近的情況：

- 7 層，每層 2000 個 neuron，得到 error rate 是 17.1
- 一層 4634 個 neuron，得到 error rate 是 22.6

即便給一個 1 層有 16k 個 neuron， error rate 也只有從 22.6 降到 22.1 而已。

結論

在一層的情況下，比較寬的 network 跟沒有那麼寬的 network 來比，因為前者的參數比較多，所以它的 performance 比較強。但比較有兩層的 network 跟只有一層的 network，即便兩層的 network 的參數遠比一層少，它的 performance 還是比，只有一層的 network 的 performance 還要好。

在很多人的的想像裡面，deep learning 之所以成功就是因為這是一個暴力輾壓的方法，利用一個大很大的 model，收集一大堆的 data，因此得到比較好的 performance。

但實際不是這樣，如果只是單純的增加 parameter，但是讓 network 長寬而不是長高的話，其實對 performance 的幫助是比較小的，反而把 network 長高，對 performance 的影響很有幫助

現在，接下來的問題就是「為什麼會這樣」。

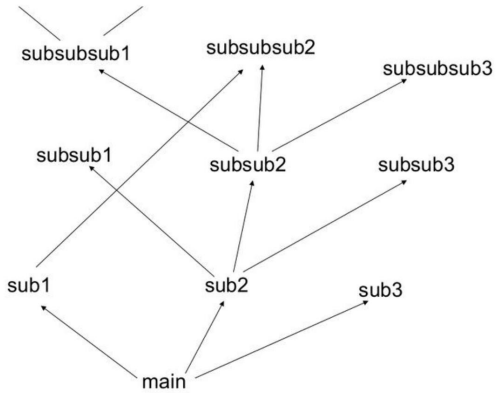
Why deep works

我們可以想像，deep learning 其實就是在做模組化，就如同我們一般寫程式時會把重複使用的部分拆分成一個個 function，而實際程式在運作時就會出現結構化的 function (A call B, B call C，如下圖)，因此我們藉著模組化 (拆分 function) 這件事讓程式減少冗餘或重複的部分。

Modularization

- Deep → Modularization

Don’t put everything in your main function.



http://rinuboney.github.io/2015/10/18/theoretical-motivations-deep-learning.html

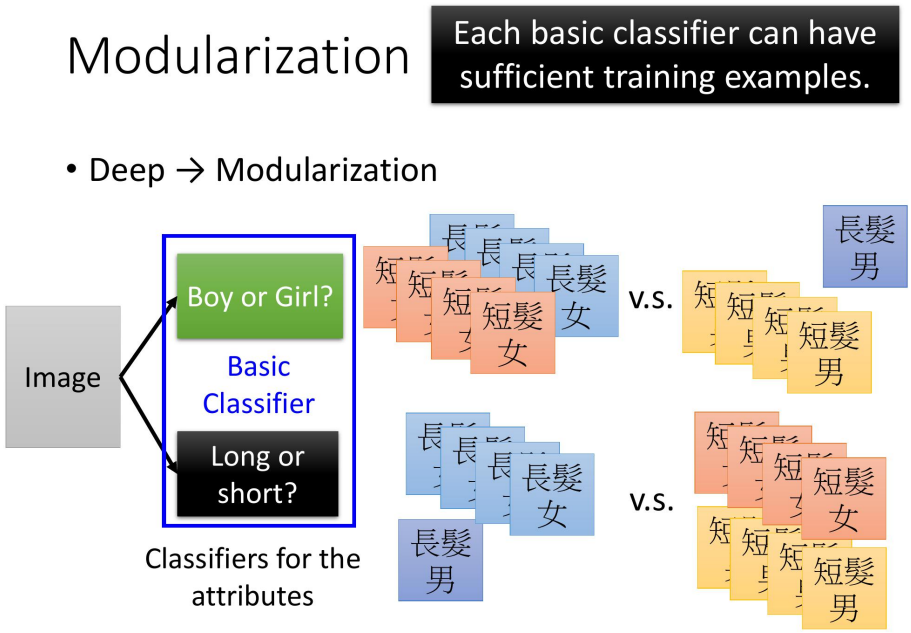
回到 machine learning 上面來做討論，假設要做影像分類並把圖片分成 4 類：

- 長頭髮女生
- 長頭髮男生
- 短頭髮女生
- 短頭髮男生

這裡出現的問題就是，長頭髮的男生的 data 可能是比較少的，在沒有太多的 training data 的情況下，訓練出來的 detect 長頭髮男生的 classifier 就比較 weak，因此 detect 長頭髮男生的 performance 就比較差。

那怎麼辦呢？模組化的概念代表不是先直接去解原本的問題，而是把問題切成比較小的問題，比如說，把剛剛的問題切成：

- 男生 or 女生
- 長頭髮 or 短頭髮



這麼一來，data 都可以 collect 到足夠的量，這些 basic classifier 都是有足夠 data 把它 train 好。

於是我們要真正處理的問題的時候，每一個 classifier 就去參考這些 basic classifier 的 output，藉此來決定這個 classifier 的輸出。

回到 deep learning 跟模組化之間的關係，每一個 neuron 其實就是一個 basic 的 classifier，所以第一層 neuron 是最 basic 的 classifier，第二層 neuron 是比較複雜的 classifier，每一層都使用前一層的 output 當作 input，也就是每一層都把前一層的 neuron 當作他的 module。而在 deep learning 的時候，怎麼做模組化是由機器自動學到的。

