影像處理100天（python）

[Day01：OpenCV介紹 1](#_Toc28162208)

[Day02：OpenCV表示 2](#_Toc28162209)

[Day03：操作圖片&直方圖均衡（convertScaleAbs()） 2](#_Toc28162210)

[Day04：圖形矩陣操作&warpAffine()用法 3](#_Toc28162211)

[Day05：在圖上畫圖 3](#_Toc28162212)

[Day06：仿射(affine transformation)&getAffineTransform() 3](#_Toc28162213)

[Day07：透視Perspective transformation，遠近操作 3](#_Toc28162214)

[Day08：取特徵值，Sobel&Gaussian Blur 3](#_Toc28162215)

[Day09-Day10：SIFT，取特徵值 4](#_Toc28162216)

[Day11：CNN 4](#_Toc28162217)

[Day12：步長(strides)&填充(Padding) 4](#_Toc28162218)

[Day13：池化(Pooling) 5](#_Toc28162219)

[Day14：Batch Normalization(BN) 5](#_Toc28162220)

[Day15：總結 5](#_Toc28162221)

[Day16：資料擴增 6](#_Toc28162222)

[Day17-18：各CNN介紹(AlexNet & VGG16,19) 6](#_Toc28162223)

[Day19&Day20：InceptionV1-V4 6](#_Toc28162224)

[Day21：CNN 經典模型應用-ResNet,VGG16 6](#_Toc28162225)

[Day22：OCR的CTC Loss function介紹 7](#_Toc28162226)

[Day23：Sliding windows detection 8](#_Toc28162227)

[環境安裝 8](#_Toc28162228)

[install pip 8](#_Toc28162229)

[Jupyterlab 9](#_Toc28162230)

[Open CV 9](#_Toc28162231)

[Keras 9](#_Toc28162232)

[Tensorflow 9](#_Toc28162233)

[Imgaug安裝 9](#_Toc28162234)

[pandas 9](#_Toc28162235)

# Day01：OpenCV介紹

1. 影像彩色代表3個channels；灰階則為1個channel
   1. Img影像陣列：

|  |
| --- |
| [[[125 137 226] #B  [125 137 226] #G  [133 137 223] #R  ...  [122 148 230]  [110 130 221]  [ 90 99 200]] |

1. cv2.imread(‘path’)
2. cv2.imshow(‘title’,img)要呈現圖，一定要配合cv2.waitKey(0) & cv2.destroyAllWindows()

# Day02：OpenCV表示

1. OpenCV中的預設圖片表示順序是BGR（Blue,Green,Red）
2. Cv2.COLOR\_BGR2HSL轉換：

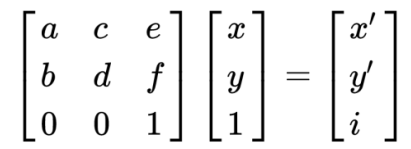
|  |
| --- |
| [[[ 4 176 162]  [ 4 176 162]  [ 1 178 149]  ...  [ 7 176 174]  [ 5 166 158]  [ 2 145 128]]  ...  [178 124 112]  [177 126 111]  [178 130 113]]] |

# Day03：操作圖片&直方圖均衡（convertScaleAbs()）

1. 3個channel轉換成陣列寫法：img[:,:,1]代表取得第1個channel，或用img[…,1]表示之。
2. Img = cv2.convertScaleAbs(img, alpha = alpha, bea=beta)，用來調整圖的對比及明亮度

# Day04：圖形矩陣操作&warpAffine()用法

1. 水平翻轉Flip：可直接對list操作，list[start:end:step]
2. Img[::-1,:,:]代表垂直翻轉，img[ h(y) : w(x) :c]，-1代表倒序
3. cv2.resize()介紹，cv2.INTER\_AREA縮小時使用；cv2.INTER\_CUBIC(速度較慢)/INTER\_LINEAR放大使用，效果較好，比較不會模糊
4. ad：縮放scale；bc：旋轉角度；ef：位移；00遠近操作
5. cv2.warpAffine()用法



