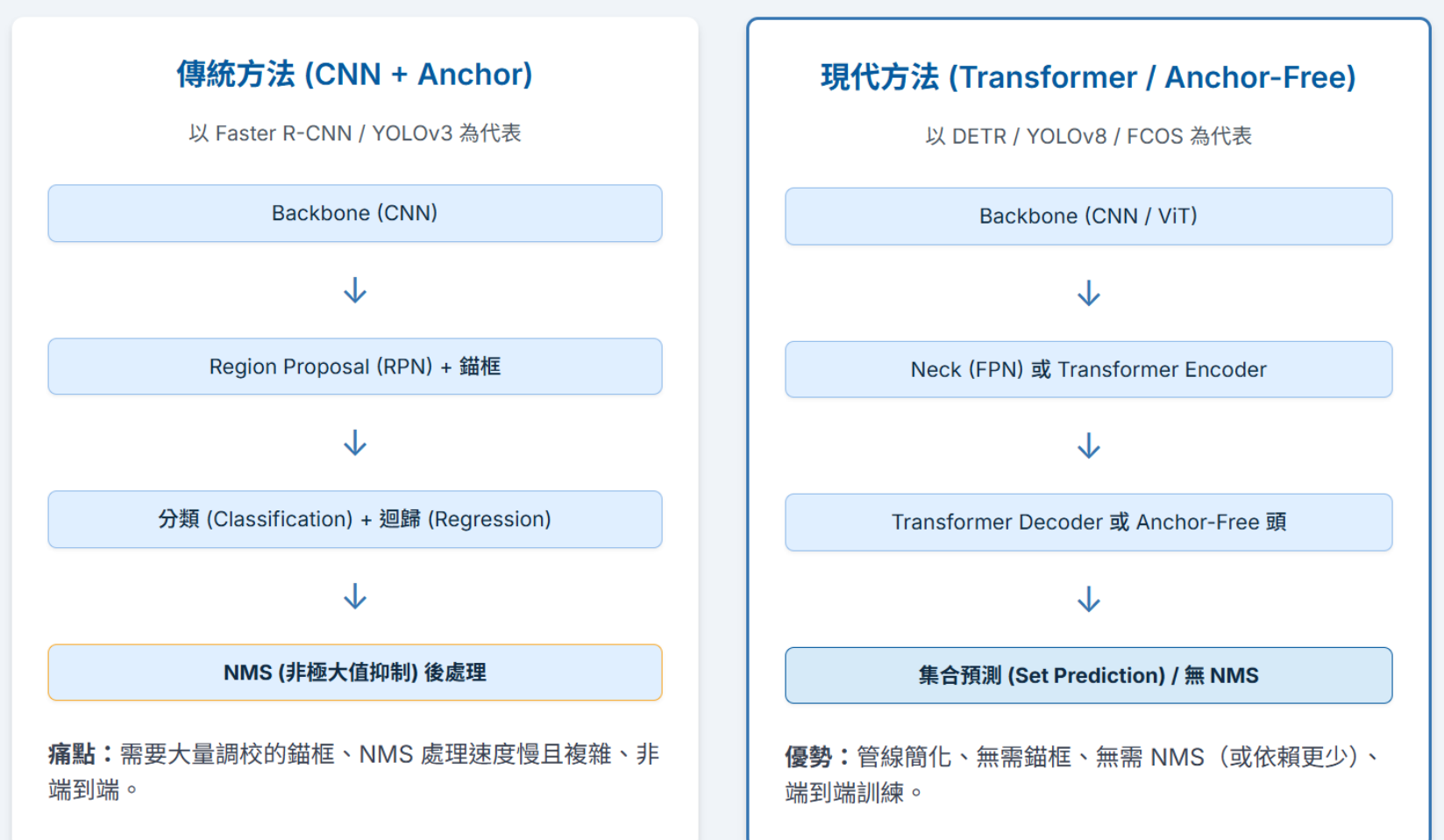
TRENDS AFTER 2020

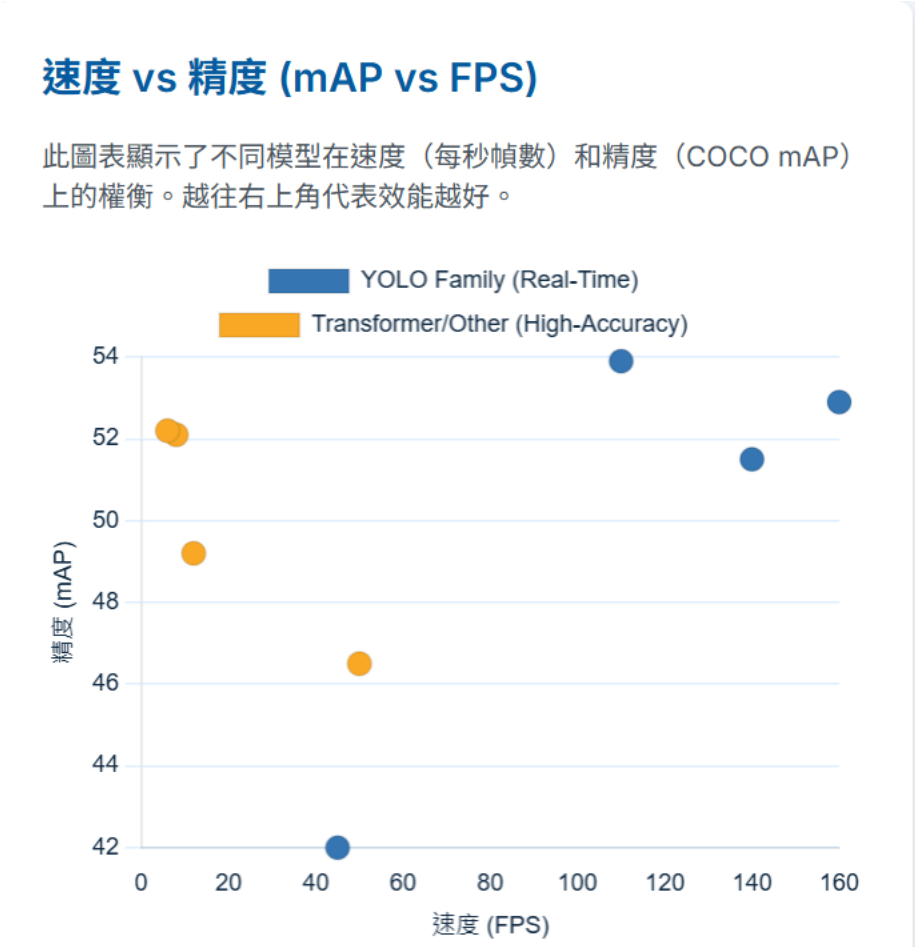
典範轉移：從錨框 (Anchor) 到 Transformer

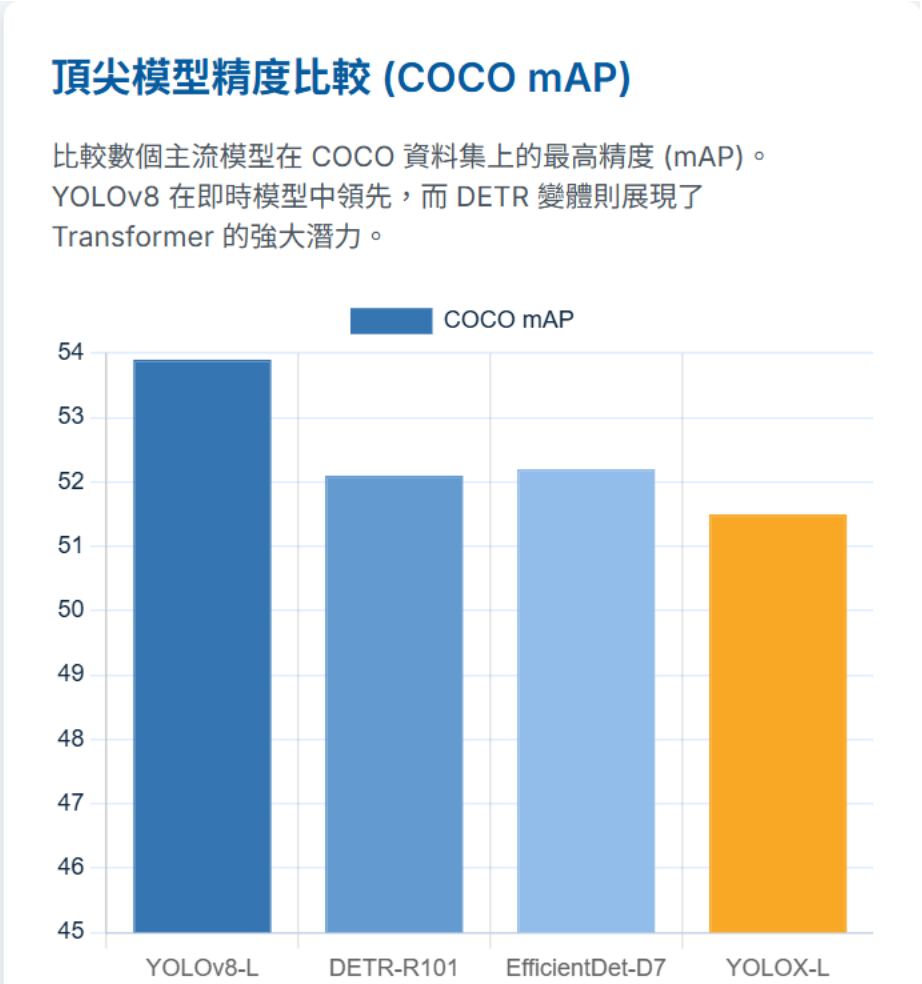
- 最大的變革之一是架構的簡化。
  - 我們正從依賴複雜手工設計（如錨框和 NMS）的流程，轉向更簡潔、端到端 (End-to-End) 的 Transformer 模型。



效能競賽：速度 (FPS) 與精度 (mAP)

- YOLO 家族持續在即時偵測領域制霸，不斷推高「效率前緣」；
- 而 Transformer 基礎的模型則在犧牲部分速度下，達到了極高的精度。





超越邊界框：新興的偵測能力

- 近年的趨勢是讓模型不僅能「偵測」，更能「理解」。模型正從辨識固定類別，轉向能理解自然語言描述、偵測新物體的能力。

開放詞彙偵測

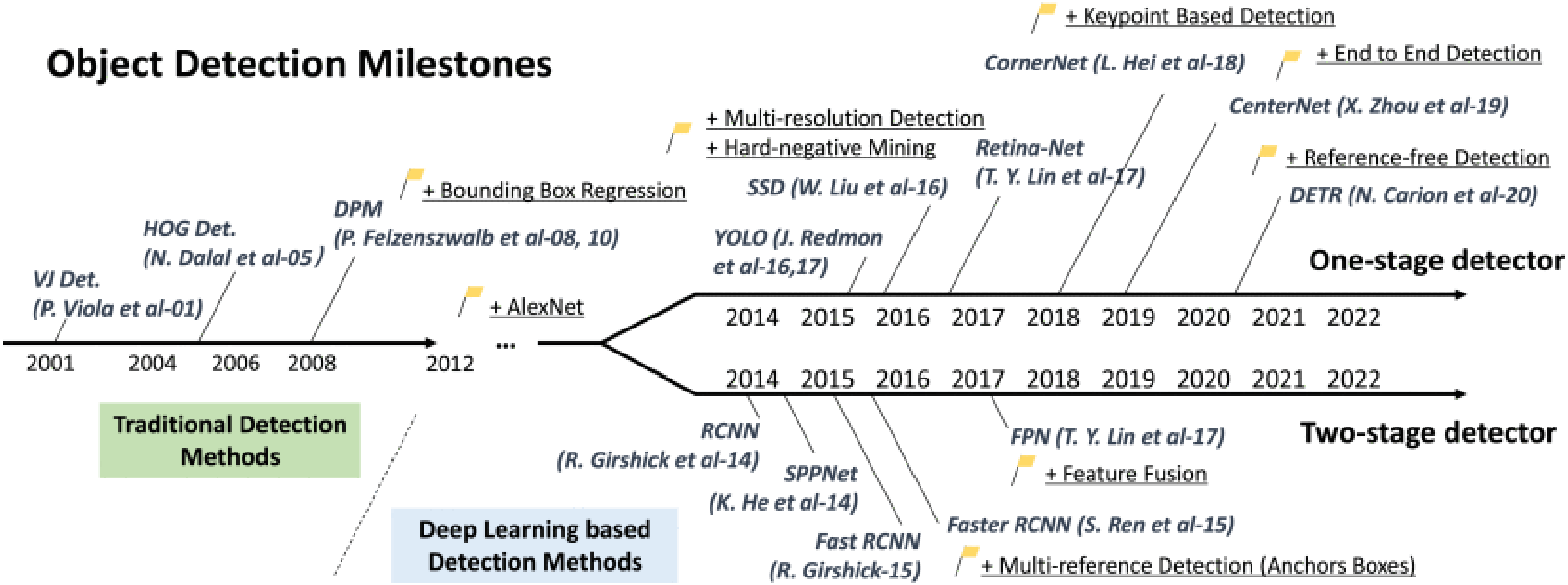
模型不再局限於訓練時的 80 個類別。得益於 CLIP 等視覺-語言模型的進展，現在可以輸入任意文字（如 "一隻正在跳躍的柯基"）來偵測對應物體。

