1. 大家好，我是碩一的蔡伊婷
2. 今天我要報的主要是在一個視覺辨識競賽中，5個冠軍代表作。在這之前我先用一個比較簡單的例子來複習一下湘元剛剛所介紹到的名詞。
3. 我們都知道CNN常常被用在被用在影像處理上，但為甚麼我們不用一般的full network? CNN其實是簡化的DNN，換句話說就是拿掉部分參數的DNN，而他拿掉參數的依據是甚麼?

* 假設在第一層的hidden layer裡的每個newrun是在偵測不同的pattern(ex)而我們不需要看整張圖。
* 同樣的pattern可能會出現在不同的圖的不同地方，像是以一張貓的圖，pattern=眼睛，用相同的newrun就可以偵測到眼睛，要求左右眼使用相同的newrun如此可以(共用參數)。
* 像素由2000\*2000所小到500\*500可能會影響到解析度，但對於我們辨識圖片來說不用有太大的影響

1. 前面兩件事是由convolution layer處理，而做subsampling是由Max pooling來做處理，convolution & Max pooling可以重複做很多次，而次數跟Deep learning的層數一樣是事先決定。
2. 假設現在圖片是一個6\*6\*1 image(黑白的)，有一組Filter，不同的Filter可以看做不同的newrun，想像成Filter1是在偵測直線，Filter2是橫線，而矩陣裡的值也就代表著不同的權重和bias。

而Filter裡的值是機器去學出來的。

而我們要偵測是否有Filter1，就像紅色框框只看這九個值，stride=1(設定值)平移一步在偵測下一個九宮格，就提到前面所說的第一件事，觀察某個pattern不需要看整張圖。

1. 接下來我們就是把圖和Filter1做內積
2. 經過第一個convolution後，可以得到一個4\*4的matrix。而我們的Filter1可以想像是在偵測斜線，在原始圖片中左邊出現了兩個這樣得矩陣，所以在右邊這個圖的左半邊為3(數值最大)同時表示有出現這樣的圖案。

同時也表示我們用同一個Filter1就偵測出來，也就是我們所表示的使用相同。

1. 不同的Filter可得不同的4\*4矩陣，合在一起就稱做是一個feature map，所以feature map是一個4\*4\*n的矩陣，而n代表的是Filter個數。
2. 回到一開始說的，CNN其實是簡化的DNN，換句話說就是拿掉部分參數的DNN，我們把圖片拉直變成36\*1，而經過第一個Filter1所得的結果我們把它變成16\*1的矩陣，而我們說Filter1也可以說是weight，把圖和Filter1個別值相乘後再做相加，我們可以發現和fully connected相比只有連接9個input，在裡就使用了**較少的參數**。我們要辨識一個pattern不需要看整張圖，如此我們也能使用較少的參數。
3. Stride=1，乘相同filter，和剛剛是乘相同的filter，也可以表示成共用參數，-1是另外一個newrun的output。
4. 接下來我們要做sampling，就是把下列的值4個一組，取框框內最大的就是max pooling。
5. 最後可以得到這個結論，一個6\*6的matrix經過一個convolution在經過一個max pooling可以得到一個新的image 2\*2深度就是filter的個數，每一個filter可以想成一個channel。
6. 校正
7. 校正
8. 比賽誰辦得從甚麼，時候開始，比賽內容，為甚麼停辦?

ImageNet Large Scale Visual Recognition Challenge (ILS-VRC) 挑選出ImageNet當中一小部分的影像做為競賽的訓練樣本。共有1000種類別、每種類別有1000張左右的影像，總計Training Set約120萬張、Validation Set 5萬張、Testing Set 1.5萬張。

1. 圖傳達錯誤率

## LeNet-5

**​**LeNet-5 架構也許是最廣為人知的CNN 架構。如前所述，它是由Yann LeCun 於1998 年創建的，廣泛用於手寫數字識別（MNIST）。

**輸入**：32\*32的手寫字體圖片，這些手寫字體包含0~9數字，也就是相當於

10個類別的圖片

**輸出**：分類結果，0~9之間的一個數

因此我們可以知道，這是一個多分類問題，總共有十個類，因此神經網絡的最後輸出層必然是SoftMax問題，然後神經元的個數是10個。LeNet-5結構：

**輸入層**：32\*32的圖片，也就是相當於1024個神經元

**C1層**：選擇6個特徵卷積核，然後卷積核大小選擇5\*5，這樣我們可以得到6個特徵圖，然後每個特徵圖的大小為32-5+1=28，也就是神經元的個數由1024減小到了28\*28=784。

**S2層**：這就是下採樣層，也就是使用最大池化進行下採樣，池化的size，選擇(2,2)，也就是相當於對C1層28\*28的圖片，進行分塊，每個塊的大小為2\*2，這樣我們可以得到14\*14個塊，然後我們統計每個塊中，最大的值作為下採樣的新像素，因此我們可以得到S1結果為：14\*14大小的圖片，共有6個這樣的圖片。

