王建堯老師 - yolov7

出自

題目

YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors

YOLOv5：auto-anchor 用 k-means 自動計算 anchor box

YOLOv6：Anchor-free

YOLOv7：speed and accuracy 強化，自帶 scaling

Introduction

背景

動機

NPU（Neural Processing Unit）：專門用來加速神經網路運算的處理器，**低功耗、高效能，長裝在**手機、邊緣設備、IoT 裝置

本篇研究希望加快這些裝置上 detect 的速度

目的

1. 將 model re-parameterization 從原本提出的 vgg 轉換到 resnet 與 densenet 中，並使用 gradient propagation path 的觀念加強學習效果 (反向傳播時，傳到該參數 w 之前經過哪些 layer, activation funciton 的 path)
2. dynamic label assignment : 如果 model 有多個 output layer ， 像是透過 FPN 後接收 p3, p4, p5 分別道不同 head ，這些 head 看到的 label 要不一樣，不然會出問題
3. ground truth 制定問題解決 :
   傳統來說會 hard label 說哪些 ground truth box 要歸類到哪些 scale ，然後每個 scale 各自處理。但是如果是 dynamic label assignment 就不會預先規定哪些 ground truth box 要歸類到哪些 scale

相關研究

YOLOX [21] and YOLOR [81] 加快了 gpu 上的 inference time

object detection 中的新方法

1. model re-parameterization :

   a. module-level ensemble :

      針對那些有 multi branch 的網路，訓練時仍然維持多分支（例如 3×3 conv、1×1 conv、identity)， inference 時則依照數學等價可以將這些分支

融合成單一 3×3 conv 而加快速度，他的 output 與 multibranch 時相同，因此不會有 mismatch 的問題

融合時 :
identity： x = conv(x) ，一個什麼都沒做的卷積層，樣子為

$$K_{identity} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

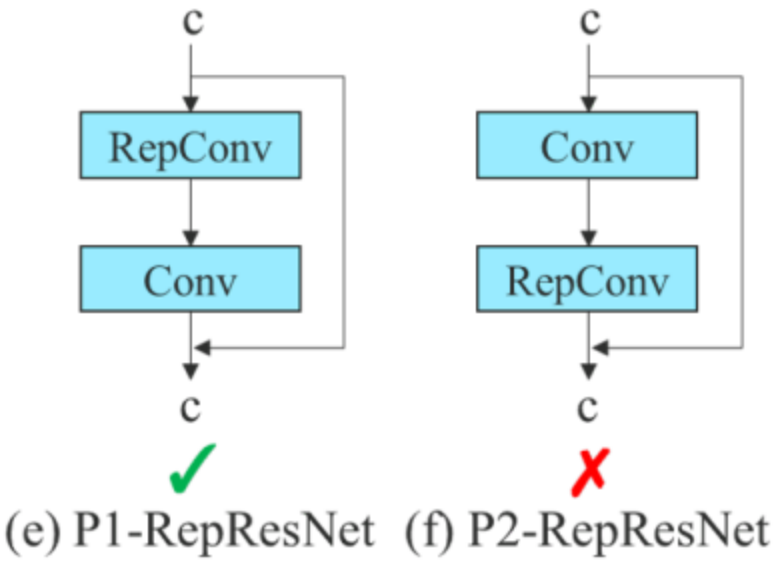
而 1 x 1 會經過 padding 成為 3 x 3 ， 接著 3 個 kernel 相加就是融合過程

RepConv (就是那個 1 * 1 conv, 3 * 3 conv + identity 的那個 re parameter) 在 vgg 上表現不錯，可是在 ResNet [26] and DenseNet 就沒有辦法，因為 resnet 上有 residual，densenet 上有 concate ， 兩者可以確保在 back propagation 時 path 是多樣的 (特別注意這邊的 residual = x + f(x), 其中 f 包含了像 relu 這樣的 non-linear)。identity 的 gradient 大，容易造成偏重，所以會減少 resnet, densenet 在 gradient 上的 diversity。 至於 vgg 則因為本來就沒有 diversity 的設計，所以不太受影響。

因此要製作 RepConvN ，如下圖中的 p1, p2 都是 RepConv 的 Conv Layer

p2 會先得到 conv(x) ，但是 repconv 的 identity 會輸出 conv(x) ，而 conv(x) 與 x 的差異並不大，所以等於是又加了一條 residual，會讓 residual 的強度過大，整體 block 都傾向於使用 residual，所以 p2 不行

p1 會將 3 * 3, 1 * 1, identity 的 output 相加之後，再交給 conv，也就是最後得到 conv(x + a + b) ，和 x 比較不像，不會額外增加 residual 強度所以可以