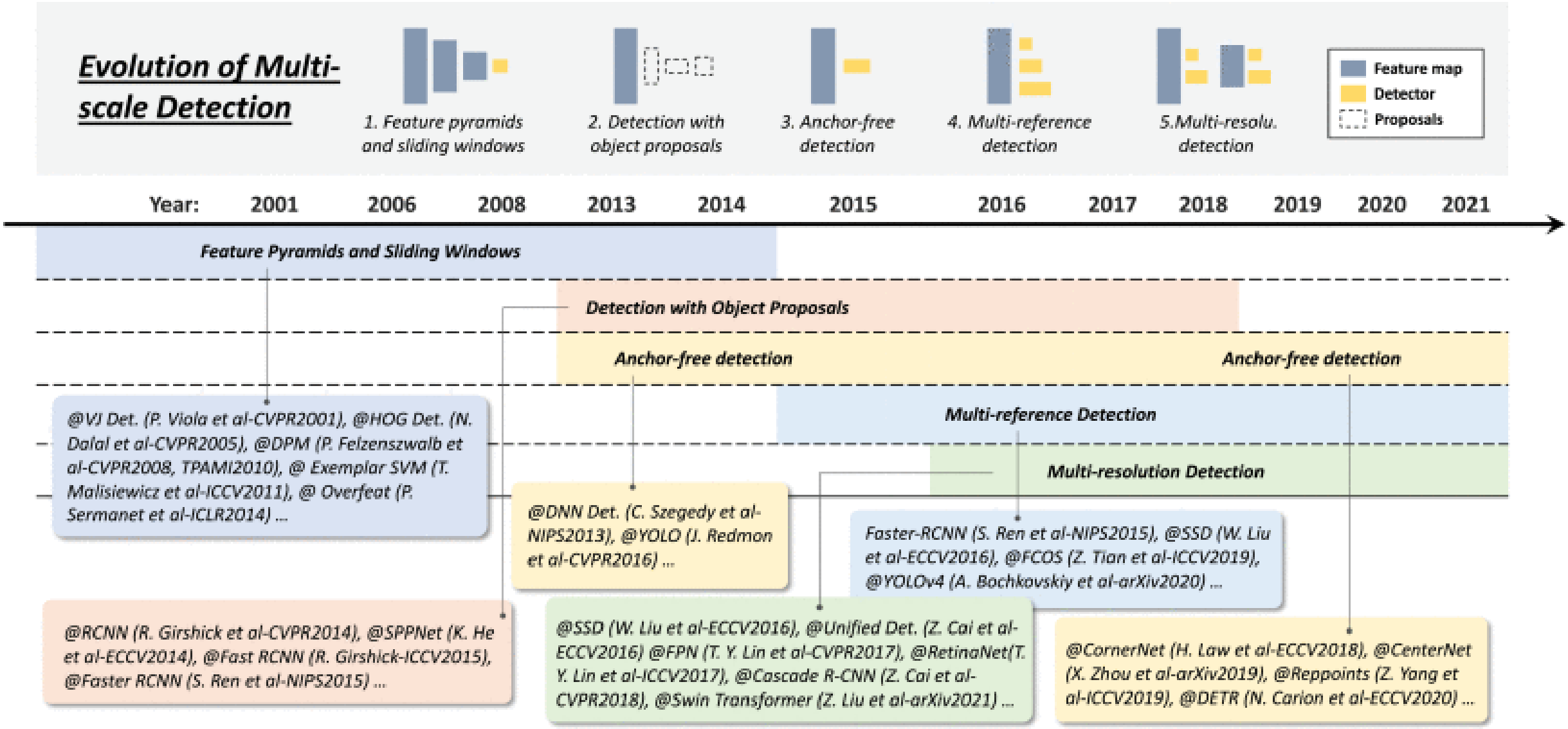
零/少樣本偵測

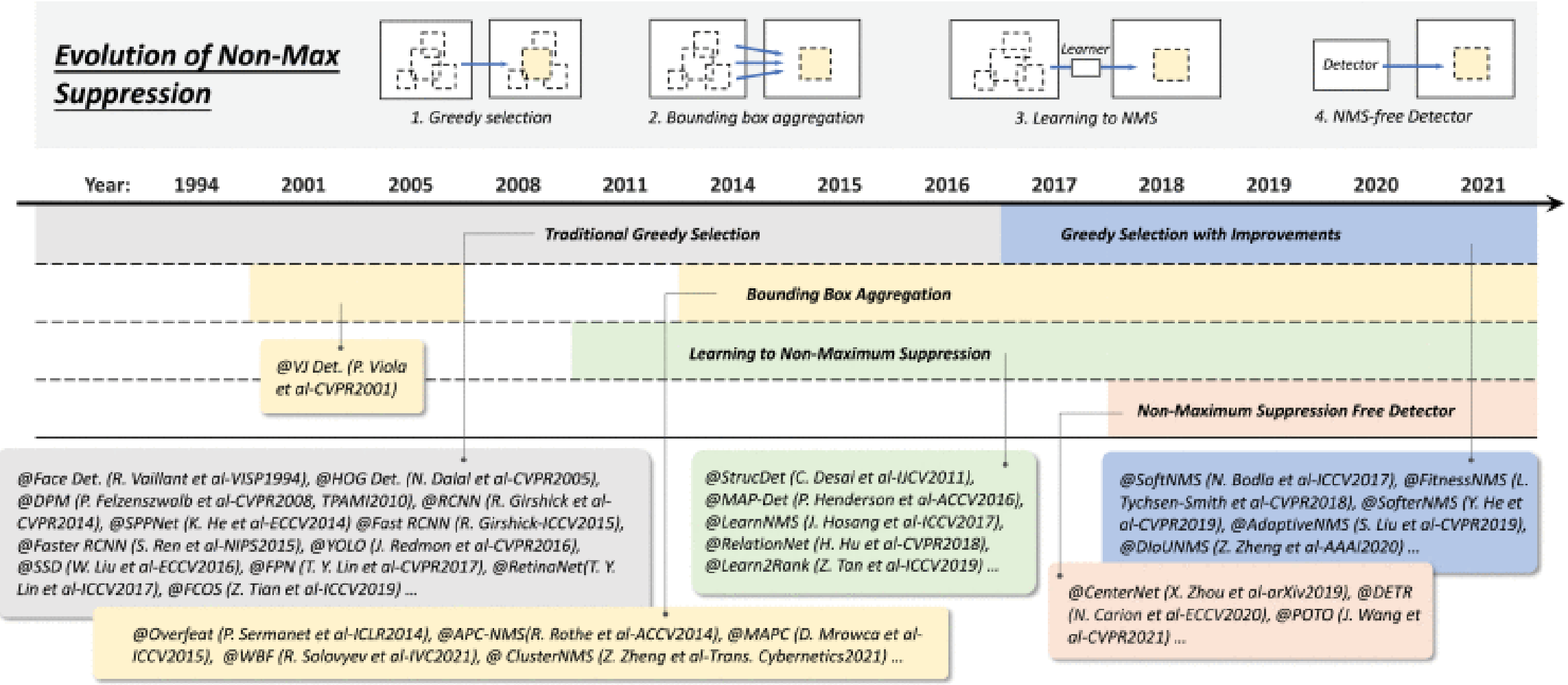
傳統模型需要數千張標註圖片才能學習一個新類別。而今，模型只需看過 0 張（Zero-Shot）或 1-10 張（Few-Shot）範例，就能開始偵測新物體。

通用基礎模型

以 SAM (Segment Anything Model) 和 Grounding DINO 為首，這些大型模型在大規模資料上預訓練，能執行多種任務，如精確分割、根據描述定位等。

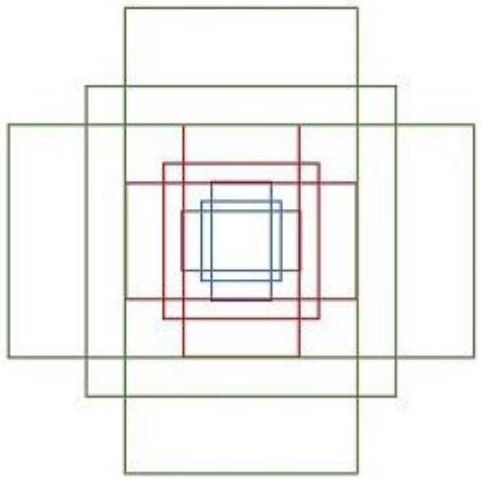

Object Detection Milestones
+ Keypoint Based Detection
CornerNet (L. Hei et al-18)
+ End to End Detection
CenterNet (X. Zhou et al-19)
+ Multi-resolution Detection
+ Hard-negative Mining
Retina-Net
(T. Y. Lin et al-17)
+ Reference-free Detection
DETR (N. Carion et al-20)
+ Bounding Box Regression
SSD (W. Liu et al-16)
DPM
(P. Felzenszwalb et al-08, 10)
HOG Det.
(N. Dalal et al-05)
YOLO (J. Redmon
et al-16,17)
VJ Det.
(P. Viola et al-01)
+ AlexNet
One-stage detector
2001 2004 2006 2008 2012 ... 2014 2015 2016 2017 2018 2019 2020 2021 2022
2014 2015 2016 2017 2018 2019 2020 2021 2022
Two-stage detector
Traditional Detection Methods
RCNN
(R. Girshick et al-14)
SPPNet
(K. He et al-14)
FPN (T. Y. Lin et al-17)
+ Feature Fusion
Deep Learning based Detection Methods
Fast RCNN
(R. Girshick-15)
Faster RCNN (S. Ren et al-15)
+ Multi-reference Detection (Anchors Boxes)


Evolution of Multi-scale Detection
1. Feature pyramids and sliding windows
2. Detection with object proposals
3. Anchor-free detection
4. Multi-reference detection
5.Multi-resolu. detection
Feature map
Detector
Proposals
Year: 2001 2006 2008 2013 2014 2015 2016 2017 2018 2019 2020 2021
Feature Pyramids and Sliding Windows
Detection with Object Proposals
Anchor-free detection
Anchor-free detection
Multi-reference Detection
Multi-resolution Detection
@VJ Det. (P. Viola et al-CVPR2001), @HOG Det. (N. Dalal et al-CVPR2005), @DPM (P. Felzenszwalb et al-CVPR2008, TPAMI2010), @ Exemplar SVM (T. Malisiewicz et al-ICCV2011), @ Overfeat (P. Sermanet et al-ICLR2014) ...
@DNN Det. (C. Szegedy et al-NIPS2013), @YOLO (J. Redmon et al-CVPR2016) ...
Faster-RCNN (S. Ren et al-NIPS2015), @SSD (W. Liu et al-ECCV2016), @FCOS (Z. Tian et al-ICCV2019), @YOLOv4 (A. Bochkovskiy et al-arXiv2020) ...
@RCNN (R. Girshick et al-CVPR2014), @SPPNet (K. He et al-ECCV2014), @Fast RCNN (R. Girshick-ICCV2015), @Faster RCNN (S. Ren et al-NIPS2015) ...
@SSD (W. Liu et al-ECCV2016), @Unified Det. (Z. Cai et al-ECCV2016) @FPN (T. Y. Lin et al-CVPR2017), @RetinaNet(T. Y. Lin et al-ICCV2017), @Cascade R-CNN (Z. Cai et al-CVPR2018), @Swin Transformer (Z. Liu et al-arXiv2021) ...
@CornerNet (H. Law et al-ECCV2018), @CenterNet (X. Zhou et al-arXiv2019), @Reppoints (Z. Yang et al-ICCV2019), @DETR (N. Carion et al-ECCV2020) ...


Evolution of Non-Max Suppression
1. Greedy selection
2. Bounding box aggregation
Learner
3. Learning to NMS
Detector
4. NMS-free Detector
Year: 1994 2001 2005 2008 2011 2014 2015 2016 2017 2018 2019 2020 2021
Traditional Greedy Selection
Greedy Selection with Improvements
Bounding Box Aggregation
Learning to Non-Maximum Suppression
Non-Maximum Suppression Free Detector
@VJ Det. (P. Viola et al-CVPR2001)
@Face Det. (R. Vaillant et al-VISP1994), @HOG Det. (N. Dalal et al-CVPR2005), @DPM (P. Felzenszwalb et al-CVPR2008, TPAMI2010), @RCNN (R. Girshick et al-CVPR2014), @SPPNet (K. He et al-ECCV2014) @Fast RCNN (R. Girshick-ICCV2015), @Faster RCNN (S. Ren et al-NIPS2015), @YOLO (J. Redmon et al-CVPR2016), @SSD (W. Liu et al-ECCV2016), @FPN (T. Y. Lin et al-CVPR2017), @RetinaNet(T. Y. Lin et al-ICCV2017), @FCOS (Z. Tian et al-ICCV2019) ...
@Overfeat (P. Sermanet et al-ICLR2014), @APC-NMS(R. Rothe et al-ACCV2014), @MAPC (D. Mrowca et al-ICCV2015), @WBF (R. Solovyev et al-IVC2021), @ ClusterNMS (Z. Zheng et al-Trans. Cybernetics2021) ...
@StrucDet (C. Desai et al-IJCV2011), @MAP-Det (P. Henderson et al-ACCV2016), @LearnNMS (J. Hosang et al-ICCV2017), @RelationNet (H. Hu et al-CVPR2018), @Learn2Rank (Z. Tan et al-ICCV2019) ...
@CenterNet (X. Zhou et al-arXiv2019), @DETR (N. Carion et al-ECCV2020), @POTO (J. Wang et al-CVPR2021) ...
@SoftNMS (N. Bodla et al-ICCV2017), @FitnessNMS (L. Tychsen-Smith et al-CVPR2018), @SofterNMS (Y. He et al-CVPR2019), @AdaptiveNMS (S. Liu et al-CVPR2019), @DIoUNMS (Z. Zheng et al-AAAI2020) ...

ANCHOR FREE OBJECT DETECTION

Anchor v.s. Anchor Free

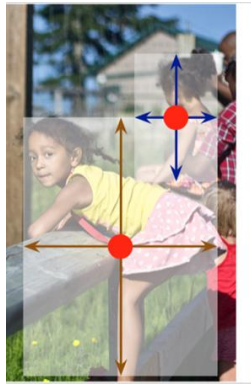


- Anchor
  - 預先定義的方框，通常是一次使用k個。比較容易收斂
  - 優點
    - 透過 不同尺度的anchor，可以減少面對物件scale和aspect ratio變化範圍大的問題。
    - 透過控制anchor數量，可以降低image pyramid層數並依情況調整運算量。
    - 可以透過調整anchor的尺度與密度，適應不同類型的任務。
  - 缺點
    - 設計anchor的hyper parameter不易調整
    - 訓練階段計算anchor的IOU（ Intersection Over Union）需要消耗大量的時間與memory
- Anchor free
  - 不事先定義方框，使用keypoint 關鍵點來預測bounding box

Anchor Free Object Detection

- 問題點
  - 不同尺度的物件辨識困難
    - 之前: image pyramid, feature pyramid
  - 訓練時過多的負樣本數過多
    - 之前:focal loss, two stages RPN
- Approaches
  - Approach: 確定物體中心和對四個邊框的預測
  - Keypoint 基於關鍵點
    - 在一個高分辨率的特徵圖上進行檢測，其優點是準確率高，但是計算量大
    - CornerNet, ExtremeNet, CenterNet,
  - anchor-point
    - 在多個分辨率上進行檢測 (RetinaNet like)，結構簡單，速度更快
    - 性能不高
    - FSAF, guided anchoring, FoveaBox, FCOS, SAPD

anchor-point



Keypoint

Trends

- Early work
  - Dense box (2015): for face detection only. Per pixel prediction
  - YOLO: general OD, per grid prediction
- Later work
  - Keypoint
    - 以Corner (使用了物件的邊緣特徵來進行預測)定義BB
      - Corner as keypoint: CornerNet
      - Extreme point as keypoint: ExtremeNet
      - 大部份使用了物件的邊緣特徵來進行預測 其容易產生一些無效框 而且由於是使用角點 也使得偵測的中心位置不在目標上
    - CenterNet
      - 在CornerNet的基礎上再加上一個**預測物件中心點來濾除無效框** 並且也使其除了**邊緣特徵外也能力用物件內部特徵**
  - 密集預測(per pixel prediction)
    - (都利用 FPN 來進行多尺度目標檢測。透過密集預測進行分類和回歸的。)
    - FSAF/FCOS
- 各種方法的關鍵在於 gt 如何定義
  - cornernet 定義為角點，extremenet 定義為極值點和中心點，FSAF、FoveaBox 定義為矩形框的中間區域，FCOS 雖然是矩形框，但是經過 center-ness 抑制掉低質量的框，其實也是一種變相的將 gt 定義為矩形框中心區域。
  - gt 重新定義之後，需要偵測的目標語意變得更加明確了，有利於分類和迴歸。

