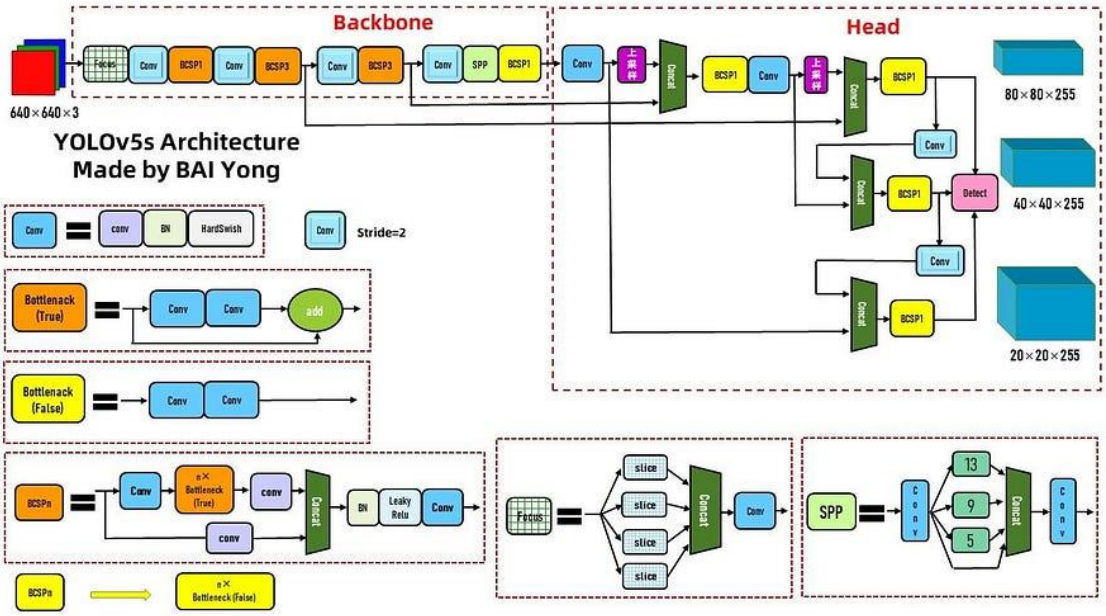
Computer Vision Practice with Deep Learning homework 1