結論

模組化的好處是讓模型變簡單了，把本來複雜的問題變得比較簡單後，就算 training data 沒有那麼多，仍然可以把這個事做好，這是模組化的精神。

也就是說，deep learning 如果是在做模組化的話，神奇的事就是deep learning 需要的 training data 是比較少的，這可能跟你的認知是相反的。

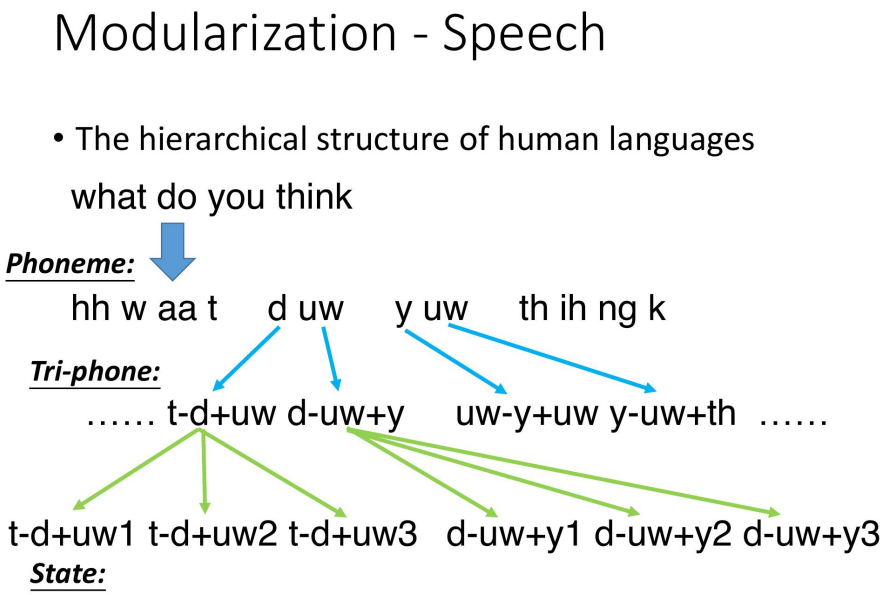
現在因為 deep learning 當紅，新聞上都有各種各式樣的說法，常見的說法有

> AI = big data + deep learning

因此造成一個誤解是 deep learning 會成功是因為 big data 的關係，沒有 big data 時 deep learning 就不會成功。

Deep Learning on Speech

背景知識



Phoneme

- 由語言學家定義出來「人類發音的基本單位」，可以把它想成是音標
- 同樣的 phoneme可能會有不太一樣的發音，當你發 d uw 和 y uw 的時候，儘管心裡想的是同一個 phoneme，但是因為人類口腔器官的限制，所以沒辦法每一次發的 uw 都是一樣的。

Tri-phone

因為人類發音器官的限制，所以 phoneme 的發音會受到前後的 phone 所影響，所以給同樣的 phoneme 不同的表示方式。

將每個 phoneme 將上它前後的 phoneme，成為三個一組的就是 tri-phone。

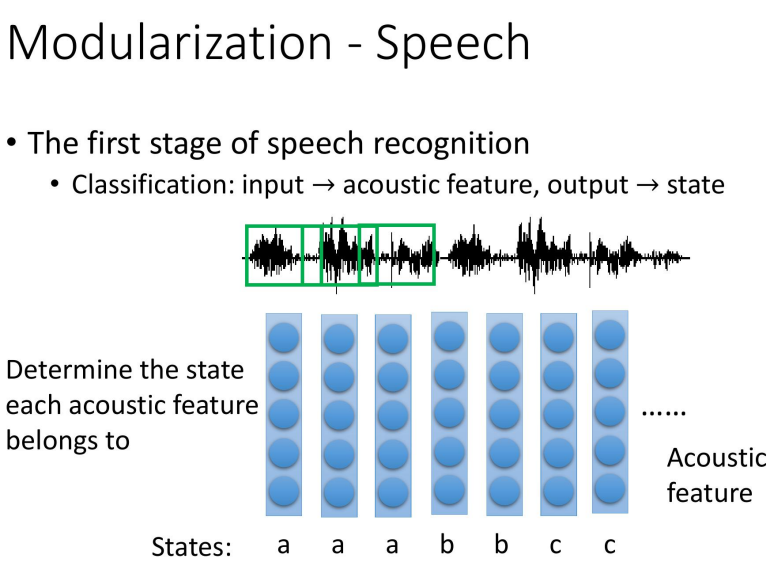
State

一個 phoneme 可以拆成一些 state，state 有幾個則是可以自訂的，通常訂成 3 個 state。

語音辨識步驟

1. 把 acoustic feature 轉成 state，這是一個單純的分類問題 (classification problem)，這個分類問題就跟圖片分類是一樣的，所做的是在建立一個 classifier。

> Acoustic feature 在邊不會細談。簡單來說就是，在聲波上取一個 window (e.g. 250 ms)，用一個 feature 來描述 window 裡的特性，就是一個 acoustic feature，一段聲音訊號就會變成一串 vector sequence，稱作 acoustic feature sequence。