**C3層**：(用第4頁解釋)前6个feature map与S2层相连的3个feature map相连接，后面6个feature map与S2层相连的4个feature map相连接，后面3个feature map与S2层部分不相连的4个feature map相连接，最后一个与S2层的所有feature map相连。

**S4層**：下採樣層，比較簡單，也是知己對C3的16張10\*10的圖片進行最大池化，池化塊的大小為2\*2。因此最後S4層為16張大小為5\*5的圖片。至此我們的神經元個數已經減少為：16\*5\*5=400。

**C5層**：我們繼續用**5\*5的卷積核**進行卷積，然後我們**希望得到120個特徵圖**。這樣C5層圖片的大小為5-5+1=1，也就是相當於1個神經元，120個特徵圖，因此最後只**剩下120個神經元了**。這個時候，神經元的個數已經夠少的了，後面我們就可以直接利用全連接神經網絡。

在現實使用中，每一層特徵圖需要多少個，卷積核大小選擇，還有池化的時候採樣率要多少，等這些都是變化的，這就是所謂的**CNN調參**，我們需要學會靈活多變。

參數個數

**C1**: 总共就有6\*（5\*5+1）=156个参数，对于卷积层C1，每个像素都与前一层的5\*5个像素和1个bias有连接，所以总共有156\*28\*28=122304个连接（connection）。

**S2**: 此处共有6\*2=12个参数。S2中的每个像素都与C1中的2\*2个像素和1个偏置相连接，所以有6\*5\*14\*14=5880个连接（connection）。

**C3**: 卷积核大小依然为5\*5，所以总共有6\*（3\*5\*5+1）+6\*（4\*5\*5+1）+3\*（4\*5\*5+1）+1\*（6\*5\*5+1）=1516个参数。而图像大小为10\*10，所以共有151600个连接。

**S4**是pooling层，窗口大小仍然是2\*2，共计16个feature map，所以32个参数，16\*（25\*4+25）=2000个连接。

**C5**是卷积层，总共120个feature map，每个feature map与S4层所有的feature map相连接，卷积核大小是5\*5，而S4层的feature map的大小也是5\*5，所以C5的feature map就变成了1个点，共计有120（25\*16+1）=48120个参数。

**F6**相当于MLP中的隐含层，有84个节点，所以有84\*（120+1）=10164个参数。

F6层采用了正切函数

输出层采用了RBF函数，即径向欧式距离函数

**AlexNet**

AlexNet CNN 架構贏得了2012 年的ImageNet ILSVRC 挑戰賽：它達到了17% 的top-5 的錯誤率，而第二名錯誤率只有26%！

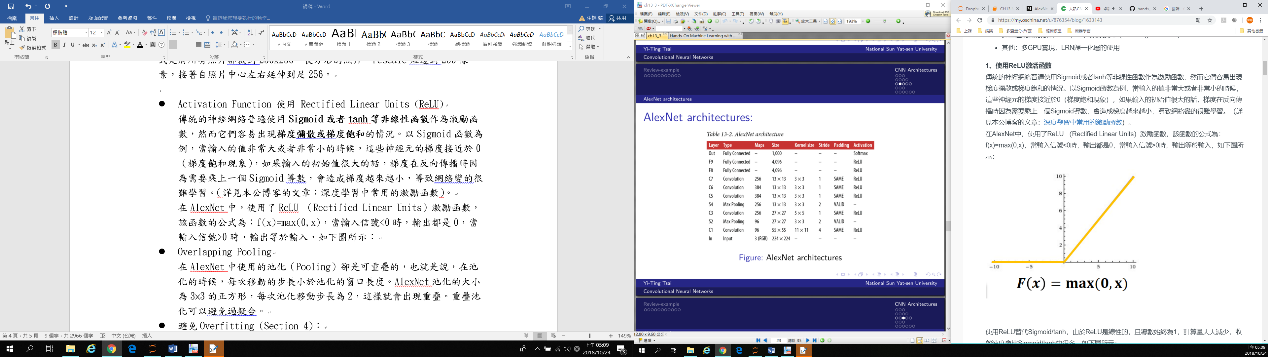
它與LeNet-5 非常相似，只是更大更深，它是第一個將捲積層直接堆疊在一起，而不是在每個卷積層頂部堆疊一個池化層。

