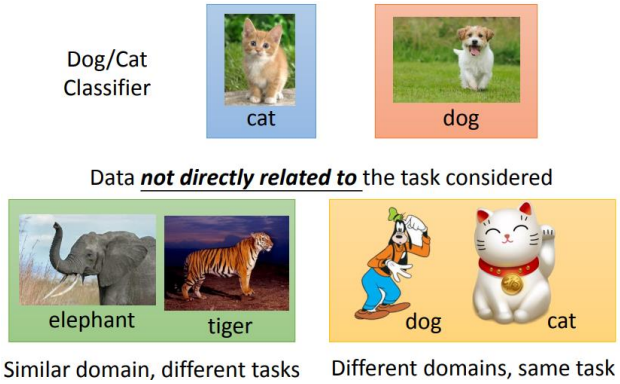
INDEX

Transfer Learning

Transfer learning 的意思是，假設手邊有一些跟現在進行的 task 沒有直接相關的 data，能否用這些沒有直接相關的 data，來幫助我們進行 training。

比如說現在做的是貓跟狗的 classifier，而 input distribution 中雖然同樣是動物的圖片，但是跟 task 的 label 無關。

還有另外一個可能是，它們的 input domain 不一樣，但 task 是一樣的。
例如一樣是做貓跟狗的分類，但是圖片卻是招財貓和高飛狗的圖片，跟原來圖片的 distribution 非常不像，但要做的 task 是一樣的。



可根據 target data （跟現在考慮的 task 是直接相關的）與 source data （跟 task 沒有直接的關係）有無 label 分成下列四個象限。

		Source Data (not directly related to the task)	
		labelled	unlabeled
Target Data	labelled	Fine-tuning Multitask Learning	Self-taught learning Rajat Raina , Alexis Battle , Honglak Lee , Benjamin Packer , Andrew Y. Ng, Self-taught learning: transfer learning from unlabeled data, ICML, 2007
	unlabeled	Domain-adversarial training Zero-shot learning	Self-taught Clustering Wenyuan Dai, Qiang Yang, Gui-Rong Xue, Yong Yu, "Self-taught clustering", ICML 2008

Fine-tuning

現在的 task 裡面，target 和 source data 通通都有 label。且 target data 的量非常少 (One-shot learning)；但 source data 非常多。因此我們想知道，在 target data 很少的情況下，如果有一大堆不相干的 source data，到底有沒有可能有幫助。

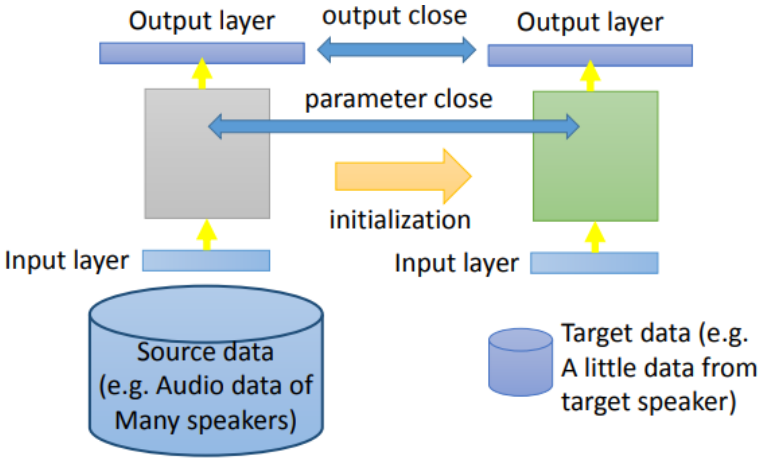
- Target data: (x^t, y^t) ⬅ Very little
- Source data: (x^s, y^s) ⬅ A large amount

因此最理想的情況是把 source data 上 train 出來的 model，當作是 training 的初始值，然後再用 target data train 下去即可。然而通常情況下都會做壞，因此必須用 target data 來 fine-tune model，可分作兩種方法。

