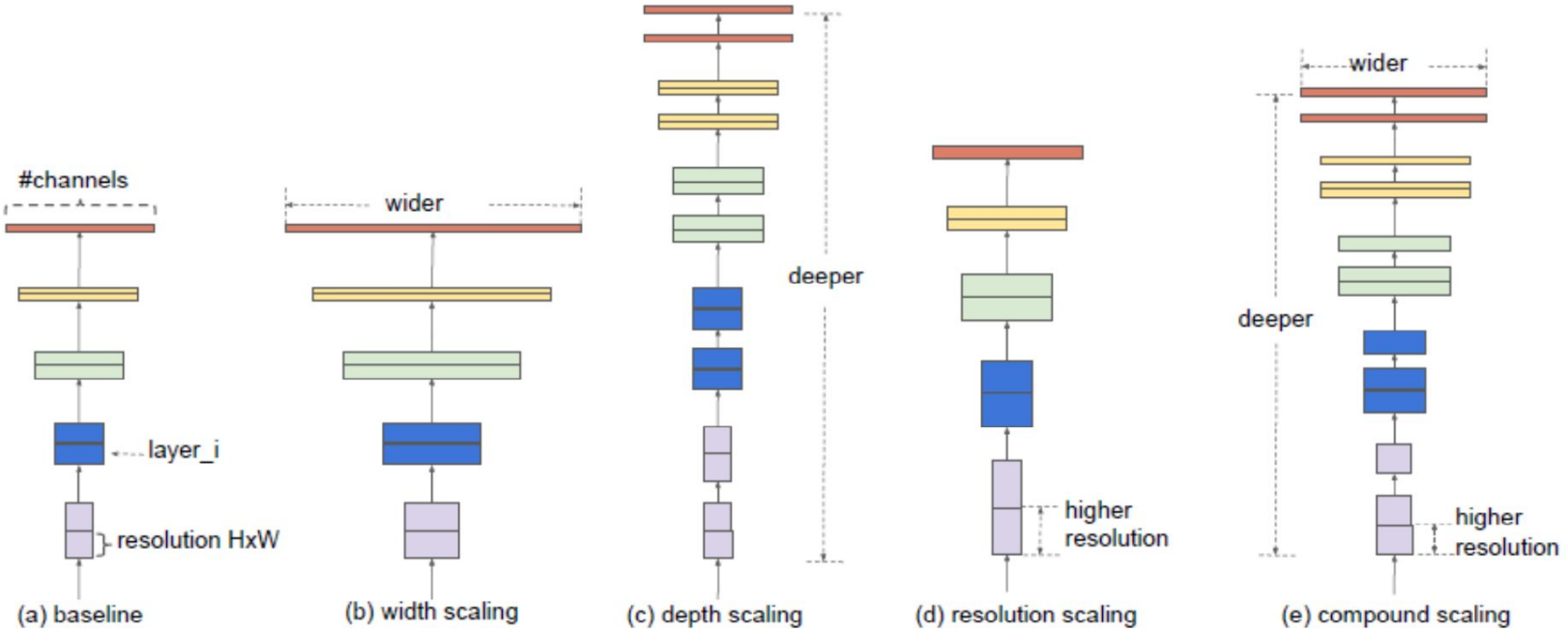
物件偵測

EfficientDet

- Recall EfficientNet, an effective scaling method for CNNs.



EfficientDet

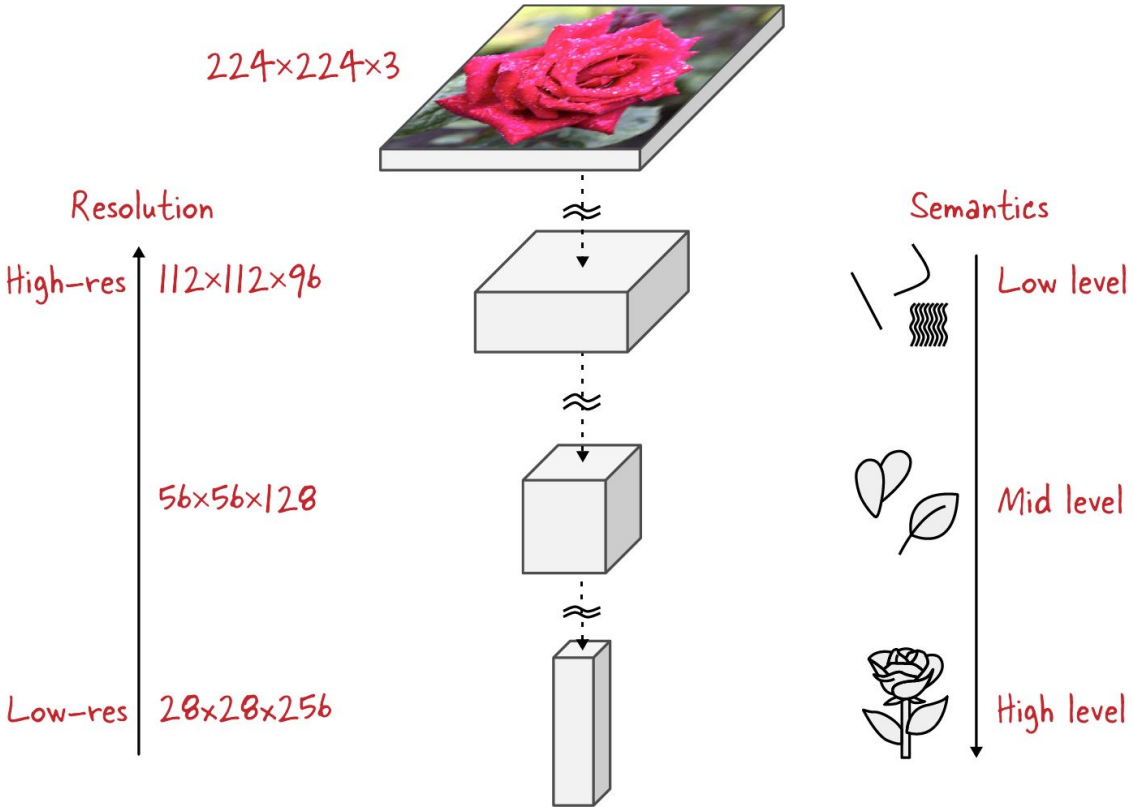
- EfficientDet 的研究人員（Google）希望了解是否有可能建立一個可擴展的物件偵測架構，在各種效能限制下都具有高精度和更高的效率。
- 和 SSD, YOLO 一樣，EfficientDet 也是 single shot detector
- 它利用 EfficientNet 的原理，並對模型縮放和多尺度特徵融合進行了改進