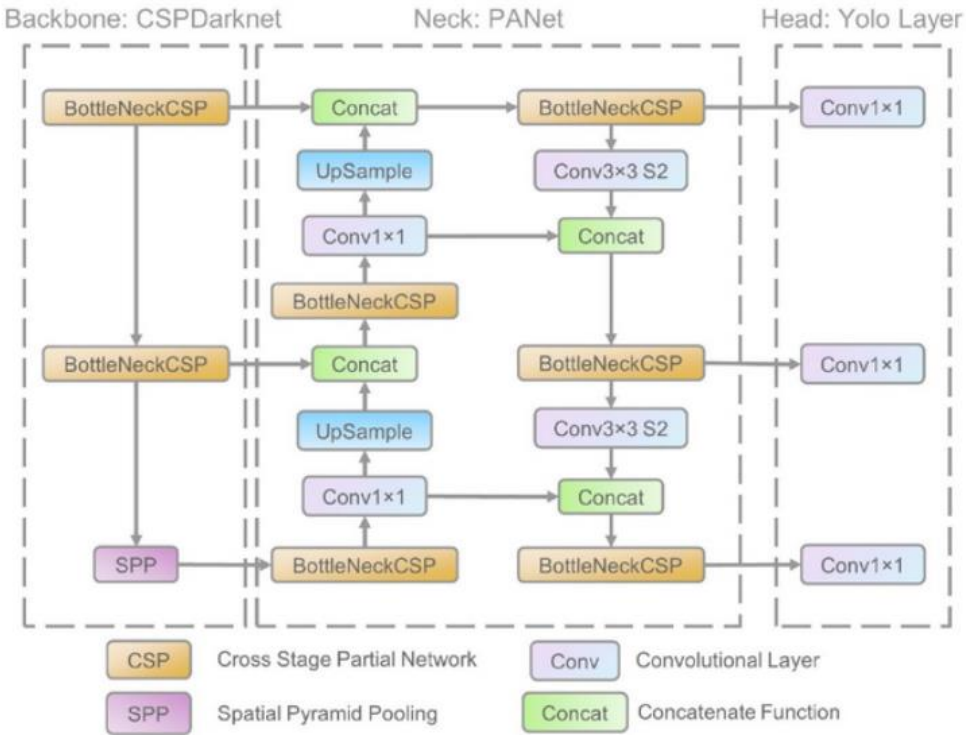
生醫電資所 R11945070 陳光唯

1. Draw the architectures for both CNN-based and Transformer-based methods:

我所選擇的 YOLOv5 (You Only Look Once)及 DETR (DEtection TRansformer) 模型皆是用於 object detection 的模型，以下我將簡單介紹他們模型架構設計上的差異

- **CNN-based: YOLOv5**





以上為 YOLOv5 模型的基本架構圖，YOLOv5 (You Only Look Once) 是屬於 single-stage 偵測的模型，直接使用 convolutional neural network (CNN) 去預測圖片中物件的 bounding boxes 以及 class probabilities 。其採用了自適應 anchor boxes 的計算方式以及 Scaled-YOLOv4 的 Backbone，並相對以前的版本進行了更多的改進，如模型的簡潔化、分離回歸、分類任務的細部設計等，使其能更快速且更有效率的執行。

YOLOv5 的架構可以分成幾個主要部分:

1. Backbone: 使用 YOLOv4 的 backbone 的修改版，此架構中含有 Focus 的結構，對圖片進行切片，並使用了一系列 convolutional layers 從不同 scales 和 resolutions 的輸入圖中提取特徵，這使的他能夠針對不同尺寸和長寬比的 object 進行檢測。

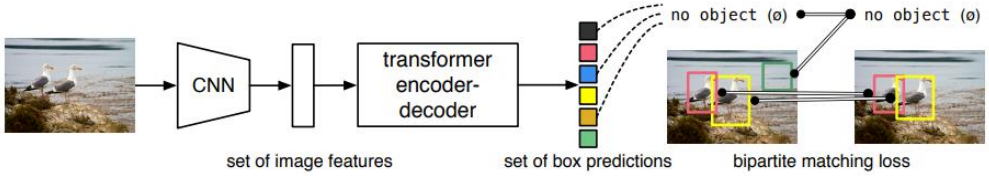
2. Neck: YOLOv5 不像其他 object detection 模型的 traditional neck，而是使用了一個 spatial pyramid pooling (SPP) module 來聚合來自不同 scales 和 resolutions 的特徵，更有效的增加特徵的提取，另外也含有 CSP-PAN 等結構。

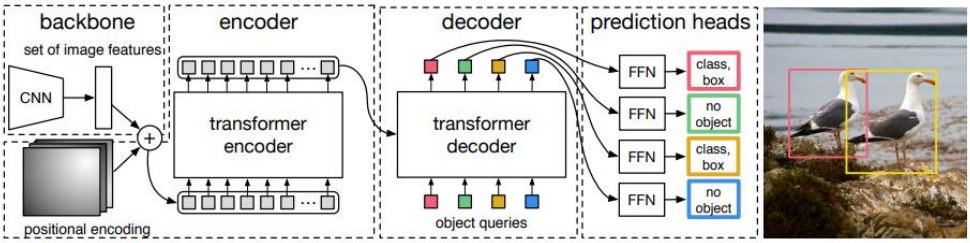
3. Head: 進行最後的檢測，在輸出的 feature map 上套上 anchor box，並預測每個目標屬於某個類別的概率、信心分數和 bounding boxes 的坐標，形成最終的輸出向量。與 YOLOv3 不同的是，YOLOv5 匹配 Anchor 不使用最大值 IOS 去判斷，而是使用 Shape 去匹配，也就是該 Ground Truth bounding boxes 和當前層的 Anchor 計算長寬比，如果長寬比例大於設定閾值，則說明該 Ground Truth bounding boxes 和 Anchor 匹配度不夠，將該 Ground Truth bounding boxes 過濾暫時丟掉，代表在該層預測中認為是背景。