(e) P1-RepResNet (f) P2-RepResNet

b. model-level ensemble

方法一、用不同資料 train 同一個結構的 model， 然後平均他們的參數作為參數

方法二、平均不同 iteration 的參數

2. dynamic label assignment : 不使用固定規則(ex：IOU > 0.5)標定正負樣本，

a. ATSS：

每個 GT box 都挑出 k 個距離最近的 anchors 當作候選 anchor，計算這些候選 anchor 與 GT box 的 IoU 分佈。設定 threshold = **平均值 + 標準差**，所有 IoU ≥ threshold 的 anchors，就標為 **正樣本**，其餘是負樣本。

b. SimOTA：

第一步、計算各個 GT 應該有多少正樣本 : 計算該 GT 的候選 anchor IOU，選擇前 M 個相加四捨五入，這個 Kg 就是正樣本個數

$$K_g = \sum_{top\ M} IoU$$

第二步、計算這些候選 anchor 的 cost :

$$\text{cost} = \lambda_1 \cdot \text{classification loss} + \lambda_2 \cdot \text{regression loss}$$

第三步、把最小 cost 的 anchors 分配給 GT，作為正樣本。

c. PAA：

每個 GT box 去計算所有候選 anchor 的 loss = classification loss + regression loss， 接著用

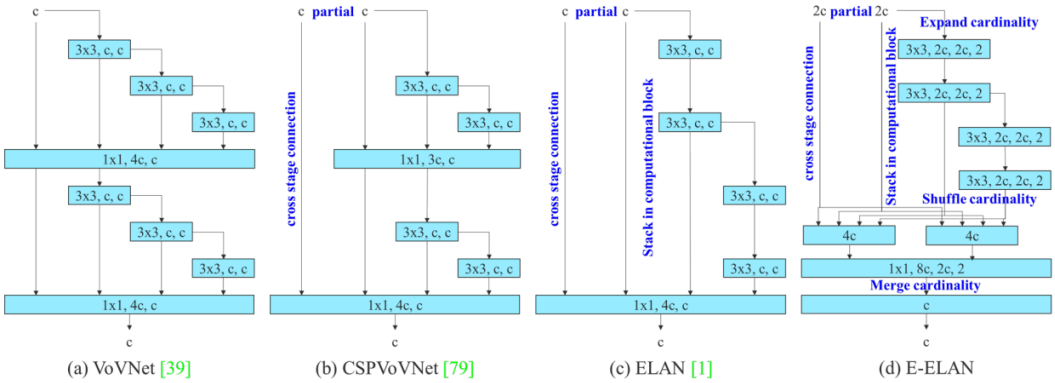
Gaussian Mixture Model (GMM)， 用兩個 gaussian 去 fit 這個 loss 分佈 (橫軸為 loss 縱軸為數量)，一個 gaussian loss 較低，是好樣本的 gaussian ，另一個是負樣本。 接著去幫每一個 anchor 看它屬於好 gaussian 的機率，如果比較大，那就是正樣本

3. NAS 可以自動找到 Network 架構，透過
   a. Reinforcement Learning
   b. 進化算法 (Evolutionary Algorithm, EA)
      i. 先生成一群候選網路，計算他們的 accuracy, 速度
      ii. 保存比較好的那些網路，然後在他們的參數附近搜索
   c. 梯度導向搜尋 (Differentiable NAS)

4. 關於 Gradient :
    a. CSPVoVNet analyzes the gradient path ， 讓 inferences 可以更快更準
    b. ELAN : 整組 model 應該要維護好，要有最短 gradient path 也要有最長 gradient path ，前者可以避免 gradient vanishing 的問題，後者可以讓 gradient 更新的效果更好 (因為經過較多層)
    c. E-ELAN : 用 group convolution 來增加不同分支，並使用 shuffle and merge 來讓不同分支彼此之間互動

    transition layer : 由 1 * 1 conv 與 pooling 組成，前者希望降低 channel 數，後者希望降低 feature map size

    e-elan 只改 conv block 沒改 transition layer



    這是一個 conv block (在論文中稱為 conv layer)，包含多層 cnn

    conv layer (在論文中稱為 conv block) 的第二個與第三個參數代表 input channel 數與 output channel 數

    e-elan 的最後一個參數是 group = 2

    而方向是 data 的方向，指向 head

    partial c 是指原本的 channel 數有 c ，我們只取部分的 channel 出來使用

stack in computational block : 將 channel stack 之後用 1 * 1 conv 再回到 channel 數 = c
將 partial c path1 與經過兩次 3 * 3 conv 的那條 path 做 stack