Highlights of EfficientDet

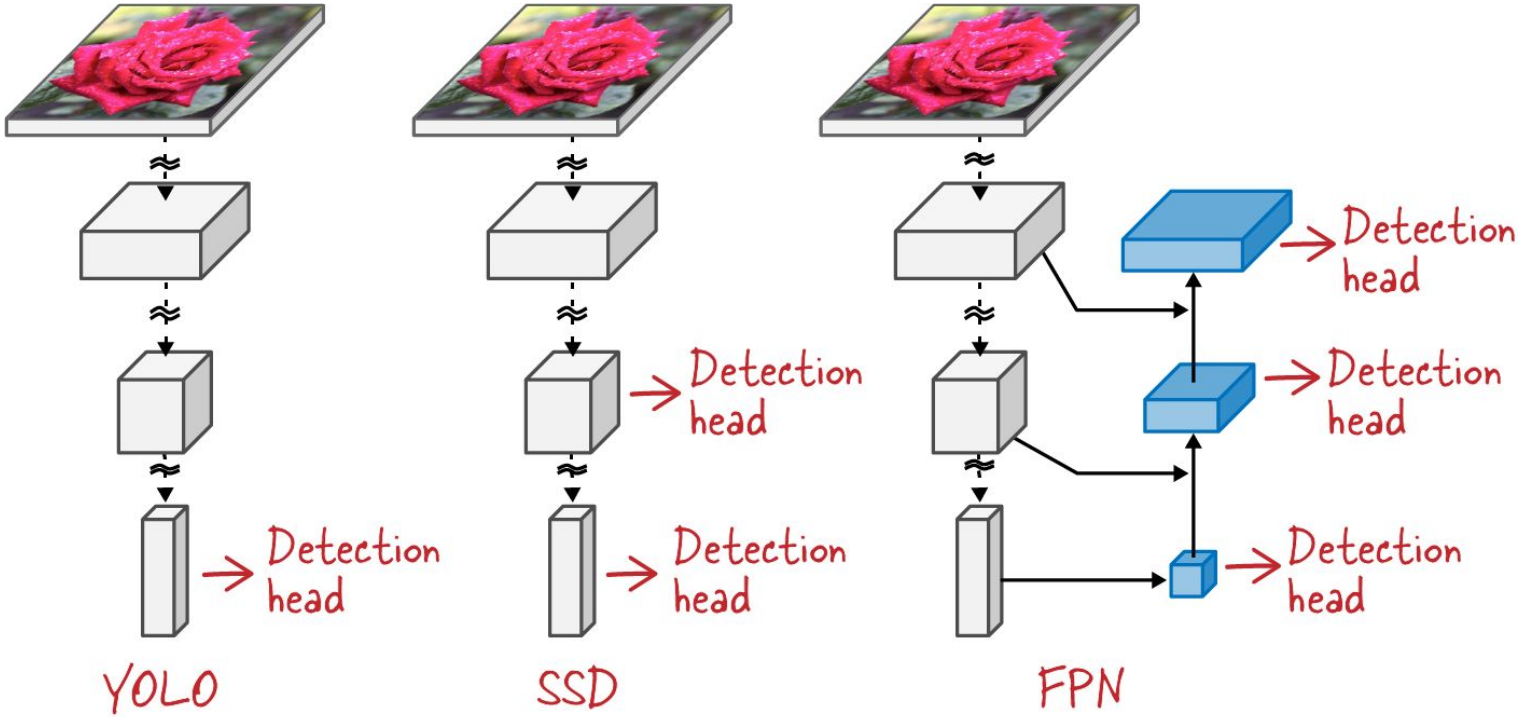
研究人員試圖解決兩個問題：

- Efficient multi-scale feature fusion：特徵金字塔網絡（FPN）已成為融合多尺度特徵的最受歡迎的方法。然而，他們只是簡單地總結它們，沒有任何區別，而且我們知道，並非所有特徵都對輸出特徵有同等的貢獻，因此需要更好的策略。
- Model Scaling：大多數偵測器依靠改進 backbone 來提高準確性，但是研究人員認為，在考慮準確性和效率時，擴展特徵網絡和偵測框/類預測網絡也很重要

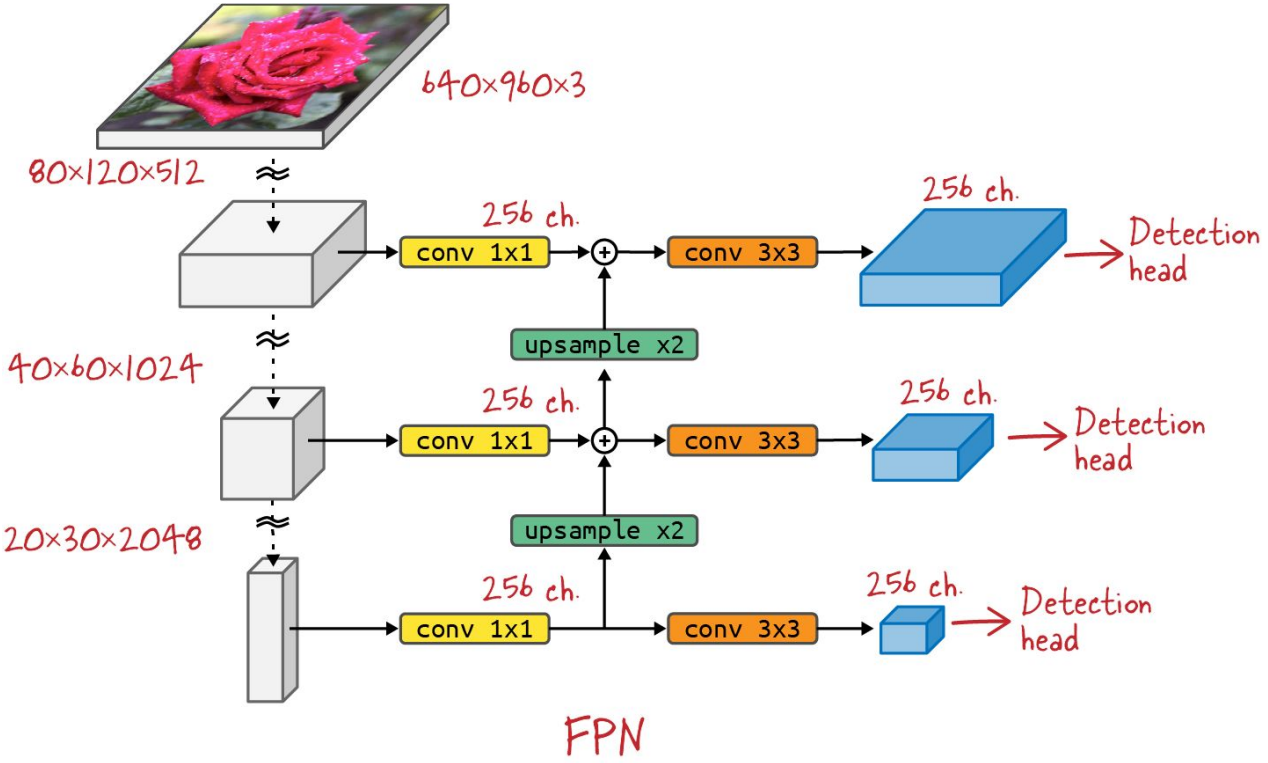
特徵金字塔網絡 (Feature Pyramid Network)