4. Output: YOLOv5 的最終輸出是一組 bounding boxes 和相關的 class probabilities。Bounding box loss function 的部分採用 IoU Loss 的改編版: GIoU Loss，而在 object detection 的後處理過程中，則通過 nms 非極大值抑制進行處理。這些 bounding boxes 是使用自適應 anchor boxes 的方法組成的，這使得該模型能夠檢測不同尺寸和長寬比的目標物。

YOLOv5 在 YOLOv4 的基礎上添加了一些新的改進思路，設計主要著重於簡潔及效率，擁有著快速、準確和易於使用的特性。採用 Scaled-YOLOv4 backbone、SPP module 等結構，最佳化性能，同時又能比其他 object detection 模型的運行速度還要快，甚至用於即時的 object detection，又能透過不同參數設定可以得到不同複雜度的模型，世芳常簡潔強大的模型。

- **Transformer-based: DETR**





以上為 DETR 模型的基本架構圖，DETR (DEtection TRansformer)，它藉由 transformer-based 的架構來生成一組 object queries，然後與輸入圖像中的特徵互相匹配，生成最終的檢測結果。 DETR 還使用了一種稱為 Hungarian matching loss 的特殊 loss function 計算方式，使其能夠在不使用 anchor boxes 的情況下進行 object detection。DETR 當時也確實在好幾個 object detection benchmarks 上取得了最為優秀的結果。

DETR 的架構可以分成幾個主要部分:

1. CNN: 由一系列 convolutional layers 組成，首先透過傳統的 CNN 負責從圖中提取特徵，再將這個 feature 直接送到以 Transformer 組成的 auto-encoder 進行後續處理。

2. Transformer Encoder: 在 backbone 後，DETR 使用一個 transformer encoder 來處理從輸入圖中提取的 feature maps。Input 通過 Input Embedding，加上人為設置的 Positional Encoding，進入會重複 N 次的 block 中運算。通過 transformer encoder 平行處理由 backbone 提取的 spatial feature maps，這使得模型能夠捕捉到 global 的資訊。

3. Query Embedding: Transformer encoder 生出一組 learnable 的 query embedding，用於 attend feature maps。每個 query embedding 都代表了模型試圖檢測圖像中的其中一個可能的 object。

4. Multi-Head Attention: Query embeddings 被用來計算使用 multi-head attention 的 feature maps 的 attention maps。Multi-head attention 使模型能夠同時搜尋到 feature maps 的不同地方。

5. Decoder: 此部分同 Transformer 的標準架構，使用 Multi-head attention 轉換成大小為 d 的 N 個 Embeddings 。與原始的 Transformer 不同的地方在於 DETR 會在此層中平行解碼出 N 個物件，而原始的 Transformer 則是

一次僅預測出一個元素的輸出序列。通過處理被鎖定的 feature maps 以生成 class probabilities 和每個 query embedding 的 bounding box 坐標。

6. Hungarian Matching: 由於 DETR 預測的 box set 沒有順序的，為了使預測的輸出與 ground-truth objects 匹配，DETR 使用了一個特別的 loss function，稱為 Hungarian matching loss，使用 Hungarian algorithm 計算出預測對象和 ground-truth objects 之間的最佳結果。 Hungarian algorithm 可以在 polynomial time 下找到最一個 (prediction, ground truth) 組合 sigma，使得最終組合的 loss 最小。

DETR 的設計理念是使用 transformers 在 single stage 進行 object detection，不同於使用 anchor boxes 或 two-stage 的傳統 object detection 模型，因此不再需要做後續處理，使此模型更加靈活，因為它只需要一個 loss function 來進行優化。此外，使用 Hungarian matching loss 使 DETR 能夠比傳統法更有效、更準確的將預測的輸出與 ground-truth objects 做比較。DETR 也確實在好幾個 object detection benchmarks 上得到了最進步的性能，雖然因為使用 transformer 架構訓練收斂速度確實慢了不少。