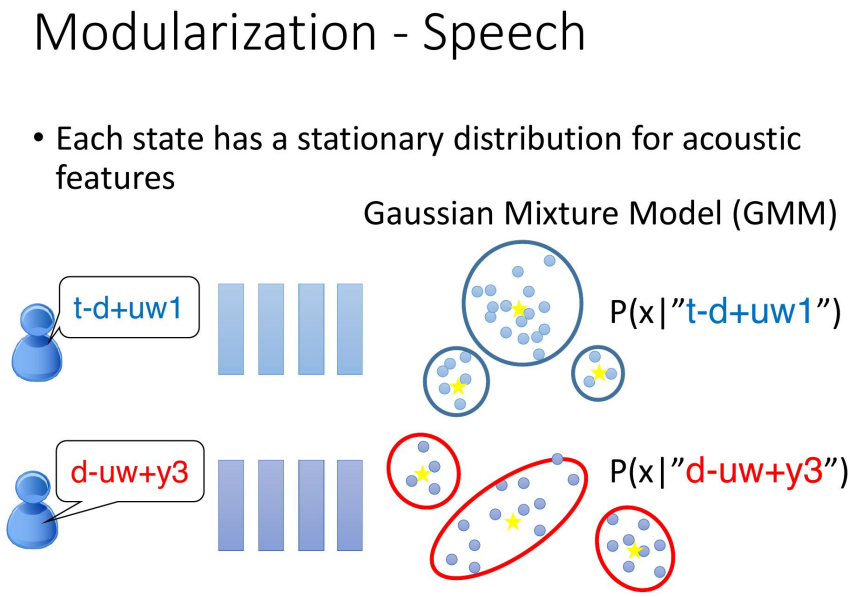
2. 把 state 轉成 phoneme
3. 再把 phoneme 轉成文字，這個階段要用 language model 考慮同音異字的問題，等等...

> 上面的步驟其實不是重點，不必充分理解

傳統語音辨識方法和 **Deep Learning** 的差別

傳統方法 (HMM-GMM)

t傳統的方法使用的是 **Gaussian mixture model (GMM)**，它假設每一個 state 是一個 stationary 的訊號分佈，也就是說，每一個 state 裡的 acoustic feature 的分布是固定的，因此只要給定一個 feature，就能算它從每個 state 出來的機率。



難點

仔細想一想，就會發現這個方法其實不太容易達成目的，為甚麼呢？因為 tri-phone 的數目太多了。一般語言都有 30 幾、將近 40 個 phoneme，假設是 30 個，那 tri-phone 的種類就會達到 30 的 3 次方個，也就是 27000 個 tri-phone。每一個 tri-phone 又有三個 state，於是有數萬個 state，而每一個 state 都需要一個 GMM 來描述，參數太多了，不太可能有足夠的訓練資料來做好這件事。

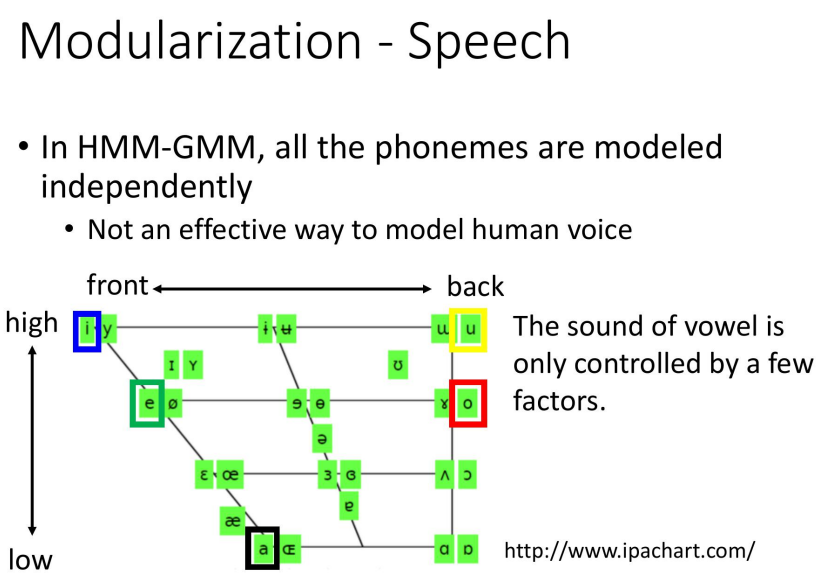
解決方法

在傳統上有 deep learning 之前，解決的方法是將一些 state 共用同樣的 model distribution，稱之為 Tied-state。哪一些要共有就要憑著經驗還有一些語言學的知識來決定，可是這樣仍然是不夠的，這個方法並不夠精細，所以有的人就會開始提出一些想法來讓它部分共用。

在 deep learning 火紅之前，比較有創新的方法叫做 subspace GMM，其實它也有 modularization 的影子，這個方法是，先把很多的 Gaussian 先找出來，成為一個 Gaussian pool，而每個 state 再從這個 Gaussian pool 裡面挑一些 Gaussian 出來，比如說可能有某一個 state 1，它挑第1, 3, 5個 Gaussian，而另一個 state 2，它挑第1, 4, 6個 Gaussian，這樣一來，這些 state 就可以 share 部分的 Gaussian，有些時候則可以完全不 share Gaussian，至於要 share 多少的 Gaussian，是可以從訓練資料去學出來的。