特徵金字塔網絡 (Feature Pyramid Network)

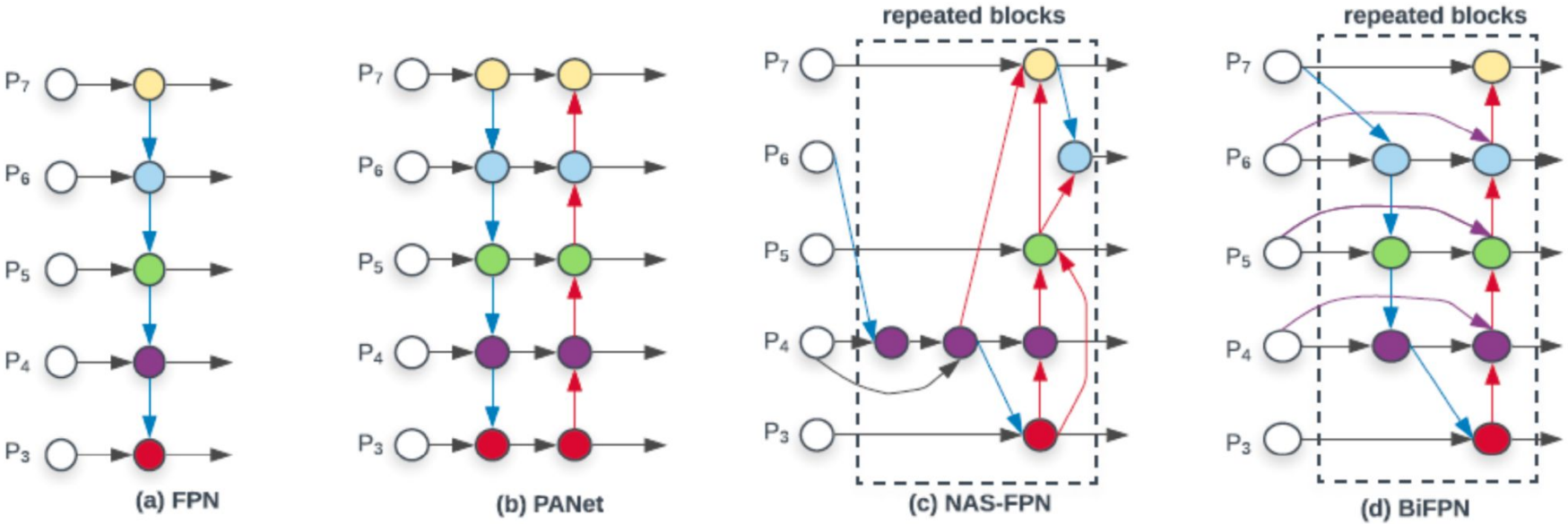


特徵金字塔網絡 (Feature Pyramid Network)



