

[論文] You Only Look Once : Unified, Real-Time Object Detection

tags: paper notes

- [\[論文\] You Only Look Once : Unified, Real-Time Object Detection](#)
 - [概要 Abstract](#)
 - [簡介 Introduction](#)
 - [統一檢測 Unified Detection](#)
 - [網路設計 Network Design](#)
 - [訓練 Training](#)
 - [推論 Inference](#)
 - [YOLO 的限制 Limitation of YOLO](#)
 - [偵測系統比較 Comparison to Other Detection System](#)
 - [Deformable parts models \(DPM \).](#)
 - [R-CNN](#)
 - [Other Fast Detectors](#)
 - [Deep MultiBox](#)
 - [OverFast](#)
 - [MultiGrasp](#)
 - [實驗 Experiments](#)
 - [現實中的即時檢測 Real-Time Detection In The Wild](#)
 - [結論 Conclusion](#)
 - [參考資料](#)
 - [註釋](#)

概要 Abstract

在這篇論文中，作者推出了一個嶄新的物件偵測方式 – YOLO。原本的物件偵測任務是利用分類器來進行，但在 YOLO 中，作者們將物件偵測視為一個回歸任務，來從空間中分割出邊界框 (Bounding Box) 並且計算出類別機率。僅利用一個神經網路進行一次計算來直接預測邊界框及類別機率，也因為整個偵測過程只有使用單一個神經網路，因此可以視為是一個 End-to-End 的優化過程。

這樣統一的架構的執行速度十分快速，YOLO 執行圖像任務上可以達到即時每秒 45 幀。而另外一個較小型的版本 Fast YOLO 不僅可以達到每秒 155 幀的執行速度，mAP (Mean Average Precision) 也是其他即時物件偵測系統的兩倍。

跟其他的物件偵測系統相比，YOLO 雖然有較高的定位誤差，但在背景的預測上不太可能出現 False Positive 的狀況。最後，YOLO 不論在自然圖像或是藝術圖像等領域上，相較於其他的偵測法，如 DPM 及 R-CNN，可以學習到泛化性更好的物體表示法。

簡介 Introduction

人類只要掃過一張圖像，就可以知道有那些物體在圖像中、它們的位置以及它們如何交互作用。人類的視覺系統非常快速且準確，使我們可以不用太多的意識思考就可以完成極為複雜的任務，如駕駛車輛。同樣的，快速且準確的物件偵測演算法可以幫助電腦駕駛車輛而不須特殊的感測器，讓輔助設備可以傳送即時資料給使用者並且釋放響應式通用機器人系統的潛力。

現今的偵測系統都以分類器來進行偵測，這些系統在測試資料上使用物件分類器的多個位置上進行評估。像是 DPM 系統就是使用一個滑窗讓分類器在固定間隔的位置上對整張圖像進行判定。

更多類似 R-CNN 的偵測系統，先使用了 Region Proposal Method^[1]來找出圖像中可能的邊界框，再來針對這些可能的區域利用分類器進行預測。分類過後，再來調整邊界框、消除重複偵測並且根據圖像中其他物件重新估算邊界框。這樣複雜的流程也因為每一個別的部分都必須要分開訓練導致速度十分緩慢解難以優化。

我們重新定義物件偵測，將其視為一個單一的回歸問題，直接從圖像的 pixel 來估算邊界框座標並且給出分類機率。利用這樣的系統，一張圖像，" You Only Look Once " (YOLO) 就可以預測物件類別及位置。

我們從下圖可以知道 YOLO 非常的簡潔，只要一個 CNN 就可以同時進行多邊界框與其類別機率的預測。YOLO 對整張圖像進行訓練並且直接進行優化。這種統一模型對比其他傳統的物件偵測方法有非常多的優點。

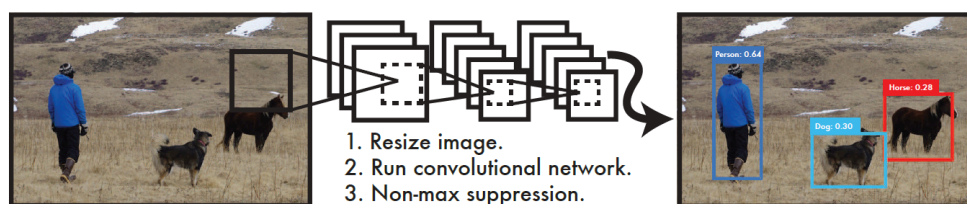


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448×448 , (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