產生難點的原因

這個方式將所有的 phone 或者是 state 視為是 independent model 的，這麼做是沒有效率的，畢竟不同的 phoneme 雖然說我們把它歸類為不同的音素，但這些 phoneme 之間並不是完全無關的，它們都是由人類的發音器官所產生的，因此根據人類發音器官發音的方式，它們是有某些關係的。



舉例來說，上圖表示了人類語言裡面所有的母音的發音，其實就只受到三件事情的影響而已，分別是是你舌頭的前後的位置、你舌頭上下的位置、和你的嘴型。

因為不同的 phoneme 之間是有關係的，所以每一個 phoneme 都各有一個 model 是沒有效率的。

Deep Learning

深度學習的做法是這樣的：

- Input: acoustic feature
- Output: 每一個 feature 屬於哪一個 state 的機率

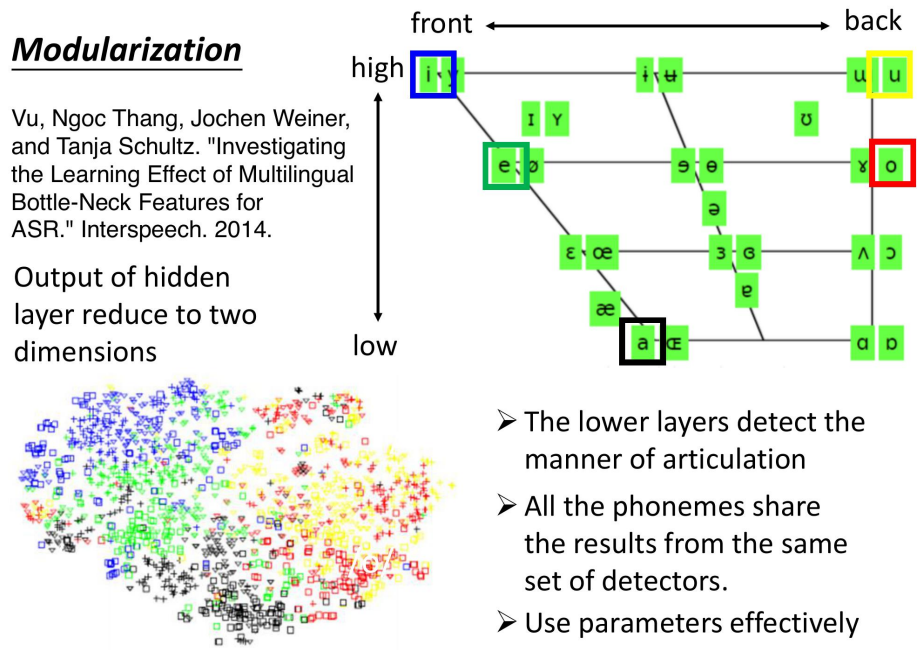
這就是一個很單純的分類問題，最關鍵的一點是**所有的 state 都共用同一個 DNN**，並非每一個 state 都有一個 DNN。

對此可能的迷思是：本來 GMM 通常最多也就做 64 個 Gaussian mixture 而已，而 DNN 有 10 層，每層 1000 個 neuron，果然參數很多，所以是用暴力輾壓的方法，其實也沒什麼。

其實 DNN 不是一個暴力輾壓的方法，在做 HMM-GMM 的時候，雖然 GMM 只有 64 個 mixture好像很簡單，但是其實每一個 state 都有一個 Gaussian mixture，所以合起來它的參數是多得不得了的。仔細去算一下，GMM 用的參數跟 DNN 用的參數，在不同的 task 上曾估測過這件事，它們用的參數其實是差不多多的，但 DNN 它只是用一個很大的 model，GMM 卻是用很多很小的 model，通通用同一個 model 來做分類會是比較有效率的做法。

為甚麼這樣是比較有效率的做法呢？

舉例來說，如果你今天把一個 DNN 它的某一個 hidden layer 拿出來，它可能有 1000 個 neuron，你沒有辦法分析它，但是把那 1000 個 layer 的 output 降到 2 維，就可以得到下圖的結果。



在上圖中，每一個點代表了一個 acoustic feature，在這個圖上的顏色代表的其實就是a, e, i, o, u 這 5 個母音。會發現神奇的事就是，這 5 個母音的分布跟這一個圖的分布，其實幾乎是一樣的。

因此，DNN 比較 lower 的 layer (也就是比較靠近 input 的 layer) 做的事情，並不是真的要要馬上去偵測現在 input 這個發音，它是屬於哪一個 phone 或哪一個 state，而是人用甚麼樣的方式在發這個聲音的，例如它的舌頭的位置在哪裡、舌頭位置是高還是低、舌頭位置是在前還是後。

Lower layer知道了發音的方式以後，接下來的 layer再根據這個結果，去決定現在的發音是屬於哪一個 state 或哪一個 phone，所以所有的 phone 會共用同一組 detector，因此這邊就有做到模組化這件事情，讓我們能用比較有效率的方式來使用參數。