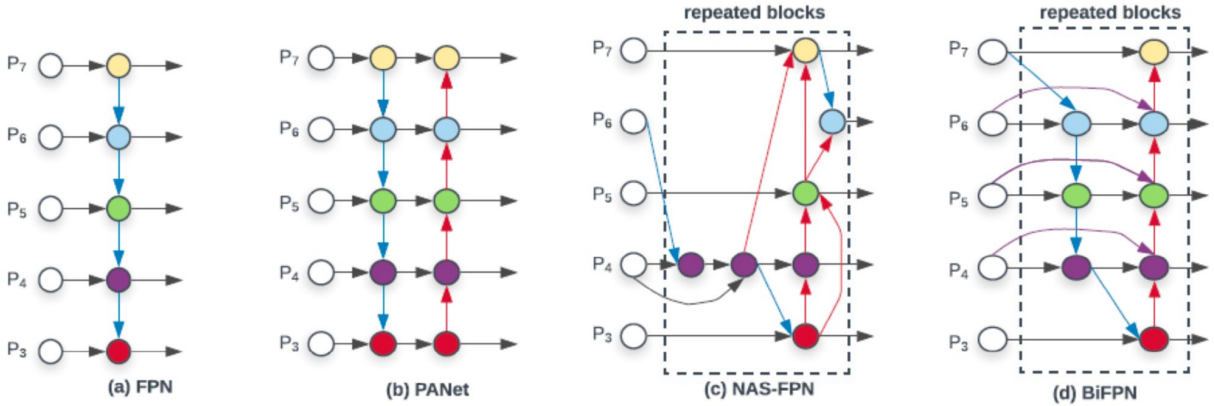
BiFPN

- EfficientDet 採用具有多尺度特徵融合的 BiFPN 架構，來結合不同解析度下的特徵



Why BiFPN

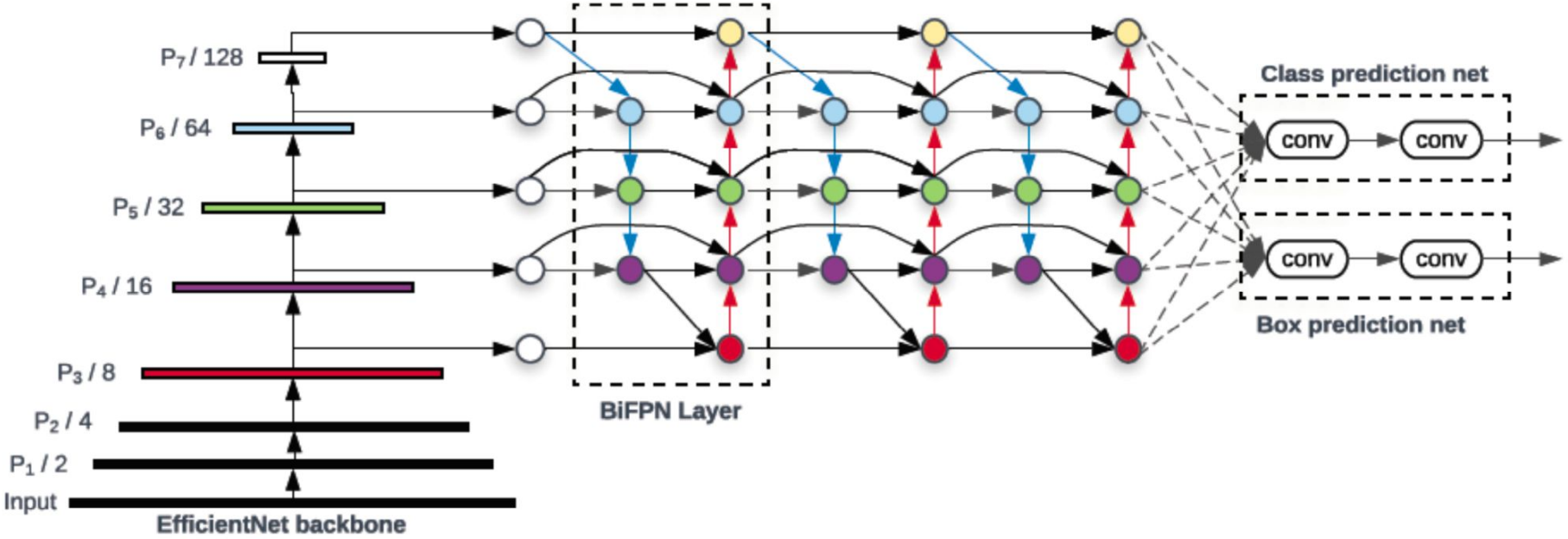
- FPN 的限制之一是它受到由上而下的資訊流的限制
- PANet 增加了額外的自下而上的資訊傳遞
- NAS-FPN 使用神經架構搜尋來找到不規則的特徵網路拓撲，然後重複套用相同的區 (Google)
- BiFPN 透過加權特徵融合、去冗余連接、和複合縮放實現了更好的準確性和效率 (Google)



FPN, PANet, NAS-FPN, BiFPN Comparison

特徵架構	訊息流方向	融合方式	優點	缺點
FPN	單向	相加	結構簡單、計算效率高	低階特徵和高階特徵融合不夠全面
PANet	雙向	相加	小物體檢測效果好、更豐富的多尺度融合	計算成本高、參數量大
NAS-FPN	自動搜索決定	自動選擇融合方式	自動搜尋最優架構、表現好	搜尋過程耗時、複雜度較大
BiFPN	雙向	加權融合 + 去冗餘	效率高、適合計算資源受限的情境	增加了加權參數

EfficientDet Architecture



EfficientDet Prediction Net

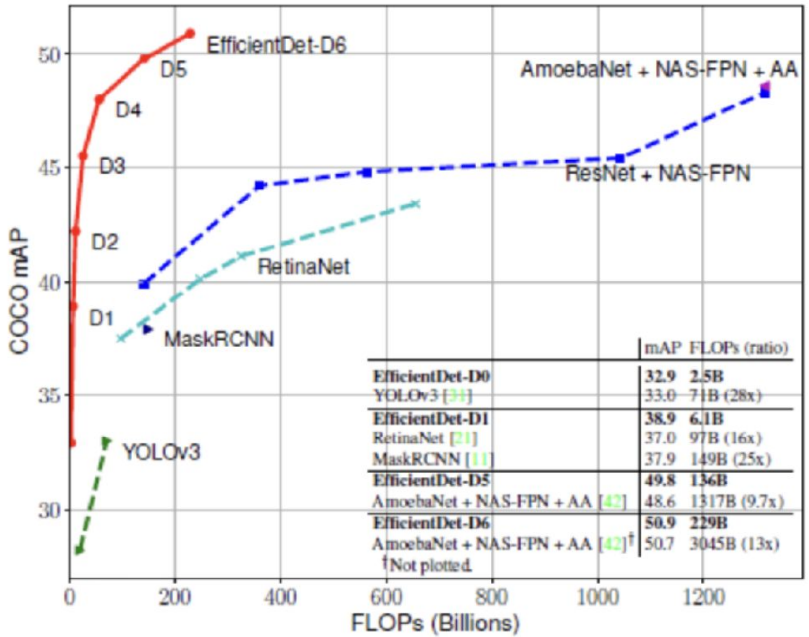
- Prediction net 是一個輕量級 CNN，生成 classifacation 和 bounding box
- Class Prediction Head
  - 通常是4層 Depthwise Separable Convolution，每層後面跟著一個 Batch Normalization 層和 ReLU 激活函數
  - 最終輸出層是一個 feature map，為每個候選區域產生類別預測
- Box Regression Head
  - 與 class prediction head 類似。最後輸出用於回歸和預測每個錨框的四個邊界框座標偏移的 feature map
- Backbone, BiFPN, Prediction Net 可以一起進行縮放

EfficientDet Version

	Input size R_{input}	Backbone Network	BiFPN #channels W_{bifpn}	BiFPN #layers D_{bifpn}	Box/class #layers D_{class}
D0 ($\phi = 0$)	512	B0	64	2	3
D1 ($\phi = 1$)	640	B1	88	3	3
D2 ($\phi = 2$)	768	B2	112	4	3
D3 ($\phi = 3$)	896	B3	160	5	4
D4 ($\phi = 4$)	1024	B4	224	6	4
D5 ($\phi = 5$)	1280	B5	288	7	4
D6 ($\phi = 6$)	1408	B6	384	8	5
D7	1536	B6	384	8	5

Table 1: **Scaling configs for EfficientDet D0-D7 –** ϕ is the compound coefficient that controls all other scaling dimensions; *BiFPN, box/class net, and input size* are scaled up using equation 1, 2, 3 respectively. D7 has the same settings as D6 except using larger input size.

EfficientDet Performance



	mAP	FLOPs (ratio)
EfficientDet-D0	**32.9**	**2.5B**
YOLOv3 [31]	33.0	71B (28x)
EfficientDet-D1	**38.9**	**6.1B**
RetinaNet [21]	37.0	97B (16x)
MaskRCNN [11]	37.9	149B (25x)
EfficientDet-D5	**49.8**	**136B**
AmoebaNet + NAS-FPN + AA [42]	48.6	1317B (9.7x)
EfficientDet-D6	**50.9**	**229B**
AmoebaNet + NAS-FPN + AA [42]†	50.7	3045B (13x)

†Not plotted.