1. YOLO 十分快速。

由於我們將整個偵測任務視為一個回歸問題，因此不需要太複雜的流程。在 GPU Titan X 上針對測試資料進行偵測任務，我們於全新的圖像上簡單的運行 YOLO，在沒有 batch 處理的狀態下可以達到每秒 45 幀，而在 Fast YOLO 上甚至可以超過每秒 150 幀。這表示我們可以處理即時的影像流，並且將延遲壓在 25 毫秒。再來，YOLO 比其他偵測系統有高出於兩倍的 mAP。作者將處理即使影像的 demo 都放在他的 project website : <http://pjreddie.com/yolo/> (<http://pjreddie.com/yolo/>).

2. YOLO 可對整張圖像進行全面性的推論。

不同於滑窗或是 Region Proposal Method 的方式，YOLO 可以在訓練/測試過程中看見整個圖像的全貌，因此 YOLO 可以將全局互相關聯的類別資訊以及外觀隱藏在編碼內。一個非常優秀的偵測系統 Fast R-CNN，因為僅會看見局部資訊，而導致它很容易將背景誤認為物件。YOLO 在這方面的背景錯誤率是 Fast R-CNN 的一半以下。

3. YOLO 可以學習到物件的泛化表徵。

當我們在真實圖像上作訓練，而在藝術作品上進行測試時，YOLO 相較於 DPM 及 R-CNN 有更傑出的表現。因此，YOLO 是一個高度泛化模型，在進行新領域或是沒接觸過的輸入圖像上比較不會出現問題。

即使有上述的優點，YOLO 在準確度上仍然落後這些先進的偵測系統。即使它可以快速的定位物件，但在一些小物體上要進行精準的定位還是有一些困難。作者會在論文中的實驗檢視這樣的權衡。

論文中的所有程式碼都已開源，多種預訓練好的模型也可供下載。

統一檢測 Unified Detection

作者將物件偵測中各個分開的部分全部整合到一個單一的神經網路中。這樣的網路結構從整張圖像上取得特徵並且用來預測邊界框。不僅如此，它還能同時在整張圖像上針對不同類別進行邊界框的預測。這表示這樣的網路可以全局理解整張圖像的物件。因此，YOLO 可以進行 End to End 的訓練並且速度接近即時，同時維持高 mAP。

YOLO 將圖像切成 $S \times S$ 個網格，如果物件的中心位於某一個網格內，則由此網格負責此物件的偵測。

每一個網格可以同時預測 B 個邊界框以及各自的信賴指數 (confidence score)。這些信賴指數反映了此模型對於邊界框包含某物件的信賴度，以及他認為這個邊界框預測的準確度。

$$Confidence = P_r(Object) \times IOU_{pred.}^{truth}$$

如果邊界框不含任何物件，則信賴指數為0，否則，我們希望信賴指數會等於 $IOU^{[2]}$ 。

每一個邊界框由五個預測值構成： $x, y, w, h, confidence$ ，其中 (x, y) 表示相對於所在網格邊界的邊界框的中心點座標。而 (w, h) 表示相對於整張圖像所預測出來的寬度及高度。最後，信賴預測值則為 IOU 。

每一個網格也會預測 C 個條件類別機率 $P_r(Class_i|Object)$ 。這些機率值是指包含此物件的網格條件下預測為 $Class_i$ 類別的機率。值得注意的是，無論預測邊界框的數量 B 有多少，每一個網格 YOLO 僅預測一組類別機率。

在測試階段，作者將條件類別機率乘上個別邊界框的信賴預測指數，這給出了一個邊界框中針對特定類別的信賴指數。

$$P_r(Class_i|Object) \times Confidence = P_r(Class_i|Object) \times P_r(Object) \times IOU_{pred.}^{truth} = P_r(Class_i)$$

這種計算方式，等同於將類別出現機率與預測邊界框的配適程度一起進行編碼。

object.