一、Conservative Training

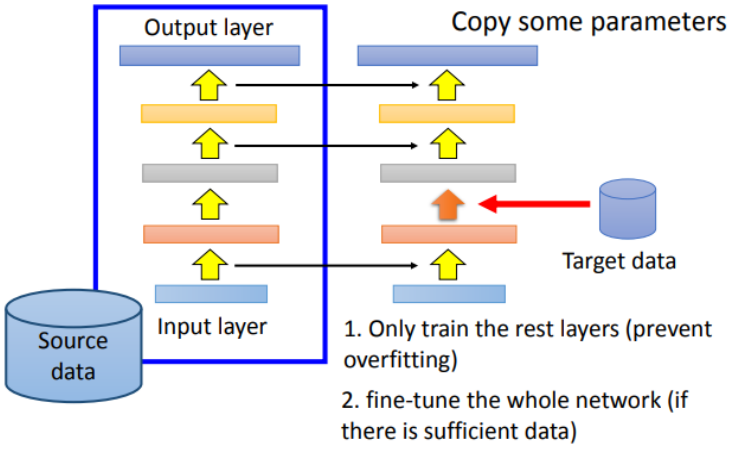
概略來說，我們希望用 source data train 出來的 model，與 target data train 出來的 model 不要差太多，因此在 training 的時候會加上一些 constrain。

如新的 model 跟舊的 model 在看到同一筆 data 時，它們的 output 越接近越好。或者是兩者之間的參數其 L2 norm 差距越小越好。如此可防止 target data 只有一點點的時候，產生 overfitting。



二、Layer Transfer

將 source data train 好的一個 model，抽出某幾層 layer 複製到新的 model，接下來用 target data 只去 train 那些沒有複製過來的 layer。



該方法的好處就是 target data 只需要考慮非常少的參數就可以避免 overfitting 的情形；若 target data 夠多亦可直接 fine-tune 整個 model。

至於要複製哪些 layer 取決於經驗與 task 的不同。
舉例而言，在語音辨識上通常是複製最後幾層，然後重新 train input 那一層，理由在於，每個人的口腔結構略有差異，同樣的發音方式但得到的聲音可能是不一樣的。
而 neural network 前面的 layer 都是在做從聲音訊號裡面，得知說話的人的發音方式。因此後面幾層是跟說話的人沒有關係，所以可以被複製。

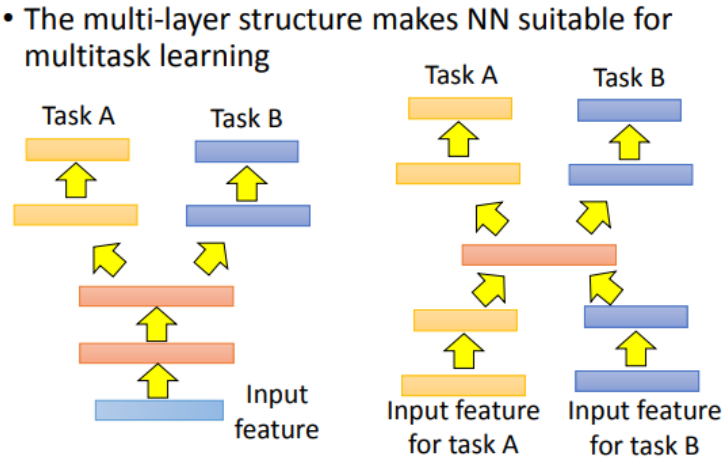
至於在 CNN 的部分則是需要複製前面幾層，重新 train 最後幾層。可想作前幾層 layer 做的就是 detect 最簡單的 pattern；但最後幾層 learn 的東西往往比較抽象，因此沒有辦法 transfer 到其他的 task 上面。

三、Multitask Learning

在 fine-tune 裡面我們 care 的是 target domain 做得好不好，就算在 fine-tune 之後 source domain 表現變差也無所謂。但 multitask learning 則是兼顧兩者的表現。

