王建堯老師 - Yolov4

出自

題目

Yolov1：全 CNN 架構， Grid : 將 input image 分成 S * S 個格子

Yolov2：

1. 引用 anchor box
2. 先用 backbone 接收高 resolution image 再用來做 object detection
3. 用改變 input 大小的方法來做 multi-scaling training
4. 每個 anchor 預測其中是否有物體

Yolov3：

1. 引用知名 backbone
2.

Introduction

背景

object detection 任務陳述：

1. 需要做到偵測 object 的 location

2. 需要判別該物件是甚麼 class ex : 人, 車 。 原因是在真實應用中，需要去 detect 出許多種類才能夠應用 ex 自駕車需要得知人, 車, 紅綠燈在哪裡
3. 會 output 一個**置信度**，如果置信度分數低於閾值，就會**捨棄這個 bounding box，因此**模型有機會判斷這張圖片中**沒有任何物體**。

zero shot object detection : 需要偵測出 training set 中沒有的那些種類

object detection 常被放在 recommendation systems ，不是推薦系統，而是說是一種輔助、給予建議的系統

提升準確率讓 recommendation systems 需要更少的人為 input 與操作，達到 stand-alone process management ，只要 AI 參與就好不用人類參與

We assume that such universal
features include Weighted-Residual-Connections (WRC),
Cross-Stage-Partial-connections (CSP), Cross mini-Batch
Normalization (CmBN), Self-adversarial-training (SAT)
and Mish-activation. We use new features: WRC, CSP,
CmBN, SAT, Mish activation, Mosaic data augmentation,
CmBN, DropBlock regularization, and CIoU loss, CBN [89], PAN [49], SAM [85]

動機

現在最強的 object detection model 訓練時需要超多 GPU 以及超大 Batch

所以要做一個 real-time object detection model

目的

1. 可以在 realtime speed (<24 FPS 人眼視覺暫留) 完成 object detection
2. 達到 43.5% AP, 65.7% AP When AP50 (見 Experiments)
3. parallel computations 優化

computation volume theoretical indicator : 計算模型在一次（`forward pass`）過程中需要進行的**浮點運算次數，理論上**需要多少運算，而不是在**實際硬體上**的運行時間

`BFLOP`（**B**illion **F**loating-point **OP**erations）: 就是一種最常見的 `low computation volume theoretical indicator`，不用甚麼運算，只需要理論推倒就可以知道效能

相關研究

1. Object Detector 優化速度分類 :

   GPU-Based Object Detection : 其架構與運算方法，有針對 GPU 特別加速

   常見的 Backbone 有 : VGG [68], ResNet [26], ResNeXt [86], or DenseNet

   CPU-Based Object Detection : 其架構與運算方法，有針對 CPU 特別加速

   常見的 Backbone 有 : SqueezeNet [31], MobileNet [28, 66, 27, 74], or ShuffleNet [97, 53]

2. Object Detector Head 分類 (by stage)

   Object Detection Model = Backbone + Head

   Backbone 是用 ImageNet 訓練的，用來抽取特徵

   Head 用來辨別 Bounding Box

   Head 則分為 one stage 或是 two stage

   one stage :

   同時做物件分類與 bounding box regression

   因為是在每個有可能的區域做 prediction，因此稱為 dense prediction

   two stage : R-CNN 系列

   coarse to fine architecture : 先生成候選區，覺得哪些區域比較可能有 object，再去精選，因為是從 selected-area 中再做 fined prediction，因此稱為 sparse prediction

   neck : 得到更多 feature map，包含 several bottom-up paths and several topdown paths.