其他例子

Universality Theorem

回到很久以前就提過的 universality theorem，它告訴我們任何的 continuous function 都可以用一層 neural network 來完成，只要那層 neural network 夠寬的話。這是在 90 年代很多人放棄做 deep learning 的一個原因，畢竟只要一層 hidden layer 就可以完成所有的 function，那做 deep learning 的意義何在呢？但是這個理論只告訴我們可能性，它沒有告訴我們要做到這件事情有多有效率，它沒有提及我們只用一個 hidden layer 來描述 function 的時候其實是沒有效率的，有 multi-layer 也就是有階層的結構時，這樣的方式來描述 function 的才是比較有效率的。

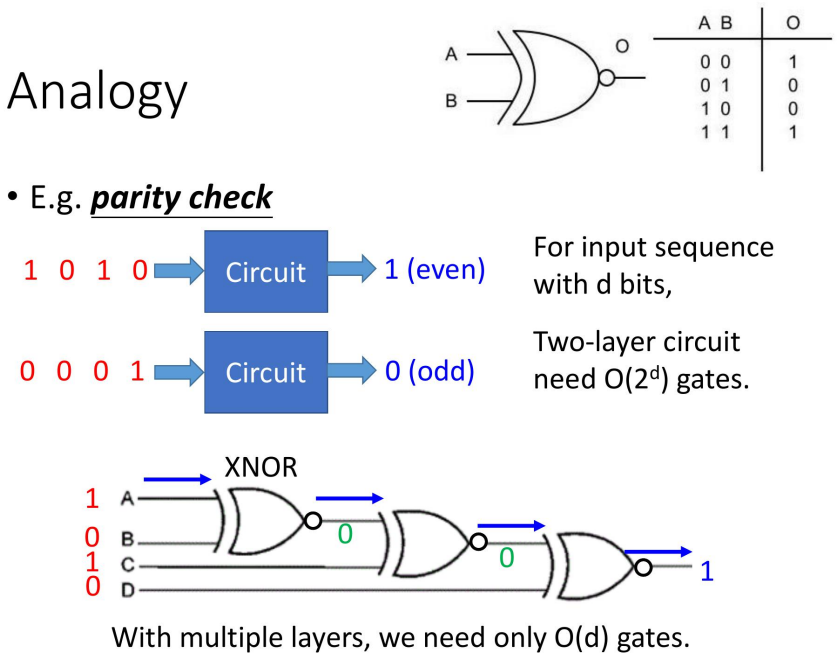
邏輯電路的例子

如果剛才模組化的概念聽得很明白的話，在此再舉另外一個例子。

如果是 EE 的背景的話，你應該修過交換電路。在邏輯電路中，其實只要兩層邏輯閘，就可以表示任何的 boolean function；而在 neural network 裡，有一個 hidden layer 的 neural network 其實也是兩層，一個 input layer、一個 output layer，所以有一個 hidden layer 的 neural network 可以表示任何的 continuous function，這件事也不會讓人特別的驚訝。

雖然我們可以用兩層邏輯閘就描述任何的 boolean function，但是實際上你在做電路設計的時候，你根本不可能會這樣做，因為當你用 hierarchy 的架構的時候，設計的電路是比較有效率的。類比到 neural network 的話，用一層 hidden layer 可以做到任何事情，但是用比較多的 hidden layer 是比較有效率的，所以可以用比較少的 neuron 就完成同樣的 function，也就意謂著比較少的參數，比較少的參數意謂著比較不容易 overfit，或者是需要比較少的 data 就可以完成你現在要訓練的任務。