Dense Box

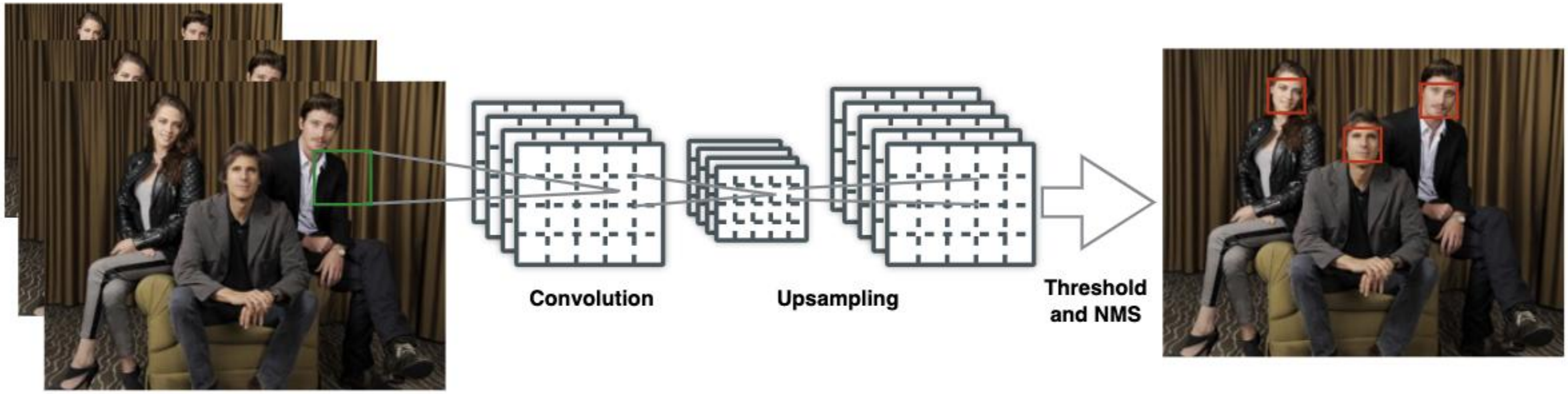
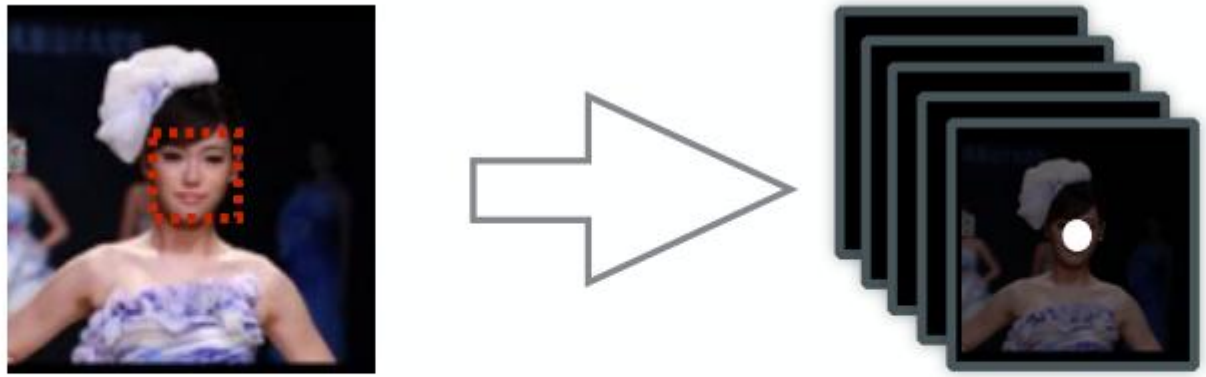


Figure 1: **The DenseBox Detection Pipeline.** 1) Image pyramid is fed to the network. 2) After several layers of convolution and pooling, upsampling feature map back and apply convolution layers to get final output. 3) Convert output feature map to bounding boxes , and apply non-maximum suppression to all bounding boxes over the threshold.

如圖 1 所示，單個 FCN 同時產生多個預測 bbox 和置信分數的輸出。測試時，整個系統將圖片作為輸入，輸出 5 個通道的 feature map。每個 pixel 的輸出 feature map 得到 5 維的向量，包括一個置信分數和 bbox 邊界到該 pixel 距離的 4 個值。最後輸出 feature map 的每個 pixel 轉化為帶分數的 bbox，然後經過 NMS 後處理。

https://arxiv.org/abs/1509.04874



第一個通道的 ground truth map 的正標籤區域由半徑為 r 的圓填充，圓的中心點位於 bbox 的中點。而剩下的 4 個通道由 bbox 的 2 個角點決定

Figure 2: **The Ground Truth Map in Training .** The left image is the input patch, and the right one is its ground truth map.

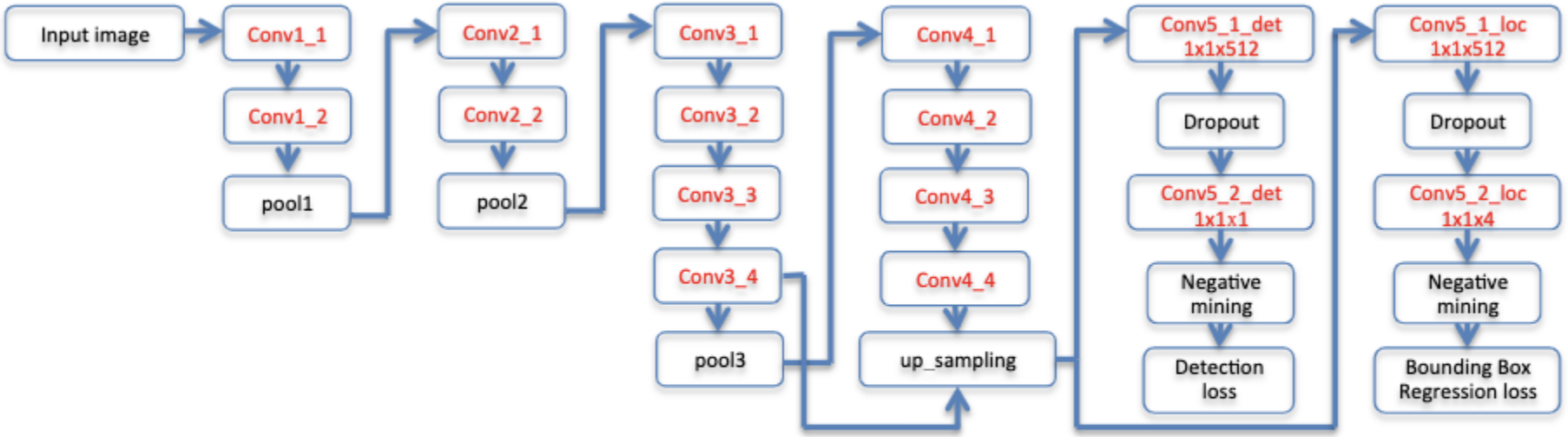
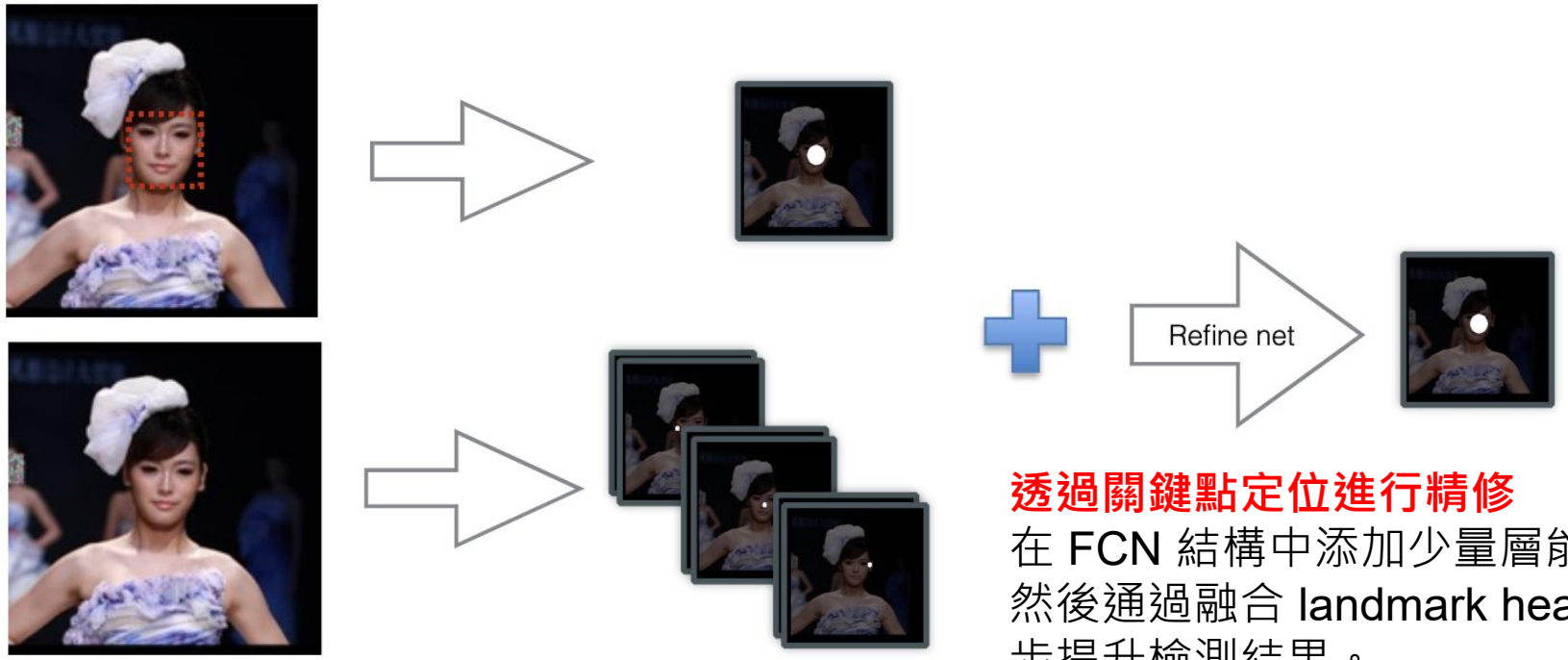


Figure 3: **Network architecture of DenseBox.** The rectangles with red names contain learnable parameters.



透過關鍵點定位進行精修

在 FCN 結構中添加少量層能夠實現 landmark localization，然後通過融合 landmark heatmaps 和 score map 可以進一步提升檢測結果。

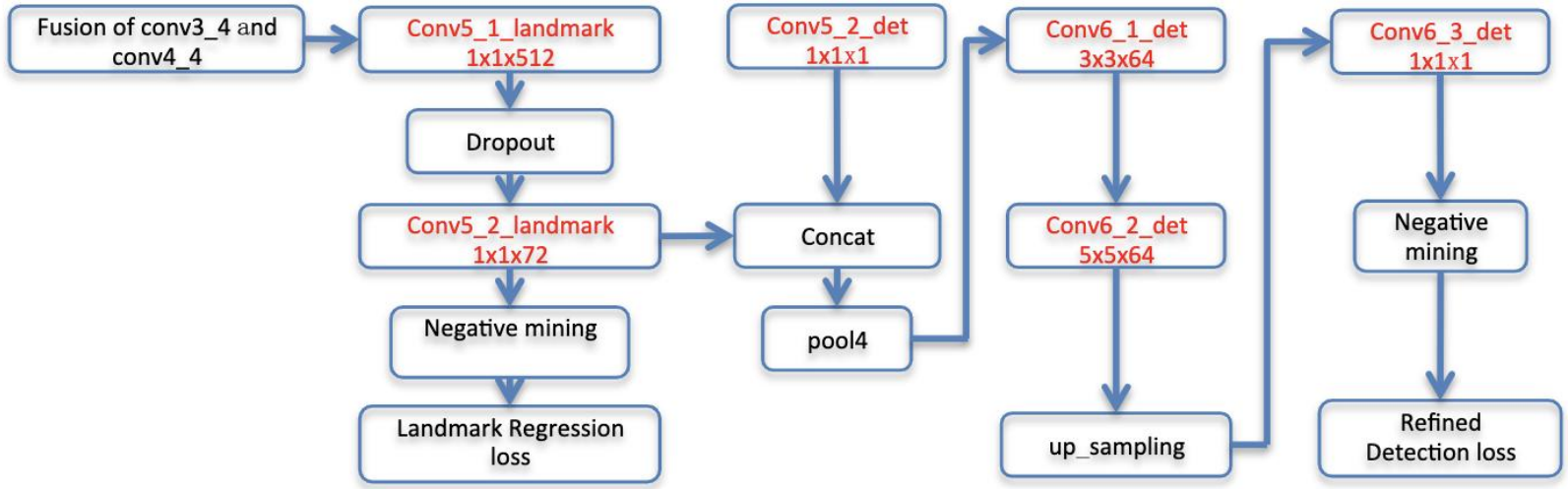


Figure 4: **Top:** The pipeline of DenseBox with landmark localization. **Bottom:** The network structure for landmark localization.

CornerNet

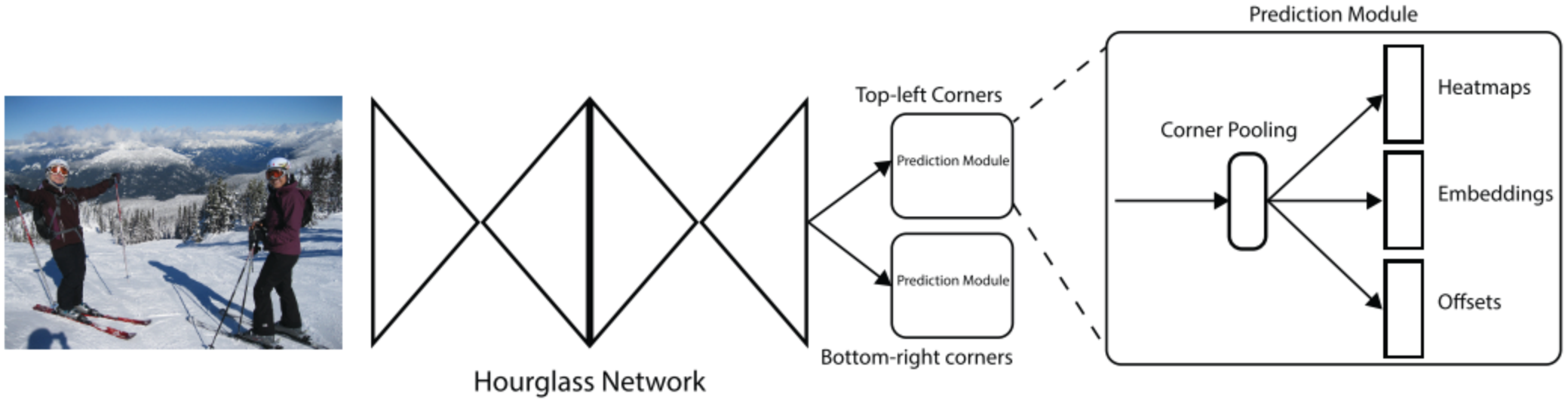


Fig. 4. Overview of CornerNet. The backbone network is followed by two prediction modules, one for the top-left corners and the other for the bottom-right corners. Using the predictions from both modules, we locate and group the corners.