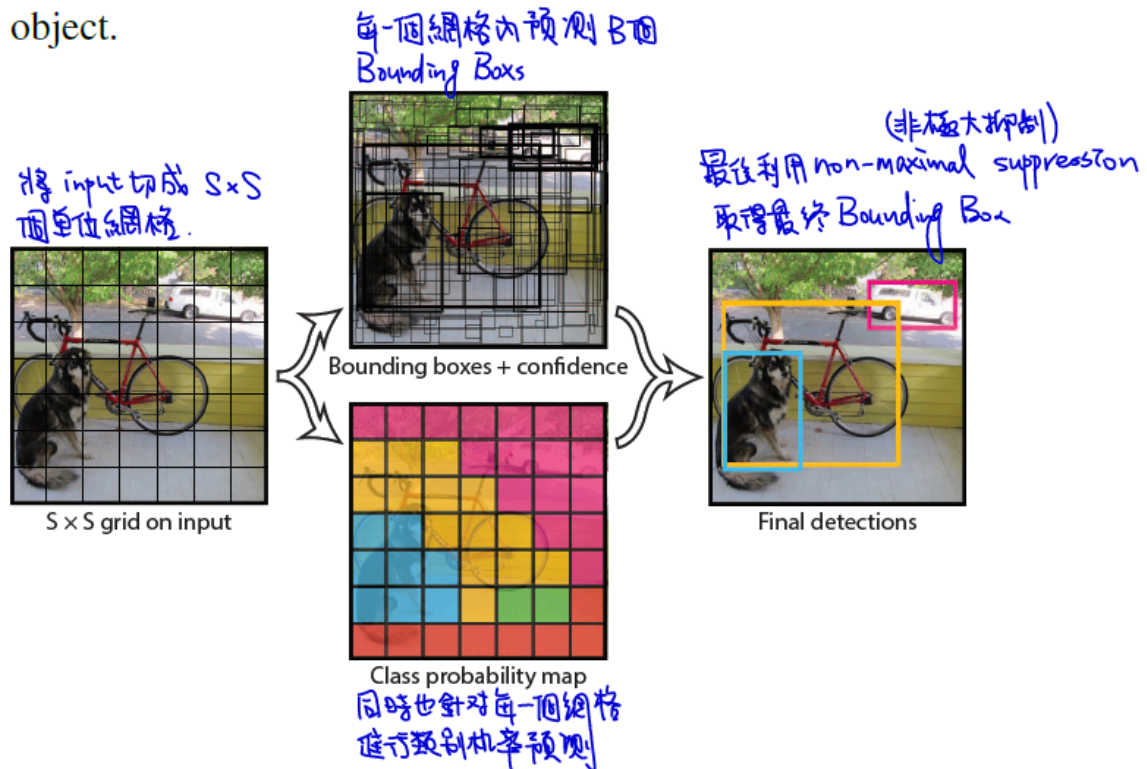


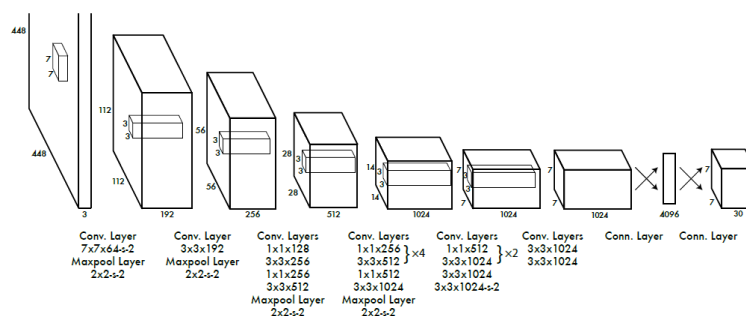
Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

$\rightarrow x, y, w, h, confidence$

論文內，利用 PASCAL VOC 資料集來評估 YOLO，作者們使用的參數： $S = 7$ ， $B = 2$ ， $C = 20$ ，因此最後會產生出一個 $7 \times 7 \times 30$ 的張量。

網路設計 Network Design

作者利用一個 CNN 結構實現這樣的 model，並且在 PASCAL VOC 資料集上做評估。初始的卷積層用來進行特徵萃取，而最後的全連接層則是輸出機率與座標。模型中的網路架構受到 GoogLeNet 所激勵，其中具有 24 個卷積層以及 2 個全連接層。作者也使用簡單的 3×3 卷積層再接一個 1×1 卷積層進行降維，用以取代 GoogLeNet 中的 Inception 模組。



研究團隊也訓練了一組快速版本的 YOLO 用以推進快速物件偵測的界線。Fast YOLO 使用一個僅有 9 層卷積層的 CNN，並且減少 filters 數量。除了網路尺寸的差別外，其餘的參數設計都與 YOLO 相同。一樣最後的輸出都是一個 $7 \times 7 \times 30$ 的張量。