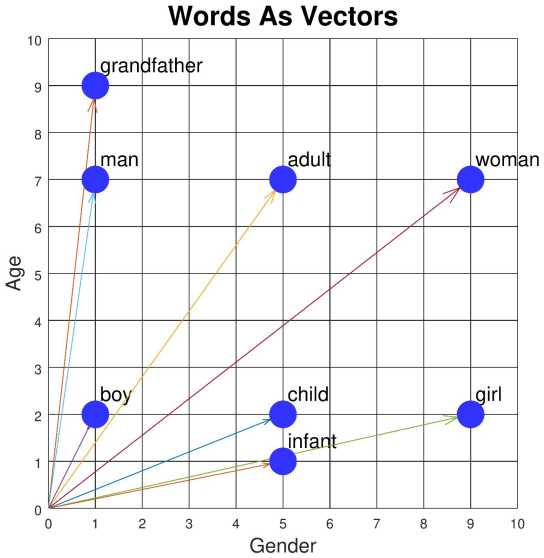
Model FLOPs vs COCO accuracy – All numbers are for single-model single-scale. Our EfficientDet achieves much better accuracy with fewer computations. In particular, EfficientDet-D6 achieves new state-of-the-art 50.9% COCO mAP with 4x fewer parameters and 13x fewer FLOPs than prior models.

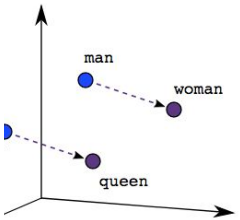
Vision Transformer (ViT)

- Transformer 在2017年著名論文《Attention is all you need》發表後在NLP中廣泛應用
- Transformers use something called self-attention
- 讓模型可以關注非局部的特徵

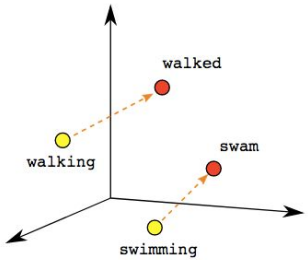
What is Transformer?

- In NLP, we use embeddings to represent the meaning of a word or a sentence
-

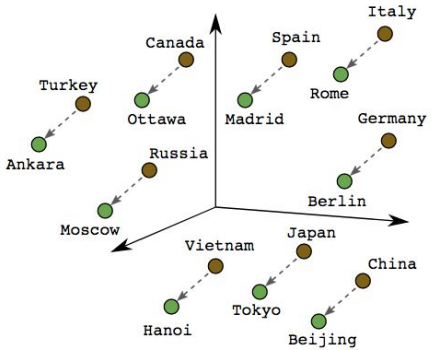




Male-Female



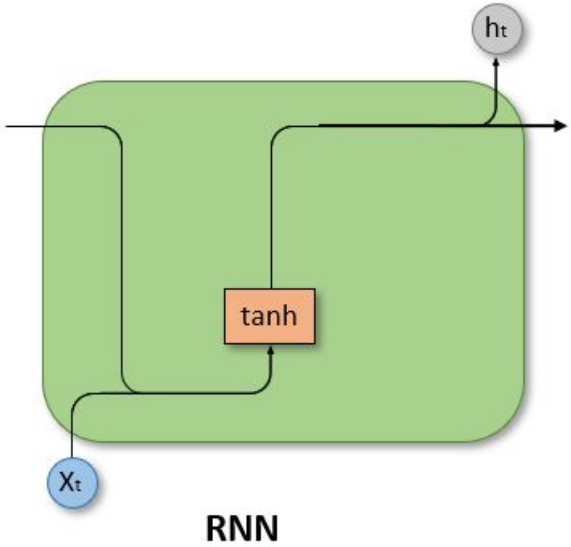
Verb Tense



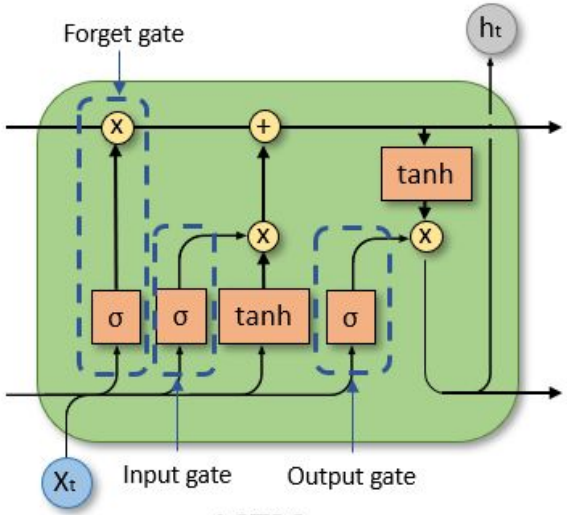
Country-Capital

What is Transformer?

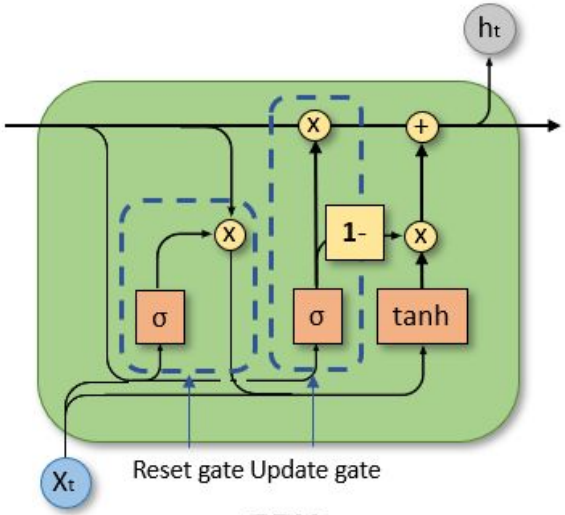
- RNN 和 LSTM 使用線性的方式處理一個句子



RNN



LSTM



GRU

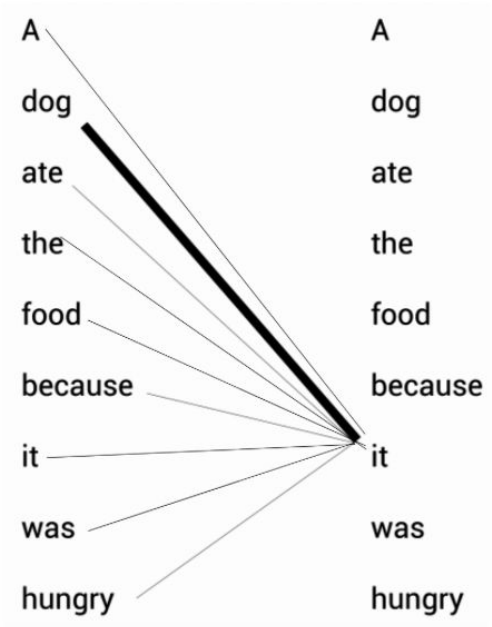
What is Transformer?