簡單對兩者做一下比較的話: YOLOv5 是 one-stage 偵測的模型，使用 CNN 架構和自適應 anchor boxes 的方法直接預測 bounding boxes 以及 class probabilities，而 DETR 則是使用 transformer-based 的架構和新的 loss function 計算方式來匹配 object queries 與圖像的特徵。

2. Report and compare the performance of two methods on validation set: (include mAP[50:5:95], mAP50, mAP75) – on 127 validation data

- **CNN-based: YOLOv5: yolov5x/batch size=12 / epochs=100**
  Pre-trained on the COCO (Common Objects in Context) dataset，此資料集包含超過 330000 張 labeled images，有 80 個不同的 object 類別

YOLOv5 原生 mAP 列表:

Class	Precision	Recall	mAP50	mAP75	mAP[50:5:95]
All	0.844	0.758	0.803	0.537	0.515
Fish	0.859	0.771	0.858	0.585	0.539
Jellyfish	0.892	0.897	0.946	0.62	0.581
Penguin	0.69	0.817	0.736	0.324	0.359
Puffin	0.757	0.588	0.623	0.319	0.353
Shark	0.893	0.719	0.815	0.624	0.565
Starfish	1	0.666	0.811	0.566	0.616
Stingray	0.818	0.848	0.836	0.723	0.592

Result of check_your_prediction_valid.py:

Class	mAP_small	mAP_medium	mAP_large
All	0.1093	0.3693	0.5851
	mAP50	mAP75	mAP[50:5:95]
	0.7454	0.4857	0.4729

- **Transformer-based: DETR: DETR-R50/batch size=2 / epochs=150**
  Pre-trained on the COCO dataset，但 DETR 沒有直接使用 labeled images，而是使用了一種叫做 "pre-training with instance segmentation"的技術，這種方法首先將 COCO 資料集分割成 individual objects，訓練模型預測 object bounding boxes 及 categories 以外的 object masks。這種方式的目的是為了提高模型在復雜情境中準確定位和分類 object 的能力

DETR 原生 mAP 列表:

	IoU	Area	maxDets	Score
AP	0.50:0.95	All	100	0.332
AP	0.50	All	100	0.672
AP	0.75	All	100	0.274
AP	0.50:0.95	Small	100	0.094
AP	0.50:0.95	Medium	100	0.247
AP	0.50:0.95	Large	100	0.472
AR	0.50:0.95	All	1	0.182
AR	0.50:0.95	All	10	0.387
AR	0.50:0.95	All	100	0.484
AR	0.50:0.95	Small	100	0.156
AR	0.50:0.95	Medium	100	0.394
AR	0.50:0.95	Large	100	0.61

註: Average Precision = AP / Average Recall = AR

Result of check_your_prediction_valid.py:

Class	mAP_small	mAP_medium	mAP_large
All	0.0661	0.2277	0.4543
	mAP50	mAP75	mAP[50:5:95]
	0.6409	0.2773	0.3252

從上圖可以看出 YOLOv5 比起 DETR 的表現在不同物件上都比較優秀 (mAP50 較高)，而細部分類也有如 Puffin 分數較低，可能是因為其長相比較怪異少見，容易與其他鳥類搞混所致 (如 Penguin)。

DETR 的部分可以從他的 paper 中得知，或許要使他收斂需要更強大的運算資源比較有可能成功，其中 batch size=2 的設定 (在 1080 ti 上做訓練，GRAM 已滿)，大幅度地影響了他的結果表現，而訓練的 epoch 或許也需要往上加才能得到更好的成果，不過一個有趣的現象是 DETR 在大型物件的偵測比起小型物件

優秀不少，可能跟他預訓練的 dataset，或是有限的 transformer query 有關，使其能在較大的物件偵測上有較不錯的表現。

基本上使用預訓練權重進行 fine-tune 已經成為深度學習這領域的標準做法，因為它可以大大減少使用在新任務所需的訓練數據量，以及加快訓練過程的收斂。預先訓練的權重通常是從一個龐大而多樣的數據集中學習的，並包含一組學習過的特徵，這些特徵通常可以轉移到其他類似的任務中，而我在使用 DETR 模型的時候有試過使用沒有預訓練的模型架構進行 batch size=2、epochs=150 的訓練，但結果的 mAP50 基本上根本沒有上升，所以只能改用事先經過預訓練的 DETR-R50 來進行 fine-tune，而結果也確實得到了較快速的收斂成果。