訓練 Training

在整個模型中，作者先將前面 20 層卷積層接上一個平均池化層與一個全連接層利用1000個類別的 ImageNet 資料集作預訓練。預訓練時間大約花了將近一周，在 ImageNet 2012 資料集中，驗證資料可以達到 88 的 top-5 error，這結果與 Caffe 的模型集中的 GoogLeNet 相當。

接著轉換模型來進行偵測。在 Ren 等人的論文 "*Object Detection Networks on Convolutional Feature Maps*" 中有提到，在預訓練模型中加入卷積層以及連接層可以提升模型表現。依據他們論文中的範例，作者在預訓練模型中加入隨機初始化權重的 4 層卷積層以及兩層全連接層。由於物件偵測通常需要細粒度 (精細化) 的視覺資訊，因此增加網路輸入解析度，由 224×224 改變成 448×448 。

模型中最後的全連接層用來預測類別機率與邊界框的座標。研究團隊藉由全圖像的寬度與高度來標準化邊界框的寬度與高度使其值介於 0 到 1 之間。同時也利用特定網格位置來參數化邊界框的中心座標 (x, y) ，使其值也介於 0 到 1 之間。

除了最後一層使用 linear activation function 外，其餘都使用 leakly ReLU activation function。

$$\phi(x) = \begin{cases} x, & \text{if } x > 0 \\ 0.1x, & \text{otherwise} \end{cases}$$

YOLO 利用誤差平方和 (sum-square error) 做為 Loss function，使用這樣的方式計算 Loss 是因為這是非常容易優化的，但只有這樣卻無法符合我們要最大化平均精度的目標。如果只有這樣的計算，它會將定位誤差與分類誤差的權重調整成一樣，這樣的結果並不理想。另外一個角度來看，大部分的圖像中不含任何物件的網格通常都非常多，如果照上面的 Loss function 來看，最後 machine 會將這樣的網格計算出來的信賴指數都壓到 0，導致整個梯度下降的過程中，權重會偏向將所有網格都預測成不含物件的網格。這會導致整個模型的不穩定以及造成早期發散。

這裡論文中說了這麼多，其實簡單來說就是一個資料不平衡的狀況，在機器學習中如果沒有處理資料不平衡的問題，那麼機器會傾向於往數量的爆炸多的那個類別做預測，這樣還能夠得到高準確度。

為了解決這種資料不平衡問題，論文中的做法是在針對「有物件」及「沒有物件」的網格誤差另外給予一個權重做比例調整。加強有物件網格的 Loss，而壓低那些沒有物件網格的 Loss。作者們給了兩個參數 $\lambda_{coord} = 5$ 及 $\lambda_{noobj} = 0.5$ 來進行這樣的比例調整。

除了上述這種不平衡的狀況，單純的誤差平方和的計算方法其實是不管邊界框的大小，僅單純將重點放在誤差上面。但是我們想要的誤差評估應該是要讓大邊界框的小誤差的影響小於小邊界框的小誤差。為了解決這個問題，論文中採取的方式是直接將寬度跟高度取平方根來進行計算。

舉例來說，若我們有兩個真實邊界框長寬各為 16×16 以及 1×1 ，若我們有針對這兩個物件做出了兩個預測邊界框長寬各為 17×17 以及 2×2 ，若兩個預測按照我們的計算誤差都是 2，但我們知道這樣的誤差在這兩個大小不同的邊界框上應該有不同程度上的影響。但藉由取平方根，便可以將這兩者的影響表現出來，在大邊界框上的誤差變為 0.0303，而小邊界框上的誤差則變成 0.343。

YOLO 在每一個網格中會進行多個邊界框的預測。在訓練階段，我們只會需要一個預測子 (predictor, 其實就是邊界框) 來負責一個物件的偵測。決定預測子的方式就是看哪一個預測子可以得到最高的 IOU 就讓其來負責此物件的偵測。這樣的做法，可以讓整個邊界框的預測與預測子之間的關係專項化。每一個預測子可以都可以針對大小尺寸、方向角、物件種類都有更好的預測能力，也可以增進整體的 recall。

最後，我們在訓練階段要優化的 Loss Function 如下：

$$\begin{aligned}
& \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{I}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] && \text{邊界框中心誤差項} \\
& + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{I}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] && \text{邊界框長寬誤差項} \\
& + \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{I}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 && \text{含有物件邊界框的信心指數 (IOU) 誤差項} \\
& + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{I}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 && \text{不含物件邊界框的信心指數 (IOU) 誤差項} \\
& + \sum_{i=0}^{S^2} \mathbb{I}_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 && \text{含有物件網格的分類誤差項}
\end{aligned}$$

