|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 模型 | 問題 | 解決方法 | 衍伸模型 | |
| Preceptron | 1. 只能解決非線性的問題, 但大部分的問題都是非線性  2. 無法解決 XOR 的問題 | 1. 線性組合後透過非線性的激活函式(acctivation), logestic function (sigmoid 等)  2. 利用 SVM 核函數(常用 radial basis function)使維度上升的方法做區分,      解決非線性及 XOR 的問題  3. 發現 MLP(Multiple Layers Preceptron) 模仿人類神經元,      可以有效解決這兩個問題 |  |
| MLP  (Multiple Layers Preceptron) | 單層的 w(權重) , 往負梯度的方向找, 多層呢? | 多層 w 可裡用連續偏微分 : dy/dx = dy/da \* da/db \* db/dx  => 只需專注單層的函數關係 |  |
| 連續偏微分太多層, 太遠的層別權重影響性太低  => 0 x logestic(0~1) => 0 => 形成梯度消失 | 1. 中間層的acctivation微分 = 1(不折扣) => y= x , 但會使 x<0 的部分也會打折, => x<0, y=0 => Relu |  |
| 2. 不要讓跨層的這麼容易忘 => 跨層神經元連接 => RNN, LSTM, ResNet | RNN |
| 1. 資料預處理 : 讀取, 正規化(0~1), one-hot-encoding  2. 建立模型(Sequential-基礎, Model-可以用平行合併) : model, Layers(Dense 全連接層), 神經元數, (第一層要輸入 input w個數)      acctivation(中間層 : relu, 最後一層 : 2元 - sigmoid, 多元 - softmax), 要知道怎麼算 parm 個數  3. 確認模型訓練方式(compile)  : loss(交叉商, 最後輸出一個: binary\_crossentropy, 最後輸出多個: categorical\_crossentropy),      \*\*\*\* 交叉商 v.s. 梯度下降 : 用 crossentropy 時機 : 1. 機率相加 = 1, 2. one-hot encoding      optimizer : 無腦選 adam, 利用隨機梯度下降搭配動能的優化      metrics : 評分方式 [“accuracy”]正確率, 也可加入其他評分方式(categorical\_accuracy 等), 主要看 val\_loss (test 的梯度損失減緩程度)  4. 訓練模型(fit) : mlp.fit(batch: 看幾筆(200)做一次梯度下降(修正) , epochs: 整份考古題看到最好, 看幾次(10))      # 10個 epochs, 200batch: 60000 \* 10 / 200 -> 300(次), 停下的時機: 只要看 val\_loss 變平滑  5. 計算最佳解 mlp.evaluate(x\_test\_shape, y\_test\_cat)  6. 利用 pre\_class 和 evaluate 做出混淆矩陣並抓出辨識錯誤的圖 | |  |
| MLP對於複雜的圖像產生的問題 :每一個圖像攤開, 計算量大, 訓練太多時間, 難以建立大量資料 (人類不是用像素在看是用顏色外框等特徵) | | CNN |
| CNN / LeNet | CNN (Convolutional Neural Network) 又稱 LeNet ,1998 年由 LeCun 提出被 Hinton 發揚光大, 概念為利用卷積窗在圖面上移動將像素或分數變成分數圖, 不再是直接丟進MLP => 先抓取重要的(特徵值):萃取 => 再丟進MLP  濾鏡 filter : 先經過濾鏡把不重要的過濾掉 => 再進入大腦, \*\*\*卷積=過濾的動作 | |  |
| 5 x 5 算完變 3 x 3 , 圖片變小無法做多層 | 圖片外加一圈 0 使之後分數圖根元圖一樣大(padding=”same”) |  |
| 有多個濾鏡 :  28x28x1 -- 32個濾鏡(3x3) --> 28x28x32 次計算   => 計算量太大 | 池化 (Macspooling 2x2 最大值池化) : 28x28x32個分數圖 =>14x14x32  池化 (Averagepooling 2x2平均值池化 , 會使圖片太模糊, 不常用)  \*\* 新一代模型主要以改變步長來取代池化 |  |
| 為避免過擬合, 導致機器背答案 | Dropout : 每一次隨機丟掉一些神經元(同隨機森林), \*\* 新一代模型不太用 |  |
| Relu 死亡 : CNN 進來的負太大 => 落於 activation 屏蔽區 => MLP 神經元死亡 => 導致辨識度極差 | BatchNormalization (常態化): 把整個資料調整到合理, 符合 keras 可接受的範圍  1. 把整個 Batch 常態化: 平均 = 0, 標準差 = 1,  2. ? 所有 -> 常態 XX => r: 標準差 放大 , B: 平移, |  |