1.透過檢測 bbox 的一對角點來檢測出目標。

2.提出 corner pooling，來更好的定位 bbox 的角點。

https://arxiv.org/abs/1808.01244

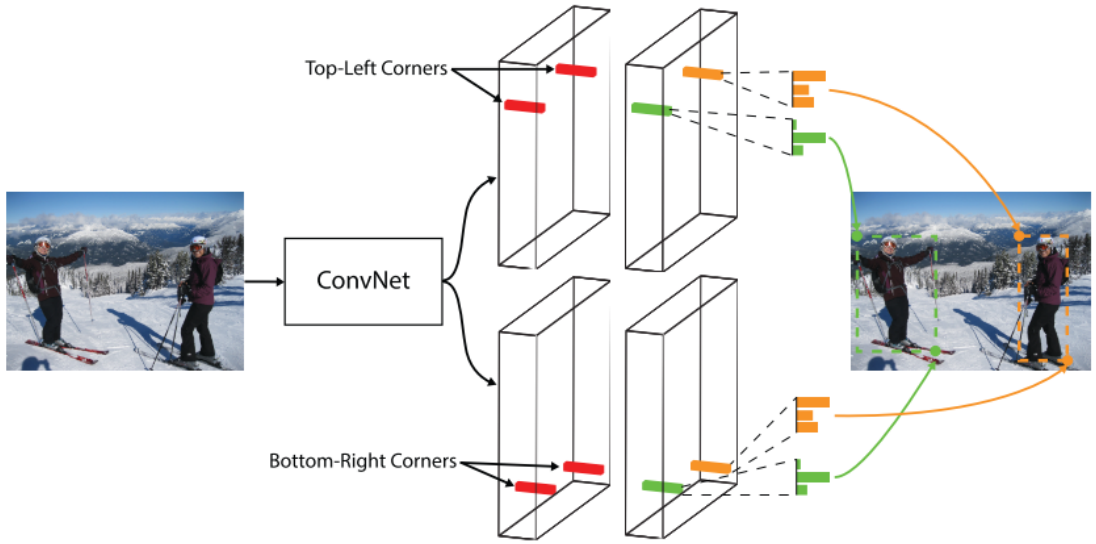


Fig. 1. We detect an object as a pair of bounding box corners grouped together. A convolutional network outputs a heatmap for all top-left corners, a heatmap for all bottom-right corners, and an embedding vector for each detected corner. The network is trained to predict similar embeddings for corners that belong to the same object.

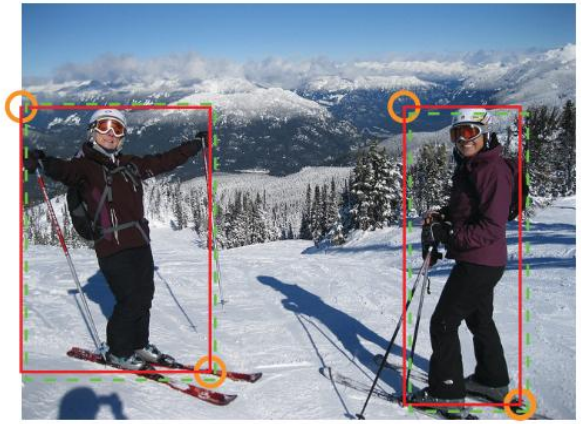


Fig. 5. "Ground-truth" heatmaps for training. Boxes *(green dotted rectangles)* whose corners are within the radii of the positive locations *(orange circles)* still have large overlaps with the ground-truth annotations *(red solid rectangles)*.

偵測角落

首先預測出兩組 heatmaps，一組為 top-left 角點，另一組為 bottom-right 角點。每組 heatmaps 有 C 個通道，表示 C 個類別，尺寸為 HxW。每個通道是一個 binary mask，表示一個類的角點位置。

對於每個角點來說，只有一個 gt 正例位置，其他都為負例位置。訓練時，以正例位置為圓心，設置半徑為 r 的範圍內，減少負例位置的懲罰(採用二維高斯的形式)，如上圖所示。

受到多人姿態估計論文的啟發，基於角點 embedding 之間的距離來對角點進行分組。

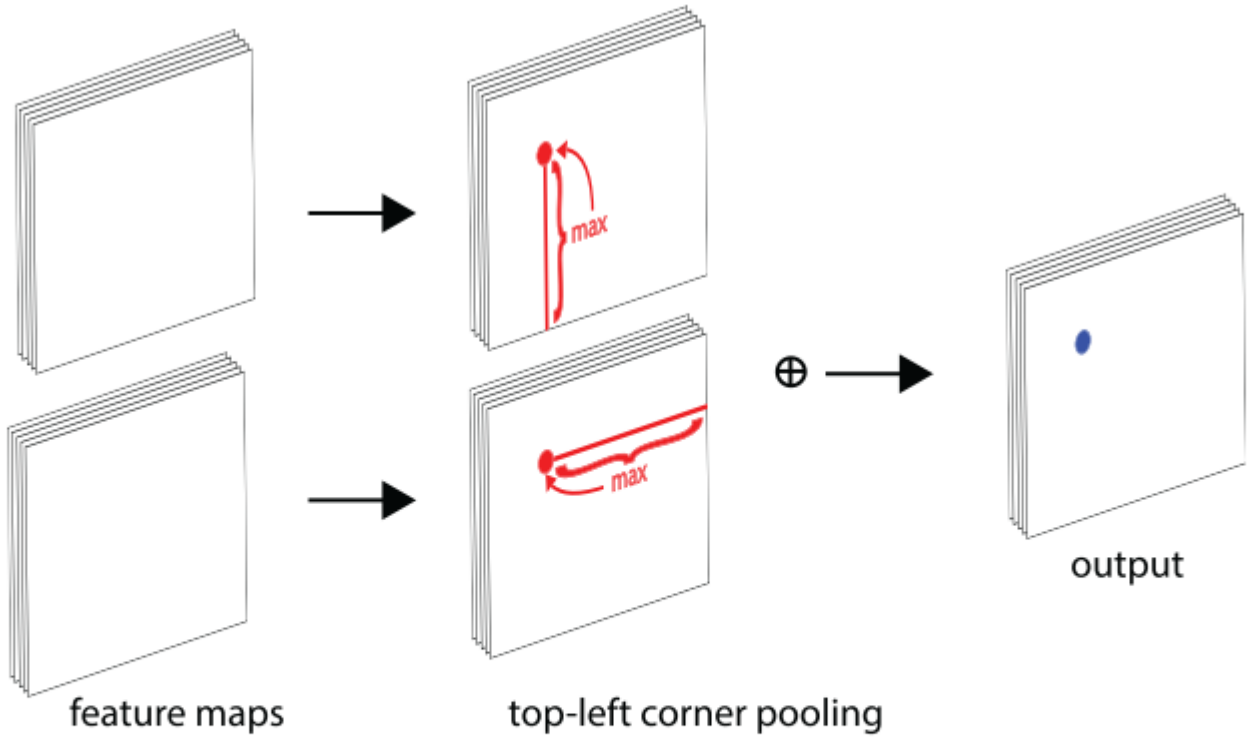


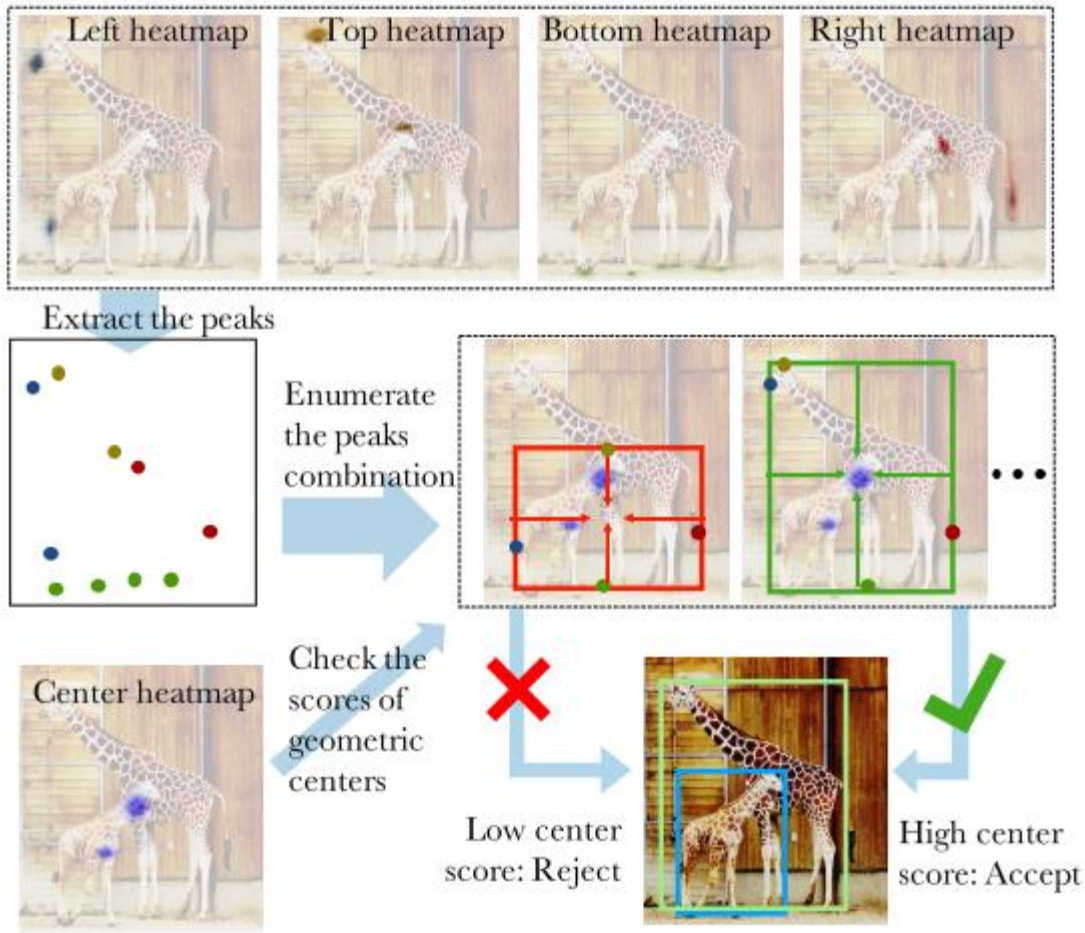
Fig. 3. Corner pooling: for each channel, we take the maximum values *(red dots)* in two directions *(red lines)*, each from a separate feature map, and add the two maximums together *(blue dot)*.

Corner Pooling

在每個像素位置，對第一個特徵圖右方的所有特徵向量進行最大池化，對第二個特徵圖下方的所有特徵向量進行最大池化，然後將兩個池化結果相加。

提出 corner pooling，來更好的定位 bbox 的角點。

ExtremeNet



1.將關鍵點定義為極值點。
2.根據幾何結構對關鍵點進行分組。

作者使用了最佳的關鍵點估計框架，透過對每個目標類別預測 4 個多峰值的 heatmaps 來尋找極值點。

分組演算法的輸入是每個類別的 5 個 heatmaps，一個 center heatmap 和 4 個 extreme heatmaps，透過偵測所有的峰值來提取出 5 個 heatmaps 的關鍵點。

給出 4 個極值點，計算幾何中心，如果幾何中心在 center map 上對應高響應，那麼這 4 個極值點為有效檢測。

作者使用暴力列舉的方式來得到所有有效的 4 個關鍵點。

CenterNet

- CornerNet
  - 使用角點來偵測物體，代表他大部份使用了物件的邊緣特徵來進行預測 其容易產生一些無效框。
  - 而且由於是使用角點 也使得偵測的中心位置不在目標上
- CenterNet
  - 在CornerNet的基礎上再加上一個預測物件中心點來濾除無效框 並且也使其除了邊緣特徵外也能力用物件內部特徵
  - Corner pooling => cascade corner pooling

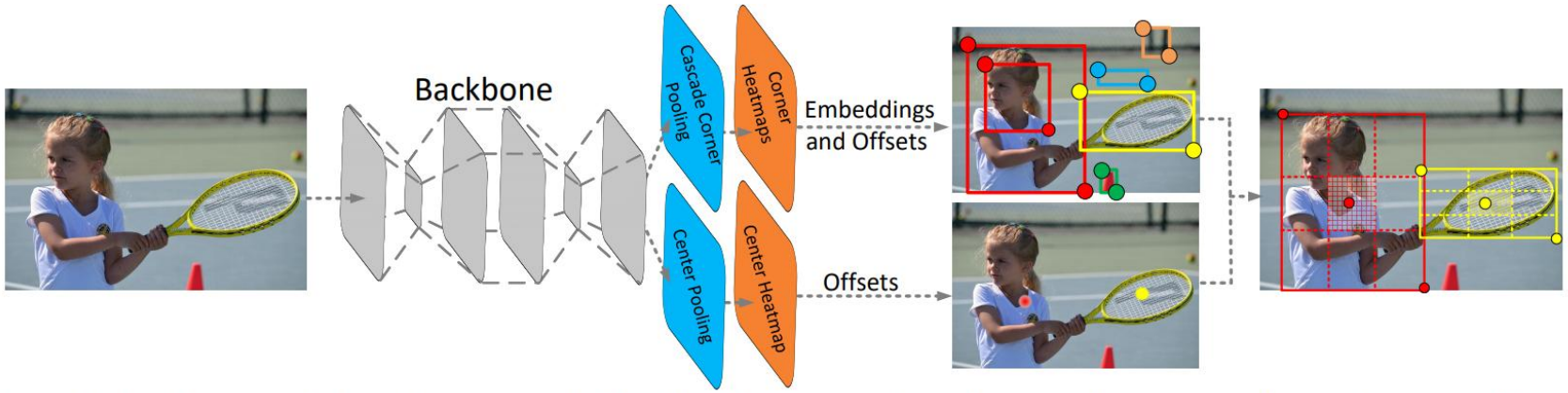


Figure 2: Architecture of CenterNet. A convolutional backbone network applies cascade corner pooling and center pooling to output two corner heatmaps and a center keypoint heatmap, respectively. Similar to CornerNet, a pair of detected corners and the similar embeddings are used to detect a potential bounding box. Then the detected center keypoints are used to determine the final bounding boxes.