3. Object Detector Head 分類 (by method)

   anchor-based object detection :

   會先定義一些 anchor-box ，例如 $1 * 1$, $3 * 3$ ，並在 input image 的每一個 pixel 上展開，所以假設我們有九種 anchor-box ， 那在 input image 的某個像素上，就可以展開九種 anchor-box。

   接著 input image 經過 CNN 抽成 Feature map 後，會依據 receptive field 找回當初的 anchor-box

每一個 anchor-box 都 output 置信度與 offset，接著經過 NMS 找到 Bounding box

非人為 anchor-box 制定方法 : 將 training data 中的所有正確 bounding box (w, h) 找出來做 K-mean，這個 K 就是 anchor-box 的種類數量

anchor-free object detection :

用 Fully-CNN Architecture 輸出 feature map， 這個 feature map 就代表了 heat map，上面的 pixel 表示該 local 有物體中心點的機率。接著將這些數值高的 pixel 丟入 model，因為他有 receptive field 的關係，他可以正確地預測出 bounding box 的長寬

two stage anchor-free object detector : RepPoints

one stage anchor-free object detector : CenterNet [13], CornerNet [37, 38], FCOS

4. Bag-of-Freebies : **只在 training 時**使用，不會影響到 inference 時的時間，常見的有 Data Augmentation, Regularization, 標籤平滑（Label Smoothing）: 將硬性標籤（如 0 或 1）轉換為軟性標籤（如 0.05 或 0.95），有助於防止模型過度自信，也可以用 knowledge distilling。
    a. Data Augmentation :
        a. photometric distortions
        b. geometric distortions
        c. random erase [100] and CutOut : 選一塊矩形全部填 0，單體遮蔽
        d. hide-and-seek [69] and grid mask : 選多塊矩形全部填 0，多體遮蔽
        e. DropOut [71], DropConnect [80], and DropBlock : 遮蔽部分 feature maps

            drop path : 常用在有 residule 的結構，他會將 layer 的 output 直接變成 0，讓 model 模擬當某些層停止運作後，是否還是有辦法做好

Spatial Dropout : feature map 上選擇某些 channel 丟掉，讓 model 不要太依賴某一個 kernel

drop block : 針對 cnn 上的 drop out ，因為一般的 drop out 只丟一個資訊太少了，很容易被補足，所以她是先在 feature map 上選中心點，然後將 鄰近 b *b 區域全部填 0

f. MixUp : 重疊兩張圖片
g. CutMix : 從圖片 `A` 中裁剪一塊矩形區域，**覆蓋**到圖片 `B` 的一個隨機位置上，而 label 會變成混和標籤 ex : `0.4(A area 大小) * 1(貓) + 0.6(B area 大小) * 0(狗) = 0.4`
h. transfer GAN : 生成更多 image
i. Mosaic : 4 圖組合 ⇒ 讓訓練時 batch size 可以不用開那麼大
j. SAT :

   第一階段 : adversarial attack 自己，生成一個 noise 使得原本的圖片雖然有物體，但是會讓當前 model 的 confidence score 很低

   第二階段再用這個圖片訓練

b. 解決 正負樣本 inbalance

negative example : 當 bounding box 與 ground truth bounding box 的 iou 小於**閾值**（例如，0.5）那就是 negative example

Hard Negative Example : 那些置信度很高的 negative example，因為 置信度與 bounding box 是分開偵測的，所以可能會有不一致的問題

category inbalance : category 的樣本數之間有 inbalance ， 例如狗有 10000 貓有 100 張

正負樣本 inbalance : 一張 input image 中，正確的 bounding box 少, 錯誤的 bounding box 多

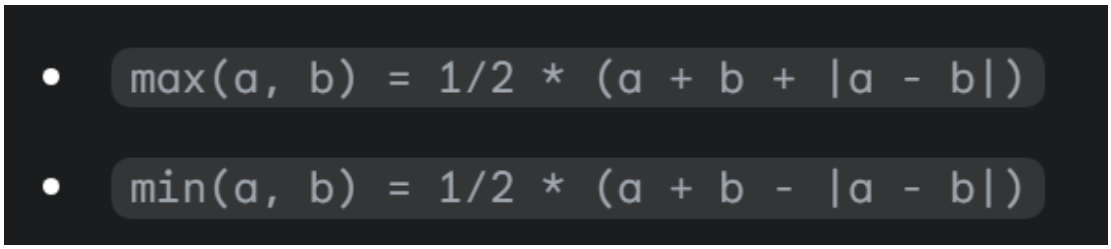
online hard example mining ：用來解決 正負樣本 inbalance (針對 2 stage model)