以下是一個 parity check 的實際邏輯電路的例子來佐證：

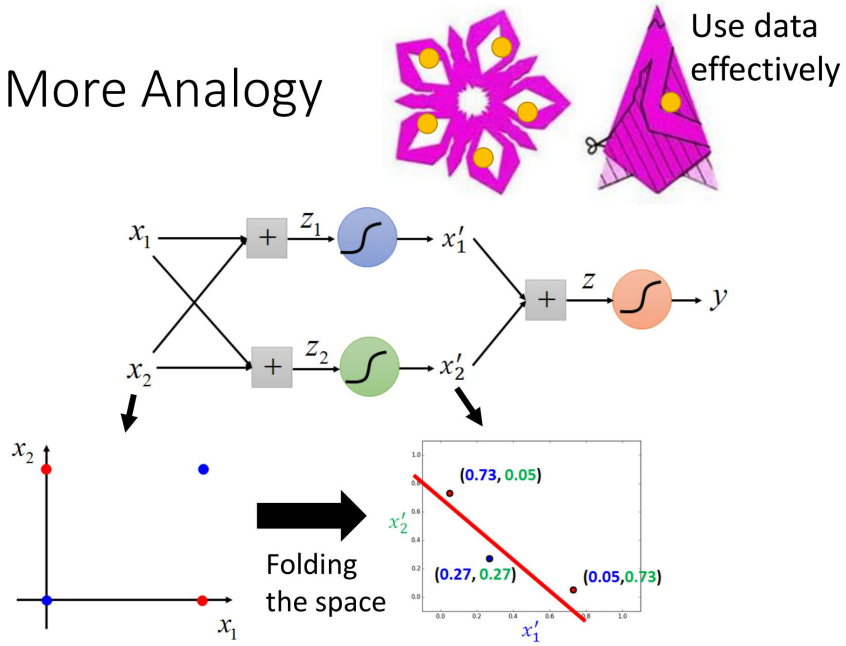


剪窗花的例子

剪窗花的時候，你並不是真的去把這個形狀的花樣剪出來，你是先把紙摺起來然後才剪。

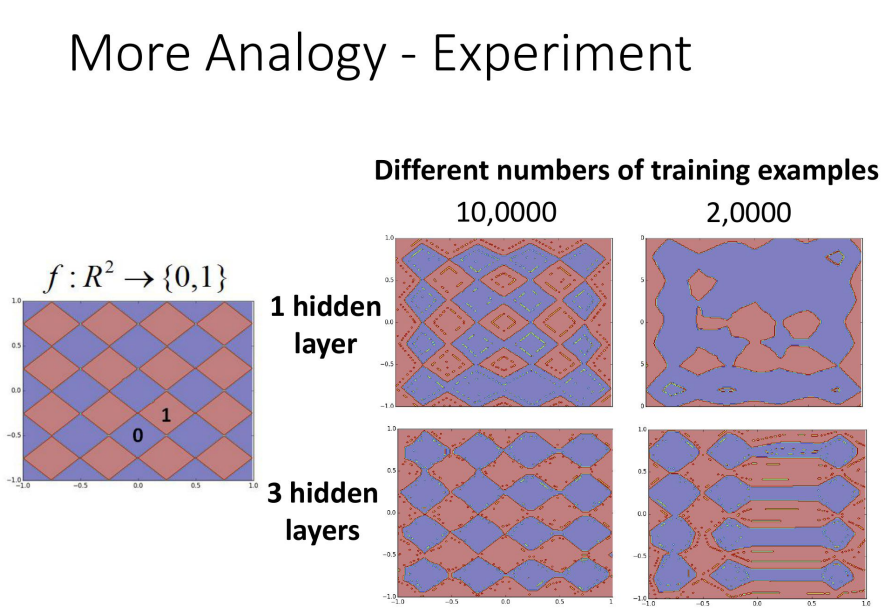
這個跟 deep learning 有什麼關係呢？

用之前講的這個例子來做比喻，下圖左下角的四個點，如果用一個 linear model，怎麼做都沒有辦法把藍紅完全分開，但是加了 hidden layer 的時候，就做了一個 feature transformation，這兩個藍色的點就重合在一起了，所以通過一個 hidden layer 變到這裡的時候，其實就好像是把原來的這個平面對折了一樣，這就好像剪窗花的時候先把色紙對折一樣。



在深度學習裡，只要告訴我們這個小的區間裡面的 data 怎麼分類，展開以後我們就可以做出複雜的圖樣，所以一筆 data 在上圖右上角的例子中，就可以發揮五筆 data 的效用，讓 deep learning 成為比較有效率使用 data 的方式。

Another example



這裡有一個 function，它的 input 是二維實數平面的點，它的 output 是 0 跟 1，而這個 function 是一個地毯形狀的 function。

如果我們用了不同量的 training example，分別在一個 hidden layer 跟三個 hidden layer 的時候，會看到甚麼樣的情境？

> 這邊要注意一下就是，我們有特別調整一個 hidden layer 和三個 hidden layer 的參數，他們的參數數量是調整到接近的。

十萬筆 data 的話，那這兩個 network 都可以學得不錯，因為對一個 hidden layer 來說，它只要它夠寬就可以模擬任何 function。

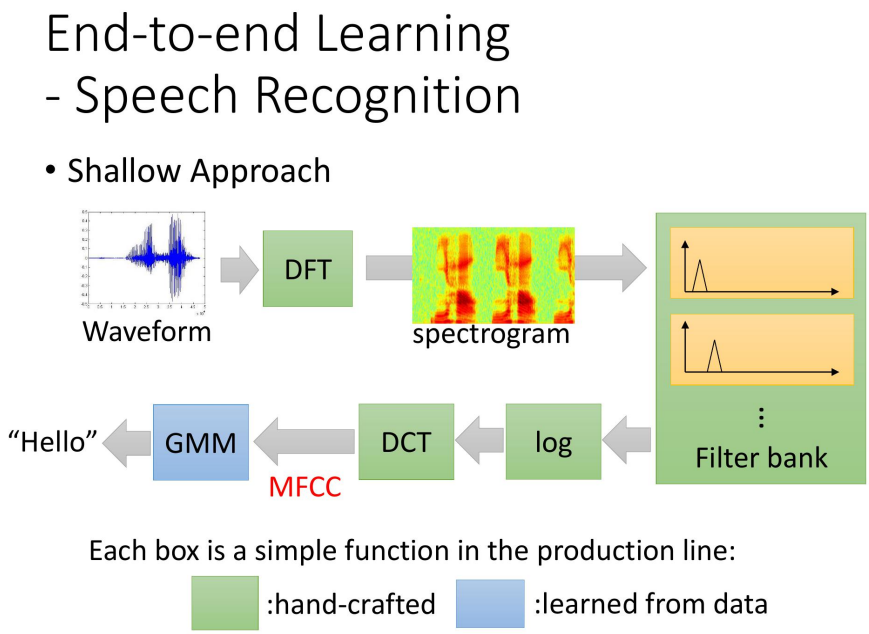
現在，如果我們減少參數的量到只用兩萬筆，這個時候只有 1 個 hidden layer 的結果就崩掉了，但是 3 個 hidden layer 的結果雖然也是變得比較差，但整體而言還是比一層的表現好。