- CornerNet 為什麼容易產生很多的誤檢
  - CornerNet 通過檢測角點確定目標，而不是通過初始候選框 anchor 的回歸確定目標，由於沒有了anchor 的限制，使得任意兩個角點都可以組成一個目標框，這就對判斷兩個角點是否屬於同一物體的算法要求很高，一但準確度差一點，就會產生很多錯誤目標框
  - 此算法在判斷兩個角點是否屬於同一物體時，缺乏全局信息的輔助，因此很容易把原本不是同一物體的兩個角點看成是一對角點，因此產生了很多錯誤目標框
  - 角點的特徵對邊緣比較敏感，這導致很多角點同樣對背景的邊緣很敏感，因此在背景處也檢測到了錯誤的角點。

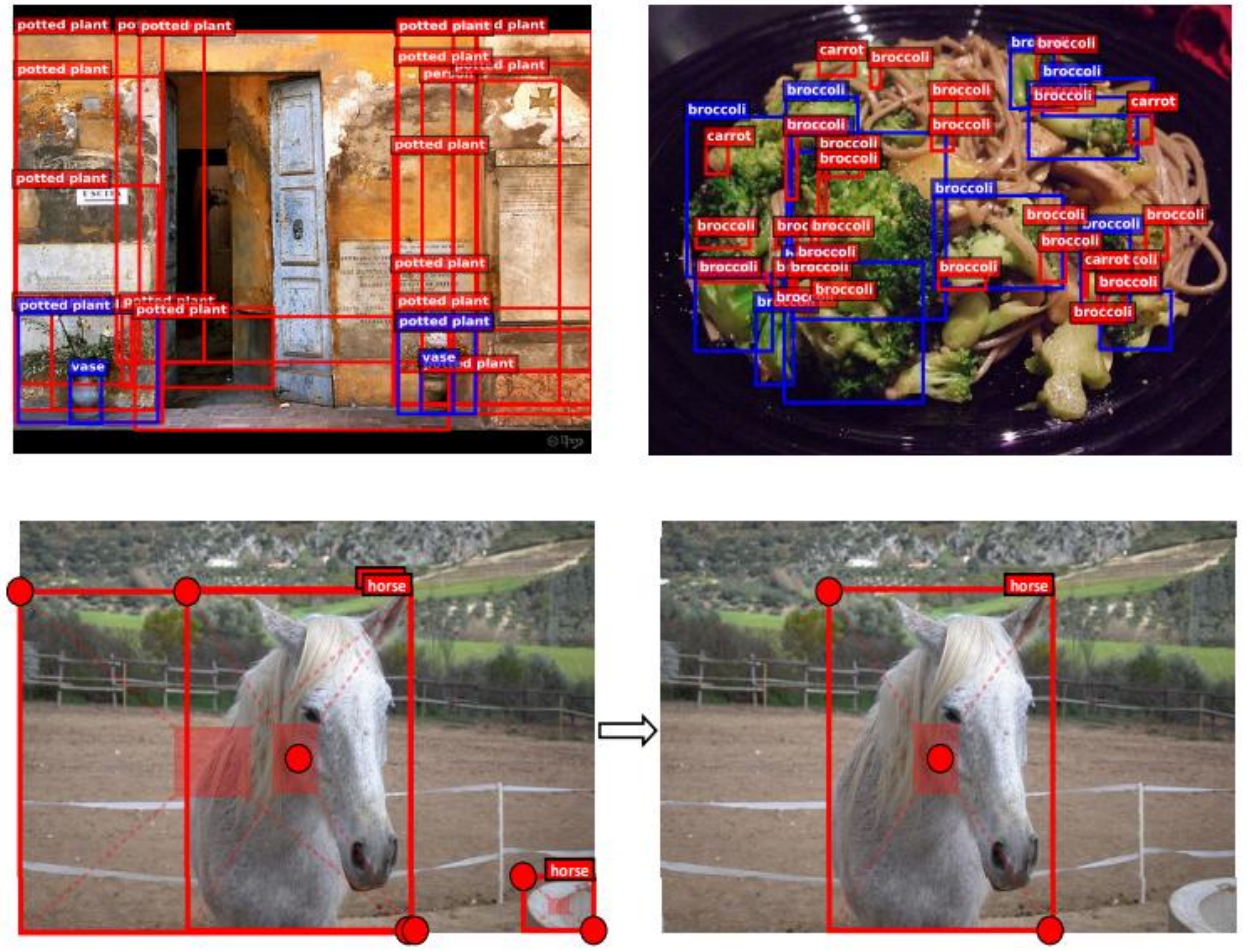


Figure 1: In the first row, we visualize the top 100 bounding boxes (according to the MS-COCO dataset standard) of CornerNet. Ground-truth and predicted objects are marked in blue and red, respectively. In the second row, we show that correct predictions can be determined by checking the central parts.

FSAF: Feature Selective Anchor-Free Module for Single-Shot Object Detection

- Based on RetinaNet

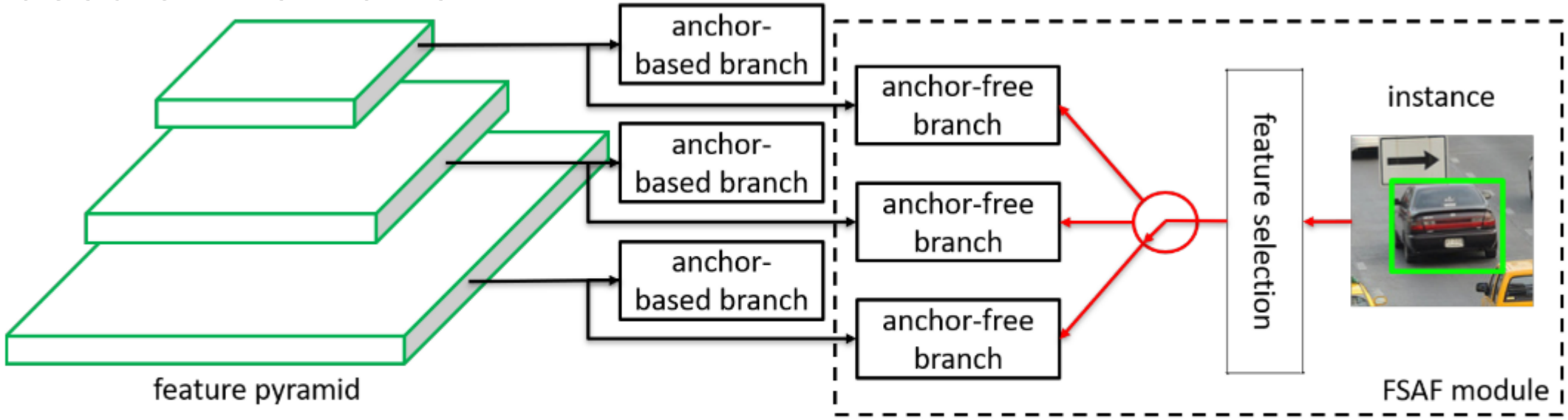


Figure 3: Overview of our FSAF module plugged into conventional anchor-based detection methods. During training, each instance is assigned to a pyramid level via feature selection for setting up supervision signals.

讓每個實例選擇最好的特徵層來優化網絡，因此不需要 anchor 來限制特徵的選擇

一個 anchor-free 的分支在每個特徵金字塔層中構建，獨立於 anchor-based 的分支。和 anchor-based 分支相似，anchor-free 分支由分類子網路和迴歸子網路。一個實例能夠被安排到任意層的 anchor-free 分支。訓練期間，基於實例的資訊而不是實例 box 的尺寸來動態地為每個實例選擇最合適的特徵層。所選的特徵層學會偵測安排的實例。推理階段，FSAF 模組和 anchor-based 分支獨立或聯合運作。

https://arxiv.org/abs/1903.00621

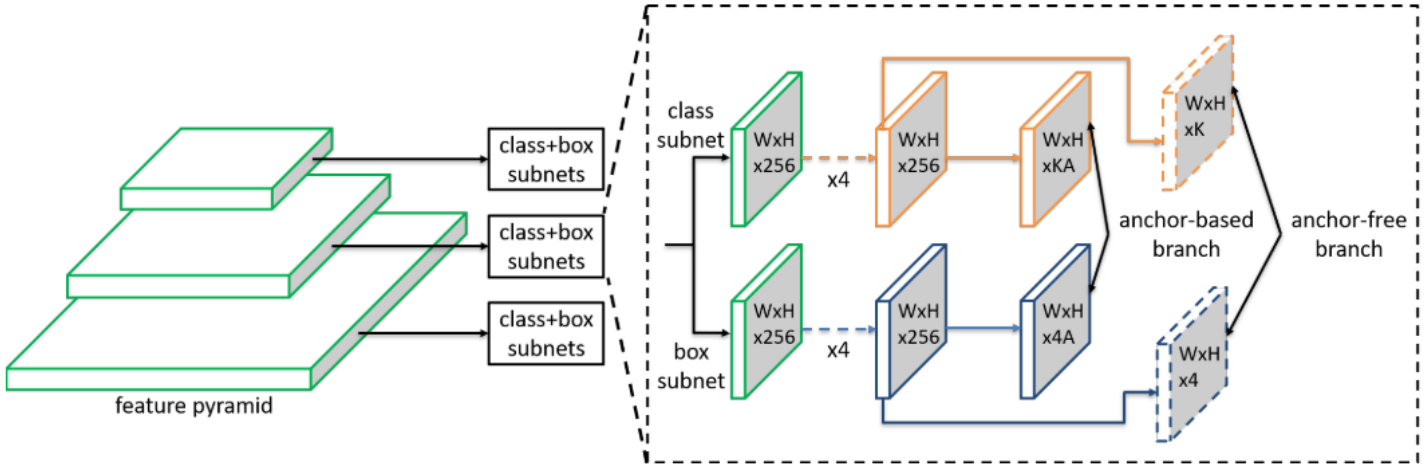


Figure 4: Network architecture of RetinaNet with our FSAF module. The FSAF module only introduces two additional conv layers (dashed feature maps) per pyramid level, keeping the architecture fully convolutional.

在 RetinaNet 的基礎上，FSAF 模組引入了 2 個額外的捲積層，這兩個卷積層各自負責 anchor-free 分支的分類和回歸預測。

具體的，在分類子網路中，feature map 後面跟著 K 個 3x3 的捲積層和 sigmoid，在回歸子網路中，feature map 後面跟著 4 個 3x3 的捲積層和 ReLU。

白色為有效區域，灰色為忽略區域，黑色為負樣本區域。

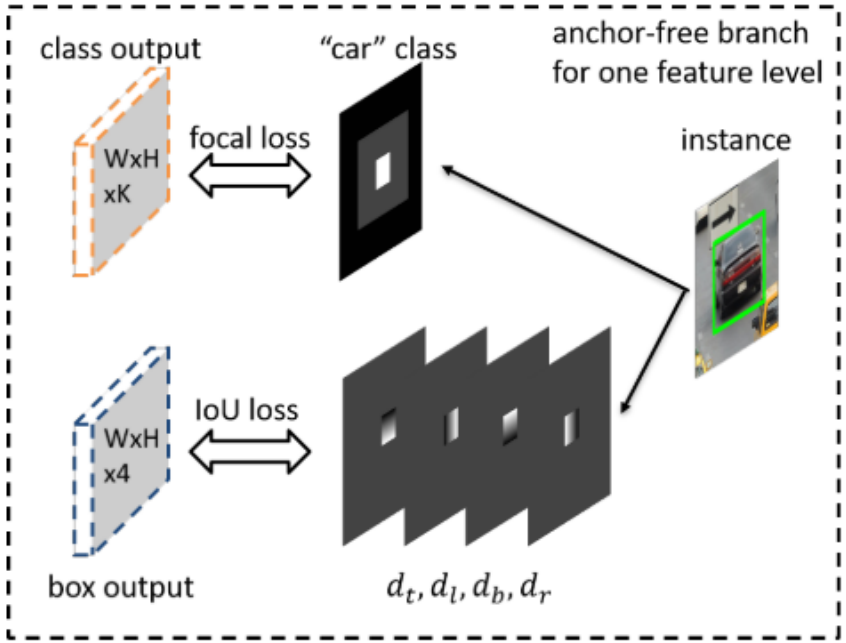


Figure 5: Supervision signals for an instance in one feature level of the anchor-free branches. We use focal loss for classification and IoU loss for box regression.

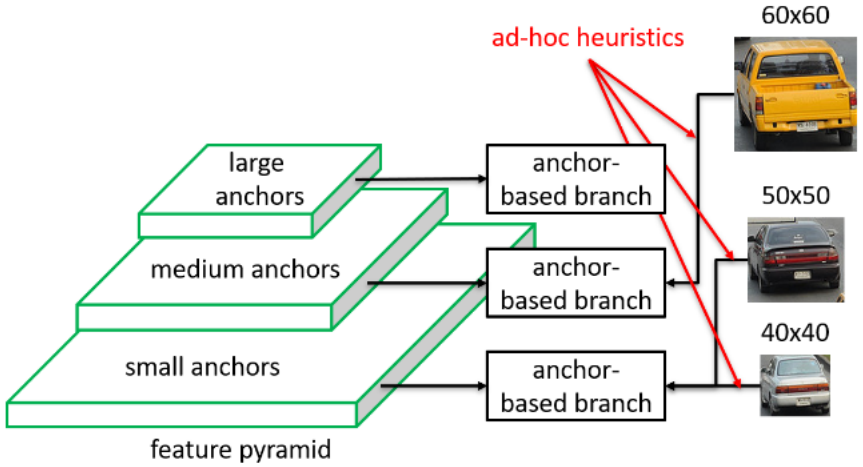


Figure 2: Selected feature level in anchor-based branches may not be optimal.

Online selection

實例輸入到特徵金字塔的所有圖層，然後得到所有 anchor-free 分支 focal loss 和 IoU loss 的和，選擇 loss 和最小的特徵圖層來學習實例。

訓練時，特徵會根據安排的實例進行更新。推理時，不需要進行特徵更新，因為最合適的特徵金字塔層自然地輸出高置信分數。

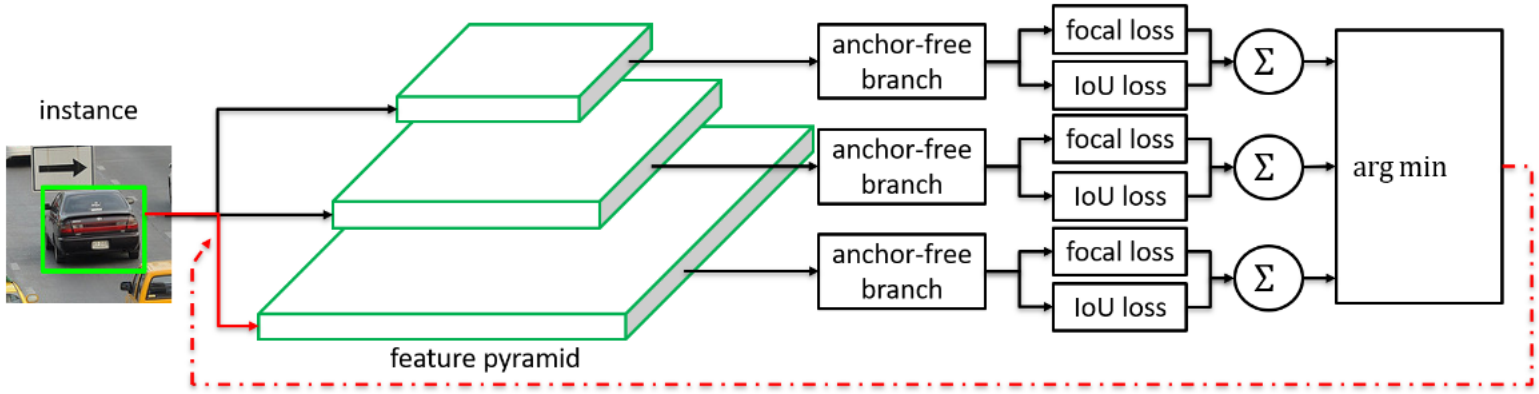


Figure 6: Online feature selection mechanism. Each instance is passing through all levels of anchor-free branches to compute the averaged classification (focal) loss and regression (IoU) loss over effective regions. Then the level with minimal summation of two losses is selected to set up the supervision signals for that instance.

a: RetinaNet (anchor-based, ResNeXt-101) b: Ours (anchor-based + FSAF, ResNet-50)

Figure 1: Qualitative results of the anchor-based RetinaNet [22] using powerful *ResNeXt-101* (left) and our detector with additional FSAF module using just *ResNet-50* (right) under the same training and testing scale. Our FSAF module helps detecting hard objects like tiny person and flat skis with a less powerful backbone network. See Figure 7 for more examples.