這裡有幾個符號是前面沒有提到的：

- $\mathbb{I}_i^{\text{obj}}$ ：在第 i 個網格中有偵測到物件
- $\mathbb{I}_{ij}^{\text{obj}}$ ：在第 i 個網格中的第 j 個偵測子(邊界框)負責偵測物件

由於 *LATEX* 支援問題，在這裡的 \mathbb{I} 指的就是上圖的空心 1。另外， \mathbb{I} 我們可以視為一個條件判斷子，當符合條件時後面的算式才會進行運算(這些部分應該也都可以用 *Iverson bracket* 來改寫)。

要注意的是，在這個 Loss Function 中，我們只對「含有物件網格的分類誤差」以及「作為預測子的邊界框座標誤差」(高 *IOU* 的邊界框)做懲罰。這邊主要講的就是前面說的 λ_{coord} 及 λ_{noobj} 在 Loss Function 中的帶來的懲罰效果。

研究團隊利用 PASCAL VOC 2007/2012 資料集，在訓練資料及驗證資料上進行 135 個 epochs 的訓練。統整一下整個訓練的參數設定如下：

```

batch_size = 64
momentum = 0.9
decay = 0.0005

Learning Rate :
1-75    epochs LR = 0.01
76-105  epochs LR = 0.001
106-135 epochs LR = 0.0001

Dropout :
dropout = 0.5

Data Augmentation :
20% scaling 縮放 / translation 平移
1.5 times exposure 曝光度 / saturation 飽和度

```

推論 Inference

跟訓練階段一樣，在測試資料上進行偵測也只需要一個網路就可以做到。在 PASCAL VOC 資料集上，同樣的網路結構可在每一張圖像上預測 98 個邊界框並且給予分類機率。YOLO 用這樣單一網路結構就可以達到非常快的預測速度，跟其他方法不一樣。

網格的設計，強制在邊界框的預測中將空間進行切割。通常哪一個物件落在哪一個網格中是很明顯的，而一個邊界框只會預測一個物件。然而很多時候我們會遇到物件很大或者是會跨越多個網格的物件，這時候就會利用 Non-maximal suppression (NMS, 非極大抑制) 來修正這種重複偵測的狀況。NSM 對 YOLO 的性能影響並不如 R-CNN 或是 DPM 般重要，但仍能增加 2 – 3% mAP。

YOLO 的限制 Limitation of YOLO

YOLO 在邊界框預測上有很強的空間限制，因為每一個網格僅能預測兩個單一目標的邊界框。這樣的空間限制，限制了 YOLO 對相鄰目標的預測。換句話說，YOLO 對於成群聚集的小物件（例如鳥群）有著預測上的困難。

再來，因為 YOLO 是利用資料來進行邊界框預測的學習，這表示模型很難泛化到新的、不常見的長寬比的物件。此外，YOLO 是利用較粗糙的特徵來進行邊界框的預測，因為 YOLO 的結構本身就對輸入圖像進行了多重的 downsampling。

最後，我們利用上述的 Loss Function 進行訓練來逼近偵測性能，但即使我們有對邊界框取根號後計算長寬誤差，但仍然對於大小邊界框中的誤差所造成的影響區別不夠明顯，也就是說，整個網路仍然對於大邊界框與小邊界框中的誤差一視同仁。最後導致我們的整體誤差來源其實只有定位上的誤差。

偵測系統比較 Comparison to Other Detection System

物件偵測是電腦視覺領域中最核心的問題，偵測流程通常始於圖像的特徵萃取（方法有 Haar, SIFT, HOG 以及 卷積特徵），再來進行分類及定位來對物體作識別。這些分類器或定位器利用滑窗的方式掃過整張圖像（或某一個子區域）。底下作者們比較了多種頂尖的物件偵測框架，來指出相似及相異處。

Deformable parts models (DPM)