1. 每次訓練時先找出正樣本與負樣本，而負樣本只取置信度高的那幾個 Hard Negative Example，再將這些樣本作 backpropagation

因為 1 stage 的計算量會太大，所以 1 stage 需要使用 focal loss : 給那些已經預測得很好的**簡單樣本**非常小的權重，並把 Hard Negative Example 的權重放大

c. 修改 objective function :

1. 過往是使用 MSE 將四個座標做 Regression ，但是這樣四個點的預測會變成獨立，但他們其實相關並與 object 的形狀有關
2. IoU loss = 1 - iou(predicted BBox, ground truth BBox)
   其中 min, max 可以微分



3. GIoU loss [65]： is to include the shape and orientation of object in addition to the coverage area

   smallest area BBox that can simultaneously cover
   the predicted BBox and ground truth BBox, and use this
   BBox as the denominator to replace the denominator originally used in IoU loss

4. DIoU loss the distance of the center of an object

$$DIoU = IoU - \frac{\text{distance between centers}^2}{\text{enclosing box diagonal}^2}$$

5. CIoU loss [99], on the other hand simultaneously considers the overlapping area, the distance between center points, and the aspect ratio

5. Bag-of-Specials : **在 training 與 inference 時**都會增加模型複雜度，增加運算時間，但可以提高準確率，常見的有 Residual Connections, Attention Mechanisms, 換 New Activation Functions

   a. enlarging receptive field

      a. SPP：把某一層 CNN 的 output 給等分成 1 * 1, 2 * 2, 4 * 4 ...，接著每一塊做 max pooling ， 會變成 1 * c, 2 * c, 4 * c ...，然後把這些 feature concate 起來得到 n * c 。 這個方法也可以用在當 input 圖片不同大小時，得到一個固定大小的 output 送入 fully convolution network

b. SPP 改 : 改良成用 k * k max pooling kernel 來做，先對 input image 做 padding，並取 這 k * k 範圍中的 max

c. ASPP：max pooling kernel 用的是 dilated 版本

d. RFB：用 dilated

b. channel-wise attention and pointwise attention

a. Squeeze-and-Excitation (SE) [29]：channel-wise attention，CPU上時間增加一點, GPU 上時間增加很多。
input : h * w * c ， 會先經過 max pooling 變成 1 * c ，接著經過 mlp 與 softmax 變成 c 個權重，接著在 h * w * c 上對每一個 c 套用自己的那個權重

b. Spatial Attention Module (SAM)： pointwise attention ，不會改變 GPU 時間

input : h * w * c ， 會先做 channel pooling 得到 h * w * 1，接著經過 conv 得到 h * w * 1 的 attention， h * w * c 上的每一個 pixel 都乘以自己的權重，不過不同 channel 同一個 pixel 乘以的是相同的 權重

c. strengthening feature integration capability

a. skip connection

b. hyper-column : 多層 CNN output 多層 feature map ，會把每個 feature map upsampling 成 h * w (與原 input image 相同)，接著將他們加起來再給予最後一層 CNN ， 原因是這樣可以混和低層 feature 與 高層 feature

c. FPN：

將一張 image 經過多層 cnn 形成多層 feature map c1, c2, c3, c4, c5

接著會把 c5 upsampling 成 p5 與 c4 的 h * w 相同， 並透過個別的 1 * 1 conv 把 p5 與 c4 的 channel 數變到相同，接著把 p5 與 c4 相加，接著