使用 validation set 來進行驗證，是為了評估模型在未見過的數據上的表現，這對於確定模型的泛化能力至關重要。它確保模型能夠很好地泛化到新的數據，而不是簡單地記憶訓練數據。以下是我認為 YOLOv5 模型在驗證過程中取得比 DETR 模型更好的表現（更高的 mAP50），可能的原因：

1. 架構: YOLOv5 是一個 anchor-based 的 object detection 架構，而 DETR 是 transformer-based 架構，不使用 anchor。 YOLOv5 的架構早已經被證明在 object detection 的領域中具有極高的效率和效果，也已經經過較多代的修正，逐漸達到穩定的模式 (目前最新已經出到 YOLOv8 了)，而且相對 DETR 擁有比較大的侷限性，可以有效防止 overfitting 的發生，而 DETR 是相對而言屬於較新的架構 (從 NLP 移到影像上來用)，雖然具有不錯的效果，但在較小的 dataset 上或是有限的 transformer query 設定下，或許還有研究及修改的空間。

2. 預訓練的權重: YOLOv5 以及 DETR 皆使用來自 COCO 的預訓練權重，而我又使用相同的資料進行後續訓練，因此這部分的影響可能不大。

3. Hyperparameter: batch size、 learning rate 和 epochs 數的選擇會影響模型的性能。 我在 YOLOv5 使用 batch size =12，訓練 100 個 epochs，而 DETR 使用 batch size =2，訓練 150 個 epochs。 YOLOv5 較大的 batch size 和較短的訓練時間可能使其獲得了更好的泛化能力，而 DETR 本身的收斂速度因為是 transformer-based 的方法，本來就比較慢，而我又因為硬體問題只能使用 batch size =2 進行訓練，因此可能收斂的成果較差。

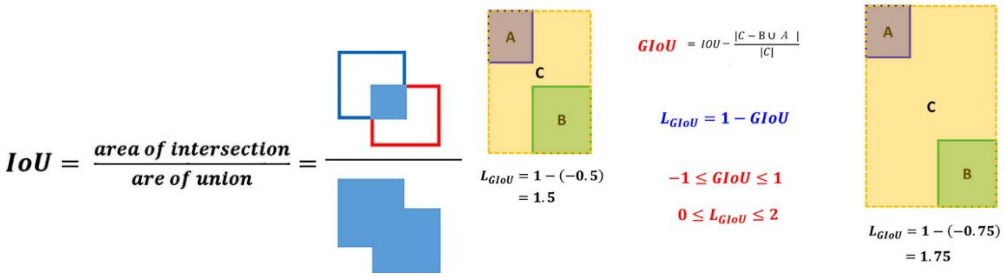
4. Datasets: 模型的性能也可能取決於用於訓練和驗證的特定數據集。可能由於本次使用的 datasets 較少，CNN 模型對於此情況的泛化能力較好，而 transformer 需要達到預期的訓練成果通常需要比較大量的訓練資料及訓練時間才能達到理想的成果，而另外 YOLOv5 本身又擁有許多 data augmentation 的手段，可能使他表現更好，因此 YOLOv5 模型可能更適合本案例中使用的特定數據集，從而在驗證集上有更好的表現。

5. 可能的改進方向: 增加訓練數據集的規模以提供更大、更多樣化的數據集，這可以幫助模型學習更廣泛的物體特徵，提高其泛化能力，或許也可以提升 DETR 的收斂表現。用不同的預訓練權重進行實驗可能在特定的數據集上能有不同的表現，或許性能能得到改善。微調 hyperparameter 也可能提高模型的性能，但這可能需要一些實驗來找到最好的數值。增加 data augmentation 的技術，可能可以進一步增加訓練數據的多樣性，提高模型對新數據的適應能力。

3. Report the implementation details of both methods: 如 augmentation, loss function, cross validation method

YOLOv5 和 DETR 都是用於 object detection 的最先進有效的深度學習模型，但它們所使用的方法和架構各不相同。以下將從不同方面比較這兩個模型