DMP 是使用滑窗的方法來進行物件偵測。DPM 利用幾個獨立的流程來進行偵測：

1. 特徵萃取
2. 對區域做分類
3. 利用區域分數來進行邊界框預測

YOLO 則利用單一個 CNN 結構來取代上述的全部過程。這一個 CNN 會進行特徵萃取、邊界框預測、NMS 並同時進行上下文推理。跟 DPM 萃取的靜態特徵不同，這個 CNN 結構可以對特徵進行動態的持續性的訓練，並且再這個偵測任務中進行優化。YOLO 這個統一的結構不論就速度或是準確度而言都勝過 DPM。

R-CNN

R-CNN 及其變體都使用了 region proposals 來取代滑窗偵測。Selective Search (SS) 生成出許多潛在的邊界框，其次使用 CNN 進行特徵萃取，用 SVM 進行邊界框的評估，一個線性模型來調整邊界框，最後再使用 NMS 來消除重複的預測。這複雜流程中的每一個階段都必須要獨立進行精準的參數調整，導致最後的偵測系統速度緩慢，再測試階段偵測每一張圖像都要超過 40 秒的時間。

YOLO 跟 R-CNN 有跟 R-CNN 有一些相似性，每一個網格利用卷積結構提出潛在邊界框並且給予分數。然而 YOLO 在每一個網格中加上了空間限制，這有助於處理同一個物件重複偵測的問題。此外，YOLO 提出的邊界框數目每張圖像最多僅 98 個邊界框，遠小於 R-CNN 利用 SS 所提出的將近 2000 個邊界框。

最後，YOLO 結合了這些單獨的流程到一個單一的、共同優化的模型。

Other Fast Detectors

Fast R-CNN 與 Faster R-CNN 把重點放在利用運算共享以及利用神經網路來代替 SS 找出 region proposals 加速 R-CNN 上。雖然它們在速度與準確度的表現都勝過 R-CNN，但在即時的表現上仍顯不足。

許多的研究試圖要利用 HOG 的加速、使用級聯並且在 GPU 上推進運算來加速 DPM 的偵測流程，然而只有 30Hz DPM 可以進行即時偵測。

YOLO 試圖在結構設計上加速偵測，並不在各個獨立的偵測流程進行優化。

偵測器在單一類別上 (人臉、人) 是可以被高度優化的，因為需要處理的變化相對少得多。YOLO 就是一個學習怎麼同時偵測多種物件的通用偵測器。

Deep MultiBox

論文 : *Scalable object detection using deep neural networks.*

MultiBox 利用 CNN 來提出 region proposals，與 R-CNN 使用 SS 來提出 region proposal 不同。MultiBox 使用單一分類預測機率來取代信賴指數預測，可以針對單一分類進行物件偵測，但，MultiBox 無法進行廣泛的物件偵測。而且這些都還只是龐大的偵測流程中的一小部分，後續還需要更多圖像的 patch (怎麼翻譯都怪怪的XD) 分類。

YOLO 跟 MultiBox 都是使用 CNN 來進行邊界框的預測，但 YOLO 是一個更完整的偵測系統。

OverFast

論文 : *Overfeat: Integrated recognition, localization and detection using convolutional networks.*

OverFast 利用一個 CNN 來進行定位並藉此定位來偵測物件。它雖然使用有效率的滑窗偵測，但是一個獨立的系統。OverFast 在定位上取得優化，但這樣的優化卻沒有在偵測表現上。就如同 DPM，定位器在預測時只看到局部的資訊，無法總結上下文資訊，因此需要特別的後處理來進行一連串的偵測。

MultiGrasp

論文 : *Real-time grasp detection using convolutional neural networks.*

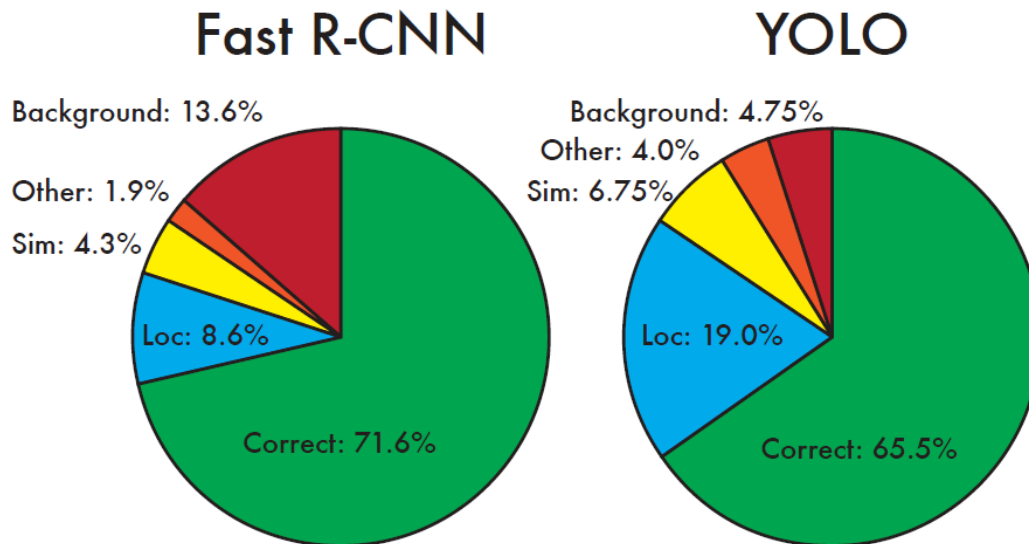
YOLO 在做的其實類似於上述論文中的偵測方式，方法就是利用 MultiGrasp 系統來進行迴歸分析以達到邊界框的抓取。