每張照片解析度不一，但AlexNet只接受單一輸入尺寸。Preprocess的方式是將所有照片都裁到256x256。長方形的照片，rescale短邊到256像素，接著自照片中心左右延伸到足256。

* Activation Function 使用 Rectified Linear Units (ReLU)

傳統的神經網絡普遍使用**Sigmoid或者tanh等非線性函數**作為激勵函數，然而它們容易出現**梯度彌散或梯度飽和**的情況。以Sigmoid函數為例，當輸入的值非常大或者非常小的時候，這些神經元的梯度接近於0（梯度飽和現象），如果輸入的初始值很大的話，梯度在反向傳播時因為需要乘上一個Sigmoid導數，會造成梯度越來越小，導致網絡變的很難學習。（詳見本公博客的文章：深度學習中常用的激勵函數）。

在AlexNet中，使用了ReLU （Rectified Linear Units）激勵函數，該函數的公式為：f(x)=max(0,x)，當輸入信號<0時，輸出都是0，當輸入信號>0時，輸出等於輸入，如下圖所示：



使用ReLU替代Sigmoid/tanh，由於ReLU是線性的，且導數始終為1，計算量大大減少，收斂速度會比Sigmoid/tanh快很多。

可以看出ReLU的潛在問題。一般實務上，hidden layer的activation function 都是使用ReLU，Output layer 在problem 若確定輸出**必大於零**，可以用ReLU。

* Overlapping Pooling

在AlexNet中使用的池化（Pooling）卻是可重疊的，也就是說，在池化的時候，每次移動的步長小於池化的窗口長度。AlexNet池化的大小為3×3的正方形，每次池化移動步長為2，這樣就會出現重疊。重疊池化可以避免過擬合。

* 避免Overfitting (Section 4)：
  + Data Augmentation(數據增加)

有一種觀點認為神經網絡是靠數據餵出來的，如果能夠增加訓練數據，提供海量數據進行訓練，則能夠有效提升算法的準確率，因為這樣可以避免過擬合，從而可以進一步增大、加深網絡結構。而當訓練數據有限時，可以通過一些變換從已有的訓練數據集中生成一些新的數據，以快速地擴充訓練數據。

其中，最簡單、通用的圖像數據變形的方式：水平翻轉圖像，從原始圖像中隨機裁剪、平移變換，顏色、光照變換，

* + Dropout

AlexNet在第一＆二的fully connected layer加入dropout。每次forward propagation的時候fully connected layer之前層的每**個神經元會以一定的機率不參與forward**，而backward propagation的時候這些單元也不參與。這種方式使得網絡強制以部分神經元來表示當前的圖片，很大限度上降低過擬合。**AlexNet的dropout rate是0.5，也就是每個神經元有50%的機率不參與下一層的傳遞**。

* Local Response Normalization (LRN)

論文作者表示，雖然ReLU函數對於很大的輸入x，仍然可以有效的學習，但是他們發現即使這樣，對數據進行正規化對於學習來說還是有幫助的。

AlexNet每層ReLU後都接一層LRN，近期的論文指出LRN對於降低誤判率沒有幫助，實際的LRN效用有待驗證。

在後來的設計中，這一層已經被其它種的Regularization技術，如dropout, batch normalization取代了。

是位於特徵映射i的神經元的標準化輸出

是在ReLU 步驟之後，但在歸一化之前的那個神經元的激活。

k，α，β和r是超參數。k稱為偏置，r稱為深度半徑。

是特徵映射的數量。

## GoogLeNet

* ​GoogLeNet 架構是由Christian Szegedy 等人開發的。來自Google Research，通過低於7% 的top-5 錯誤率，贏得了ILSVRC 2014 的挑戰賽。這個偉大的表現很大程度上因為它比以前的CNN 網絡更深（見圖13-11）。這是通過稱為初始模塊（inception modules）的子網絡實現的，這使得GoogLeNet 比以前的架構更有效地使用參數。
* **0、輸入**