FCOS: Fully Convolutional One-Stage Object Detection

和語意分割相同，偵測器直接將位置作為訓練樣本而不是 anchor。具體的，如果某個位置落入了任何 gt 中，那麼該位置就被認為是正樣本，並且類別為該 gt 的類別。基於 anchor 的偵測器，根據不同尺寸安排 anchor 到不同的特徵層，而 FCOS 直接限制邊界框回歸的範圍(即每個 feature map 負責一定尺度的回歸框)。

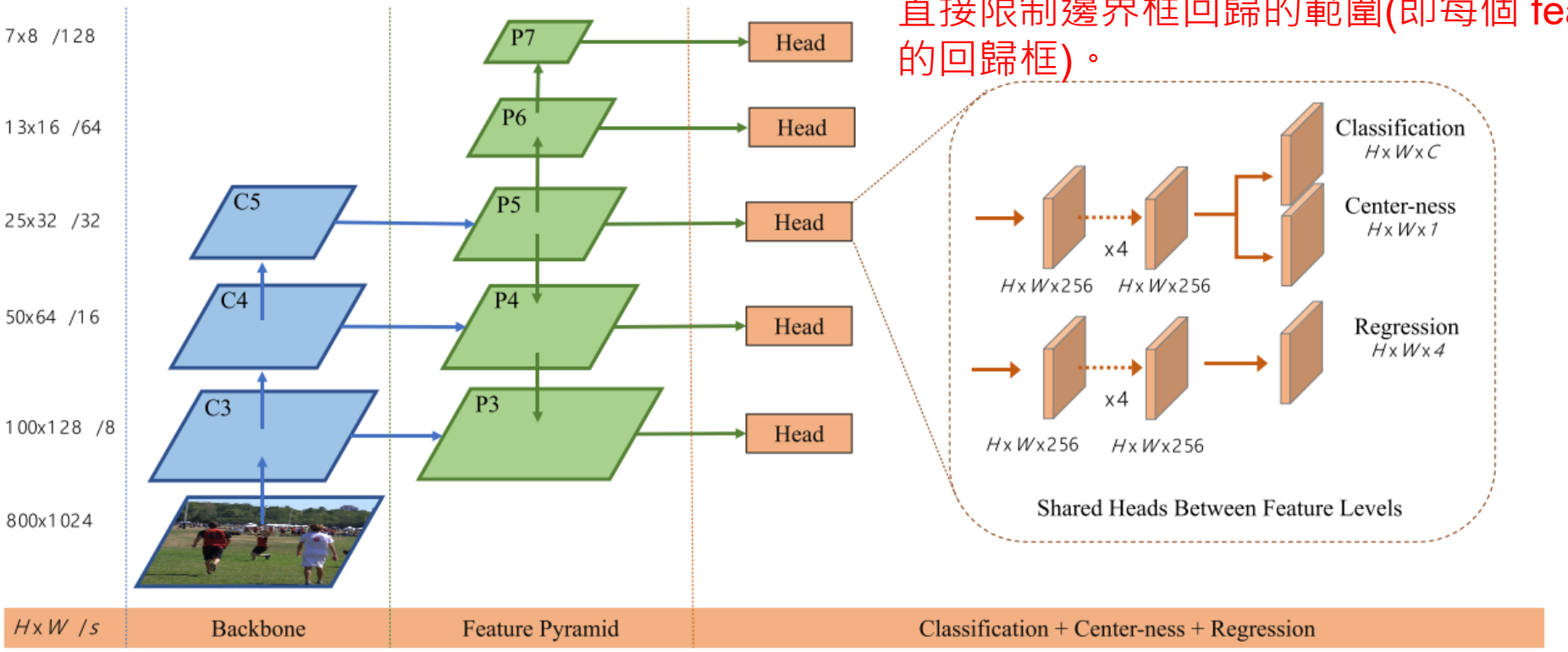


Figure 2 – The network architecture of FCOS, where C3, C4, and C5 denote the feature maps of the backbone network and P3 to P7 are the feature levels used for the final prediction. $H \times W$ is the height and width of feature maps. '/s' ($s = 8, 16, ..., 128$) is the down-sampling ratio of the level of feature maps to the input image. As an example, all the numbers are computed with an 800×1024 input.

https://arxiv.org/abs/1904.01355

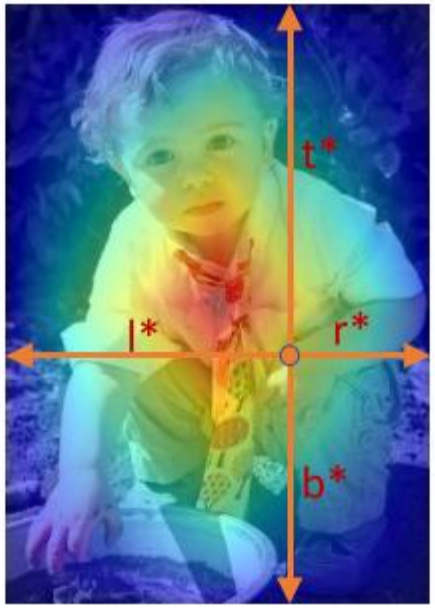


Figure 3 – Center-ness. Red, blue, and other colors denote 1, 0 and the values between them, respectively. Center-ness is computed by Eq. (3) and decays from 1 to 0 as the location deviates from the center of the object. When testing, the center-ness predicted by the network is multiplied with the classification score thus can down-weight the low-quality bounding boxes predicted by a location far from the center of an object.

$$\text{centerness}^* = \sqrt{\frac{\min(l^*, r^*)}{\max(l^*, r^*)} \times \frac{\min(t^*, b^*)}{\max(t^*, b^*)}}$$

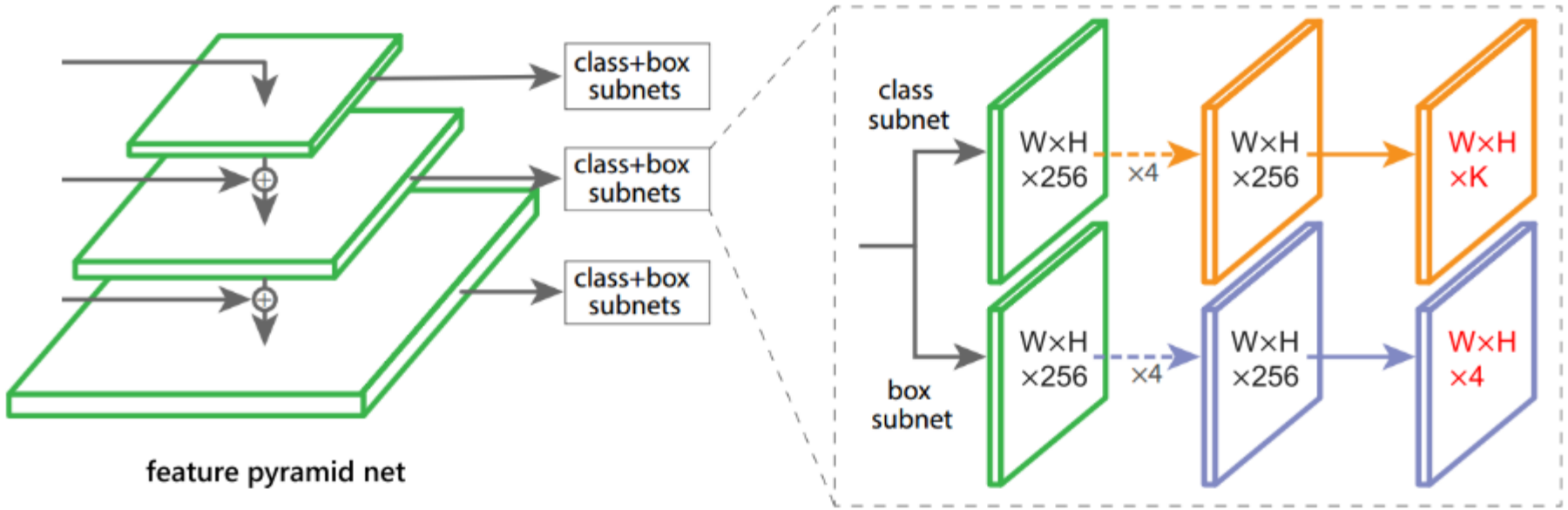
中心性

為了剔除遠離目標中心的低品質預測 bbox，作者提出了添加 center-ness 分支，和分類分支並行。

開根號使 center-ness 衰退緩慢。 center-ness 範圍為 0-1 之間，通過 BCE 訓練。

測試時，最終分數由 center-ness 預測結果和分類分數乘積得到。

FoveaBox



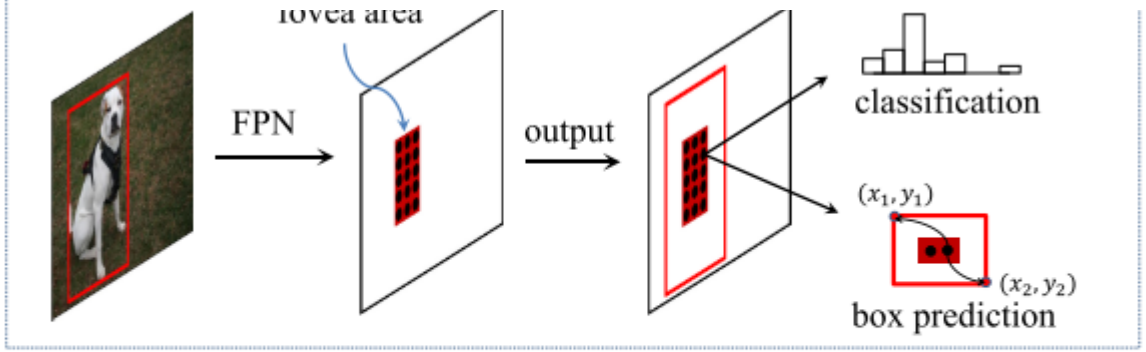


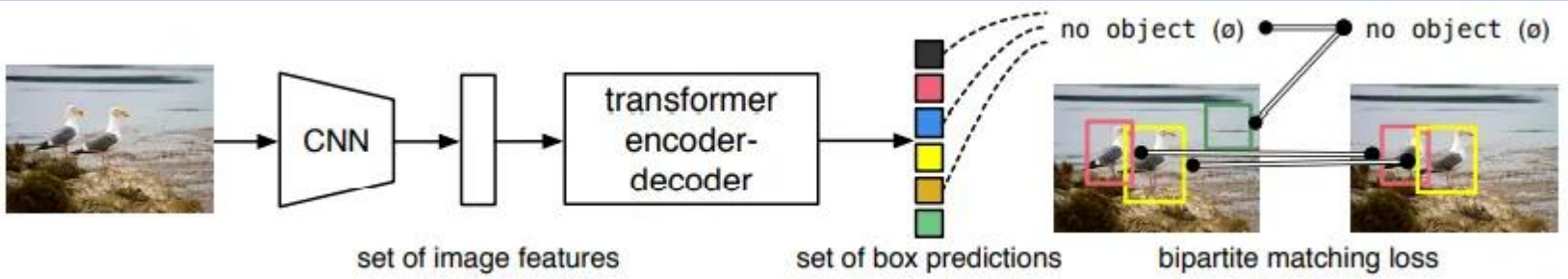
Figure 3. **FoveaBox object detector**. For each output spacial position that potentially presents an object, FoveaBox directly predicts the confidences for all target categories and the bounding box.