- Sentence 1: She poured water from the pitcher to the cup until it was full.（她將水罐中的水倒入杯子，直至它滿了為止。）
- 這裡的「it」是指杯子
- Sentence 2: She poured water from the pitcher to the cup until it was empty.（她將水罐中的水倒入杯子，直至它空了為止。）
- 這裡的「it」是指水罐。
- RNN 和 LSTM 無法考慮到 “it” 後面的資訊

What is Transformer?

- In Transformer, we use self-attention to calculate the relationship between each pair of words

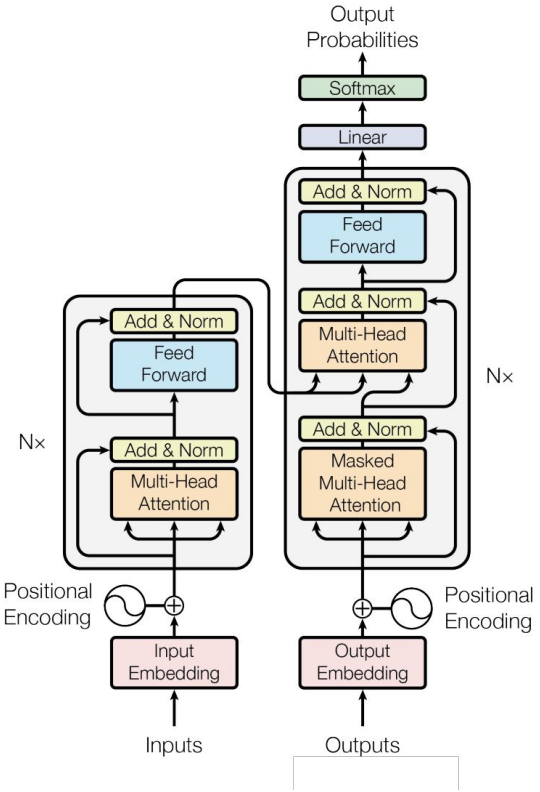
A dog ate the food because it was hungry



What is Transformer?

BERT

Encoder

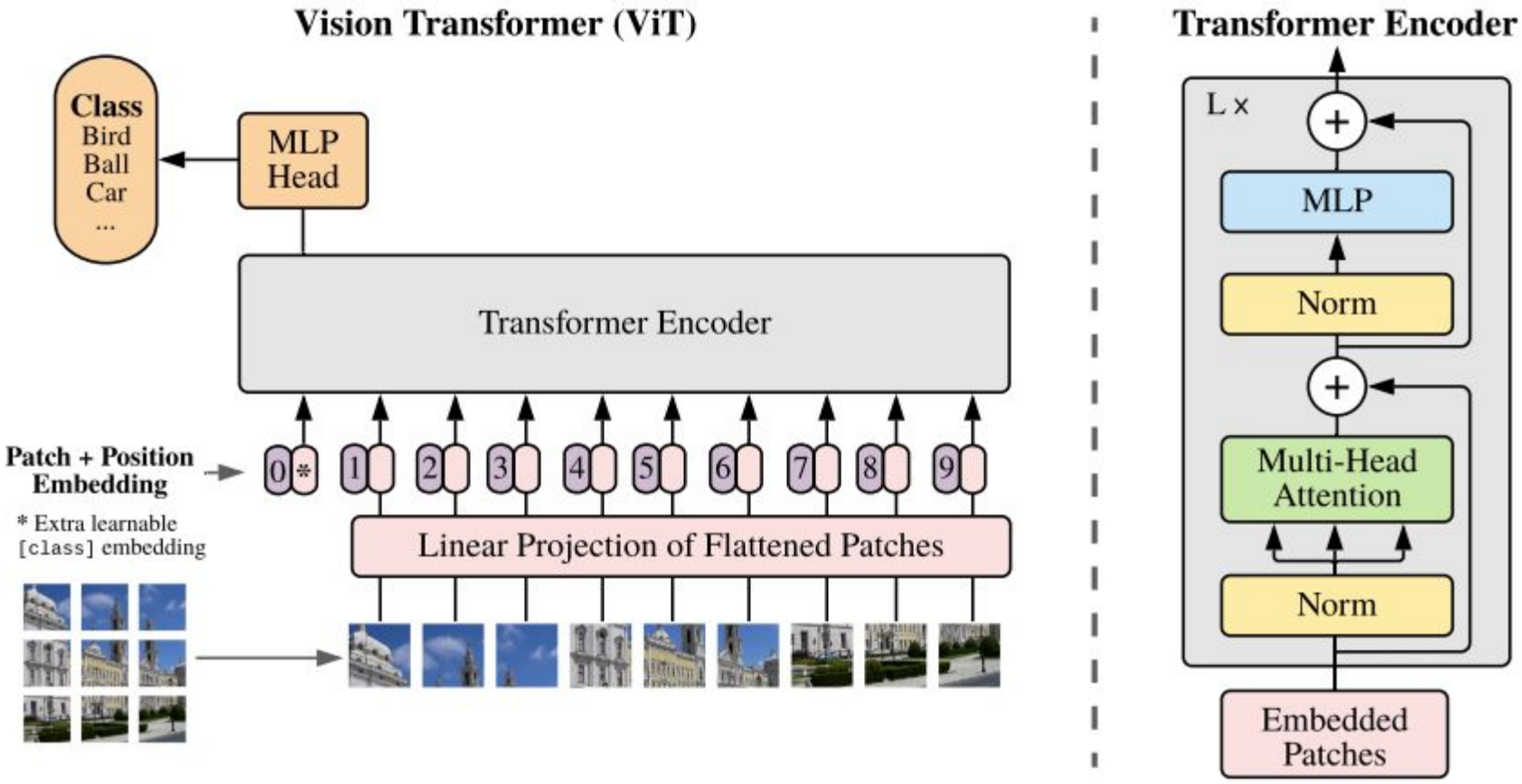


GPT

Decoder

How is this used for Images?