- **Data Augmentation**
  YOLOv5 採用了一些 data augmentation 技術，如 Mosaic 數據增強: random cropping、flipping 和 color jittering，以及 Copy paste、Random affine(Rotation, Scale, Translation and Shear)、MixUp、Albumentations、Augment HSV(Hue, Saturation, Value) 、Random horizontal flip 等多種方法，以增加訓練資料的多樣性，提高模型的泛化能力。而 DETR 則不使用任何 data augmentation 技術，它只依靠 transformer 結構來學習輸入圖像的 robust representations。
- **Loss Function**
  Loss Function 是所有深度學習模型中最關鍵的部分，其中 YOLOv5 採用了 BECLogits loss function 計算 objectness score 的損失，class probability 則採用了 binary cross-entropy loss，而 GIoU loss 則用於計算預測 object bounding boxes 與 ground truth bounding boxes 之間的相似度，它是 IoU (Intersection over Union) loss 的修改版，IoU 通常用於 object detection 模型。 GIoU loss 是對兩個 bounding boxes 之間重疊的部分進行更準確的測量，並同時考慮兩者交集和聯集的情形。如下圖所示，C 是 A, B 兩個框框可以圍出的最小的封閉矩形。可以看到左右兩張圖都沒相交，但是因為左邊的圖 A 和 B 距離比較短，所以 Loss 比較低。



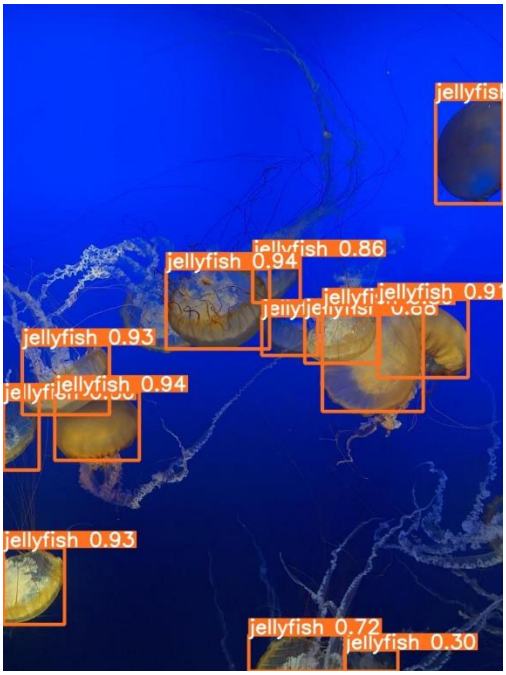
另一方面，DETR 使用了非常特別獨特的 loss function 來做計算，將要預測 object 與 ground truth object 做比較，而不需要 anchor boxes，它原理是根據預測對象和 ground truth 對象的相似度分數，找到它們之間的最

佳匹配點，這是用 Hungarian algorithm 完成的，這種演算法是一種 combinatorial optimization algorithm，在兩組對象之間找到最佳的配對。簡單來說，DETR 和 YOLOv5 使用的 loss function 有很大的不同，DETR 使用 Hungarian algorithm 來進行訓練，而 YOLOv5 則使用較為簡單的 binary cross-entropy loss 和 GIOU Loss 的組合來進行計算。