# Day05：在圖上畫圖

# Day06：仿射(affine transformation)&getAffineTransform()

1. 一張圖片只需要1組（即2個）個3個點（共6個點），就可以建構出矩陣，透過矩陣來操作圖片

# Day07：透視Perspective transformation，遠近操作

1. getPerspectiveTransform()& warpPerspective()轉換

# Day08：取特徵值，Sobel&Gaussian Blur

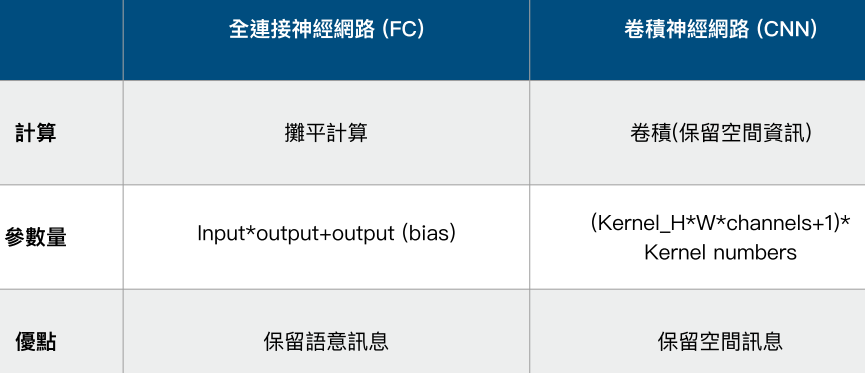
1. Filter=kernel，用來取特徵值
2. 操作方式，使用corss-correlation & convolution，convolution比較常用
3. Cv2.GaussianBlur()將圖模糊化；cv2.Sobel()邊緣檢測
4. 邊緣檢測（sobel）完畢後需要先將圖型做正規化，正整數，不得為負數

# Day09-Day10：SIFT，取特徵值

1. 使用邊緣檢測描取特徵，不是使用Sobel
2. 分為LoG & DoG(Difference of Gaussian)
   1. LoG，模糊化Gaussian Blur→二階導數取得邊緣
   2. DoG，先縮放→Gaussian Blur→合併取差值，無二階導數
3. 需將圖片轉成灰階
4. Feature Matching特徵配對(ratio test)最好設定在0.7-0.8之間

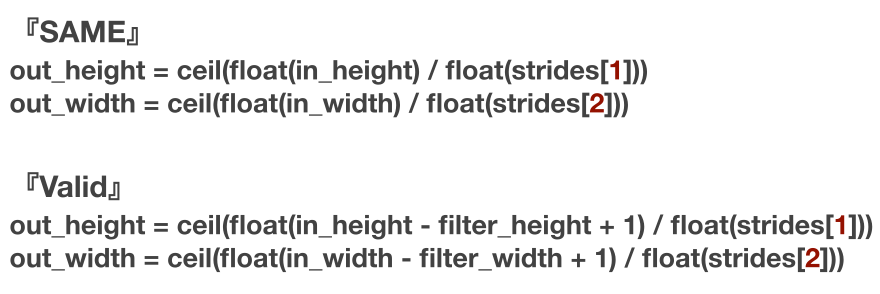
# Day11：CNN

1. 卷積核：Filter/Kernel/特徵偵測
2. 常見的kernel\_size=3\*3、5\*5
3. 特徵接受域(Receptive field)，換言之，看到特徵值大小
4. 張數通常在16、32、64