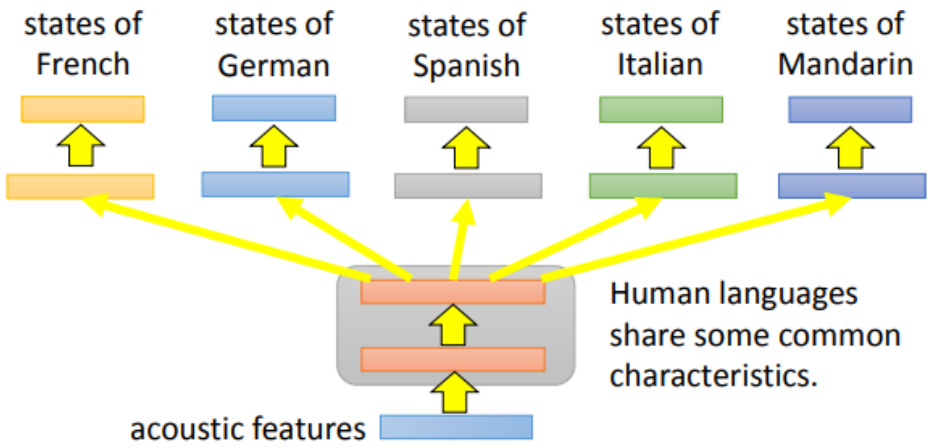
假設有兩個不同的 task，它們用的都是同樣的 feature，但在 neural network 中間分叉出來 output 不同 task 的內容。比如說一部分的 network output 的可能是 task A 的答案；一部分的 network，它的 output 則是 task B 的答案。
因此它們在前面幾層 layer 都是共用，直到最後再分類。
更極端的例子中，甚至可只共用其中一層 hidden layer 去 training。

這麼做好處在於，前面幾層 layer 是用比較多的 data 去 train 的，所以表現可能會比較好。



然而該例的前提是，兩個 task 之間必須具有共通性。

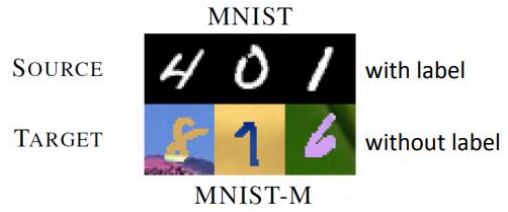
Multitask learning 一個很成功的例子便是多語言的語音辨識。在 train 不同語言的 data 時，可共用前面幾層 layer，最後再辨識不同的語言。
同樣的概念亦可套用在 translation 上。



Similar idea in translation: Daxiang Dong, Hua Wu, Wei He, Dianhai Yu and Haifeng Wang, "Multi-task learning for multiple language translation.", ACL 2015

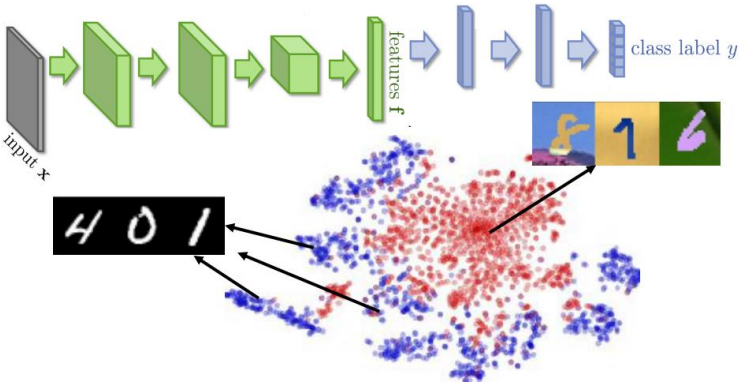
四、Domain-adversarial Training

該例中 source data 跟 target data 做的 task 相同，但兩者內容不同。
如下例 MNIST，兩者都做分類數字的 task 但背景的圖片不一樣。



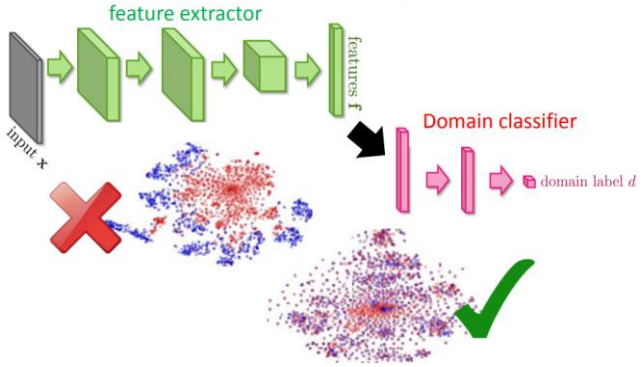
如果把一個 neural network 當作是一個 feature extractor。它的前面幾層可以看作是在抽 feature，而後面幾層則是在做 classification。
此時若把 feature 抽出來看會發現不同 domain 的 data，它們的 feature 完全不一樣。如果把 MNIST 的 feature 丟進去的話，是藍色的這些點，並且很明顯地分成十群，分別代表零到九，十個數字。