然而這種抓取偵測任務比物件偵測來的簡單，只需要為單一物件圖像預測單一個可抓取區域，不用估算物件尺寸、定位、邊界或其分類，只要找到可以抓取的區域即可。而 YOLO 則是要在多分類物件上進行邊界框及分類機率。

實驗 Experiments

(依照慣例省略，有興趣者請直接參閱論文)

實驗這一大部分我覺得比較有趣的結論是這個



YOLO 的確在準確度上不如 R-CNN 的相關變體，但是卻大大降低背景誤判的機率，犧牲一點準確率換得背景誤判降低以及速度提升，在一些現實的考量上的確非常划算。

現實中的即時檢測 Real-Time Detection In The Wild

YOLO 是一個快速、準確的物件偵測系統，因此是電腦視覺應用上的理想選擇。可以將 YOLO 接上視訊鏡頭驗證其即時偵測的表現，包含了擷取圖像的時間以及展示這些偵測。

最後產生的系統是具互動性且迷人的。雖然 YOLO 是一個獨立的圖像處理系統，但連接上視訊鏡頭後，它的功能會類似於一個追蹤系統，隨著物件移動或改變行為表現來進行物件偵測。

這樣的系統已經開源，可以於下列網址中找到：<http://pjreddie.com/yolo/> (<http://pjreddie.com/yolo/>)。

結論 Conclusion

研究團隊提出 YOLO 這樣一個統一的物件偵測模型。YOLO 本身非常容易打造並且可以直接在整張圖像上訓練。不像其他以分類器為基礎的方法，YOLO 直接使用檢測性能來對 Loss function 做訓練，而且整個模型是一起訓練的。

Fast YOLO 是目前文獻中（在當時）最快的通用檢測系統，而 YOLO 則推進了即時物件偵測的發展。且 YOLO 也可以在新的領域上泛化得很好，使得 YOLO 可以成為快速、強健的物件偵測系統的理想選擇。

參考資料

1. 深度學習-什麼是one stage，什麼是two stage 物件偵測
(<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>).
2. DPM (Deformable Part Model) 原理详解 (https://blog.csdn.net/qg_14845119/article/details/52625426).
3. 深度学习中IU、IoU(Intersection over Union)的概念理解以及python程序实现
(<https://blog.csdn.net/IAMoldpan/article/details/78799857>).

4. 機器/深度學習: 物件偵測 Non-Maximum Suppression (NMS)

(<https://medium.com/@chih.sheng.huang821/%E6%A9%9F%E5%99%A8-%E6%B7%B1%E5%BA%A6%E5%AD%B8%E7%BF%92-%E7%89%A9%E4%BB%B6%E5%81%B5%E6%B8%AC-non-maximum-suppression-nms-aa70c45adffa>).

5. 「深度學習訓練」與「推論」之間有什麼差別? (<https://blogs.nvidia.com.tw/2016/08/difference-deep-learning-training-inference-ai/>).


註釋

1. 利用特殊方式先選出物件在進行偵測，而選出物件的這個過程我們稱之為 Region Proposal Method ↵

2. intersection over union (*IOU*)，指的是真實邊界框與預測邊界框的重疊面積除以兩個邊界框的聯集面積。

$$IOU = \frac{overlap}{union}$$

IoU = $\frac{\text{Area of Overlap}}{\text{Area of Union}}$ ↵



<http://blog.csdn.net/11Wolfsong>