# Day12：步長(strides)&填充(Padding)

1. Strides影響Feature map
2. Padding週邊補0值，same指取完特徵值後，Feature map一樣大；valid則會縮小



1. Feature map的尺寸：



# Day13：池化(Pooling)

1. Max pooling(MaxPooling2D)，最常使用此方法，降低運算及加快收斂
2. Average pooling：強調特徵平滑性，所以容易把重要的特徵移除
3. CNN最後在output分類時，會用FC

# Day14：Batch Normalization(BN)

1. Activation的公式中，sigmoid方式最容易造成梯度消失，所以會讓程式跑不下去
2. Activation的公式盡量使用relu
3. 若使用sigmoid，Batch Normalization的function必須放在Activation(sigmoid)之前，效果最好
4. 若使用Relu，BN放在前後都無所謂了
5. Mini batch：一般都用32，所以很多範例都會看見batch\_size=32

# Day15：總結

1. 資料取特徵(卷積層)：Convolution + (Max Pooling/Agv Pooling)+ Activation + BN
2. 最後將特徵展開與Fully Connection連結，使用方法：Flatten or Global Average Pooling(GAP)
   1. Fully Connection的Layer層愈少愈好，一般用於最後output分類時使用 (softmax分類機率)
   2. Softmax，將所有分類的機率總合為1，例如：結果有3種，貓：0.3；狗：0.4；豬：0.3，其總合為1，代表成為狗的機率較高。
3. 影像：彩色為3 Channels，灰色為1 Channel

# Day16：資料擴增

1. Keras的ImageDataGenerator()
2. 或Imgaug擴增，將多種內容包裝在一起，用Sequential()包起來(此方法較多人使用)
   1. 也可以自訂義，打造pipeline

# Day17-18：各CNN介紹(AlexNet & VGG16,19)

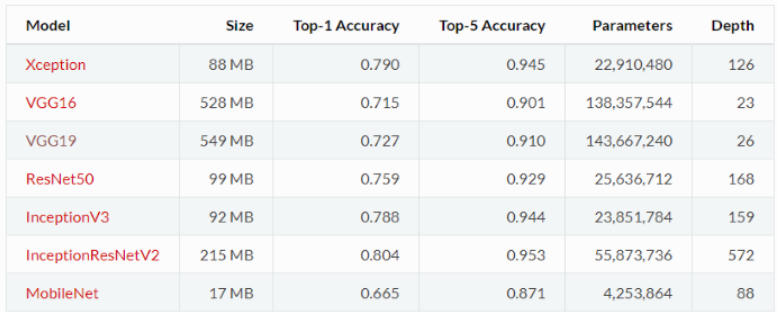
1. AlexNet提出用ReLU取代Sigmoid
2. Dropout較低Overfitting
3. Vgg介紹：1個5\*5與2個3\*3的效果是一樣的，但2\*3\*3比5\*5的參數還少，所以大家都用2個3\*3