| 建立模型 : 眼睛(CNN) -> 血管(Batch\_normalization) -> MLP 大致與 MLP 相同  確認訓練方式 : loss=SparseCategoricalCrossentropy()    # 用SparseCategoricalCrossentropy不用 one-hot encoding  訓練 : callback 可以設定停止及儲存的批次, \*\*好用\*\*  stop\_callback = EarlyStopping(patience=3, restore\_best\_weights=True)  save\_callback = ModelCheckpoint("fashion.h5", save\_best\_only=True)  # patience=3 容許震盪上升次數, restore\_best\_weights=True 自動回到最好的點  # moelCheckpoint : save\_best\_only | |  |
| 問題 : 處理速度太慢 : sliding window 大小太難決定, 整張圖都要 slidding window , 太多圖, 圖不會旋轉 | | R-CNN |
| 經典神經網路 : 由前期的形態學(邊緣檢測, 輪廓)到特徵的抓取  主要影響 :  1. ImageNet 比賽 : 李菲菲 舉辦, 2012~2017 ,  2. Hinton 實驗室首創 GPU 運算打敗吳恩達的 16000 CPU  3. 特徵群聚性可分為深層特徵與淺層特徵, 各有其重要性  4. 發現人類視覺的本質 : (1) 視線聚焦(感知域-卷積, 錐狀神經細胞-CNN) , (2) 空間可交換性(貓在角落還是貓-權值共享) | | | | |
| AlexNet | Hinton實驗室,  1. 首先以 GPU 運算的模型  2. Data Augmentation 數據增強, 避免 CNN 背答案, 增強方法包括灰度變化、水準翻轉、垂直翻轉、色階調整、裁切、旋轉  3. Relu , Dropout, Local Response Normalization, Batch Normalization(防止梯度瀰散)首發 | |  |
| Network in Network | average pooling : 透過特徵值平均結果提高泛化能力, Full connected layer : 增加卷積複雜度  首先提出網中網的概念 | | GoogLeNet |
| VGG | Visual Geometry Group, 首先出現較深的模型, 經典練習題 | |  |
| GoogLeNet | Xecption : 利用 網中網 及 1x1 卷積(升降維, 整流)首發平行式的網路, 像電影 inception 致敬, V4 不對稱交替式卷積, ResNet 前身 | | ResNet |
| ResNet | 何愷明 原 微軟亞洲研究院,後來去 Facebook  又稱殘差神經網路, 透過結構的巧妙逼殘差收斂為 0 ,  透過快捷連結傳遞梯度( y=f(x) => y=f(x)+x )  思路 :  1. 疊加 : (長, 寬, 高) 都一致  2. 利用殘差, 明確好收斂方向(shortcut connection)  再利用 1x1 卷積, 降為及恢復, 稱 Bottle Net 設計  注意 :  batch\_normalization 跟前層參數傳遞結合後才可以 Relu  相關結構 : Highway Network , ResNet 表現較佳  參考網站 : <https://kknews.cc/zh-tw/tech/l6jlv3b.html> | | ResNeXt  DenseNet  YOLO |
| 新一代卷積神經網路 : 更準, 更快, 更大的感知域, 更省算力, 模型更小, 行動裝置, 跨框架, 跨平台  結構要點改變 :  1. 小卷積為主流 : 大尺寸冗餘跟小卷積效果相當, 5x5 比 3x3 清楚, 但也只差一次池化,      1x1 卷積 => 升降維調整器, 用於整流, => battle net 結構  2. 不要用池化(pooling, 尤其 tan-h),避免資訊流失 => 用步長(2x2) 或 深度無損池化(同 reshape) 再加 1x1 卷積  3. 越深層圖越小 => 相同卷積窗看到得越多 => 感知域越大,      但 不要放大卷積層 => 計算量太大, 128 算很大了  4. 盡量不要用全連接層(Dense), 避免權重等比例上升, 導致模型不好收斂 => 改用 1x1 的卷積 (ex. 512x1x1) | | | | |
| 空洞卷積  (Dilated Convolution) | 池化雖然增加感知域，但是也造成空間信息的丟失，  分類問題無所謂，但是像素級任務就很嚴重了  空洞卷積能夠在不增加權重的基礎下擴大感知域  \*很常用, 必備  注意 : 主要用於上半段, 不要接 Dropout | |  |