去融合 c3, c2, c1 ...，最後得到 P5, P4, P3 ... 交給 Head

d. PAN：FPN 得到 P5, P4, P3, P2 之後，再藉由 sum(P2_down, P3) → 更新 P3，依序更新 P4, P5 然後交給 Head 使用

e. SFAM：把每一層 feature map 都 upsampling 到相同 h * w 然後拼接成 h * w * c′，並使用 SE (Channel Attetion)

f. ASFF：先 upsampling 到同一個尺寸，然後 h * w 經過 softmax 學得權重乘以自己的 h * w ，然後相加

g. BiFPN：feature map c2, c3, c4, c5 ，目的是得到混和過後的 p2, p3, p4, p5 接著做後續處理，其中一個 p 會是自己與鄰近 scale 的混和，舉例來說 p4 = c4 + up-sampling(c5) + down-sampling(c3)

d. Activation Function :

a. LReLU and PReLU：處理 RELU 在 input < 0 時 gradient < 0 的問題

b. ReLU6 and hard-Swish : 放到 quantization network (低精度) 上。quantization network 是將數值映射到 8bit 或 16 bit 上的技術，所以如果像 relu 一樣沒有上界會導致一格 8 bit 代表的數值太多，所以 relu 6 會將數值壓到 0 ~ 6 再用 8 bit 代表

c. parametric-ReLU：a 是參數，可以學習小於 0 時的斜率為何

$$f(x) = \begin{cases} x & x > 0 \\ ax & x \leq 0 \end{cases}$$

d. Self-normalizing neural network : 是一種 neural network，符合以下特徵，讓輸出的 mean = 0 std = 1

1. 每層 activation function 都是 SELU
2. 經過 LeCun normal 初始化
3. 每一層都是 Dense 連接

e. NMS 後處理

a. greedy NMS：多次迭代，每次先選出 classification confidence score 最高的 box 作為 reference（box_max），計算其他 box 與 box_max 的 IoU，**如果 IoU > threshold，就直接刪掉這些 box**。到下一次迭代後，原本這些 box_max 會移除，再找一波新的 box_max

b. soft NMS ：**如果 IoU > threshold ，則下降這些 box 的 confidence score 而不是移除**

f. Batch Normalization :

a. Filter Response Normalization (FRN) [70]：因為一般的 batch normalization 太過依賴當下 batch 的分布情形了，所以需要一個不用 batch 的方法 ⇒

假設 feature map x 的 shape = b * c * h * w ，根據每一個 b * c 先算出 v = 該 channel 下 feature map 元素總和，也就是說現在每一個 b * c 都對應一個 v ，這個 v 是該 feature map 的總和，接著 x 的每一個元素都去除以對應 b c 的 v ，然後參數學習平移與伸縮

$$\hat{x}_{c,i,j} = \frac{x_{c,i,j}}{\sqrt{\nu_c^2 + \epsilon}}$$

接著，像 BN 一樣有可學習的縮放和平移：

$$y_{c,i,j} = \gamma_c \hat{x}_{c,i,j} + \beta_c$$

b. Cross-Iteration Batch Normalization (CBN)：保存過去的多次 iteration 的 mean 與 varience 來做更穩定的 normalization。

假設我們保存了最近 K 個 iteration 的 batch statistics：
$\{(\mu^{(t)}, \sigma^{2(t)})\}_{t=T-K+1}^{T}$

那麼 CBN 的均值與方差定義為：

$$\mu_{CBN} = \frac{1}{K}\sum_{t=T-K+1}^{T}\mu^{(t)}, \quad \sigma^2_{CBN} = \frac{1}{K}\sum_{t=T-K+1}^{T}\sigma^{2(t)}$$

最後用來 normalize：

$$y_i = \gamma\frac{x_i - \mu_{CBN}}{\sqrt{\sigma^2_{CBN} + \epsilon}} + \beta$$

c. CmBN Cross mini-Batch Normalization :

因為真正放到 GPU 中的是 mini batch (一個 batch = 4 個 mini batch)，為了讓這四個 mini batch 有更穩定的訓練，因此用這四個 mini batch 當成 t-3, t-2, t-1, t 來做 CBN 而讓 mini batch 可以有 batch 的效果

