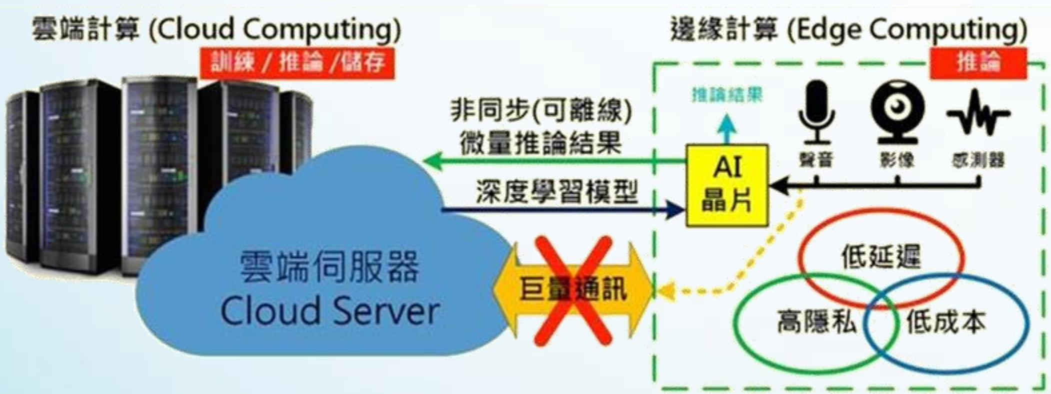
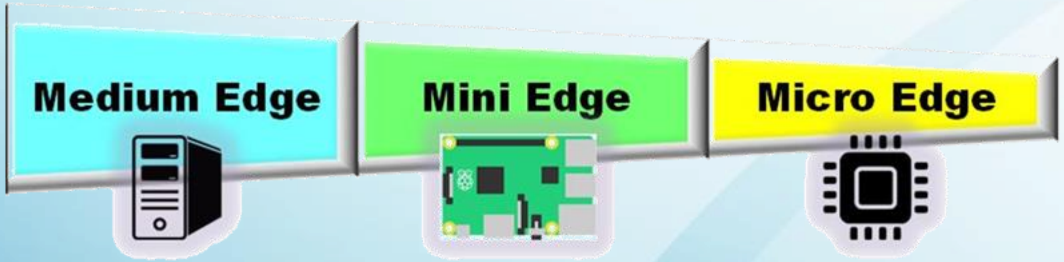
# OmniXRI's Edge AI & TinyML 小學堂





## 【第9講】
## 實作案例─物件偵測

歐尼克斯實境互動工作室 (OmniXRI Studio)
許哲豪 (Jack Hsu)

# 簡報大綱

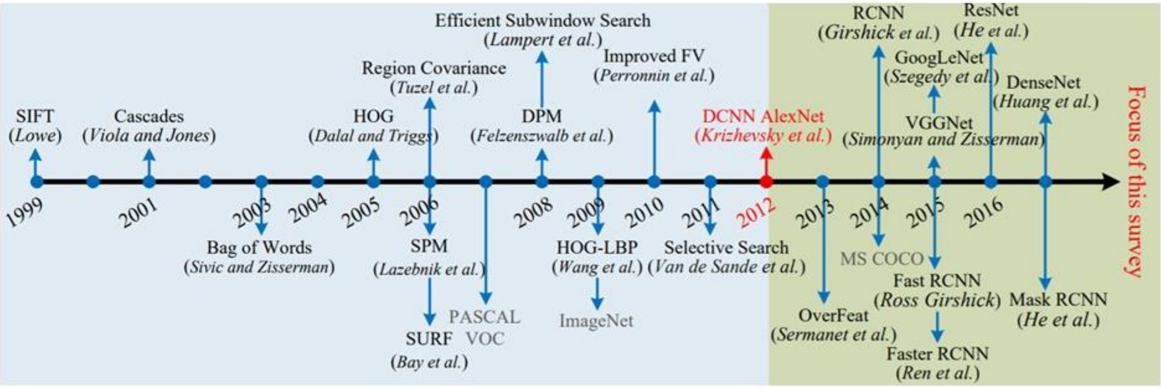
- 9.1. 物件偵測簡介
- 9.2. 物件偵測模型
- 9.3. 物件偵測評量
- 9.4. 物件偵測實作

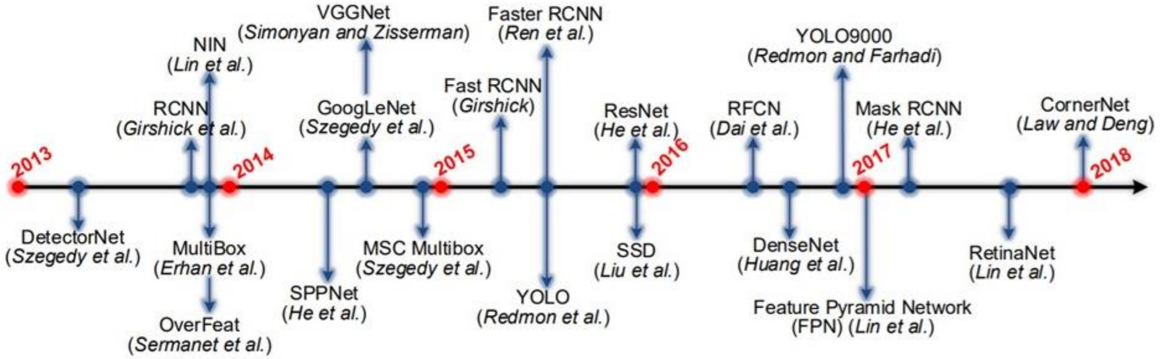
**本課程完全免費，請勿移作商業用途！**
**歡迎留言、訂閱、點讚、轉發，讓更多需要的朋友也能一起學習。**

完整課程大綱： https://omnixri.blogspot.com/2024/02/omnixris-edge-ai-tinyml-0.html
課程直播清單： https://www.youtube.com/@omnixri1784/streams

# 物件偵測演進史

特徵點提取算法 深度學習算法





精度很低 速度很快

## 特徵式

抽取影像少數特徵進行比對

精度較高 速度較低

## 二段式

先找出可能是物件的邊界框再進行分類

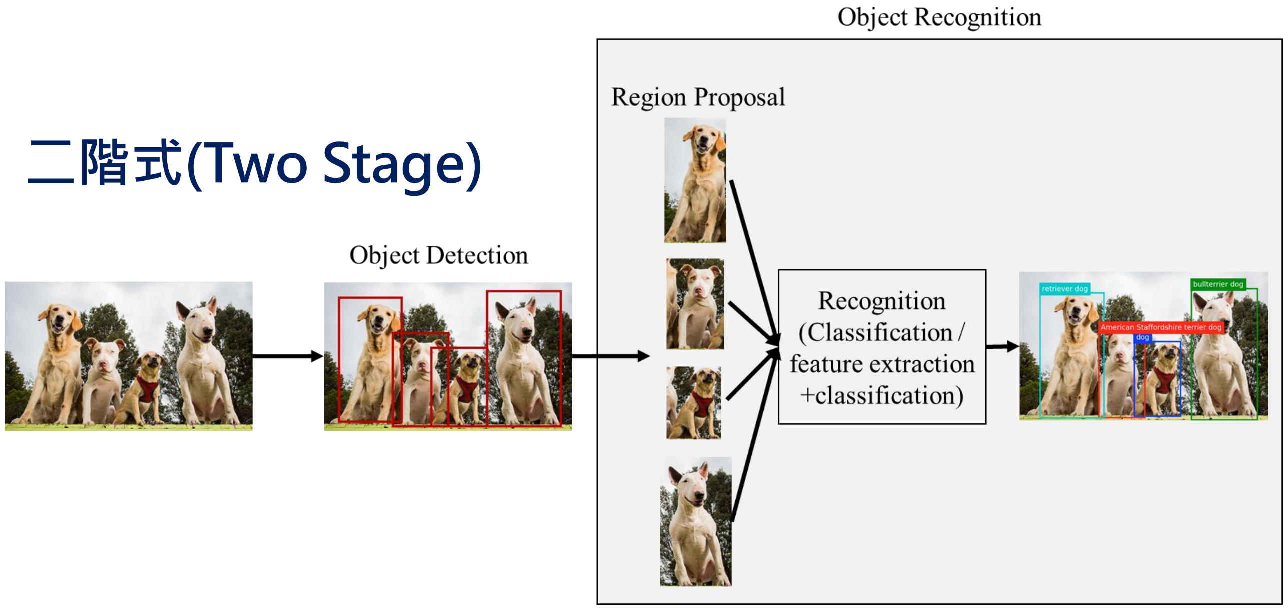
精度較低 速度較快

## 一段式

直接對特定位置邊界框預測尺寸及中心位置再進行分類

# 二階式 vs. 一階式 物件偵測

## 二階式(Two Stage)


Object Recognition

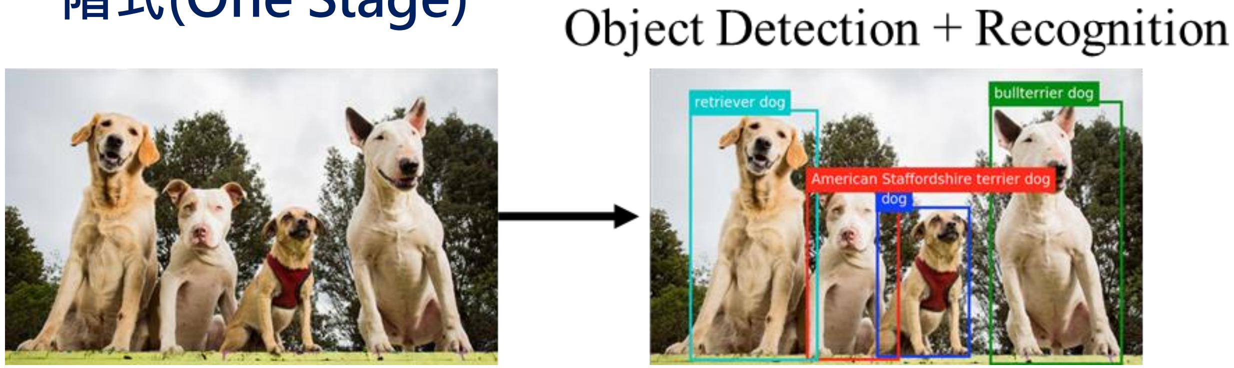
Object Detection

Region Proposal

Recognition (Classification / feature extraction +classification)

先找出物件可能區域(Region Proposals)再進行物件辨識，如
R-CNN
Fast R-CNN
Faster R-CNN
…

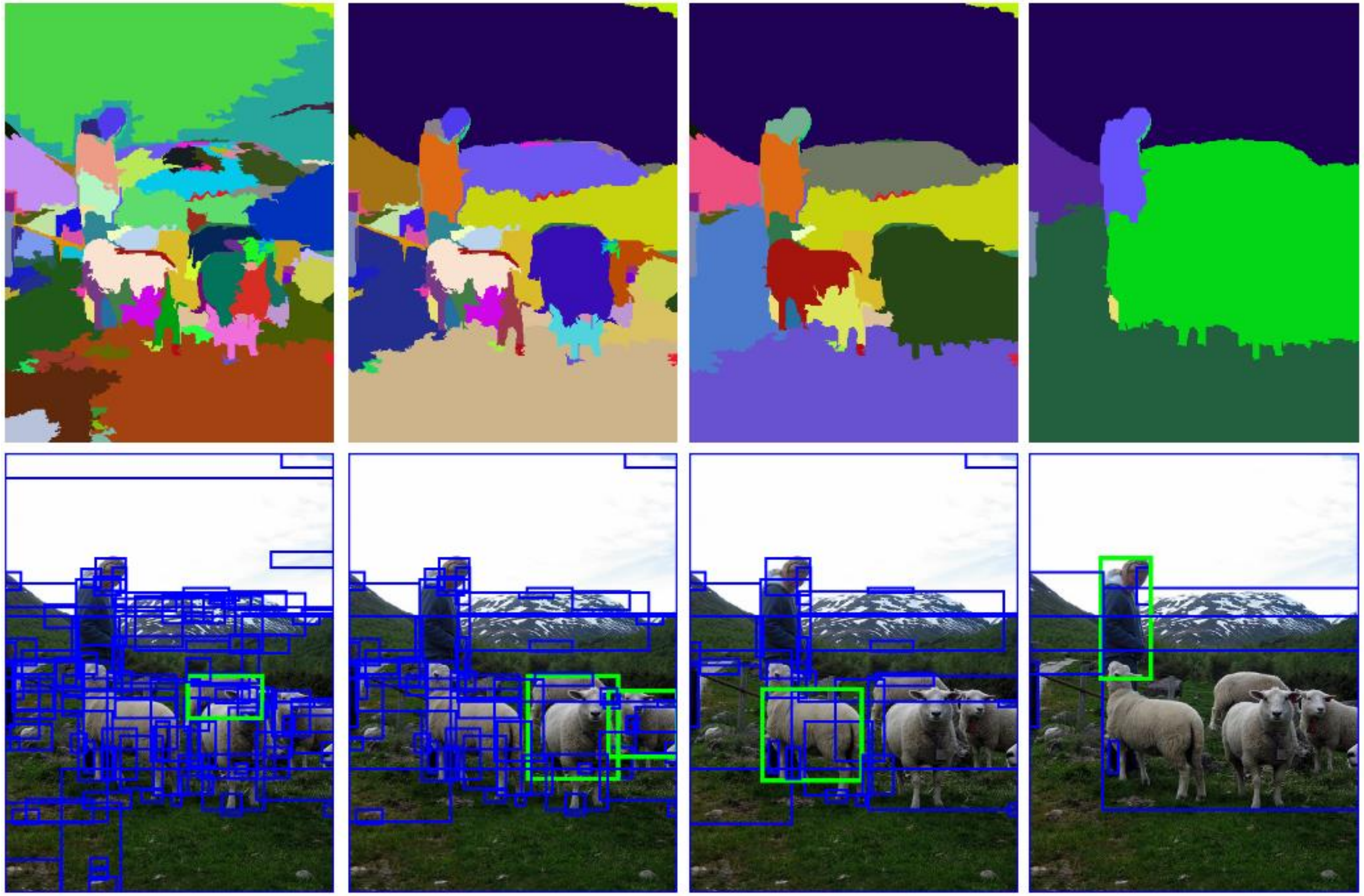
## 一階式(One Stage)


Object Detection + Recognition

物件偵測及辨識一起完成，如
SSD
YOLO
…

# 候選區域(Region Proposals)

Selective Search for Object Recognition



**二階式**
**物件偵測**

**第一階：**
產生候選區域再合併。

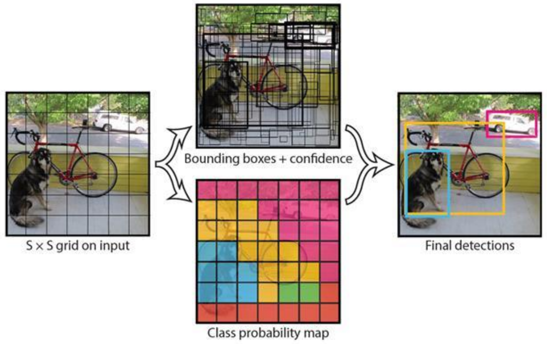
**第二階：**
影像分類。

資料來源：http://www.huppelen.nl/publications/selectiveSearchDraft.pdf

# YOLO vs. YOLO

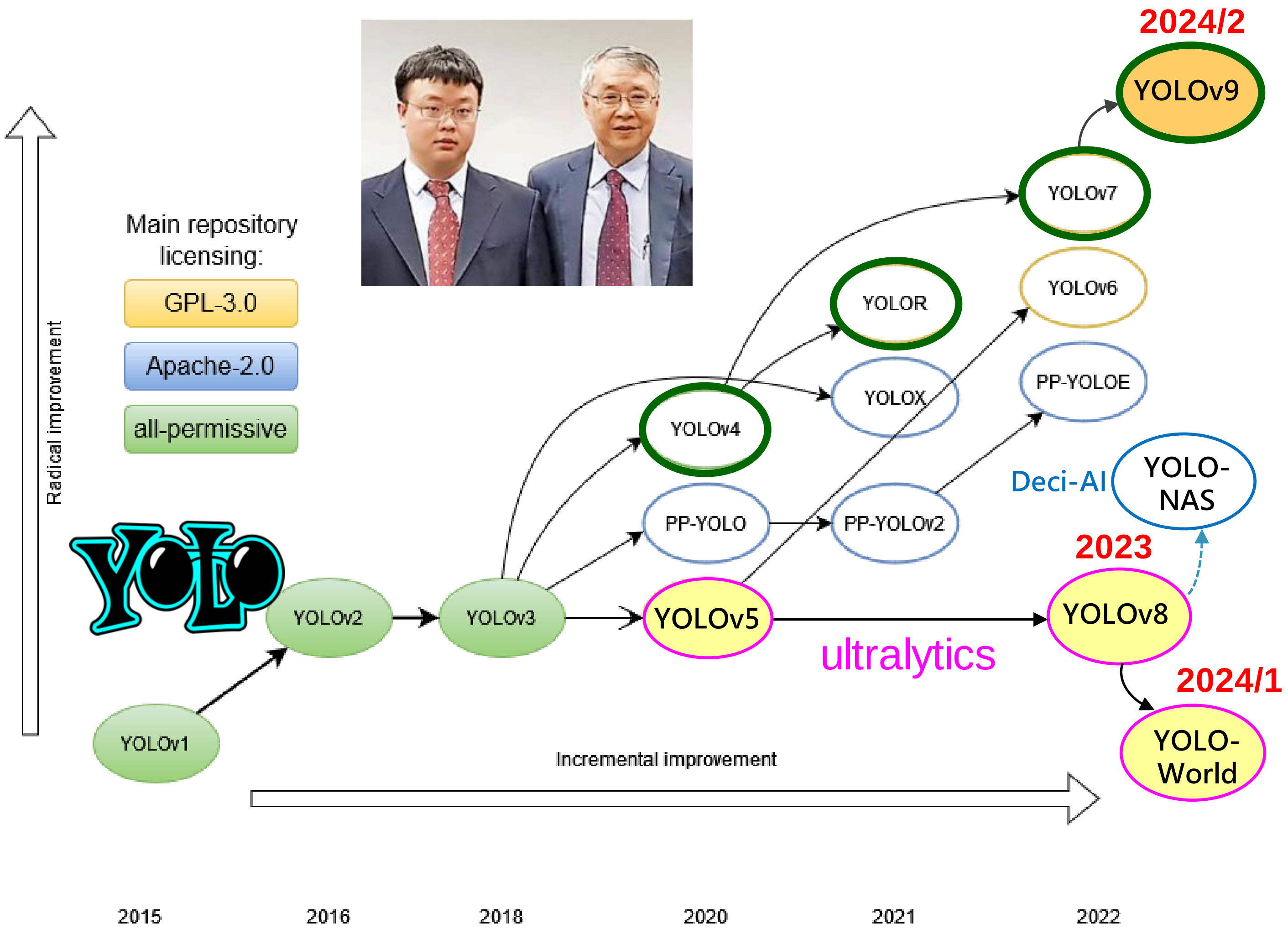
**You**
**Only**
**Live**
**Once**

別再傻傻搞不清，此YOLO非彼YOLO！



**You**
**Only**
**Look**
**Once**

# YOLO 物件偵測模型發展歷程



影像來源：https://medium.com/deelvin-machine-learning/the-evolution-of-the-yolo-neural-networks-family-from-v1-to-v7-48dd98702a3d

## You Only Look Once

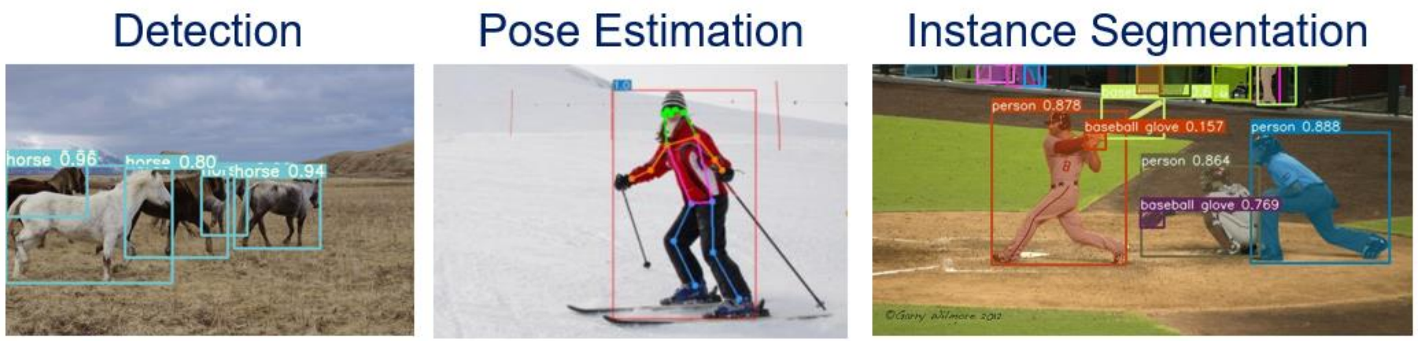
簡稱**YOLO**，目前已發展至9代(v9)，但其中v5, v6,v8為商用版本，非原團隊發表之學術論文。

其中**v4, v7, v9**為中研院資訊所廖弘源所長及其學生王建堯博士和原團隊Alexey Bochkovskiy共同開發，深獲全世界肯定，大量使用在各個領域。

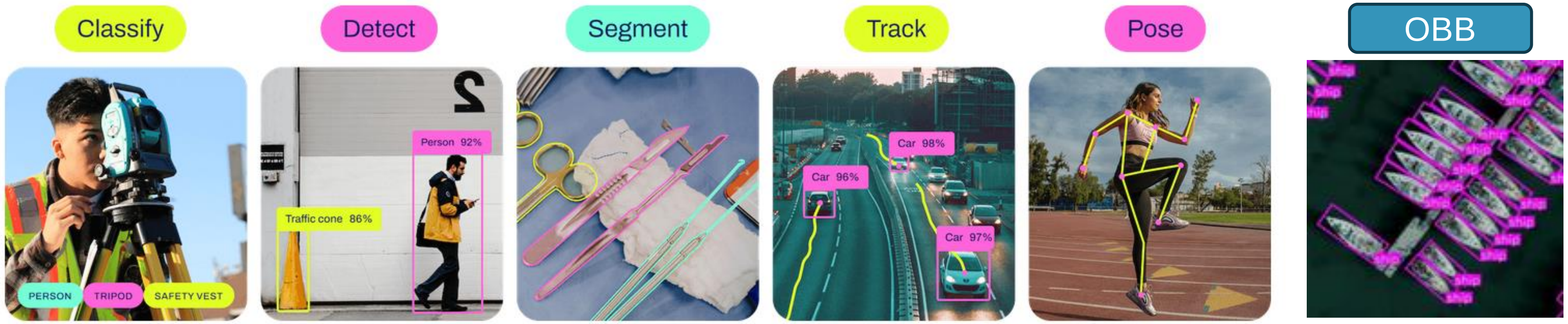
# YOLOv7 / v8 / v9 功能擴增

**COCO (80類)**

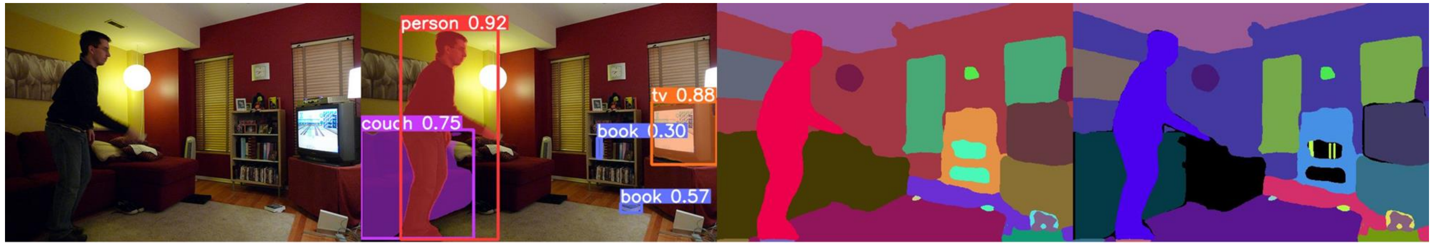
**YOLOv7**



**YOLOv8**



**YOLOv9**



Input Image

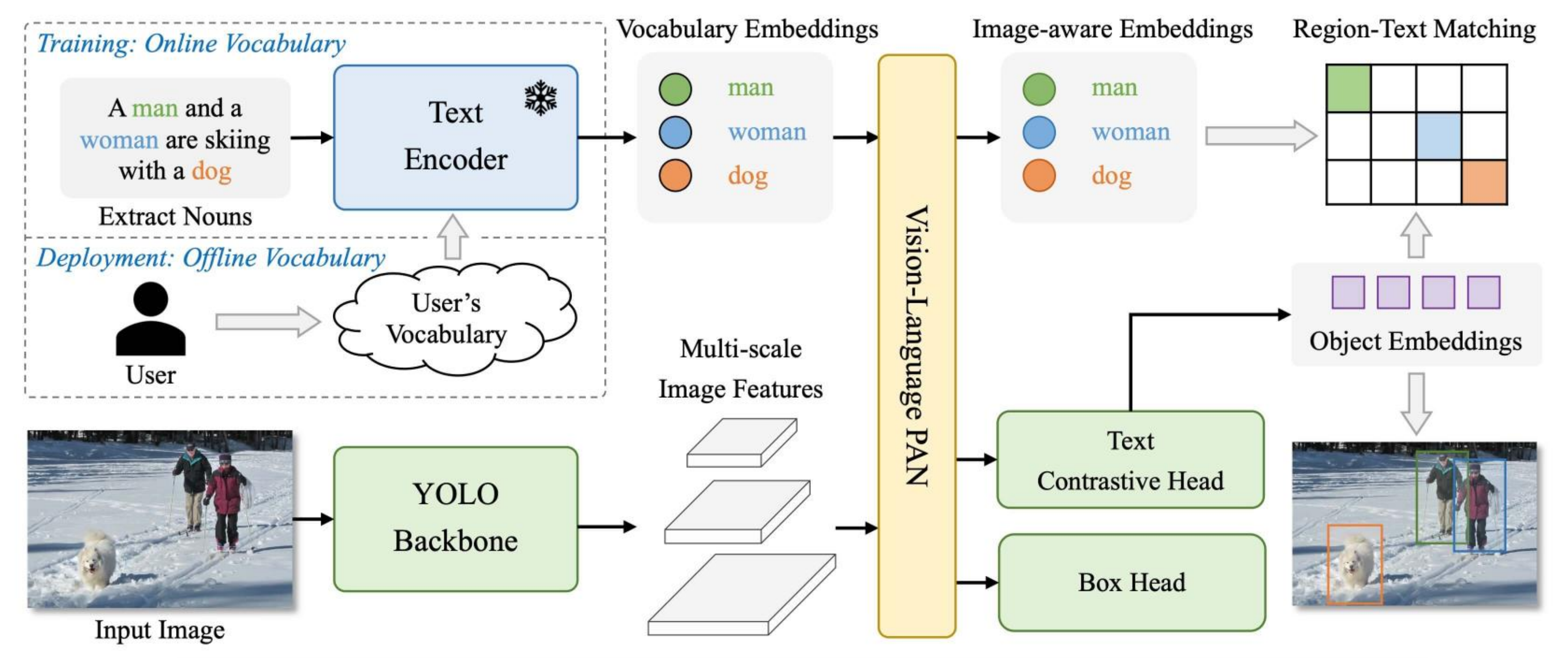
Object Detection & Instance Segmentation

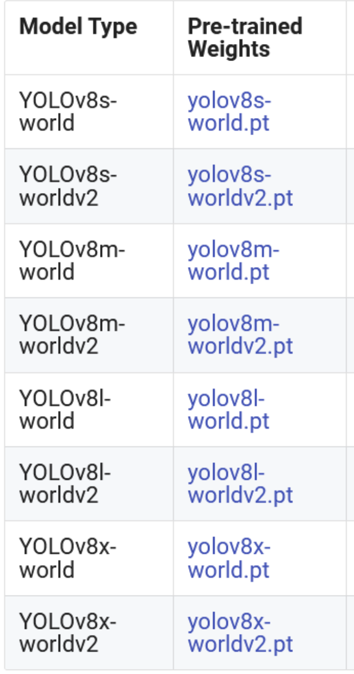
Semantic Segmentation

Panoptic Segmentation

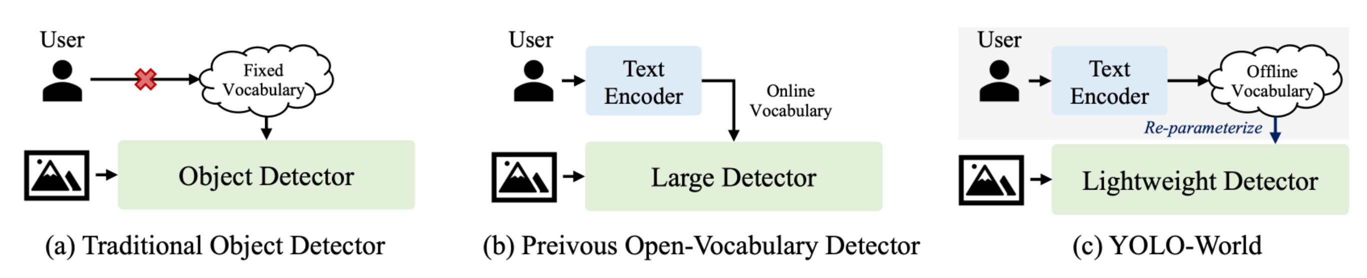
Image Captioning
(not yet release)

# Yolo-World 基本概念





| Model Type | Pre-trained Weights |
|---|---|
| YOLOv8s-world | yolov8s-world.pt |
| YOLOv8s-worldv2 | yolov8s-worldv2.pt |
| YOLOv8m-world | yolov8m-world.pt |
| YOLOv8m-worldv2 | yolov8m-worldv2.pt |
| YOLOv8l-world | yolov8l-world.pt |
| YOLOv8l-worldv2 | yolov8l-worldv2.pt |
| YOLOv8x-world | yolov8x-world.pt |
| YOLOv8x-worldv2 | yolov8x-worldv2.pt |



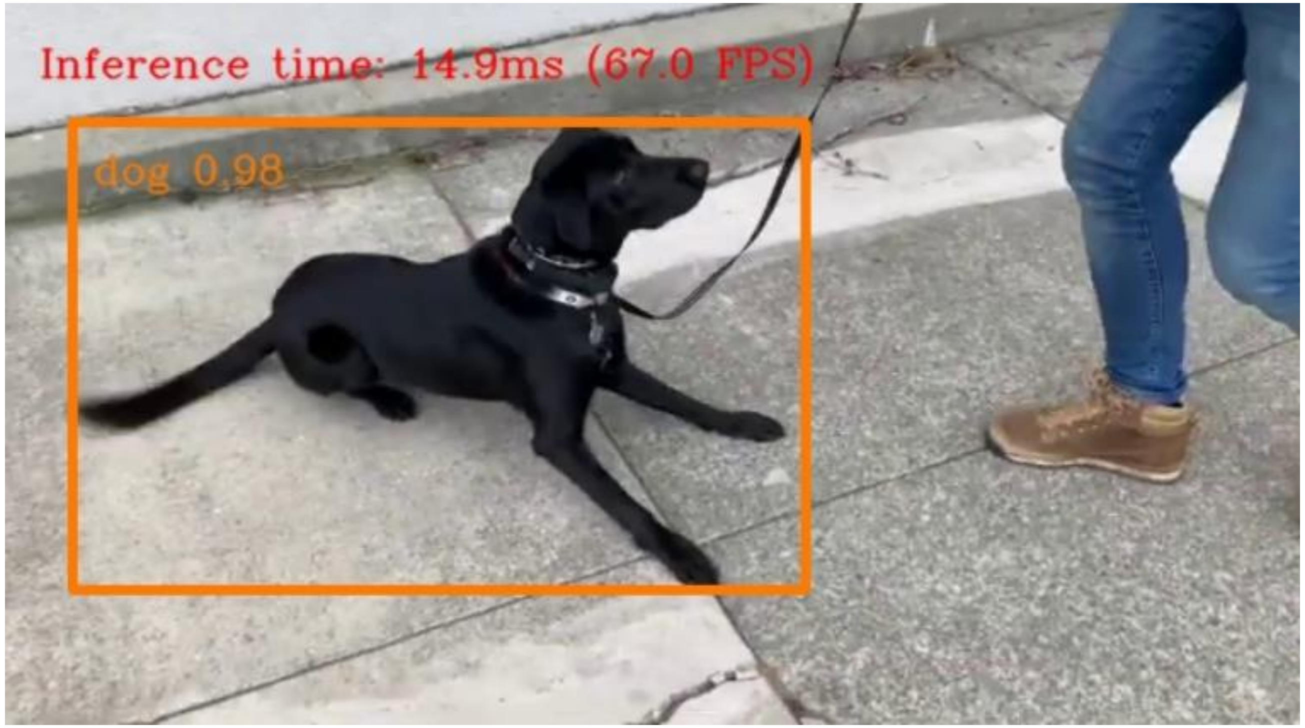
(a) Traditional Object Detector (b) Preivous Open-Vocabulary Detector (c) YOLO-World

基於Yolov8及開放詞彙檢測任務的先進實時方法。可根據描述性文本檢測圖像中的任何物體。

資料來源：https://docs.ultralytics.com/models/yolo-world/

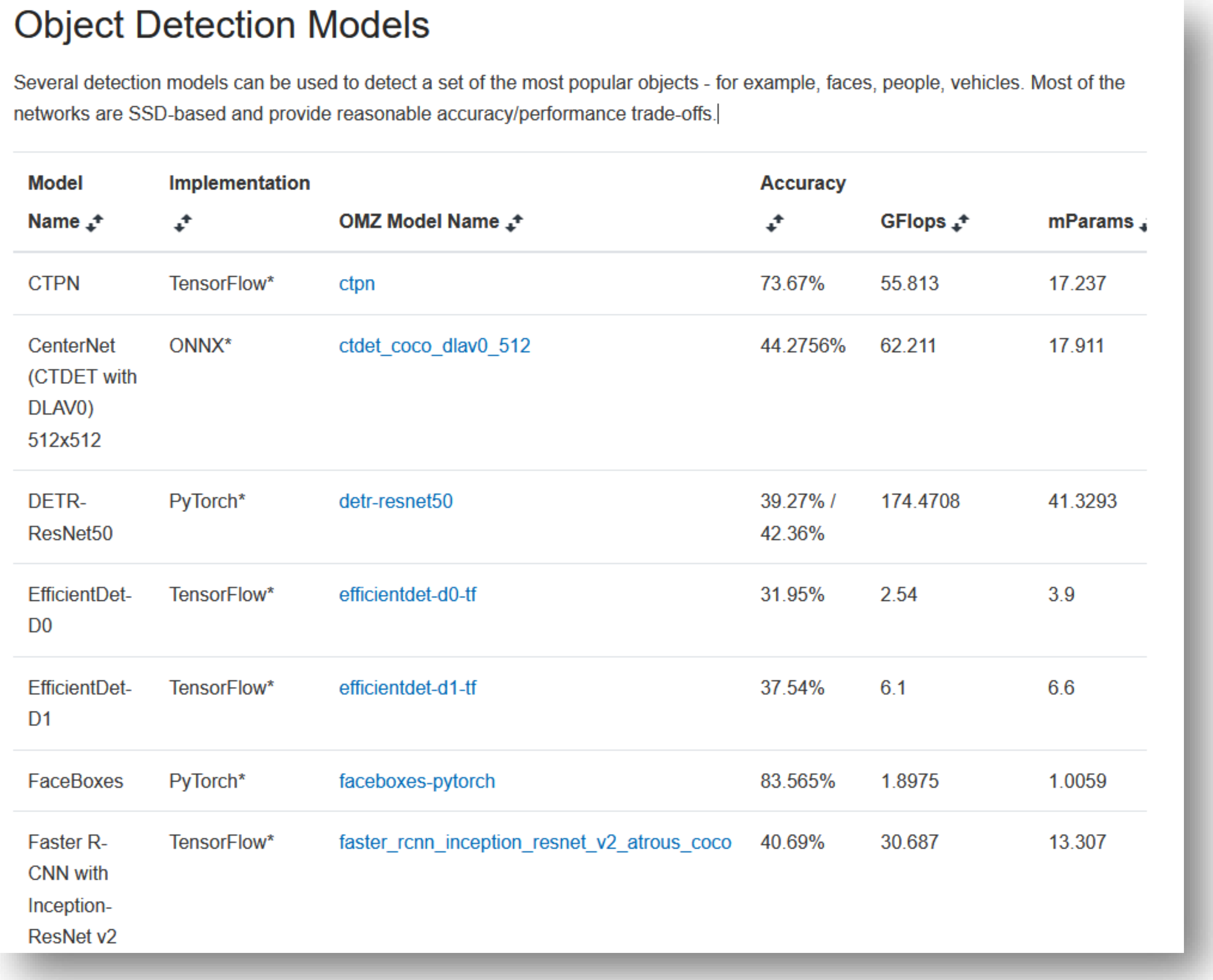
# 常見物件偵測模型

- CTPN
- CenterNet
- DETR
- EfficientDet
- Faster R-CNN
- SSD
- YOLO



物件偵測，找出場景中所有（數量不据）已知物件的邊界框位置、尺寸、類別，邊界框允許部份重疊。

# OpenVINO Open Model Zoo 範例



## Object Detection Models

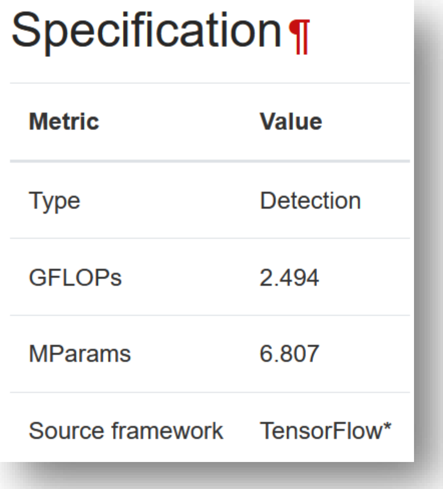
Several detection models can be used to detect a set of the most popular objects - for example, faces, people, vehicles. Most of the networks are SSD-based and provide reasonable accuracy/performance trade-offs.

| Model Name | Implementation | OMZ Model Name | Accuracy | GFlops | mParams |
|---|---|---|---|---|---|
| CTPN | TensorFlow* | ctpn | 73.67% | 55.813 | 17.237 |
| CenterNet (CTDET with DLAV0) 512x512 | ONNX* | ctdet_coco_dlav0_512 | 44.2756% | 62.211 | 17.911 |
| DETR-ResNet50 | PyTorch* | detr-resnet50 | 39.27% / 42.36% | 174.4708 | 41.3293 |
| EfficientDet-D0 | TensorFlow* | efficientdet-d0-tf | 31.95% | 2.54 | 3.9 |
| EfficientDet-D1 | TensorFlow* | efficientdet-d1-tf | 37.54% | 6.1 | 6.6 |
| FaceBoxes | PyTorch* | faceboxes-pytorch | 83.565% | 1.8975 | 1.0059 |
| Faster R-CNN with Inception-ResNet v2 | TensorFlow* | faster_rcnn_inception_resnet_v2_atrous_coco | 40.69% | 30.687 | 13.307 |

- DOCUMENTTATION\Legacy Features \ Open Model Zoo \ Overview of OpenVINO Toolkit Public Pre-Trained Models #Object Detection
- 提供30多種預訓練模型
- **Open Model Zoo 2023.0 版後雖不再更新維護但可使用。**

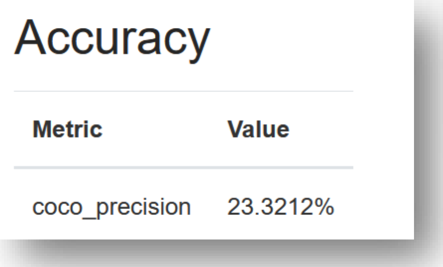
https://docs.openvino.ai/2024/omz_models_group_public.html#object-detection-models

# Open Model Zoo – ssd_mobilenet_v1



## Specification¶

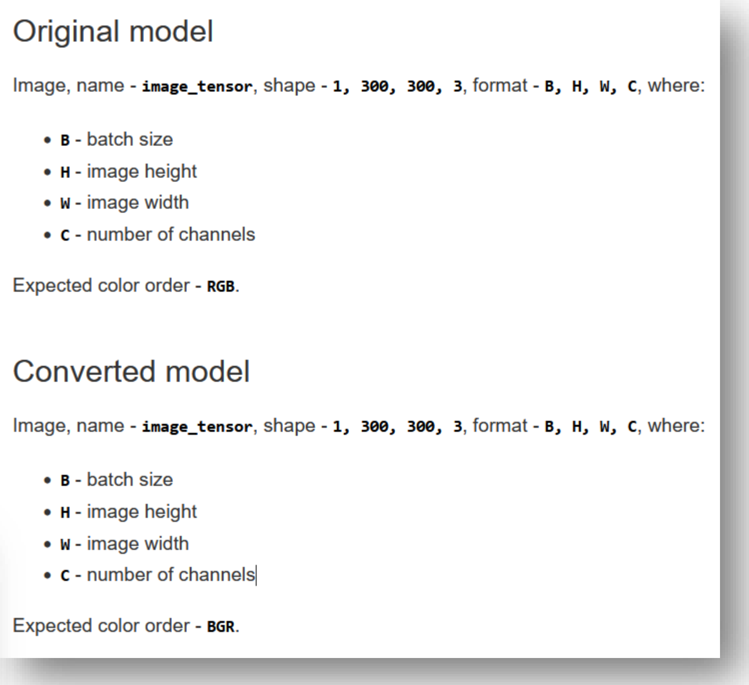
| Metric | Value |
|---|---|
| Type | Detection |
| GFLOPs | 2.494 |
| MParams | 6.807 |
| Source framework | TensorFlow* |



## Accuracy

| Metric | Value |
|---|---|
| coco_precision | 23.3212% |

## Input



### Original model

Image, name - `image_tensor`, shape - `1, 300, 300, 3`, format - `B, H, W, C`, where:

- `B` - batch size
- `H` - image height
- `W` - image width
- `C` - number of channels

Expected color order - `RGB`.

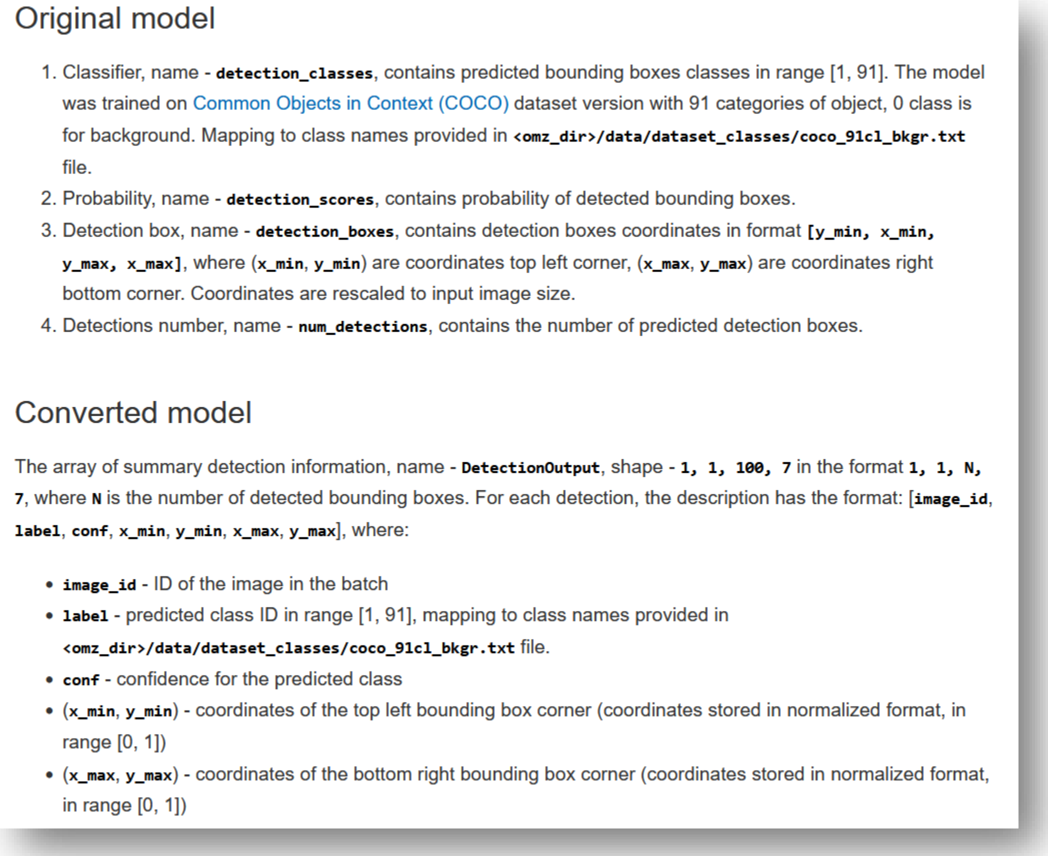
### Converted model

Image, name - `image_tensor`, shape - `1, 300, 300, 3`, format - `B, H, W, C`, where:

- `B` - batch size
- `H` - image height
- `W` - image width
- `C` - number of channels

Expected color order - `BGR`.

## Output



### Original model

1. Classifier, name - `detection_classes`, contains predicted bounding boxes classes in range [1, 91]. The model was trained on Common Objects in Context (COCO) dataset version with 91 categories of object, 0 class is for background. Mapping to class names provided in `<omz_dir>/data/dataset_classes/coco_91cl_bkgr.txt` file.
2. Probability, name - `detection_scores`, contains probability of detected bounding boxes.
3. Detection box, name - `detection_boxes`, contains detection boxes coordinates in format `[y_min, x_min, y_max, x_max]`, where (`x_min`, `y_min`) are coordinates top left corner, (`x_max`, `y_max`) are coordinates right bottom corner. Coordinates are rescaled to input image size.
4. Detections number, name - `num_detections`, contains the number of predicted detection boxes.

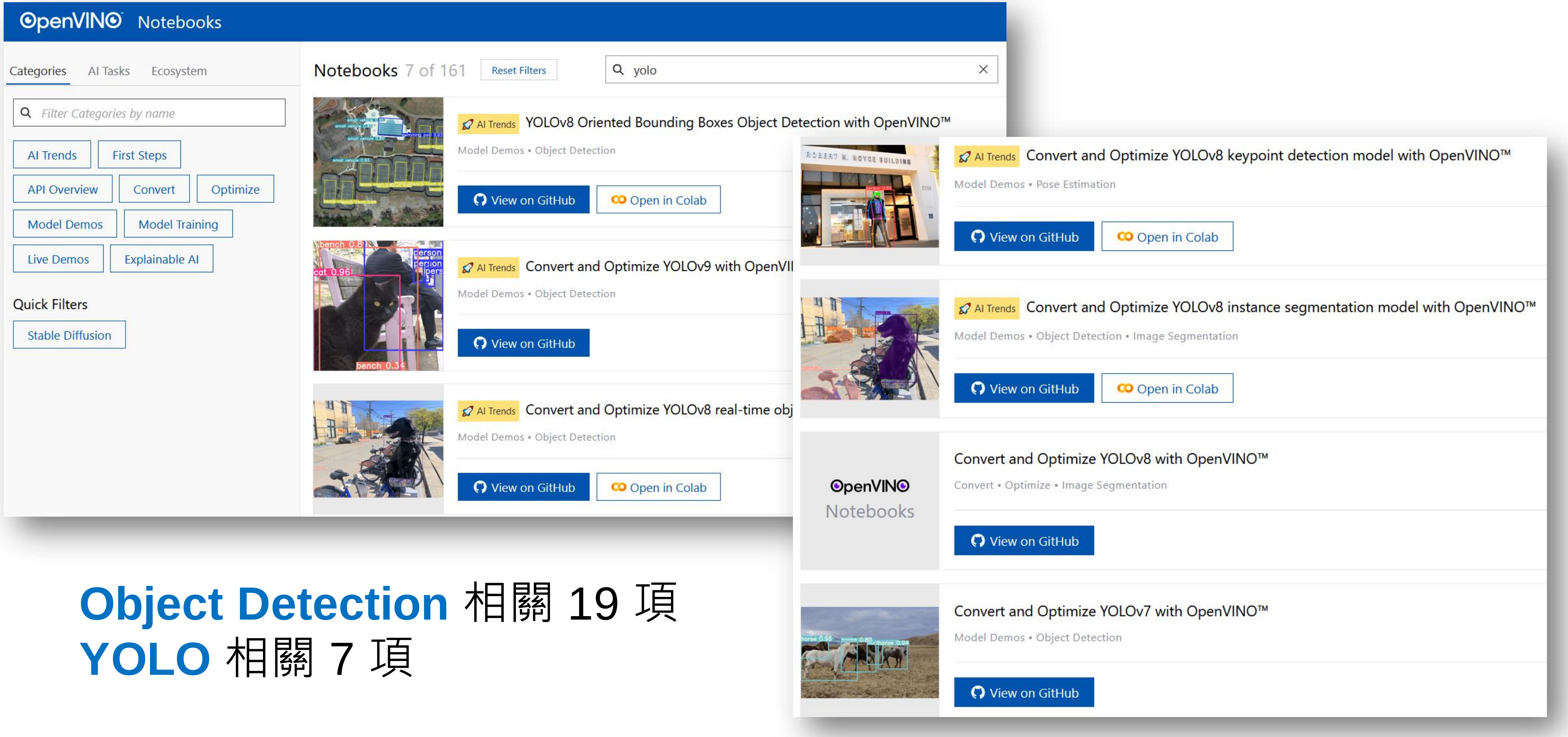
### Converted model

The array of summary detection information, name - `DetectionOutput`, shape - `1, 1, 100, 7` in the format `1, 1, N, 7`, where `N` is the number of detected bounding boxes. For each detection, the description has the format: [`image_id`, `label`, `conf`, `x_min`, `y_min`, `x_max`, `y_max`], where:

- `image_id` - ID of the image in the batch
- `label` - predicted class ID in range [1, 91], mapping to class names provided in `<omz_dir>/data/dataset_classes/coco_91cl_bkgr.txt` file.
- `conf` - confidence for the predicted class
- (`x_min`, `y_min`) - coordinates of the top left bounding box corner (coordinates stored in normalized format, in range [0, 1])
- (`x_max`, `y_max`) - coordinates of the bottom right bounding box corner (coordinates stored in normalized format, in range [0, 1])

資料來源：https://docs.openvino.ai/2024/omz_models_model_ssd_mobilenet_v1_coco.html

# OpenVINO Notebooks 範例



OpenVINO Notebooks

Categories | AI Tasks | Ecosystem

Filter Categories by name

AI Trends | First Steps | API Overview | Convert | Optimize | Model Demos | Model Training | Live Demos | Explainable AI

Quick Filters

Stable Diffusion

Notebooks 7 of 161 | Reset Filters | yolo

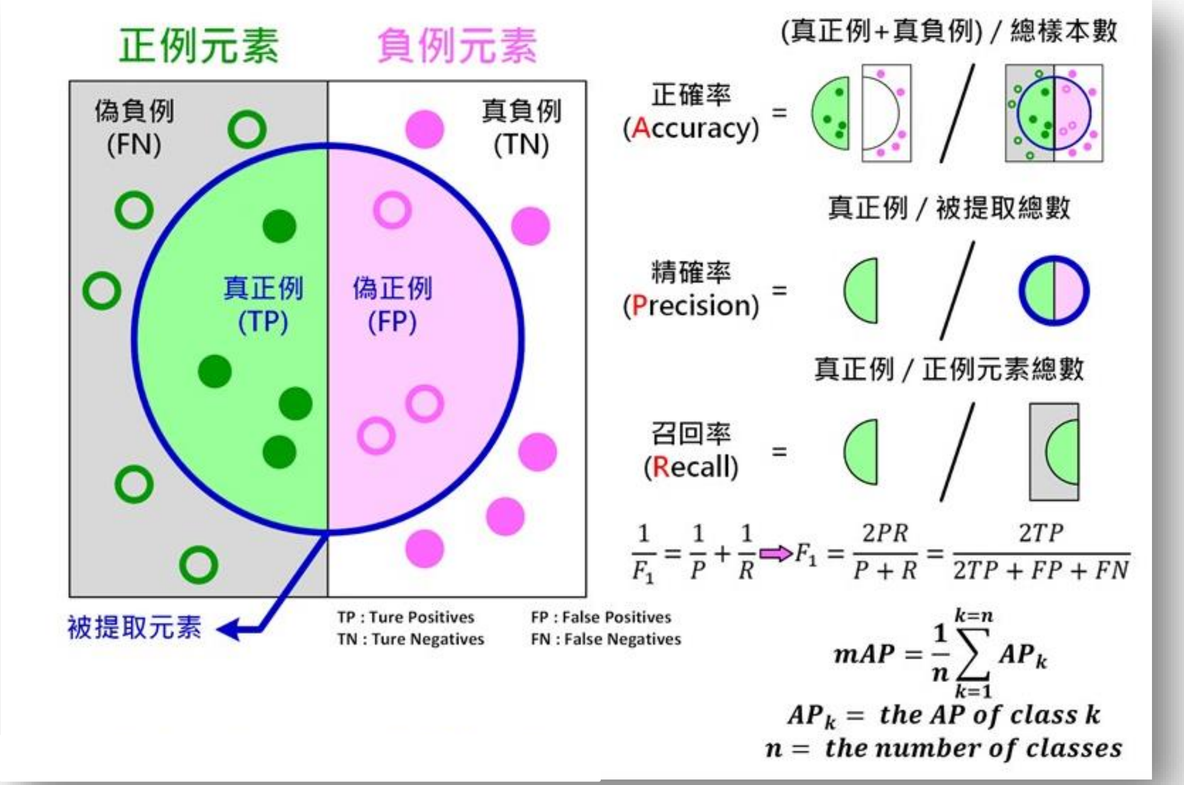
- AI Trends — YOLOv8 Oriented Bounding Boxes Object Detection with OpenVINO™ — Model Demos • Object Detection — View on GitHub | Open in Colab
- AI Trends — Convert and Optimize YOLOv9 with OpenVI — Model Demos • Object Detection — View on GitHub
- AI Trends — Convert and Optimize YOLOv8 real-time obj — Model Demos • Object Detection — View on GitHub | Open in Colab
- AI Trends — Convert and Optimize YOLOv8 keypoint detection model with OpenVINO™ — Model Demos • Pose Estimation — View on GitHub | Open in Colab
- AI Trends — Convert and Optimize YOLOv8 instance segmentation model with OpenVINO™ — Model Demos • Object Detection • Image Segmentation — View on GitHub | Open in Colab
- Convert and Optimize YOLOv8 with OpenVINO™ — Convert • Optimize • Image Segmentation — View on GitHub
- Convert and Optimize YOLOv7 with OpenVINO™ — Model Demos • Object Detection — View on GitHub

**Object Detection** 相關 19 項

**YOLO** 相關 7 項

https://openvinotoolkit.github.io/openvino_notebooks/

# 物件偵測評量

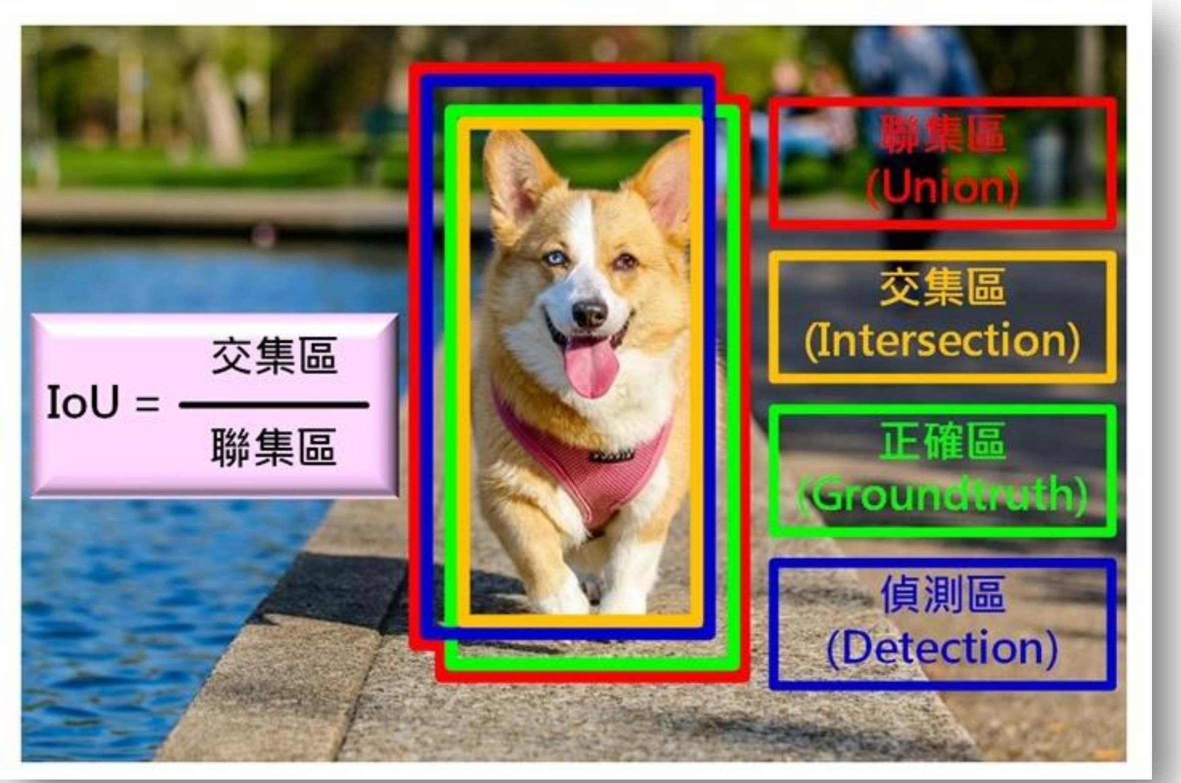


$$\frac{1}{F_1}=\frac{1}{P}+\frac{1}{R}\Rightarrow F_1=\frac{2PR}{P+R}=\frac{2TP}{2TP+FP+FN}$$

$$mAP=\frac{1}{n}\sum_{k=1}^{k=n}AP_k$$

$AP_k =$ the AP of class k

$n =$ the number of classes

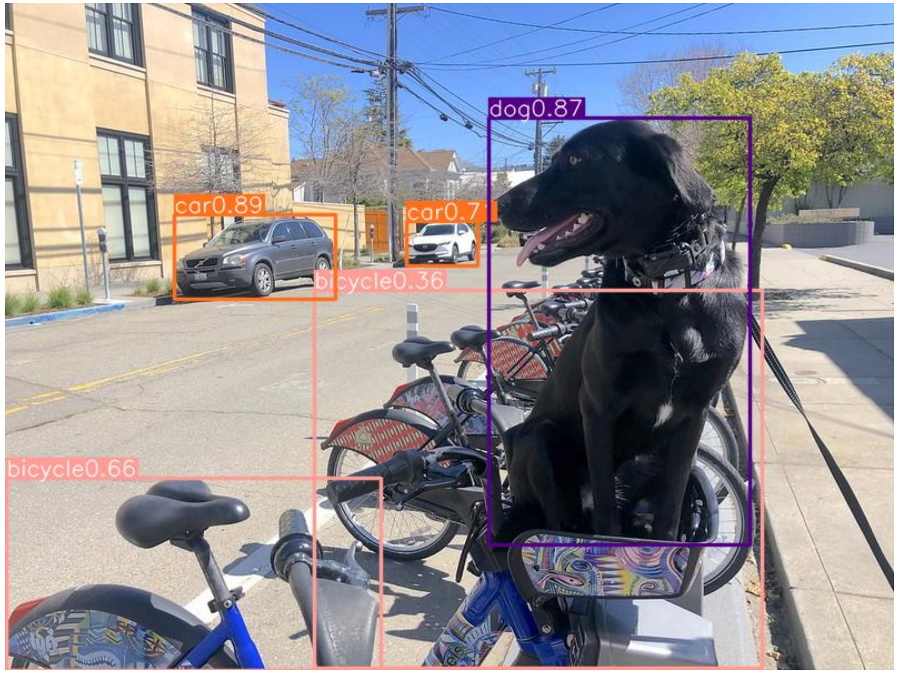


$$\text{IoU}=\frac{交集區}{聯集區}$$

影像來源：https://omnixri.blogspot.com/p/ntust-edge-ai-ch4-1.html

# 實作範例1 – YOLOv8物件偵測 (Colab)

## Convert and Optimize YOLOv8 real-time object detection with OpenVINO™



### 程式導讀

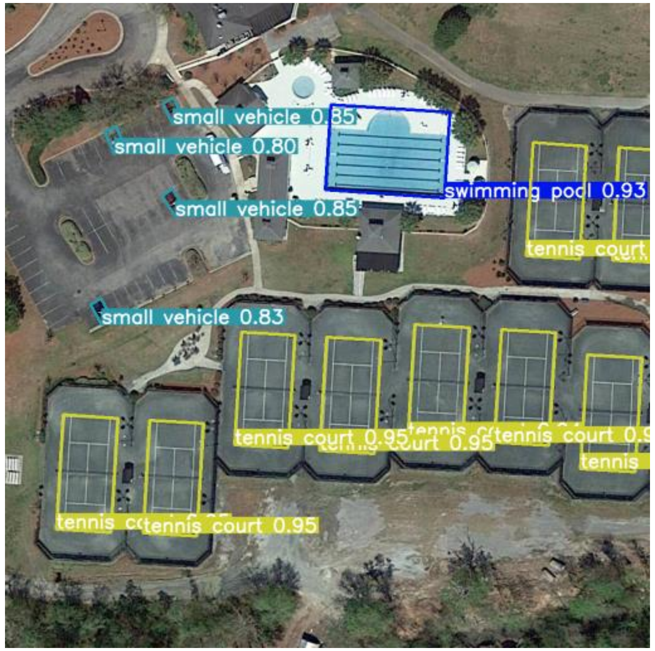
（原230-yolov8-object-detection）

- Prepare the PyTorch model.
- Download and prepare a dataset.
- Validate the original model.
- Convert the PyTorch model to OpenVINO IR.
- Validate the converted model.
- Prepare and run optimization pipeline.
- Compare performance of the FP32 and quantized models.
- Compare accuracy of the FP32 and quantized models.
- Other optimization possibilities with OpenVINO api
- Live demo

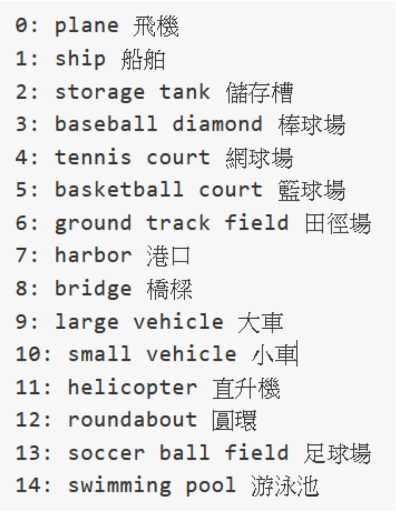
https://colab.research.google.com/github/openvinotoolkit/openvino_notebooks/blob/latest/notebooks/yolov8-optimization/yolov8-object-detection.ipynb

# 實作範例2 – YOLOv8 OBB (Colab)

## YOLOv8 Oriented Bounding Boxes Object Detection with OpenVINO™



空拍影像及旋轉物件資料集（DOTA）



```
0: plane 飛機
1: ship 船舶
2: storage tank 儲存槽
3: baseball diamond 棒球場
4: tennis court 網球場
5: basketball court 籃球場
6: ground track field 田徑場
7: harbor 港口
8: bridge 橋樑
9: large vehicle 大車
10: small vehicle 小車
11: helicopter 直升機
12: roundabout 圓環
13: soccer ball field 足球場
14: swimming pool 游泳池
```

https://captain-whu.github.io/DOTA/

### 程式導讀

**(Notebooks 2023.1 ~ Latest)**

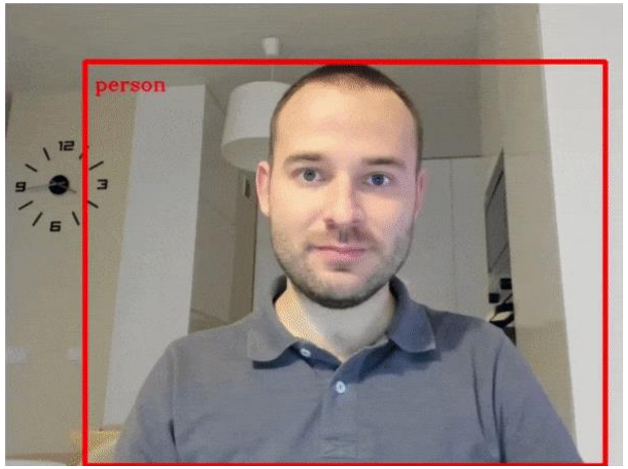
- Prerequisites
- Get PyTorch model
- Prepare dataset and dataloader
- Run inference
- Convert PyTorch model to OpenVINO IR
  - Select inference device
  - Compile model
  - Prepare the model for inference
  - Run inference
- Quantization
- Compare inference time and model sizes

https://colab.research.google.com/github/openvinotoolkit/openvino_notebooks/blob/latest/notebooks/yolov8-optimization/yolov8-obb.ipynb

# 實作範例3 – 即時影像辨識 (Colab)

## Live Object Detection with OpenVINO™ （原401-Object-Detection-Webcam）

1. 本程式使用微軟COCO資料集（80類）預訓練之SSDLite MobileNetV2模型。
2. device 預設為 AUTO, 本地端時亦可切換至 iGPU / NPU Colab 時只能選 CPU / AUTO。
3. 本程式支援使用 webcam 或 mp4 影片檔



**video_file = "https://storage.openvinotoolkit.org/repositories/openvino_notebooks/data/data/video/Coco%20Walking%20in%20Berkeley.mp4"**

**run_object_detection(source=video_file, flip=False, use_popup=False)**

ps. source=**0** 時，表示使用電腦上第一組網路攝影機。

若發生顯示會閃爍時，
將use_popup設為True

**https://colab.research.google.com/github/openvinotoolkit/openvino_notebooks/blob/latest/notebooks/object-detection-webcam/object-detection.ipynb**

# 參考文獻

- 許哲豪，臺灣科技大學資訊工程系「人工智慧與邊緣運算實務」（2021~2023）

https://omnixri.blogspot.com/p/ntust-edge-ai.html

- pyimagesearch - Introduction to the YOLO Family

https://pyimagesearch.com/2022/04/04/introduction-to-the-yolo-family/

- 許哲豪，【vMaker Edge AI專欄 #16】AIPC開箱實測 ─ Yolov8斜物件偵測

https://omnixri.blogspot.com/2024/04/vmaker-edge-ai-16aipc-yolov8.html

# 延伸閱讀

- 許哲豪，如何以Google Colab及Yolov4-tiny來訓練自定義資料集─以狗臉、貓臉、人臉偵測為例

https://omnixri.blogspot.com/2021/05/google-colabyolov4-tiny.html

- 李謦伊，謦伊的閱讀筆記（YOLO）

https://medium.com/ching-i/tagged/yolo

歐尼克斯實境互動

歐尼克斯實境互動工作室
(OmniXRI Studio)
許哲豪 (Jack Hsu )

Facebook : Jack Omnixri
FB社團： Edge AI Taiwan邊緣智能交流區
電子信箱：omnixri@gmail.com
部落格：https://omnixri.blogspot.tw
開　源：https://github.com/OmniXRI
YOUTUBE 直播： https://www.youtube.com/@omnixri1784/streams