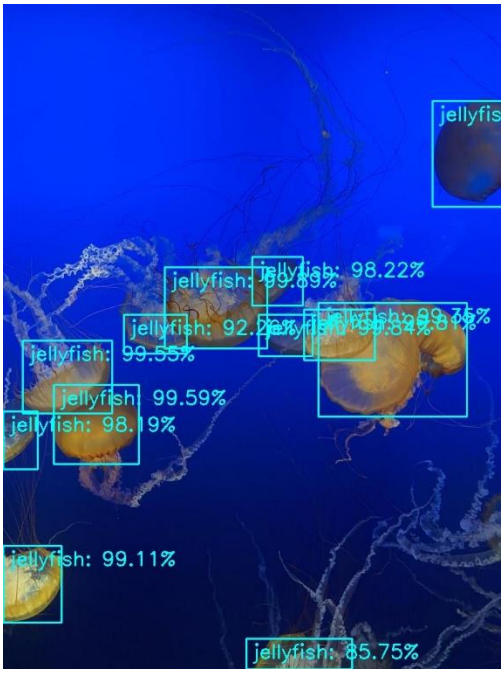
- **Cross-validation**
  YOLOv5 和 DETR 都使用 cross-validation 來評估模型在未見過的數據上的表現。 YOLOv5 使用 k-fold cross-validation，其中訓練資料被分成 k 個子集，模型在 k-1 個子集上進行訓練，在剩下的子集上進行評估，每個子集皆被用作一次驗證集，最後再求出平均的性能。 DETR 也使用 k-fold cross-validation，但也可以視情況使用 leave-one-out cross-validation 或是 stratified cross-validation 等其他方法進行評估。
- **Architecture**
  YOLOv5 是使用自適應 anchor boxes object detection 的方法，其中模型根據預先定義好的 anchor boxes 來調整預測 bounding boxes。相比之下，DETR 是一種 anchor-free object detection 的方法，其中模型不依賴預先定義好的 anchors 來預測 bounding boxes，相反的，他直接根據輸入的圖像來預測 bounding boxes，並透過學習的方式使用 transformer 架構將 object 與它們各自的 bounding boxes 接起來。
- **Performance**
  在性能方面，YOLOv5 以其超高速度和高精確度聞名。它可以在單個 GPU 上實現即時的 object detection，並在好幾個 object detection benchmarks 上取得了最佳的結果，如 COCO 和 PASCAL VOC，而且與以前的 YOLO 版本相比，YOLOv5 的結構更為簡單，訓練速度更快且檢測精度更高。另一方面，與 YOLOv5 相比，DETR 的 inference 時間較慢。然而，它在準確度方面確實也有不錯的結果及發展的可能性，也在好幾個 object detection benchmarks 取的了不少佳績，如 COCO 和 LVIS。
- **總結**
  YOLOv5 和 DETR 在 object detection 的方法和架構設計上有所不同。YOLOv5 使用了自適應 anchor boxes object detection 的方法，而 DETR 使用 anchor-free object detection 的方法。 YOLOv5 採用了 data augmentation 技術來提高模型能力，而 DETR 則依靠 transformer 架構來學習 robust representations。我認為這兩個模型都有各自的優點，YOLOv5 以其高速和高精確度而聞名，而雖然 DETR 在 COCO 上的實驗數據並不是特別突出，小物件的偵測能力偏弱，不過令人感受到了全新的可能性。

4. Visualization: draw the bounding boxes of two methods on this test image:

- **CNN-based: YOLOv5**



- **Transformer-based: DETR**



5. Reference:

(1)End-to-End Object Detection with Transformers. Computer Vision and Pattern Recognition (cs.CV). 28 May 2020 (v3)

(2)https://allen108108.github.io/blog/2020/07/27/[%E8%AB%96%E6%96%87]%20End-to-End%20Object%20Detection%20with%20Transformers/

(3) https://github.com/ultralytics/yolov5

(4) https://github.com/facebookresearch/detr

(5) https://iq.opengenus.org/yolov5/

(6) https://medium.com/ai-blog-tw/detr%E7%9A%84%E5%A4%A9%E9%A6%AC%E8%A1%8C%E7%A9%BA-%E7%94%A8transformer%E8%B5%B0%E5%87%BAobject-detection%E6%96%B0pipeline-a039f69a6d5d

(7) https://medium.com/@_Xing_Chen_/yolov5-%E8%A9%B3%E7%B4%B0%E8%A7%A3%E8%AE%80-724d55ec774

(8) https://blog.csdn.net/qq_16792139/article/details/114310670

(9)https://www.researchgate.net/publication/363824867_A_comparative_study_of_YOLOv5_models_performance_for_image_localization_and_classification

(10) Evolution of YOLO-V5 Algorithm for Object Detection: Automated Detection of Library Books and Performace validation of Dataset. 2021 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES)

(11) https://zhuanlan.zhihu.com/p/267156624

(12) https://blog.csdn.net/wusar/article/details/125477140