| 可變形卷積神經網路 | 可變形卷積神經網路(Deformable Convolutional Networks) :  卷積神經網路的神效來自於感知域，但是這個感知域卻是固定大小而且是在超參數中被固定的  所以在卷積中加入了隨機位移的概念讓卷積形狀可變  v2在此基礎上還允許調節每個採樣位置或者bin的特徵的amplitude | |  |
| 模型壓縮 | 模型壓縮 : 稀疏化(L1正則) , 精細模型(事前設計) , 模型裁切(事後裁切) , 教師-學生方法(蒸餾, GAN)  2 個 3x3 = 5x5 , 用Bottleneck替代一個Resnetblock | |  |
| 深度可逐步分離卷積 | 深度可逐步分離卷積(Depth-wise separable) :  1. 引入深度可分離卷積的 Inception，稱之為 Xception  （Extreme Inception）的終極版  2. 把每一個深度逐步拆分卷積, 再由 1x1 卷積相互融合      => 較輕, 稍可運用於手機上 | |  |
| ShuffleNet | Face++推出的可在移動裝置運作的輕量級架構,  主要講述特徵的可交換性, 特徵洗牌,  其結構同 RestNet | |  |
| S-E net | S-E net (Squeeze-and-excitation) :  1. 2017 最後一屆ImageNet冠軍  2. 把特徵壓縮 => 篩出重要的 => 再放大疊加  3. 把每個單元都當作最後一個來判斷      誰才是最重要的特徵 | |  |
| DenseNet | 概念 : 重要的特徵就要一遍又一遍的重溫，絕不會丟失  DenseNet-FCN這結構又被稱為提拉米蘇  會使模型變得太重, 主要用於模型的前半部分 | |  |
| ResNext | 概念 : 除了深度、寬度還有分支基數(cardinality)  1. battle net \* 分支數 => 權值共享 => 再加上跳轉  2. battle net 右圖結構好壞 : a > b > c  3. 提出 轉置卷積(Transpose Convolution) 及      轉置池化(Transpose Pooling) 只是把操作反過來  4. 特徵抓取 :      上採樣 : 讓圖變小( stride(2,2) , pooling )      下採樣 : 把圖放大(卷積 , 轉置池化)  5. 棋盤效應解決 :      1. 最後一層轉至池化      2. 每一層空洞卷積 | |  |
| 模型跨平台 | 1. ONNX : CNTK, PYTOCH, Chainer, maxnet, Caffe2      tensorflow : 新版 dropout 不會自動關 => 大家想換 pytoch  2. 手機框架 : ncnn (騰訊)  3. ONNX runtime, Win MLTools (ML), 做 TFZONNX      => apple 的 core ML => core ML 指令集 | |  |
| R-CNN  (Region - CNN) | R-CNN 物件偵測(2013 年) :  1. 利用 Select Serch 產生 2000 個可能區域  2. 用 CNN 哲取特徵值  3. 用 SVM 分類器區分物件或背景  4. 用線性迴歸校正 bounding box 位置  Select Serch :   R-CNN 用來篩選 Region Proposals,  將 Segment(分割) 的結果各自劃出 bounding box,  用迴圈每次合併相似高的兩個,直到整張圖合併成一個 box  SPP Spatial Pyramid Pooling (不能反向傳播) | |  |
| 問題 : 1. 2000 個區域 => 至少 2000 次 CNN => 太慢            2. 分三部分(CNN, SVM, bounding box 迴歸) => 不易訓練            3. 有沒辦法不管圖片尺寸多大都能生成固定特徵大小? | | Fast  -R-CNN |
| Fast  -R-CNN | 連上 FC 網路 => softmax 分類 => bounding box 線性迴歸  1. Select Serch 後只做一次 CNN, 特徵2000區域共用,      (將proposal的region映射到CNN的最后一層的特徵圖,      以減少重複的特徵提取  2. 直接使用softmax取代svm  3. 將SPP層的概念轉化為ROI Pooling (SPP層無法反向傳播) | | Faster  R-CNN |
| 問題 : 還是太慢, 因為 Select Serch 太耗時 | |
| Faster  R-CNN | 直接解決proposal生成的問題  加入全卷積, 聯合訓練, 預埋錨點(anchor)  RPN (Region Proposal Network) :  不同大小的候選框  => 先定位 => 再分類 => 挑預選框  R-FCN : 沒有全連接層 | | Light-Head  R-CNN  Mask R-CNN |
| 問題 : 預選框, 錨點(anchor) 太多, 太肥 => 太慢 | |
| Light-Head  R-CNN | 類似 Google Net以合併的方式減輕模型 | |  |
| 所有 R-CNN 系列皆為二階段模型 => 少用, 雖然精準, 但太慢 => 盡量用一階段模型(快 , 準度不會太差) | | YOLO |