原始輸入圖像為224x224x3，且都進行了零均值化的預處理操作（圖像每個像素減去均值）。

**1、第一層（卷積層）**

使用7x7的捲積核（滑動步長2，padding為3），64通道，輸出為112x112x64，卷積後進行ReLU操作

經過3x3的max pooling（步長為2 ），輸出為((112 - 3+1)/2)+1=56，即56x56x64，再進行ReLU操作

**2、第二層（卷積層）**

使用3x3的捲積核（滑動步長為1， padding為1），192通道，輸出為56x56x192，卷積後進行ReLU操作

經過3x3的max pooling（步長為2），輸出為((56 - 3+1)/2)+1=28，即28x28x192 ，再進行ReLU操作

**3a、第三層（Inception 3a層）**

分為四個分支，採用不同尺度的捲積核來進行處理

（1）64個1x1的捲積核，然後RuLU，輸出28x28x64

（2） 96個1x1的捲積核，作為3x3卷積核之前的降維，變成28x28x96，然後進行ReLU計算，再進行128個3x3的捲積（padding為1），輸出28x28x128

（3）16個1x1的卷積核，作為5x5卷積核之前的降維，變成28x28x16，進行ReLU計算後，再進行32個5x5的捲積（padding為2），輸出28x28x32

（4）pool層，使用3x3的核（padding為1），輸出28x28x192，然後進行32個1x1的捲積，輸出28x28x32。

將四個結果進行連接，對這四部分輸出結果的第三維並聯，即64+128+32+32=256，最終輸出28x28x256

* 1x1的捲積核有什麼用呢？

1x1卷積的主要目的是**為了減少維度**，還用於修正線性激活（ReLU）。比如，上一層的輸出為100x100x128，經過具有256個通道的5x5卷積層之後(stride=1，pad=2)，輸出數據為100x100x256，其中，卷積層的參數為128x5x5x256= 819200。而假如上一層輸出先經過具有32個通道的1x1卷積層，再經過具有256個輸出的5x5卷積層，那麼輸出數據仍為為100x100x256，但卷積參數量已經減少為128x1x1x32 + 32x5x5x256= 204800，大約減少了4倍。

## ****ResNet****

* ​最後是，2015 年ILSVRC 挑戰賽的贏家Kaiming He 等人開發的Residual Network（或ResNet），該網絡的top-5 誤率低到驚人的3.6%，它使用了一個非常深的CNN，由152層組成。能夠訓練如此深的網絡的關鍵是使用跳過連接（skip connection，也稱為快捷連接）：一個層的輸入信號也被添加到位於下一層的輸出。
* 從第一個圖可以知道

網絡深度非常重要，但也存在一些問題，這個問題的障礙之一是臭名昭著的所謂梯度消失(爆炸)問題

* 目標是使其模擬一個目標函數h(x) 如果將輸入x添加到網絡的輸出中（即添加跳過連接），那麼網絡將被迫模擬f(x)= h(x) - x而不是h(x)
* 举个例子：

输入 x = 2.9 , 经过拟合后的输出为 H(x)=3.0

那么残差就是 F(x)=H(X)-x=0.1

如果拟合的是恒等变换，即输入 x=2.5 ，输出还是 H(x)=2.5

那么残差就是 F(x)=H(X)-x=0.0

而如上图所示，假设 x 从 2.9 经过两层卷基层(conv)之后变为 3.1 ，

平原网络的变化率 \Delta=\frac{3.1-3.0}{3.0}=3.3\text{%}

而残差模块的变化率为 \Delta=\frac{0.2-0.1}{0.1}=100\text{%}

##補充

padding必须是"VALID"或"SAME"：

如果设置为"VALID"，卷积层不使用零填充，并且可能会忽略输入图像底部和右侧的某些行和列，具体取决于步幅，如图 13-7 所示（为简单起见， 这里只显示水平尺寸，当然，垂直尺寸也适用相同的逻辑）

如果设置为"SAME"，则卷积层在必要时使用零填充。 在这种情况下，输出神经元的数量等于输入神经元的数量除以该步幅，向上舍入（在这个例子中，ceil(13/5)= 3）。 然后在输入周围尽可能均匀地添加零。