目標的中央凹如上圖所示。目標中央凹只編碼目標物件存在的機率。為了確定位置，模型要預測每個潛在實例的邊界框。

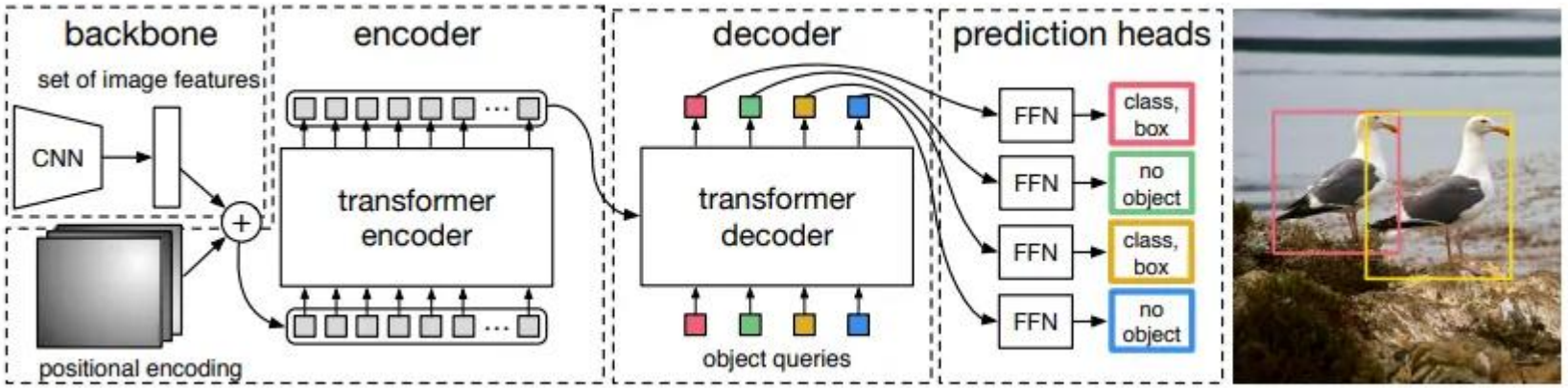
https://arxiv.org/abs/1904.03797

TRANSFORMER BASED OD

DETR



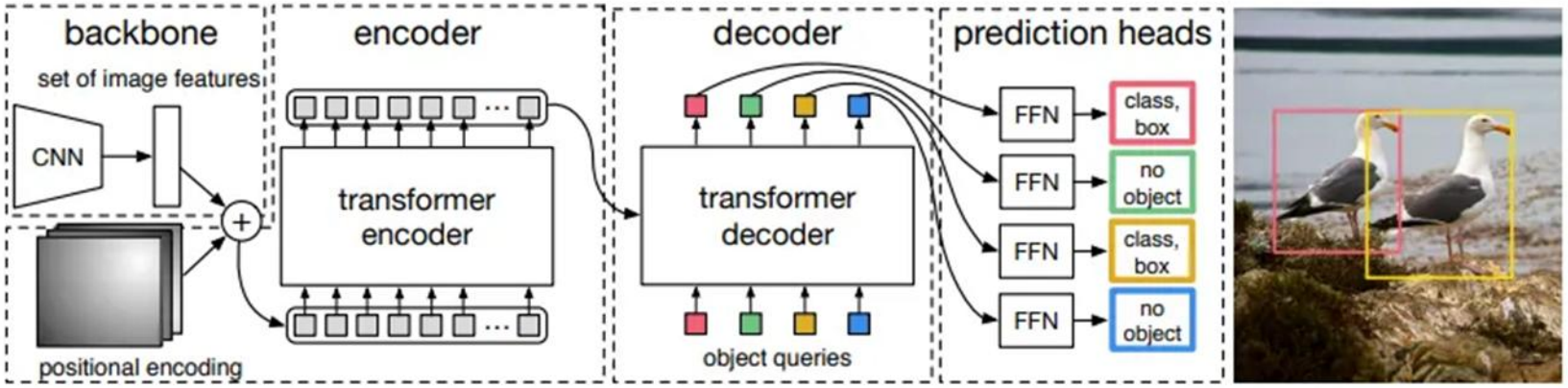
- NMS
  - （包括 FCOS）會產生大量冗餘的預測BB，靠NMS 處理
  - NMS:啟發式演算法，它不是端到端可學習的，並且有自身的超參數和問題
- 引入了 Transformer 架構 ，
  - 將物件偵測重新定義為一個「端到端的集合預測 set prediction」問題。
    - 創新處在於給予object detection一個更簡單乾淨的pipeline：在網路上不需要proposal、anchor與window center；在後處理上不需要non maximum suppression。這個新的pipeline得利於DETR將object detection視為set prediction問題，並且在訓練時要求predict set與ground truth set間的bipartite matching。



https://arxiv

DETR 的解決方案

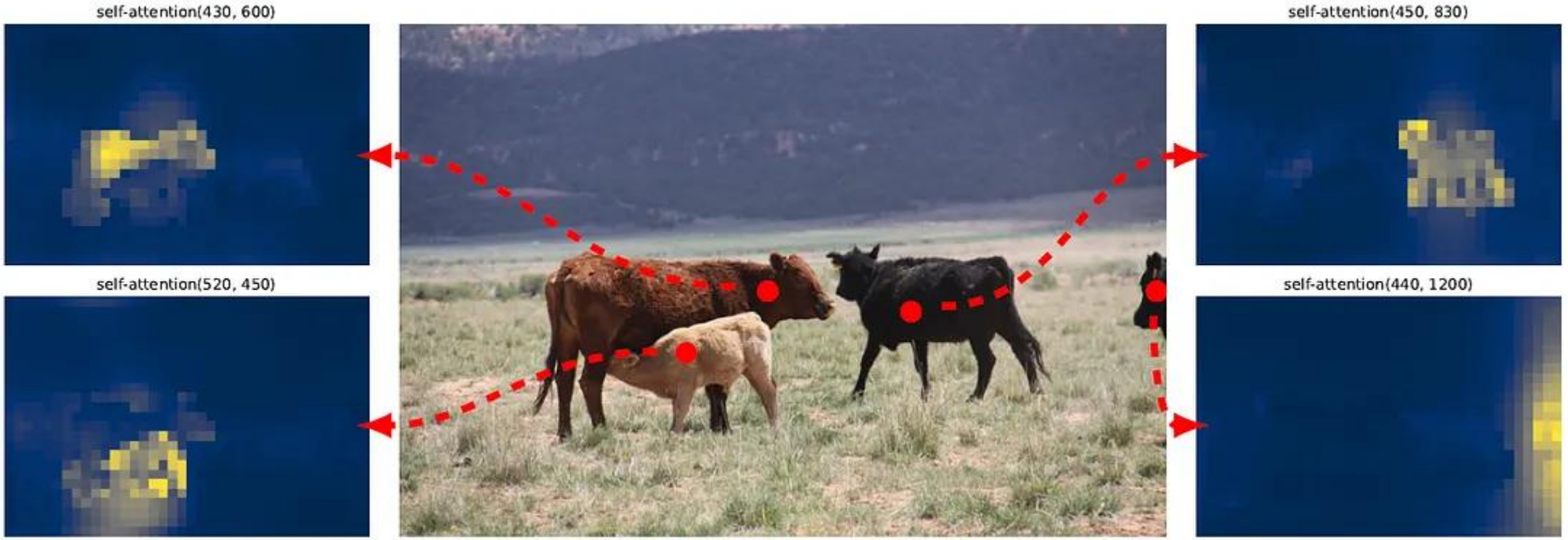
- CNN + Transformer:
  - DETR 首先使用 CNN 骨幹網路提取圖像特徵。然後，將這些特徵與一組固定的、可學習的 "Object Queries"（物體查詢，例如 N=100 個）一起輸入到 Transformer 的 Encoder-Decoder 架構中 。
- Set Prediction (集合預測):
  - Transformer 的 Decoder 並行輸出一個固定大小的預測集合（N 個預測）。
- Bipartite Matching (二分圖匹配):
  - 為了訓練模型，DETR 使用「匈牙利演算法」(Hungarian Algorithm) 來執行一次性的二分圖匹配，在 N 個預測和 M 個真實物體 (ground truth) 之間找到一個唯一的、成本最低的匹配 。
- Set-Based Global Loss:
  - 僅對匹配上的預測計算損失（分類損失 + Bounding Box 損失）。



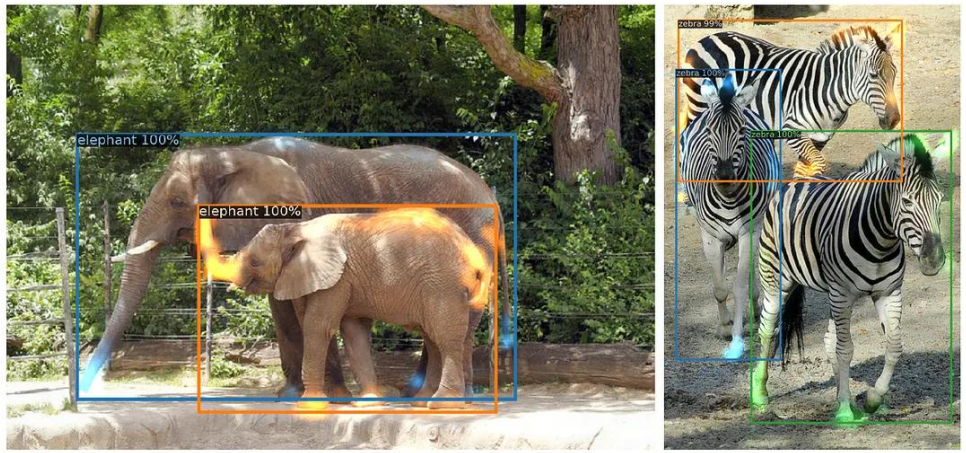
object queries可以視為某種非幾何上的deep anchor，在training的過程中自動學得。事實上，Object queries並不是直接作為decoder的輸入，而是透過learnable positional encoding影響Transformer的decoder。

Model	GFLOPS/FPS	#params	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Faster RCNN-DC5	320/16	166M	39.0	60.5	42.3	21.4	43.5	52.5
Faster RCNN-FPN	180/26	42M	40.2	61.0	43.8	24.2	43.5	52.0
Faster RCNN-R101-FPN	246/20	60M	42.0	62.5	45.9	25.2	45.6	54.6
Faster RCNN-DC5+	320/16	166M	41.1	61.4	44.3	22.9	45.9	55.0
Faster RCNN-FPN+	180/26	42M	42.0	62.1	45.5	26.6	45.4	53.4
Faster RCNN-R101-FPN+	246/20	60M	44.0	63.9	**47.8**	**27.2**	48.1	56.0
DETR	86/28	41M	42.0	62.4	44.2	20.5	45.8	61.1
DETR-DC5	187/12	41M	43.3	63.1	45.9	22.5	47.3	61.1
DETR-R101	152/20	60M	43.5	63.8	46.4	21.9	48.0	61.8
DETR-DC5-R101	253/10	60M	**44.9**	**64.7**	47.7	23.7	**49.5**	**62.3**

DETR在大物件的偵測能力優於Faster RCNN，但在小物件上卻也有明顯的劣勢。



論文認為Transformer的encoder主司於將instance從image中分離，而decoder則關注於local資訊。除了透過視覺化encoder與decoder的attention map外，他們也發現拔除encoder會使大物件偵測能力下降。



Attention map of the decoder

DINO (DETR with Improved deNoising anchOr boxes)

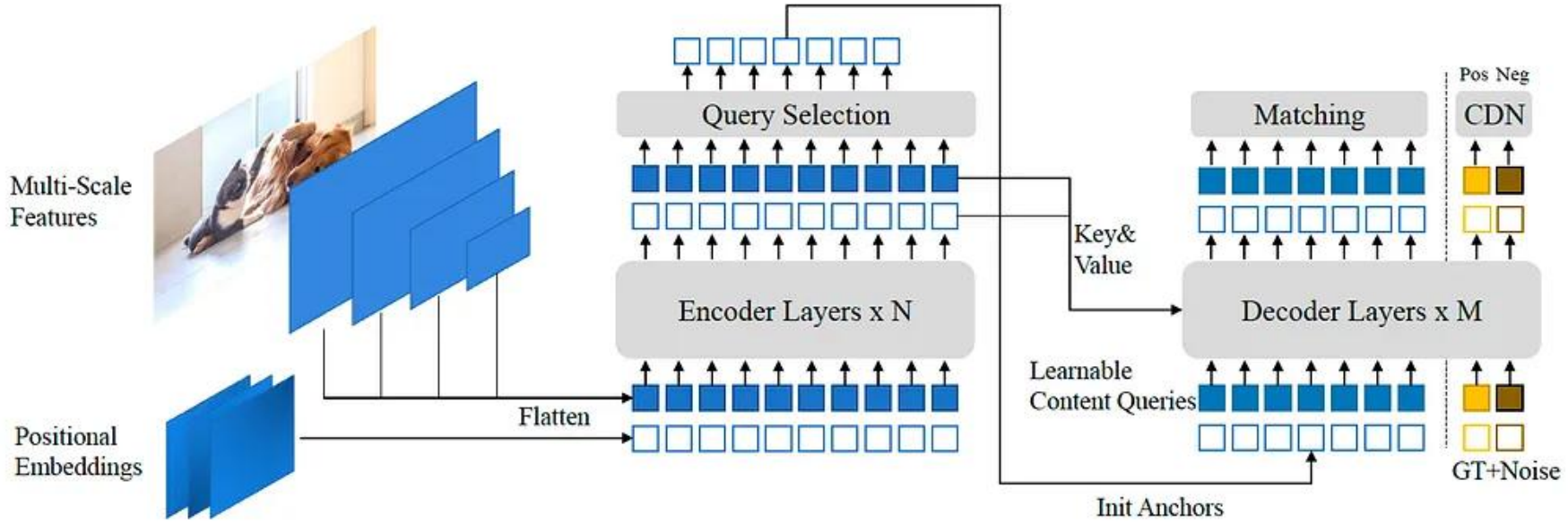


Fig. 2. The framework of our proposed DINO model. Our improvements are mainly in the Transformer encoder and decoder. The top-K encoder features in the last layer are selected to initialize the positional queries for the Transformer decoder, whereas the content queries are kept as learnable parameters. Our decoder also contains a Contrastive DeNoising (CDN) part with both positive and negative samples.

- Contrastive Denoising (CDN) 訓練:
  - 這是對 DN-DETR 的改進。訓練時，除了匹配真實物體外，還向解碼器 (Decoder) 餵入「加入噪聲的真實邊界框」。模型被要求「去噪」(denoising) 以重建原始邊界框 。這個「修正」任務比「從零猜測」更容易，因此極大加速收斂。
  - CDN 進一步引入了對比學習，不僅要求模型重建正樣本（加了小噪聲的框），還要求模型將負樣本（加了大噪聲的框）區分為「無物體」。

- Mixed Query Selection (混合查詢選擇):
  - 為了改善 Query 的初始化，DINO 結合了兩種方式：部分 Query 仍然是可學習的（用於捕捉全局上下文），部分 Query 則根據 Encoder 輸出的特徵圖來動態選擇（用於關注高可能性的物體位置）。這有助於更好地初始化 anchor boxes 。

- Look Forward Twice (前瞻兩次):
  - 一種優化策略，允許在更新 box 參數時，能參考來自後續層的梯度，從而更精確地優化邊界框回歸 。

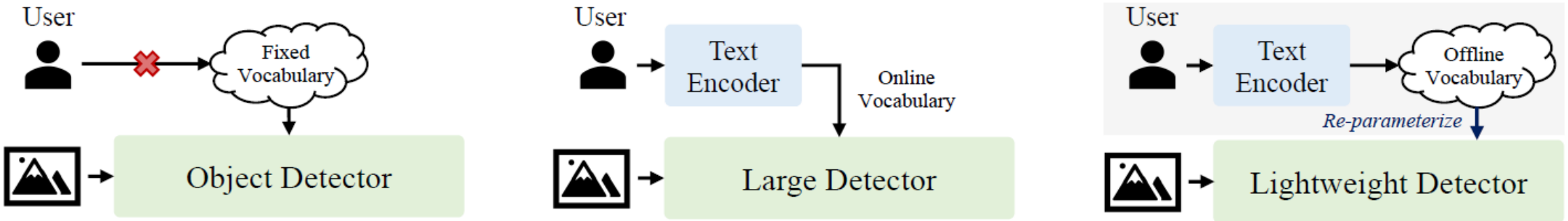
影響

- DETR: Single NN pipeline for OD
  - 透過消除 NMS ，DETR 將物件偵測從一個「CNN 特徵提取 + 啟發式後處理」的拼湊系統，轉變為一個「單一、可微分、端到端」的神經網路。
  - 開啟了「統一視覺任務」的可能性
    - DETR 和後來的 Mask DINO 可以非常容易地擴展到實例分割 (Instance Segmentation) 和全景分割 (Panoptic Segmentation)，只需在 Transformer Decoder 上增加一個 mask 預測頭即可
- Dino:「混合式設計」，即「顯式先驗」與「端到端學習」的結合
  - 原始 DETR 試圖用一種「純粹」的端到端學習（從隨機 queries 開始）來解決問題，結果導致收斂極慢 。
  - DINO 的核心創新——Denoising (去噪) ——本質上是*重新引入*了「錨框先驗」(anchor prior)，只不過是以一種更「軟」的、可學習的方式（即「加噪聲的真實框」）。這為解碼器提供了一個更容易的學習目標（從「猜測」變為「修正」），從而極大地加速了收斂 。