但如果把另外一個 corpus 的 image 丟進去的話，抽出來的 feature 就是紅色這群（皆經 t-SNE 降維）。

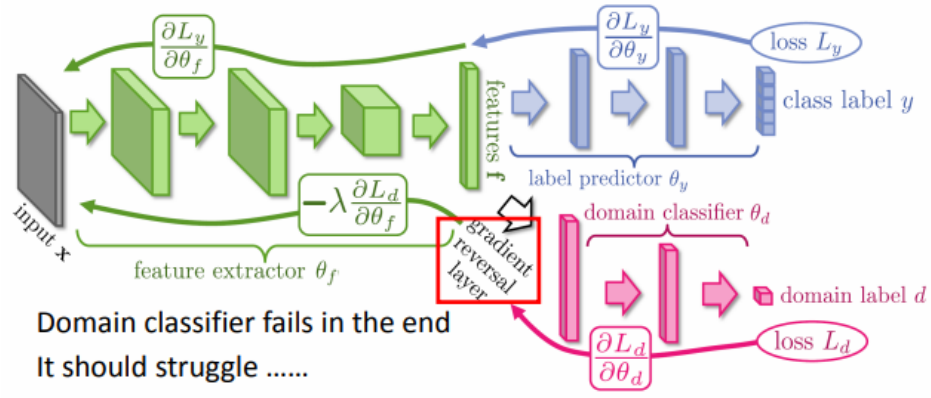


因此會發現 source domain 跟 target domain 根本就不在同一個位置裡面，且抽出來的 feature 也完全不一樣。

後面的 classifier 雖然可以把藍色的部分做好，但對於紅色的部分就無能為力。
因此希望做到在前面進行 feature extractor，把 domain 的特性去除掉。



在此方法中我們會將 source data 當作 training data；target data 當作 testing data。並將 network 分成兩個分支。
紅色的部分做的是將 data 的 domain 分類（分出 source data 跟 target data）；
藍色的部分則是進行 predict label 的任務，
最後綠色的部分必須騙過紅色的 classifier，讓它誤認 source data 跟 target data 為同一個 domain，並同時使藍色的 predictor 的正確率變高。



Yaroslav Ganin, Victor Lempitsky, Unsupervised Domain Adaptation by Backpropagation, ICML, 2015

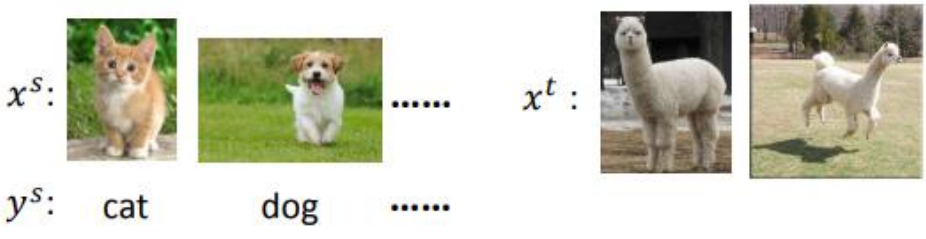
Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, Domain-Adversarial Training of Neural Networks, JMLR, 2016

至於在 train 該 model 時，我們可以將紅色做 back propagation 得到的 value 直接乘上一個負號在傳給綠色，可想成 feature extractor 做的事完全跟 domain classifier 相反。

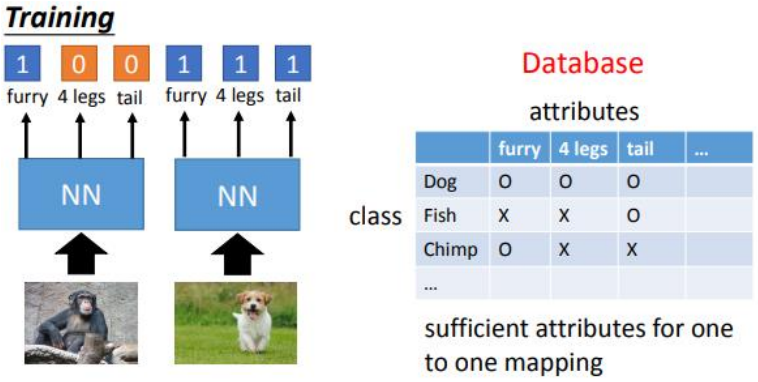
該方法的缺點在於不好 train，必須想辦法讓 domain classifier 在放棄分辨 domain （output 變成永遠都是 0 的情況）與持續分辨 domain 之間取得平衡，最終再讓他 fail。