4c 表示蒐集 4 個 c 集合成 4c ，不是 conv layer

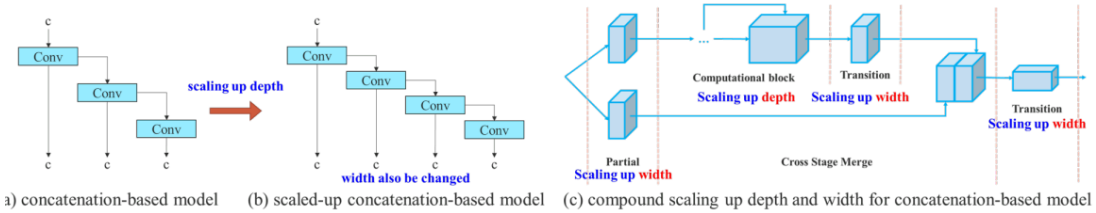
4. 關於 Scaling (Model 依照需求變大變小) :

   過去調整不同項 scaling 參數是分開來調整的，因為他們覺得這些項目之間是 independent。

   但如果是 all concatenation based models, such as DenseNet [32] or VoVNet [39]，那就必須要用心的設計 scaling method

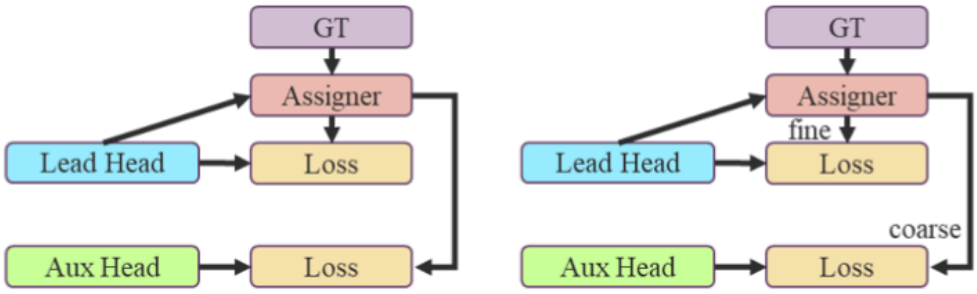
   concated-based model scaling 時不能直接調整深度，否則會對不齊

   所以需要深度與 width 一起調整



a) concatenation-based model (b) scaled-up concatenation-based model (c) compound scaling up depth and width for concatenation-based model

6. Deep supervision : 在 model 的中間加幾層 head ，這些 head (auxiliary head)的任務目標與最終的那個 head (lead head)相同，也會得到一樣的 loss ，這些 loss 可以幫助 model 的前幾層進行學習



(d) Lead guided assigner (e) Coarse-to-fine lead guided assigner

1. soft label : 我們會去拿 anchor box 與 ground truth box 的 iou 值去當作 soft label ，而不是直接用 1 與 0 的 hard label 。 在本研究中稱為 fine label
2. Lead head guided label assigner : 用 lead head 的 output 與 ground truth 形成訓練資料，區分正負樣本，同時給 auxiliary head 以及 lead head 使用，這樣整體 model 會有 residual 的效果
3. coarse label : 放寬 iou 標準的 fine label 。 coarse 的 soft label 需要被弱化 (ex: 原本是 0.6 要弱化成 0.4) ，避免 model 太 fit coarse 而不 fit fine，特別注意 : 如果只讓 coarse 的 weight 小 fine 的 weight 大還是無法處理這個問題

7. Batch normalization in conv-bn-activation topology : inference 時把 batch normalization 丟到 conv 中，這樣 inference 速度更快

BN 在一個輸出特徵圖 y 上的運算是：

$$y_{BN} = \gamma \frac{y - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta$$

其中 $y = Wx + b$，是 convolution 輸出的值。
我們可以把這個式子改寫成等效的一個新的 convolution：

$$y_{BN} = W'x + b'$$

其中

$$W' = \frac{\gamma}{\sqrt{\sigma^2 + \epsilon}} W$$

$$b' = \frac{\gamma}{\sqrt{\sigma^2 + \epsilon}} (b - \mu) + \beta$$

8. 借用 YOLOR 的 Implicit knowledge：YOLOR 會學習兩個 encoding (數值) y_mul 與 y_add，在 backbone 最後幾層 cnn 中，假設某層的 output feature map = y ，那他會利用 implicit knowledge 來做調整，最後輸出 y′ = (y * y_mul) + y_add 。 inference 時，這個 encoding 不會受到 input 影響，是一個固定的值。

9. EMA model (Exponential Moving Average)：「平滑」的更新模型參數
   訓練時 : 模型都使用 theta 以及更新 theta，這樣訓練會很快。
   inference 時 : 不使用 theta，而用 theta_ema

$$\theta_{EMA}^{(t)} = \alpha \cdot \theta_{EMA}^{(t-1)} + (1 - \alpha) \cdot \theta^{(t)}$$

方法

Experiments

可以學習的地方

問題