- Split an image into patches
- Flatten the patches
- Create a lower-dimensional linear embeddings (Conv) from the flattened patches
- Add positional embeddings (trainable position embedding used typically)
- Feed the sequence as an input to a standard transformer encoder
- Pre-train the model with image labels (fully supervised on a huge dataset)
- Fine tune on the downstream dataset for image classification

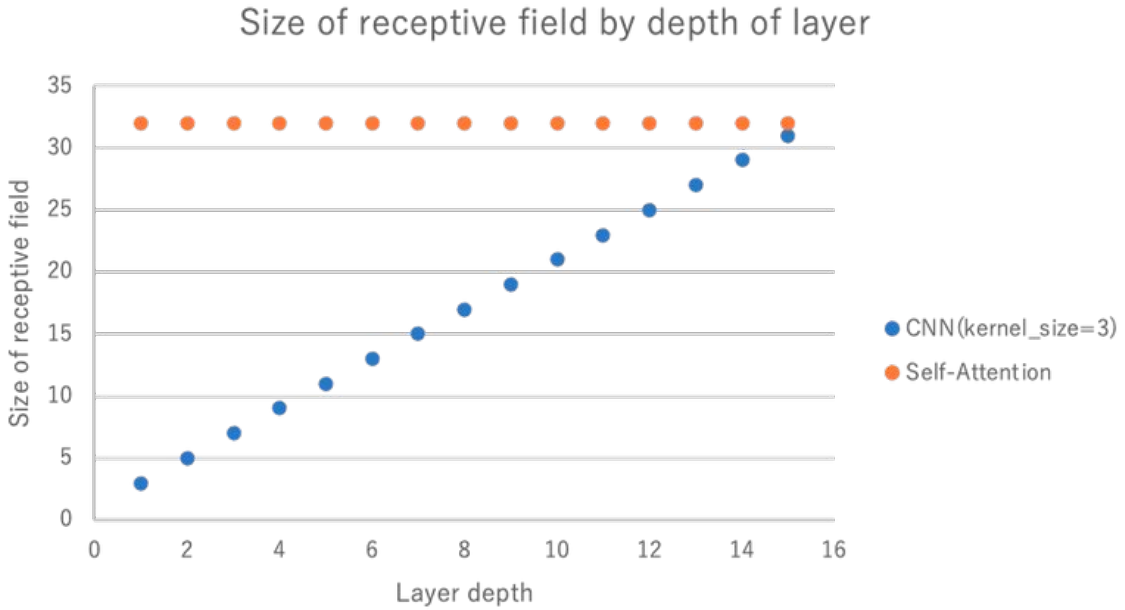

Vision Transformer (ViT)
Class
Bird
Ball
Car
...
MLP
Head
Transformer Encoder
Patch + Position
Embedding
* Extra learnable
[class] embedding
0*
1
2
3
4
5
6
7
8
9
Linear Projection of Flattened Patches
Transformer Encoder
L x
MLP
Norm
Multi-Head
Attention
Norm
Embedded
Patches

More about Vision Transformers

- Image patches are equivalent to sequence tokens (words in NLP)
- We can vary the number of blocks in th encoder to get allowing for deeper networks
- ViTs require A LOT of data to be trained to beat SOTA CNNs
- However you can pre-train on a larger dataset and fine-tune on smaller ones (change the MLP head)

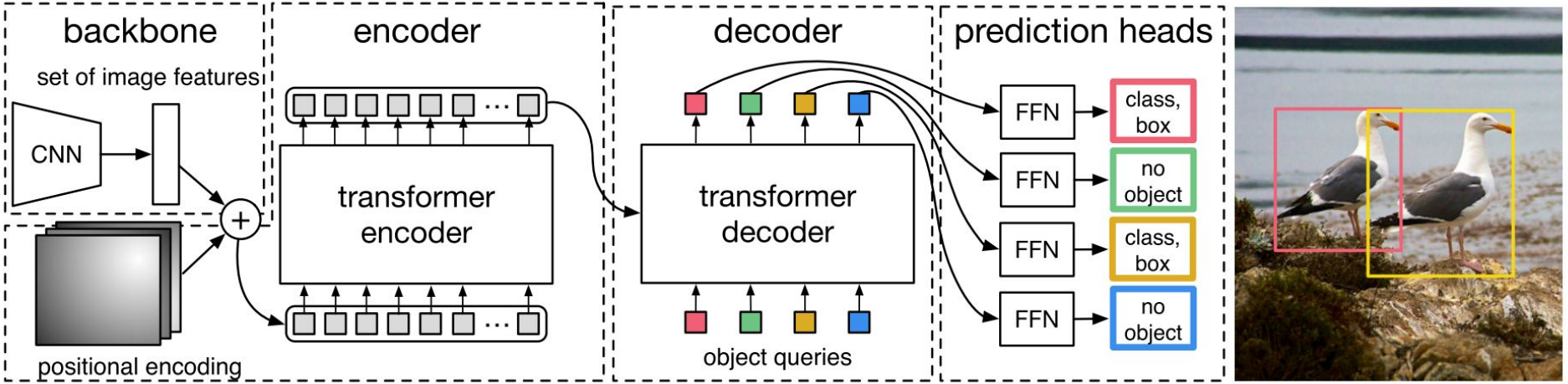
ViT vs CNN (Receptive Field)

- Receptive field: the portion of the input image that a neuron (or a filter) “looks at” or is connected to



Detection Transformer (DETR)

- 發表於2020年,由Facebook AI Research團隊提出
- 將Transformer架構應用於物件偵測任務



- Object queries 是可學習的嵌入向量 ,數量固定(通常為100)
- 通過自注意力和交叉注意力機制 ,將 queries 轉換為object representation

DETR Key Features

- 集合預測: 直接輸出物件集合,無需 NMS 後處理
- 物件查詢(Object Queries): 學習固定數量的物件表示
- 雙向匹配損失: 透過 Hungarian Matching 確保每個目標物件只被預測一次
- 位置編碼: 加入空間信息

Advantages

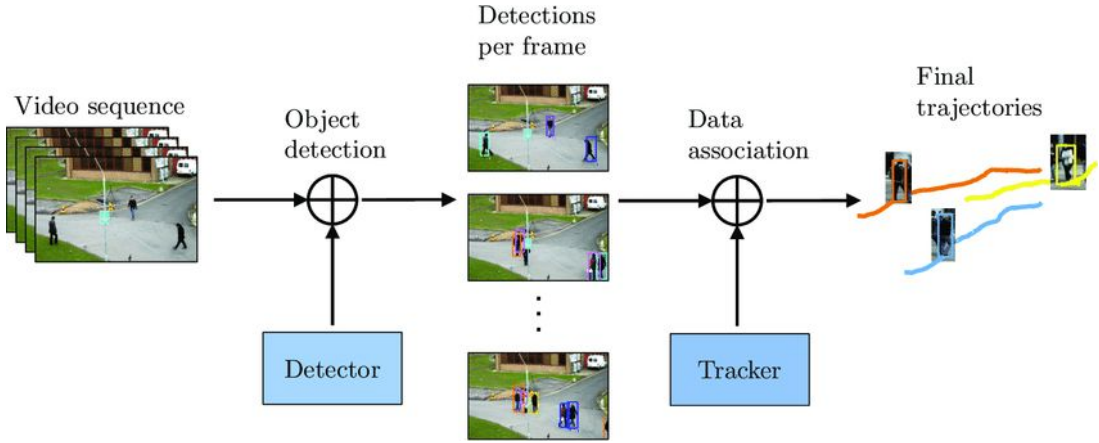
- 端對端訓練：DETR不需要複雜的後處理（例如 NMS）來產生預測結果。這種端到端的設計簡化了模型架構和訓練過程。
- 不依賴錨框：傳統偵測模型（如 Faster R-CNN 和 YOLO）依賴錨框，而 DETR 直接透過可學習 query 來偵測物件，消除了手動設計錨框的複雜性。
- 全域注意力機制：Transformer 的自注意力機制使模型能夠從全局角度理解影像資訊並捕捉長距離依賴關係，對於識別某些結構複雜或形狀不規則的物體非常有幫助。

Disadvantages

- 訓練時間長：DETR通常需要很長的訓練時間才能收斂，因為 Transformer 需要在大量資料上進行訓練。
- 小物體的偵測較弱：DETR 對大物體表現良好，但對小或密集物體效果較差。後續提出的 Deformable DETR 能夠改善此問題。
- 對大規模資料的需求：DETR的架構通常在大規模資料集（如COCO資料集）上表現較好，而在小資料集上模型效果可能不如傳統方法。

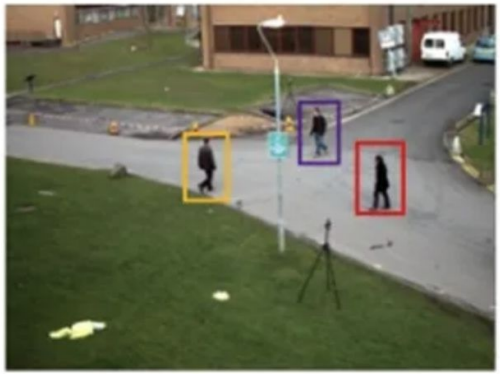
Object Tracking

- 模型大致分成兩個部分: 偵測 + 追蹤
- 判斷不同 frame 之間的物件是否為同一個

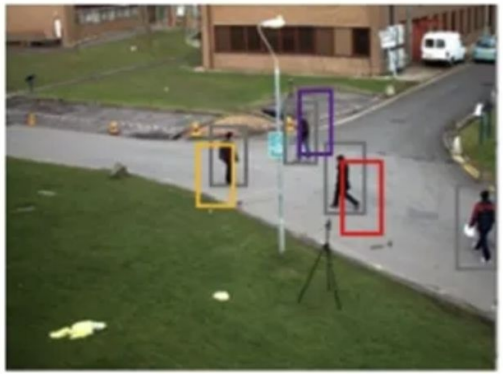


Central Tracker

- 最簡單的方法就是直接指定距離最近的框為同一個物件 (euclidean distance or IOU)
- 不適合用於複雜、物件會交錯的環境
- https://youtu.be/vPuuY-QMZxg



Frame t



Frame t+1

DeepSORT

- Introduced in 2017, 由SORT演化而來
- SORT = Kalman Filter + Hungarian Algorithm
- DeepSORT = SORT + 外觀特徵提取

Key Features

- 外觀特徵提取
  - 透過深度學習模型 (e.g. ResNEt) 來獲取每個目標的外觀特徵 (embedding)，從而在遮擋、身份切換或短暫丟失目標時保持追蹤的穩定性
- Kalman Filter
  - 根據前幀的位置資訊來估計目標的未來位置，即便在部分幀中目標未被檢測到，也可以利用 Kalman Filter 推算位置。
- Hungarian Algorithm
  - 用於解決檢測結果與已有追蹤目標的匹配問題，同時考慮了物體的空間距離和外觀特徵，從而實現更精確的多物體匹配

Procedure

- 物體偵測：在每一幀圖像中，首先利用物件偵測測模型（如YOLO、Faster R-CNN等）來獲取目標物的邊界框（bounding boxes）及其類別。
- 外觀特徵提取：對於每個檢測出的物體，使用 CNN-based 的特徵提取器來提取物體的外觀特徵。
- Kalman Filter 進行狀態預測：對於每個追蹤目標，使用 Kalman Filter 來估計其未來位置。
- Hungarian Algorithm 進行匹配：在每一幀中，通過 Hungarian Algorithm 對新的檢測目標與現有的追蹤目標進行匹配。
- ID管理與更新：根據匹配結果更新追蹤目標的位置和ID，對於無法匹配上的檢測目標會新增一個新的追蹤ID，對於長時間沒有更新的目標則刪除。

Hungarian Algorithm

Matching Cost Matrix

		Detection_1	Detection_2	Detection_3	Detection_4
1	Target_1	7	15	11	8
2	Target_2	7	19	11	11
3	Target_3	4	8	3	2
4	Target_4	12	6	2	1

Optimal Matching Results

	Previous_Target	Current_Detection	Matching_Cost
1	Target_1	Detection_4	8
2	Target_2	Detection_1	7
3	Target_3	Detection_3	3
4	Target_4	Detection_2	6

Cost Function

- DeepSORT 的 cost function 為空間距離成本和外觀特徵成本的加權總和
- 空間距離成本：Mahalanobis Distance
- 不確定性較大的方向「允許」較大的誤差，而不確定性較小的方向「容忍」較小的誤差
- 外觀特徵成本：比較由深度學習模型得到的 embeddings 之間的相似度
- $Cost_{ij} = \alpha \cdot Distance_{ij} + (1 - \alpha) \cdot Appearance_{ij}$

Implementation: https://github.com/MuhammadMoinFaisal/YOLOv8-DeepSORT-Object-Tracking?tab=readme-ov-file

Implementation

Download notebook at:
https://github.com/albert831229/nchu-computer-vision/tree/main/113/night