五、Zero-shot Learning

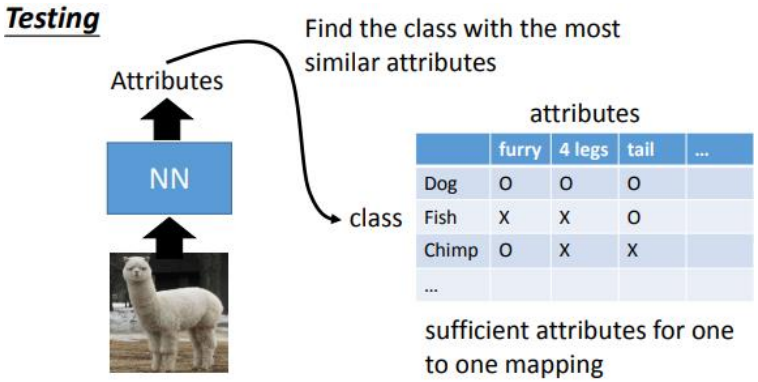
前述提到只有在 source data 有 label，但 target data 沒有 label 的時候，才可以把 source data 當 training data。但實際上 Zero-shot learning 的限制又更嚴苛一點。它限定其 source data 與 target data 的 task，必須都是不一樣的。
像是下例圖像辨識中，source data 的 task 是分辨貓與狗；而 target data 的 task 卻是分類毫不相干的羊駝。



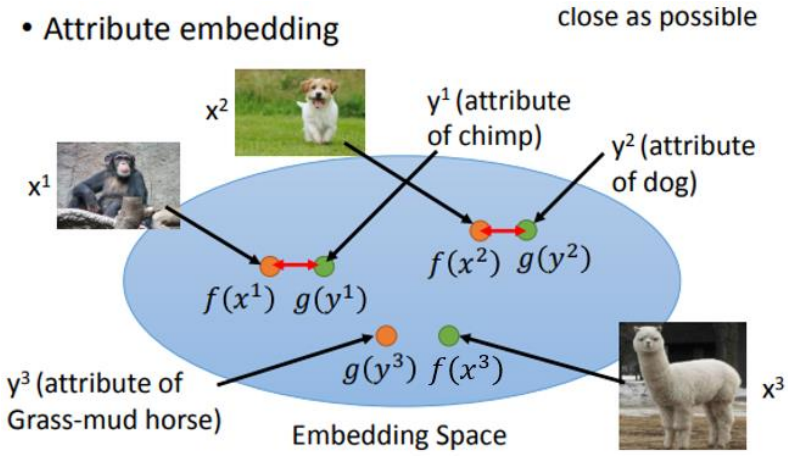
以圖像辨識動物為例，此方法不直接辨識每一張圖片於哪一個 class，而是改成抽出它們的 attributes，並另建一 database 記載所有動物的 attributes。
這裡的 attribute 必須要要定的夠豐富，使每一個 class 都要有獨一無二的 attribute，如果有兩個 class 的 attribute 一模一樣的話，該方法就會 fail 掉。



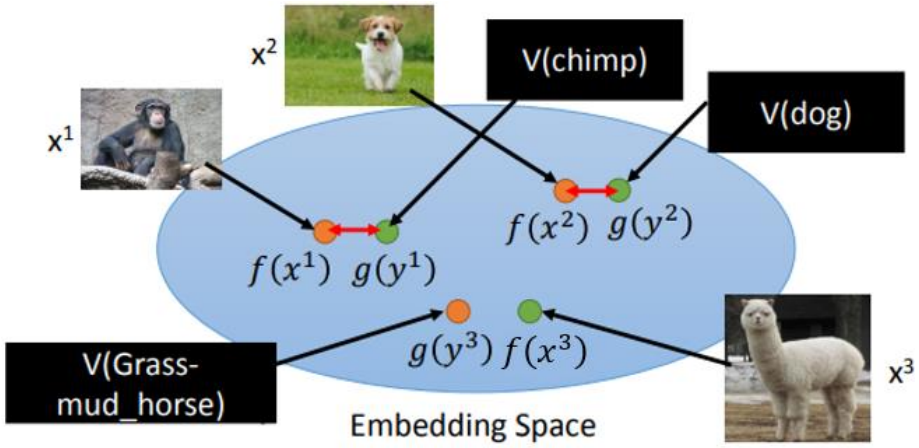
在 testing 的時候，即便來了一張從來沒有看過的動物圖片，只要抽出它的 attributes 之後對照 database 便可知道其類別為何。