# Day19&Day20：InceptionV1-V4

主要有的圖主角有的大，有的小，所以很難定義kernels的大小，因此使用InceptionV1(像全面啟動一樣)，先將kernel設定為1\*1

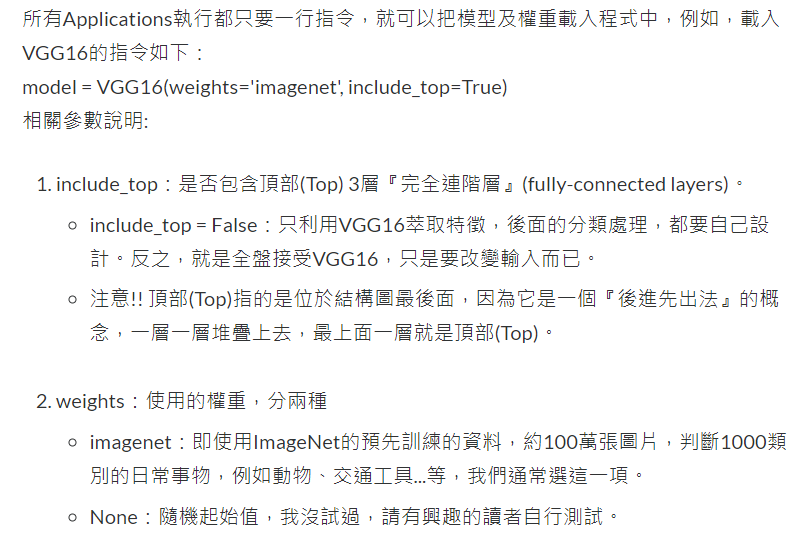
1. 控制chnnels的深度？同時增加模型的非線性？

# Day21：CNN 經典模型應用-ResNet,VGG16

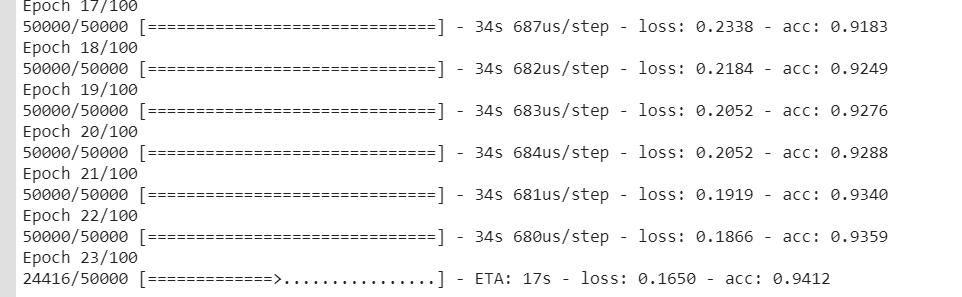


1．weight=’imagenet’，指用ImageNet協會所預先訓練的權重

<https://ithelp.ithome.com.tw/articles/10192162>



1. VGG的使用時機：VGG 使用ImageNet 100萬張圖片，共 1000 種類別，幾乎涵蓋日常生活看到的事物，例如動物、交通工具...等，訓練出來的模型，就變成一種『通用解決方案』(Generic Solution)，如果要辨識照片內事物屬於這1000類，例如貓、狗、大象等，就可以直接拿VGG模型來用了
2. 在自己電腦跑不動，需將程式移到colab執行，約訓練23次時的結果如下：

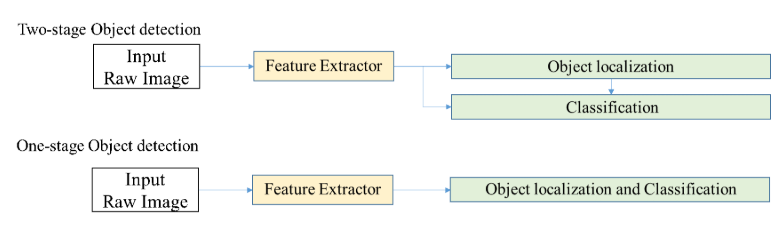


# Day22：OCR的CTC Loss function介紹

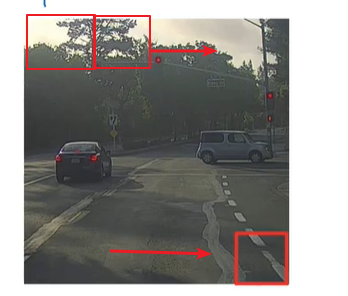
1. 作業主要是要我們自己用CNN來train ctc的loss function，讓loss值愈來愈小
2. <https://ypw.io/captcha/>
3. Lambda說明：<https://zhidao.baidu.com/question/366336612089024492.html>