從CLIP到YOLO-WORLD, YOLOE：統一圖像空間和文字空間的目標檢測問題

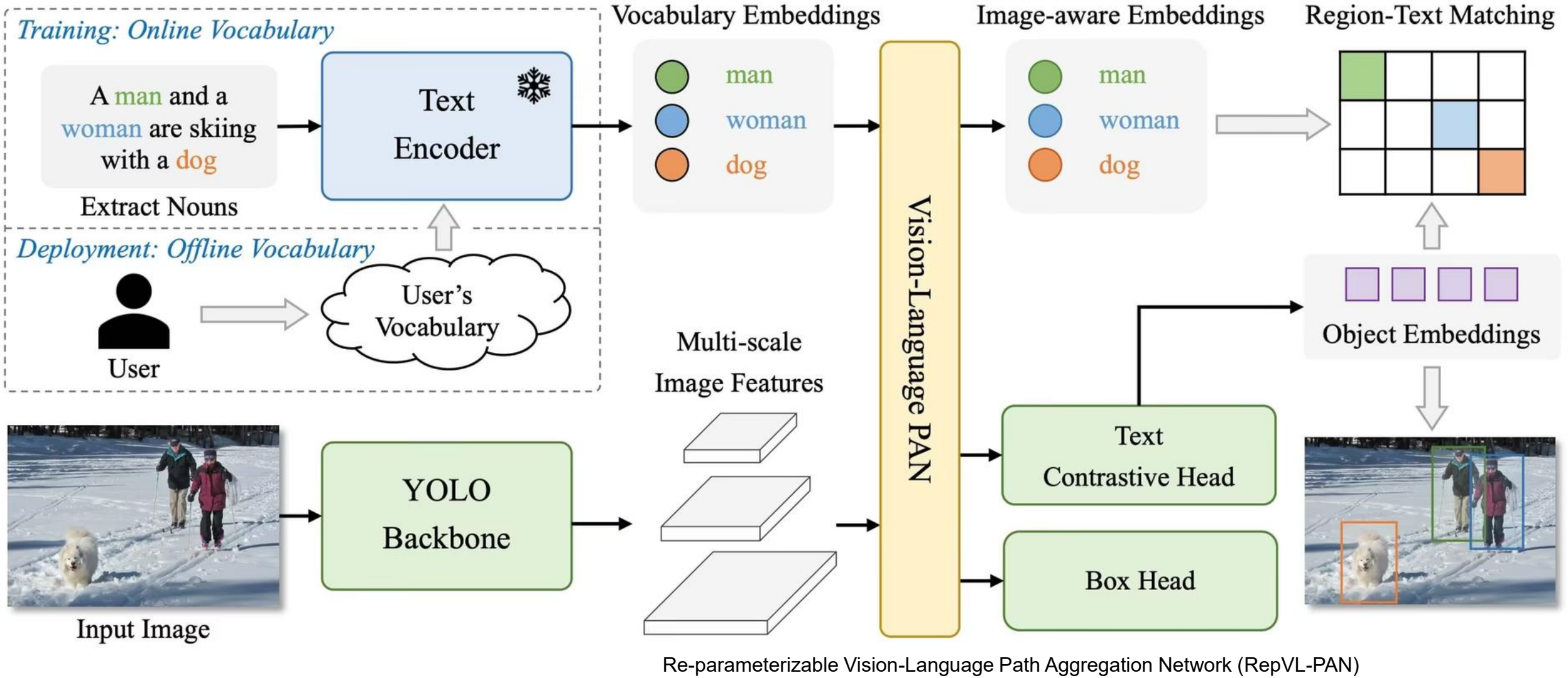
OD DURING THE FOUNDATION MODEL AGE

YOLO-World



(a) Traditional Object Detector (b) Preivous Open-Vocabulary Detector (c) YOLO-World

Figure 2. **Comparison with Detection Paradigms. (a) Traditional Object Detector**: These object detectors can only detect objects within the fixed vocabulary pre-defined by the training datasets, *e.g.*, 80 categories of COCO dataset [26]. The fixed vocabulary limits the extension for open scenes. **(b) Previous Open-Vocabulary Detectors:** Previous methods tend to develop large and heavy detectors for open-vocabulary detection which intuitively have strong capacity. In addition, these detectors simultaneously encode images and texts as input for prediction, which is time-consuming for practical applications. **(c) YOLO-World:** We demonstrate the strong open-vocabulary performance of lightweight detectors, *e.g.*, YOLO detectors [20, 42], which is of great significance for real-world applications. Rather than using online vocabulary, we present a *prompt-then-detect* paradigm for efficient inference, in which the user generates a series of prompts according to the need and the prompts will be encoded into an offline vocabulary. Then it can be re-parameterized as the model weights for deployment and further acceleration.



Re-parameterizable Vision-Language Path Aggregation Network (RepVL-PAN)

Figure 3. **Overall Architecture of YOLO-World.** Compared to traditional YOLO detectors, YOLO-World as an open-vocabulary detector adopts text as input. The *Text Encoder* first encodes the input text input text embeddings. Then the *Image Encoder* encodes the input image into multi-scale image features and the proposed *RepVL-PAN* exploits the multi-level cross-modality fusion for both image and text features. Finally, YOLO-World predicts the regressed bounding boxes and the object embeddings for matching the categories or nouns that appeared in the input text.

https://arxiv.org/abs/2401.17270

YOLO-world：利用文字prompt進行目標檢測

- Pretraining
  - large-scale detection，grounding和 image-text
- Model
  - 目前的大部分開放詞彙檢測器主要是在分類頭上做功夫：
    - 將最後那個固定的全連接頭換掉。之前的全連接頭會給出一個概率預測值：有多少個類別就有多少個對應的概率預測。
    - 現在則是讓圖像（或者更準確地說是檢測框之內的ROI）進行embedding得到vector，所有的text prompt也會通過CLIP得到對應的embedding vector，我們通過計算相似度爲每個圖像**vector匹配**最適合的text prompt，從而進行分類。
      - 注意，上述過程中，text encoder是凍結住不進行訓練的，因爲它已經在數百萬對caption-image上進行了預訓練。
  - 檢測頭對檢測框的回歸預測過程基本不變。
  - 對於圖像部分，
    - yolo-world使用yolov8的backbone對輸入圖像進行多尺度特徵圖提取，得到multi-scale image feature；
  - text prompts 則會進行embedding處理。
    - multi-scale Image feature和text embeding會一起送入vision-language PAN進行交互，得到帶有圖像感知的text embedding (image-aware embeding)；經過處理的特徵圖會分別送入box head 和text contrastive head。前者會進行檢測框預測，後者會得到image 的embedding。image embedding 會和text embeding計算相似度從而預測類別。

Method	Backbone	Params	Pre-trained Data	FPS	AP	AP_r	AP_c	AP_f
MDETR [21]	R-101 [15]	169M	GoldG	-	24.2	20.9	24.3	24.2
GLIP-T [24]	Swin-T [32]	232M	O365,GoldG	0.12	24.9	17.7	19.5	31.0
GLIP-T [24]	Swin-T [32]	232M	O365,GoldG,Cap4M	0.12	26.0	20.8	21.4	31.0
GLIPv2-T [59]	Swin-T [32]	232M	O365,GoldG	0.12	26.9	-	-	-
GLIPv2-T [59]	Swin-T [32]	232M	O365,GoldG,Cap4M	0.12	29.0	-	-	-
Grounding DINO-T [30]	Swin-T [32]	172M	O365,GoldG	1.5	25.6	14.4	19.6	32.2
Grounding DINO-T [30]	Swin-T [32]	172M	O365,GoldG,Cap4M	1.5	27.4	18.1	23.3	32.7
DetCLIP-T [56]	Swin-T [32]	155M	O365,GoldG	2.3	34.4	26.9	33.9	36.3
YOLO-World-S	YOLOv8-S	13M (77M)	O365,GoldG	74.1 (19.9)	26.2	19.1	23.6	29.8
YOLO-World-M	YOLOv8-M	29M (92M)	O365,GoldG	58.1 (18.5)	31.0	23.8	29.2	33.9
YOLO-World-L	YOLOv8-L	48M (110M)	O365,GoldG	52.0 (17.6)	35.0	27.1	32.8	38.3
YOLO-World-L	YOLOv8-L	48M (110M)	O365,GoldG,CC3M†	52.0 (17.6)	35.4	27.6	34.1	38.0

Table 2. **Zero-shot Evaluation on LVIS.** We evaluate YOLO-World on LVIS `minival` [21] in a zero-shot manner. We report the *Fixed AP* [4] for a fair comparison with recent methods. † denotes the pseudo-labeled CC3M in our setting, which contains 246k samples. The FPS is evaluated on one NVIDIA V100 GPU w/o TensorRT. The parameters and FPS of YOLO-World are evaluated for both the re-parameterized version (w/o bracket) and the original version (w/ bracket).

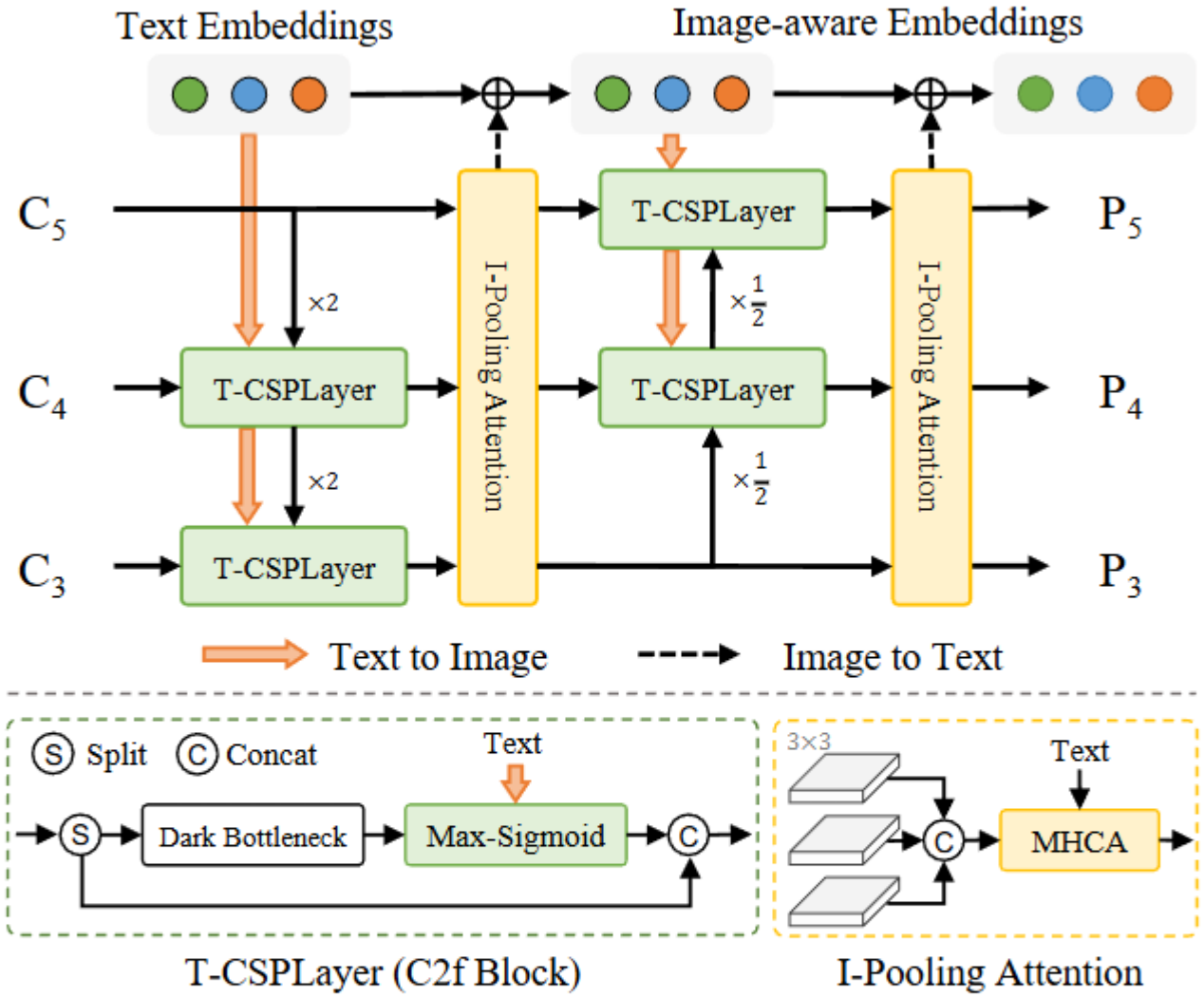


Figure 4. **Illustration of the RepVL-PAN.** The proposed RepVL-PAN adopts the *Text-guided CSPLayer* (T-CSPLayer) for injecting language information into image features and the *Image Pooling Attention* (I-Pooling Attention) for enhancing image-aware text embeddings.

A.1. Re-parameterization for RepVL-PAN

During inference on an offline vocabulary, we adopt re-parameterization for RepVL-PAN for faster inference speed and deployment. Firstly, we pre-compute the text embeddings $W \in \mathbb{R}^{C\times D}$ through the text encoder.

Re-parameterize T-CSPLayer. For each T-CSPLayer in RepVL-PAN, we can re-parameterize and simplify the process of adding text guidance by reshaping the text embeddings $W \in \mathbb{R}^{C\times D\times 1\times 1}$ into the weights of a 1×1 convolution layer (or a linear layer), as follows:

$$X' = X \odot \texttt{Sigmoid}(\texttt{max}(\texttt{Conv}(X, W), \texttt{dim=1})), \quad (4)$$

where $X\times \in \mathbb{R}^{B\times D\times H\times W}$ and $X' \in \mathbb{R}^{B\times D\times H\times W}$ are the input and output image features. $\odot$ is the matrix multiplication with `reshape` or `transpose`.

Re-parameterize I-Pooling Attention. The I-Pooling Attention can be re-parameterize or simplified by:

$$\tilde{X} = \texttt{cat}(\texttt{MP}(X_3, 3), \texttt{MP}(X_4, 3), \texttt{MP}(X_5, 3)), \quad (5)$$

where `cat` is the concentration and $\texttt{MP}(\cdot, 3)$ denotes the max pooling for 3×3 output features. $\{X_3, X_4, X_5\}$ are the multi-scale features in RepVL-PAN. $\tilde{X}$ is flattened and has the shape of $B \times D \times 27$. Then we can update the text embeddings by:

$$W' = W + \texttt{Softmax}(W \odot \tilde{X}), \texttt{dim=-1}) \odot W, \quad (6)$$