至於比對 attributes 跟 database 的步驟，稱作 attribute embedding。
將 source data 的圖片與所有的 attribute 投影到相同的 space (embedding space) 中，進行 training 的時候使兩者距離越近越好；
當 input 為 target data 時 (testing)，同樣也是投影至 space 中，再比對哪個 attribute 與它最近。



如果沒有 database 的情況下，也可以把 attribute 換作 word vector，進行 word embedding。



1. Training Criterion

將上述化成數學式的話，可表示為：我們希望 x^n 通過 f(x)與 y^n 通過 g(y)之後，它們的距離越接近越好。因此就是找一個 f 跟 g 可以 minimize $f(x^n)$與 $g(y^n)$ 的距離。

$$f^*, g^* = arg \min_{f,g} \sum_n \|f(x^n) - g(y^n)\|_2$$

然而實際上這樣做是會有問題，因為這樣 model 就只會把所有不同的 x 跟所有不同的 y，都投影到同一點就停住了。因此需要重新設計一下 loss function。

我們必須除了考慮到 x^n 跟 y^n 是一個 pair，要讓它們越接近越好外，若 x^n 跟另外一個 y^n'不是同一個 pair 的話，它們的距離應該要被拉大。

定義如下：

$$f^*, g^* = \arg \min_{f,g} \sum_n \max(0, k - f(x^n) \cdot g(x^n) + \max_{m \neq n} f(x^n) \cdot g(x^n))$$

其中 k 是一個 constant，training 的時候必須要自己 define。

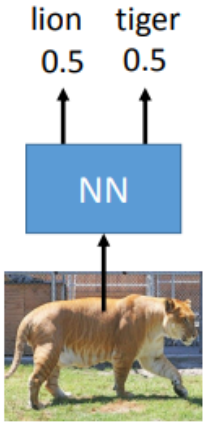
當下式小於 0 的時候，就會是 zero loss 的情形。

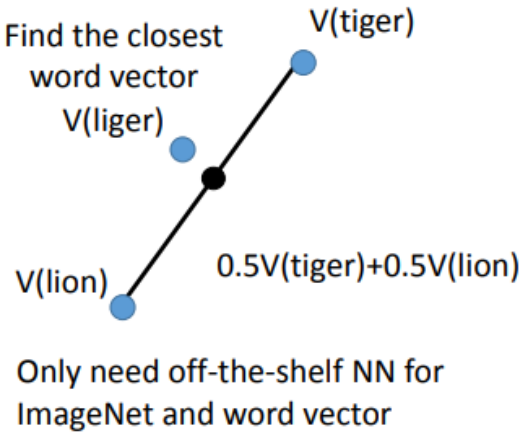
$$k - f(x^n) \cdot g(x^n) + \max_{m \neq n} f(x^n) \cdot g(x^n)$$

可以想作當 $f(x^n)$跟 $g(y^n)$的內積值大過所有其他的 $g(y^m)$跟 $f(x^n)$的內積一個 margin k 時，就代表這個式子不只是把 pair 起來的 attribute 跟 image 拉近，同時也要把那些不成 pair 的東西拆散。

Convex Combination of Semantic Embedding

該方法使用現成的 neuron network 進行操作，省去了做 learning 的步驟。
假設將「liger」的圖片丟進現成的圖像辨識 network，其 output 為 0.5 屬於「lion」；有 0.5 的機率是「tiger」。
接著將「lion」跟「tiger」的 word vector 用上述比例組合為新的 vector，並進行 embedding 找出最相近的 word vector 即為所求。





Only need off-the-shelf NN for
ImageNet and word vector