End-to-end Learning

所謂 End-to-end learning 的意思，就是只給你的模型 input 跟 output，不告訴它中間每一個 function 要怎麼分工，然後讓它自己去學。

語音辨識

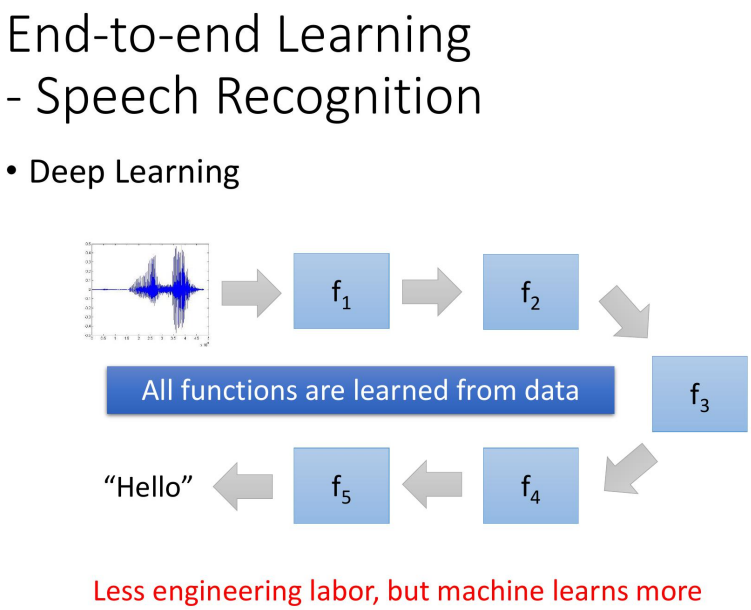
以語音辨識為例，語音辨識是一個非常複雜的問題，所以它的 model 裡面，它會變成一個生產線，原本複雜的 hypothesis function 是由很多比較簡單的 function 串接在一起，下圖就描述了語音辨識需要經過的諸多步驟。在這些步驟中，只有 GMM 是透過訓練資料學出來的，其他都是過去的專家研究了各種知識後得出的方法，很難替每個步驟再找到一個更好的方法來優化表現。



有了 deep learning 以後，我們可以把這些東西用 neural network 取代，也就是說，可以把 deep neural network 多加幾層，然後就把 DCT 拿掉，就不用再使用 MFCC 這個 feature，這已經是現在通行的做法了。把 neural network 疊深一點，直接從 log 的 output 開始做，會得到比較好的結果。

現在你甚至可以從 spectrogram 開始做，把 filter bank 的部分也拿 deep neural network 來取代，也可以得到更好的。Filter bank 是模擬人類的聽覺器官所製定出來的 filter，如果你分析 deep neural network 學出來的 weight，可以發現它自動學到要做 filter bank 這件事情。

當然有人就會想要挑戰，能不能夠疊一個很深很深的 neural network，直接 input 就是 time domain 上的聲音訊號，然後 output 直接就是文字，中間完全不要做 feature transform 之類的，做出如下圖般的 model。



最後，Google 有一篇 paper 是拚死去學了一個很大的 neural network，input 就是原始的聲波訊號，最後可以做到跟有做 feature transform 的結果打平，但也僅止於打平而已，或許 Fourier transform 就已經是訊號處理的極限了。

影像辨識

在影像的部分其實就跟剛才是相似的，有很多人訂的、人工設計的 feature，於是有幾個人工設計的 block 去處理影像的 feature，只在最後一步用一個 shallow 的 classifier。

現在，就直接用一個很深的 network，input 就直接是 pixel，output 就直接是裡面影像含有的內容，不需要抽 feature 了。

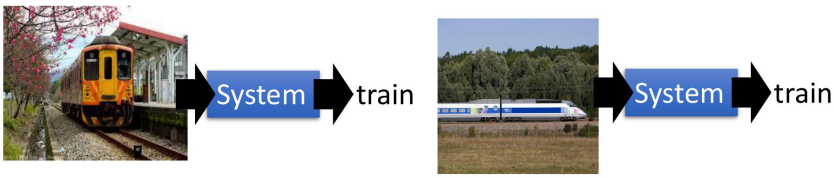
Deep Learning 的其他好處

有時候會有非常像的 input，它會有很不一樣的output；有時候看起來很不一樣的東西，其實是一樣的，比如說，這個是火車，如下圖的例子。

Complex Task ...

- Very similar input, different output

- Very different input, similar output



只有一層的 network 的話，只能夠做簡單的 transform，沒有辦法把一樣的東西變得很不一樣，把不一樣的東西變得很像。要讓原來 input 很像的東西，結果看起來很不像，需要做很多層次的轉換。

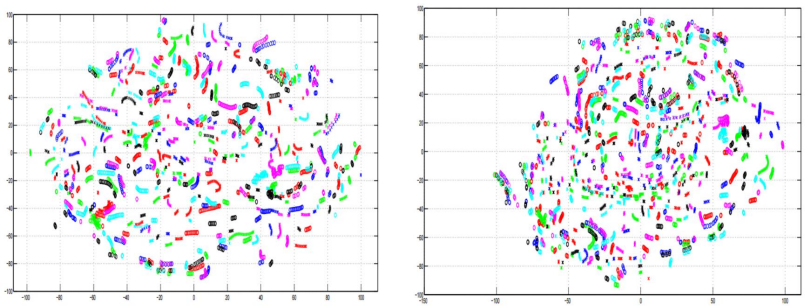
語音

另外一個與語音相關的例子，這是來自於 ICASSP 2012 的一篇 paper。這個圖是將 MFCC 投影到二維平面上，不同的顏色代表的是不同的人說的話，這些人說的是一樣的句子。你會發現不同的人說同樣的句子，它的聲音訊號看起來是非常不一樣的，但如果你今天學一個 neural network，雖然只看第一層的 hidden layer 的 output，會發現看起來還是很不一樣，但看到第八個 hidden layer 的 output，會發現不同的人說的同樣的句子被自動 align 在一起。

Complex Task ...