方法

1. 分 Group :

   GPU : channel 分 group 計算 ⇒ 當 group = 2 時，我們會把 input 依照 channel 分成 2 組，每一組會用各自的 kernel 來處理。因為兩組彼此之間沒有交互所以效果會弱一點，可是計算速度可以兩倍。

   VPU : 專門為 CNN 設計的 GPU ⇒ 依照 channel 分 group ， 每一 group 用 Global Average Pooling 獨立計算權重

2. detector 需求:
    a. input image 要大才能 detect small object
    b. 要有 more layer 才能有大 receptive field

3. Model 架構 :
    a. data augmentation： Mosaic, and Self-Adversarial Training (SAT)
    b. Random training shapes : 隨機改變 input 的大小，讓 model 不要只 fit 在同一種大小
    c. Backbone : **CSPDarknet53**
    d. Neck：PAN
    e. **增加 receptive field :** 加入 **SPP block**
    f. Batch Normalization：CmBN
    g. Cosine annealing scheduler： 學習率從初始值慢慢下降到最小值，呈現「cosine 曲線」
    h. Eliminate grid sensitivity : 利用 anchor box + offset 使得不受限於 CNN 的 receptive field ⇒ 如果該 receptive field 只截到物體的一半那就做不出來

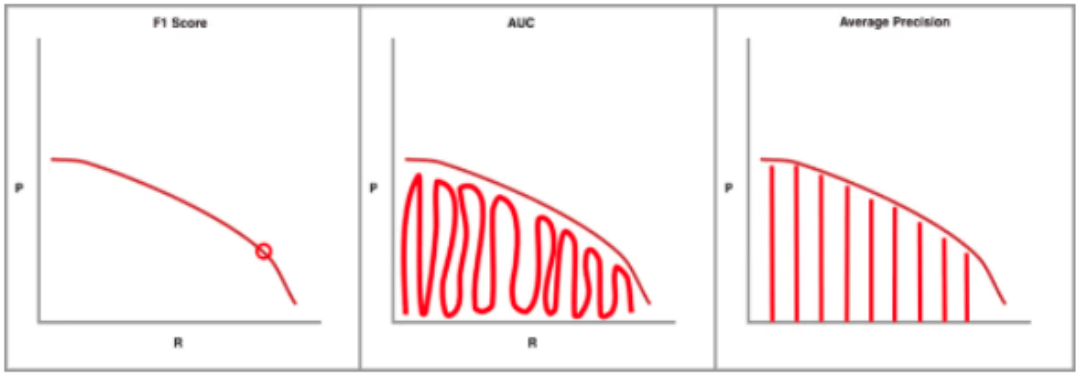
Experiments

Precision : 正確框到的數量 / model 框數 ⇒ 越高代表 machine 出手必得

Recall : 正確框數 / ground truth 框數 ⇒ 愈高表示該找的 object 都有被找到，躲貓貓達人

AP：透過下方排序的圖表畫出 AP 曲線(橫軸 recall, 縱軸 precision)，算出線下面積除以最大面積(1)，範圍介於 0 ~ 100 %

排名	預測框	置信度	判斷結果	當前 TP	當前 FP	當前 Precision	當前 Recall
1	P1	0.95	TP	1	0	1/1 = 1.0	1/3 = 0.33
2	P2	0.90	TP	2	0	2/2 = 1.0	2/3 = 0.67
3	P3	0.85	FP	2	1	2/3 = 0.67	2/3 = 0.67
4	P4	0.80	TP	3	1	3/4 = 0.75	3/3 = 1.0
5	P5	0.70	FP	3	2	3/5 = 0.60	3/3 = 1.0



AP50：表示取 IOU = 0.5 時的 AP 值，只要 Predict 框與真實框 union 佔 0.5 以上就算正確

$$IoU = \frac{\text{交集面積}}{\text{聯集面積}}$$

可以學習的地方

問題