# Day23：Sliding windows detection

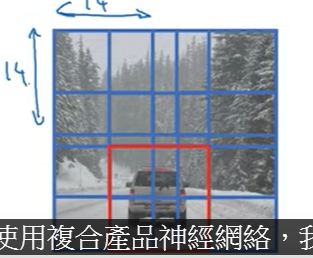
1. One Stage & Two Stage



1. Selective Search：有很多方法，Hierarchical Grouping Algorithm、Diversification Strategies…RCNN使用Sliding window
2. Two Stage：Region Proposal（選出物件）→再進行分類，這種稱為Two Stage，缺點當一張圖片很多物件時，需要很強大的GPU運算
   1. 比較有名的方法：RCNN、fast RCNN、faster RCNN
3. One Stage：物件位置偵測和物件辨識一步到位，通常速度很快，但整體辨識精度可能沒有two stage的方法來的好（Object Detection+Recognition）
   1. YOLO（You only look once）、SSD（Single Shot Detector）
4. <https://medium.com/@chih.sheng.huang821/%E6%B7%B1%E5%BA%A6%E5%AD%B8%E7%BF%92-%E4%BB%80%E9%BA%BC%E6%98%AFone-stage-%E4%BB%80%E9%BA%BC%E6%98%AFtwo-stage-%E7%89%A9%E4%BB%B6%E5%81%B5%E6%B8%AC-fc3ce505390f>
5. <https://blog.gtwang.org/programming/selective-search-for-object-detection/>
6. Sliding windows detection滑動窗口偵測的最大問題是計算成本，尤其是看設定的框大小，愈小，每次都要偵測，成本愈大，現在有個滑動視窗識別器
7. <https://www.youtube.com/watch?v=5e5pjeojznk>

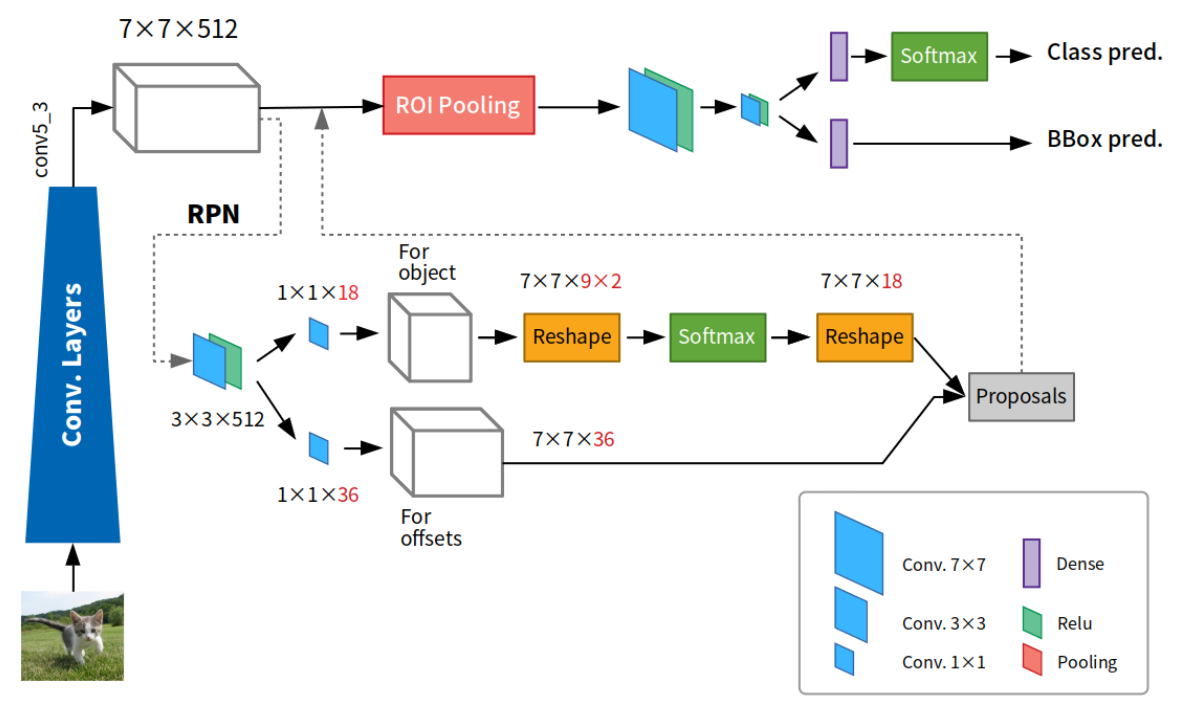


1. 滑動視窗識別器是將切割的固定大小重疊



# Day24：RCNN&YOLO

1. R-CNN：Selective Search取1000-2000上到下的RP（候選區域）→每個RP用CNN取特徵→SVN分類RP→使用迴歸調整RP區域
   1. <https://medium.com/@ywchiu84021121/object-detection-s1-rcnn-%E7%B0%A1%E4%BB%8B-30091ca8ef36>
2. Fast R-CNN：整個圖+ROI（Regions of Interest）→CNN計算特徵→feature map→ROI Pooling（M\*N進行max pooling）→FC→2個output，一類為Softmax分類 + 另一類是BBOX迴歸
   1. <https://medium.com/@ywchiu84021121/obeject-detection-s2-fast-rcnn-%E7%B0%A1%E4%BB%8B-40cfe7b5f605>
3. Faster-RCNN：RPN（取代Selective Search）計算特徵→feature map→POI Pooling→轉化為256D／512D的向量數→送入2個FC→BBOX迴歸 + BOX分類
   1. 多了一個分支來處理region proposals的小網路（RPN）
   2. <https://medium.com/@ywchiu84021121/object-detection-s3-faster-rcnn-%E7%B0%A1%E4%BB%8B-5f37b13ccdd2>



1. SSD（Single Shot MultiBox Detector）：不同scale的feature map都做偵測（CNN），但速度沒有Faster RCNN快
2. YOLO V1
   1. <https://medium.com/@chih.sheng.huang821/%E6%B7%B1%E5%BA%A6%E5%AD%B8%E7%BF%92-%E7%89%A9%E4%BB%B6%E5%81%B5%E6%B8%AC-you-only-look-once-yolo-4fb9cf49453c>
3. YOLO V2

# Day25：IOU計算

1. 計算交集面積：<https://blog.csdn.net/u012476249/article/details/53318891> & <http://www.csie.ntnu.edu.tw/~u91029/Shape.html>
2. 以pixel來計算，所以應該要+1

|  |
| --- |
| So, yes, in general, the top-voted answer is correct and is a good way to calculate the IoU. But (as other people have pointed out too) its math is completely incorrect for computer screens. You cannot just do (x2 - x1) \* (y2 - y1), since that will not produce the correct area calculations whatsoever. Screen indexing starts at pixel 0,0 and ends at width-1,height-1. The range of screen coordinates is inclusive:inclusive (inclusive on both ends), so a range from 0 to 10 in pixel coordinates is actually 11 pixels wide, because it includes 0 1 2 3 4 5 6 7 8 9 10 (11 items). So, to calculate the area of screen coordinates, you MUST therefore add +1 to each dimension, as follows: (x2 - x1 + 1) \* (y2 - y1 + 1). |

# 環境安裝

## install pip

download:get-pip.py

#>python get-pip.py

## Jupyterlab

pip install jupyterlab

## Open CV

pip install –user opencv-python==3.4.2.16

pip install –user opencv-contrib-python==3.4.2.16

## Keras

pip install Keras

## Tensorflow

pip install tensorflow

注意：乃森建議安裝1.8.0版，但python 3.7無1.8.0版，故可直接安裝新版即可

pip install scikit-learn (day15)

## Imgaug安裝

容易卡在Shapely安裝時失敗，故必須先安裝Shapely的whl（輪子）檔

<https://www.lfd.uci.edu/~gohlke/pythonlibs/#shapely>

下載需要的套件：64bit,Python3.7(Shapely‑1.6.4.post2‑cp37‑cp37m‑win\_amd64.whl)

pip install Shapely‑1.6.4.post2‑cp37‑cp37m‑win\_amd64.whl

pip install imgaug

## pandas

pip install pandas

### Captcha

產生驗証碼使用

pip install captcha