A. Mohamed, G. Hinton, and G. Penn, "Understanding how Deep Belief Networks Perform Acoustic Modelling," in ICASSP, 2012.

- Speech recognition: Speaker normalization is automatically done in DNN



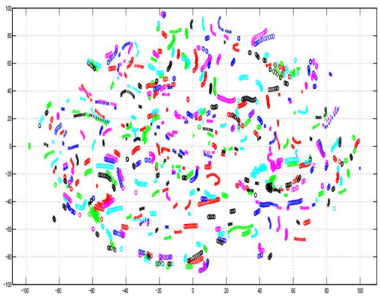
Input Acoustic Feature (MFCC)

1-st Hidden Layer

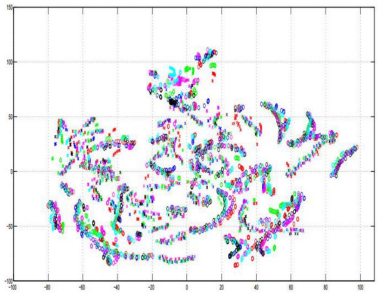
Complex Task ...

A. Mohamed, G. Hinton, and G. Penn, "Understanding how Deep Belief Networks Perform Acoustic Modelling," in ICASSP, 2012.

- Speech recognition: Speaker normalization is automatically done in DNN



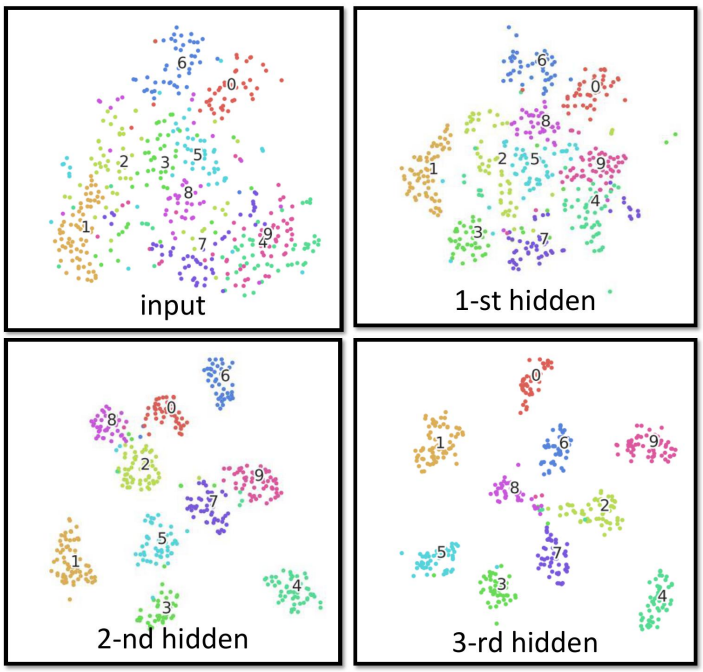
Input Acoustic Feature (MFCC)



8-th Hidden Layer

手寫辨識

MNIST



以 MNIST 手寫數字辨識為例，input 就是 28*28 個 pixel，投影到二維的平面的話，看起來 4 跟 9 幾乎是疊在一起的，但一層一層看下去後會發現 4、7、9 是逐漸被分開的。

所以如果你今天要讓不一樣的 input 被 merge 在一起，如在語音舉的例子，或者是讓原來看起來很像的 input 最後被分得很開，就需要多個 hidden layer 才能辦到這件事情。

更多用 **deep learning** 的理由

微軟的研究員 Rich Caruana 寫了一篇 paper，標題是"Do Deep Nets Really Need to be Deep？"，翻譯成中文可以翻譯成「深度學習是不是過譽了」。在這篇 paper 裡，他比較一層的 hidden layer 跟三層的 hidden layer，它們在 MNIST 還有 TIMIT 這些 benchmark corpus 上的差別，當然結果並不意外，在參數數量接近時，三個 hidden layer performance 是比較好的。

接下來，他發現說一個 hidden layer 參數怎麼增加，結果都不好，它很快就 saturate 了，他就做了一件當時看起來很匪夷所思的事。

在一個 hidden layer 的 neural network，他不要用真正的 label 當作學習的 target，反而用三層 hidden layer 的 output 當作你的 label，也就是用一層的 hidden layer 就去學三層的 hidden layer 幫你 label 的結果，最後它的 performance 反而會比較好。

這裡推出來的結論是，直接在一層 network 是訓練不起來的，要先訓練三層的 network，再用一層的 network 去模擬三層 network 的行為才學得起來。