| YOLO 1  (You Only Look Once) | 4 (4個定位點) + 1 (有沒有東西) + 1 (類別 : one-hot),  YOLO1:把1470\*1的全連結層reshape為7\*7\*30的最終特徵  相關網站 : <https://medium.com/cubo-ai/yolo-with-coreml-819799789c11> | |  |
| 問題 : 不好學 | | YOLO 2 |
| YOLO 2 (Darknet-19) /  YOLO 3  (Darknet-53) | YOLO 2 (Darknet-19) :  1. 加 batch\_normalization 加速收斂,      拿掉dropout,拿掉最後一層池化以提升解析度  2. 圖片resize為416 \* 416，產生奇數長度的center cell      passthrough layer 26\*26\*512=>13\*13\*2048  3. 借鑒了Faster RCNN產生5種Anchor，      使用FCN去掉全連接層  4. 使用了傳統的分群模型，強化PROPOSAL選取  YOLO 3 (Darknet-53) :  1. 拿掉 pooling  2. VGG => ResNet  3. 13, 26, 52 特徵圖  4. 結構像 autoencoder 的變體 | |  |
| FPN | 特徵金字塔網路( Feature Pyramid Networks) :  1. 何凱明 提出, Focal Loss  2. 二階段問題 : 慢      一階段問題 : 不準      (不是不準, 是很多框太小,柿子挑軟的吃)  3. Focal Loss :解決深度學習柿子挑軟的吃      ，造成中庸的準確率的問題      簡單的問題 FL(pi) 很小      困難的問題 FL(pi) 很大      => 大幅增加準確率 | |  |
| 語意分割 | Semantic Segmentation :  依邊界分割, 用全卷積(FCN)  b. 依類別  c. 目標檢測, 依物件  d. = b + c , 全景分割     things 可數, stuff 不可數 | |  |
| FCN | Fully Convolutional Networks全卷積網路 (2015)  FCN 沒有 Dense 等, 利用放大和縮小來合併各層,  淺層提供位置, 深層提供分類  應用 :  FCN-DenseNet  U-Net : 可用於三維 => 醫療圖片  會將原圖變成同樣大小但深度不同的圖  (看分成幾類就有多深)  DeepLab v3t  Dice coefficient :  用以衡量模型結果 | |  |
| Mask R-CNN | Mask R-CNN (2017) :  1. Faster R-CNN 中加入一層 mask層  2. ROI pooling : 利用內差做較精細的 pooling      => 得到較精細的結果 => 再做分類和回歸  需要解決的問題 :  1. 特徵圖與 Ground Truth 會有對不準的問題,       如過用像素層級分類, 效果極差  2. 一樣很慢 | |  |
| RGBD | Indoor Segmentation and Support Inference from RGBD Images  ECCV 2012  監督式, 一樣要標註, 用RGB測深度  相關網站 :  <https://cs.nyu.edu/~silberman/datasets/nyu_depth_v2.html> | |  |
| Learn to Segment Everything | Learn to Segment Everything  Ronghang Hu, Piotr Dollár,   Kaiming He, Trevor Darrell, Ross Girshick  開啟弱標註的可能性  相關網站 :  <https://arxiv.org/pdf/1711.10370v1.pdf> | |  |
| Deep Extreme Cut | Deep Extreme Cut: From Extreme Points to Object Segmentation :  Kevis-Kokitsi Maninis, Sergi Caelles, Jordi Pont-Tuset, Luc Van Gool  把逐像素多分類問題變成二分類問題, 能夠協助快速標註 => 好用的分割  相關網站 :  <https://arxiv.org/pdf/1711.09081.pdf>  <https://github.com/scaelles/DEXTR-PyTorch> | |  |
| 動作檢測 | 動作檢測 (Keypoints Recognition)  DensePose  相關網站 :  <http://densepose.org/>  <https://arxiv.org/pdf/1711.10370v1.pdf> | |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |