SimCenter NHERI
Center for Computational Modeling and Simulation

2021 AI Bootcamp

Object Detection

Barbaros Cetiner

NHERI SimCenter
University of California, Los Angeles

NSF Award: CMMI 1612843

Outline

- Why use object detection?
- Object detection algorithms: past to present
- Datasets for object detection
- A simple framework for model development

Why Use Object Detection?

What is Object Detection?

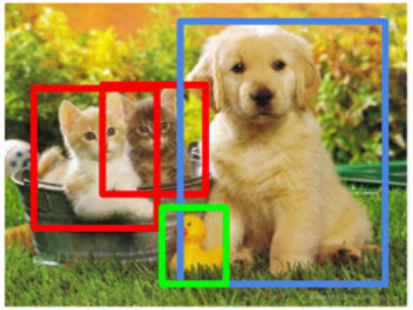
Image classification: Classifies an image into a certain category

Object detection: Identify the location(s) of objects in an image

Classification

CAT

Object Detection



CAT, DOG, DUCK

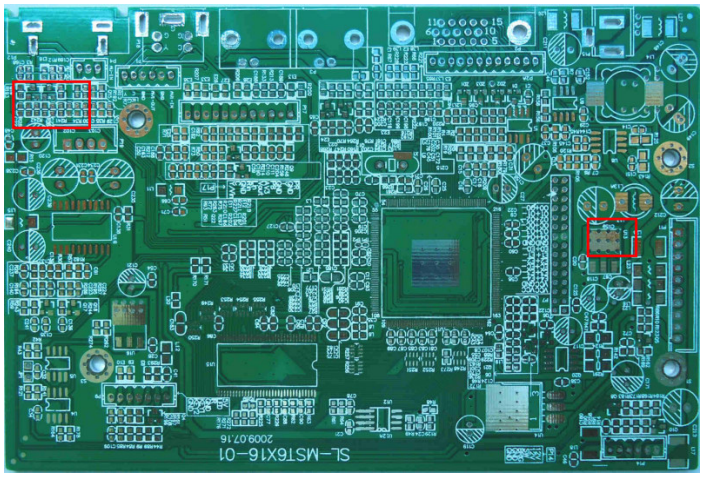
Why Use Object Detection?

Object detection comes in handy when

1. Multiple instances of objects of interest exist in input images
2. (Rough) location information is required

OR

For image classification tasks where the object or the characteristics that are required to perform categorization are not predominant



So, what is object detection and when should you use it?

In simple terms: Object detection is classification performed at sub-image level

It is great for: 1) Locating objects in multi-object images
2) Classifying images with sparse/weak features

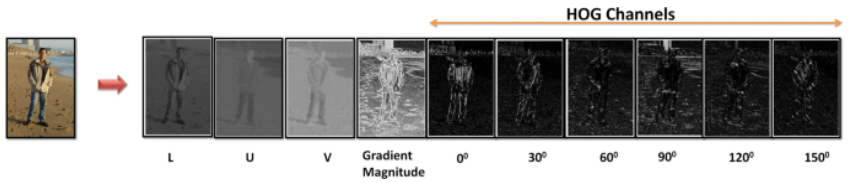
Object Detection Algorithms: Past to Present

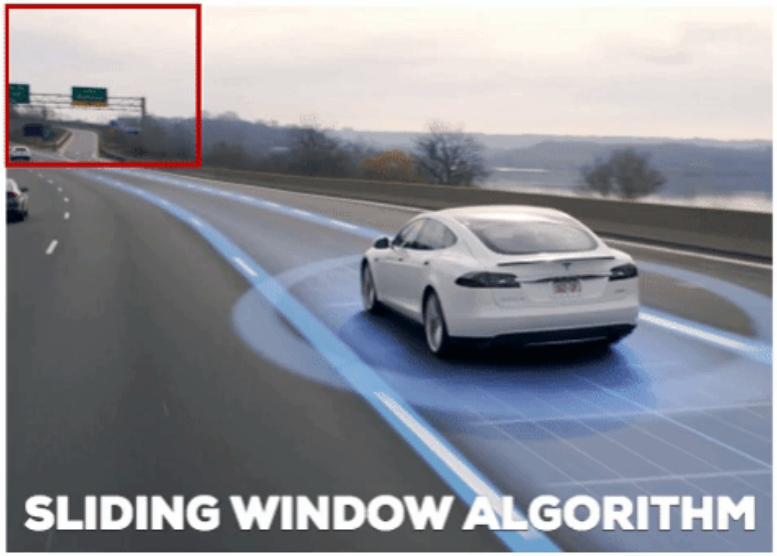
Object Detection with Sliding Windows

Early object detectors based on histogram of gradients (HOG) features, and (or) aggregate channel features (ACF) utilized sliding windows

How are ACF/HOG different than deep learning methods?

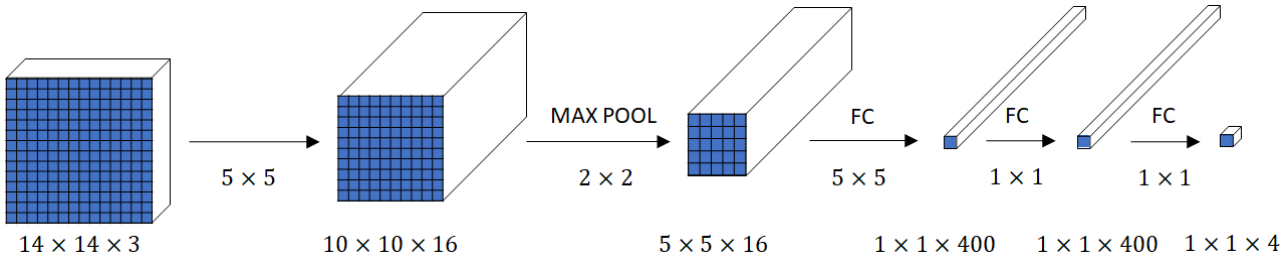
Both ACF and HOG are largely linear methods with simpler (less computationally intensive) architectures



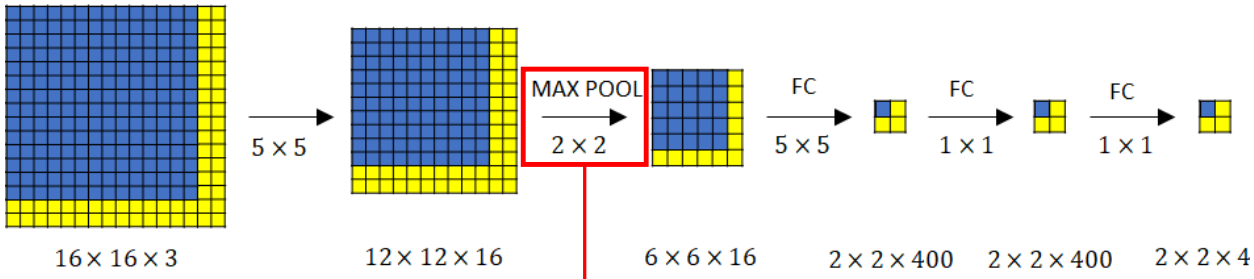


Convolutional Sliding Windows

Classifier operating on a 14x14 image:



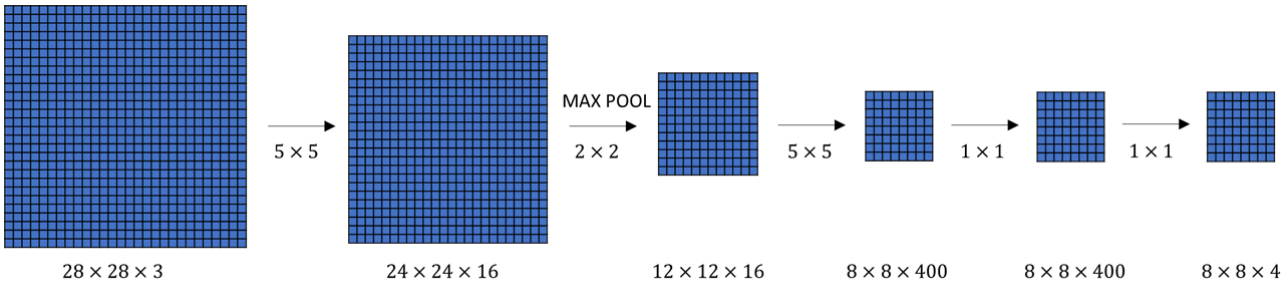
Sliding window classifier operating on a 16x16 image at a stride of 2:



ide size determined by the size of the MaxPool

Convolutional Sliding Windows: Issues

Sliding window classifier operating on a 28x28 image at a stride of 2:



For simplicity, only some of the sliding windows are shown here

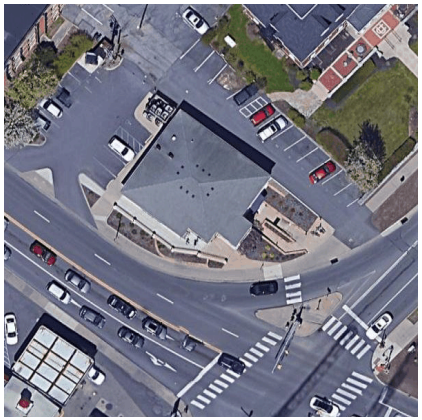
Do you see any problems with this approach?

Convolutional Sliding Windows: Issues

Issue 1

Coupling deep-learning based classifiers with sliding window-based detection is **computationally expensive** at typical image resolutions

A higher resolution image better shows object features



600x600 resolution

Downsampling

28x28 resolution

Convolutional Sliding Windows: Issues

Issue 2

Sliding windows ***cannot parse out overlapping objects***

Issue 3

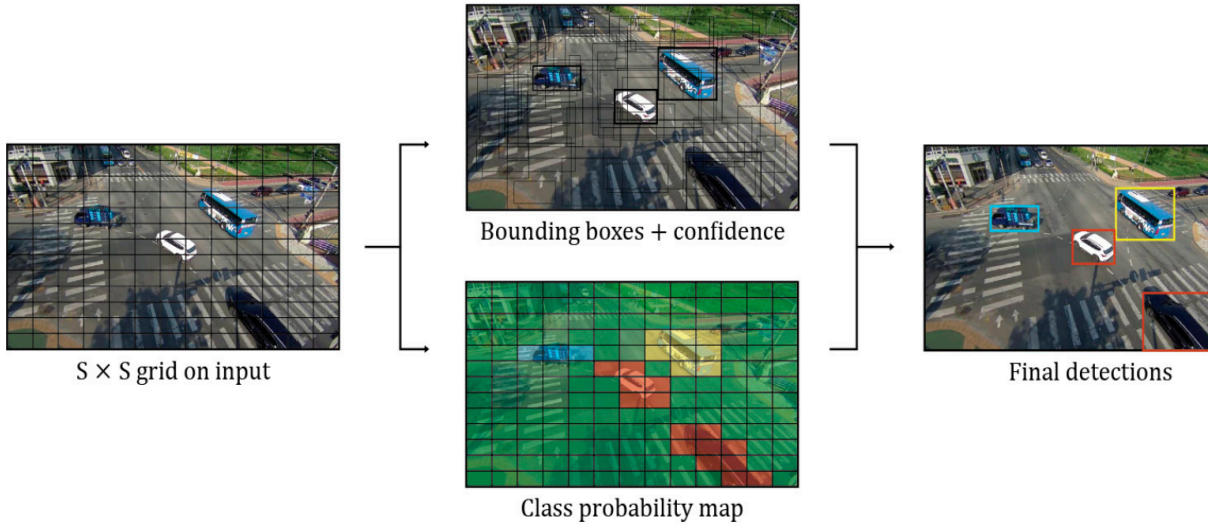
Poor bounding box fit around detected object

What are the downsides of using convolutional sliding windows methods for object detection?

1) They are **computationally expensive**
2) They **cannot parse out overlapping objects**
3) They achieve **poor bounding box fit around detected objects**

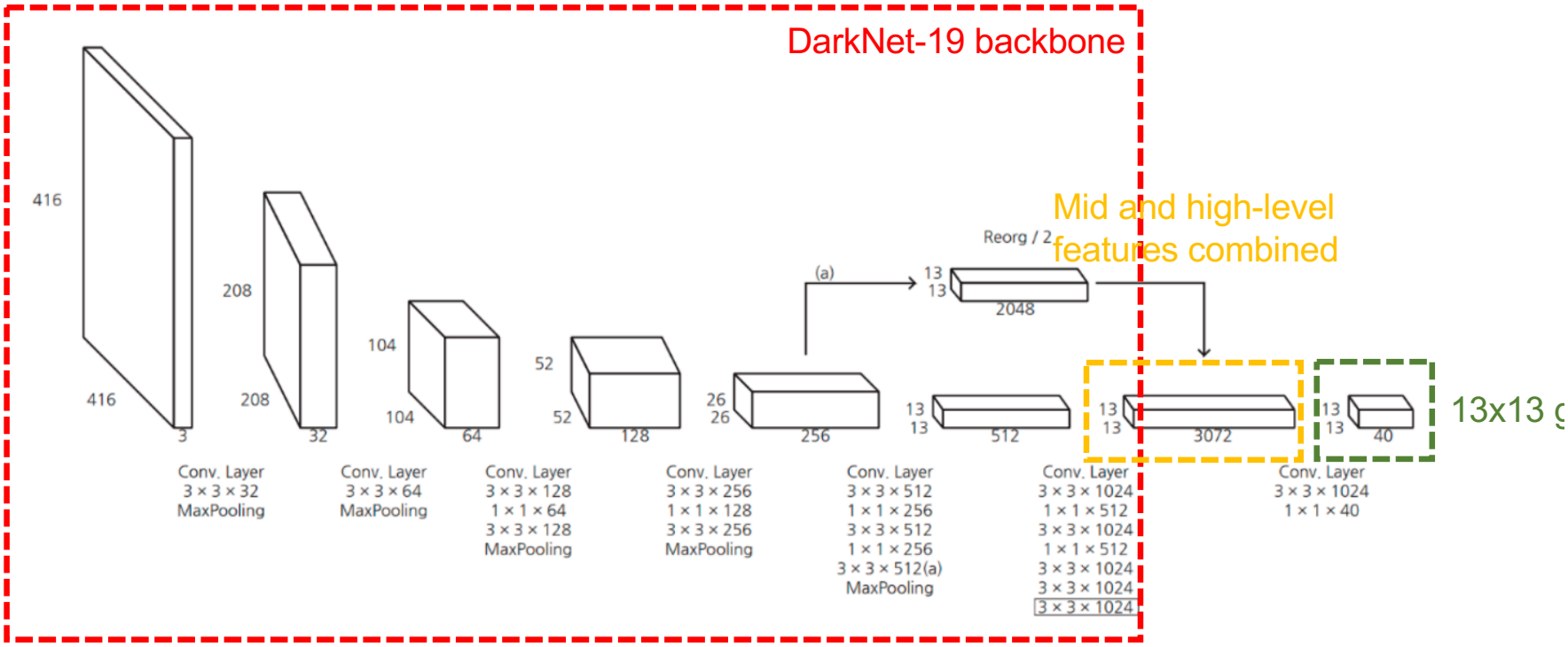
Object Detection with Anchor Boxes

1) Image is divided into a grid (size of the grid depends on input image size)
2) For each grid cell, bounding box predictions of different aspect ratios (anchor boxes) and their objectness (confidence) score & class probabilities are calculated
3) For each object detected, weaker (non-max) boxes are removed (suppressed)

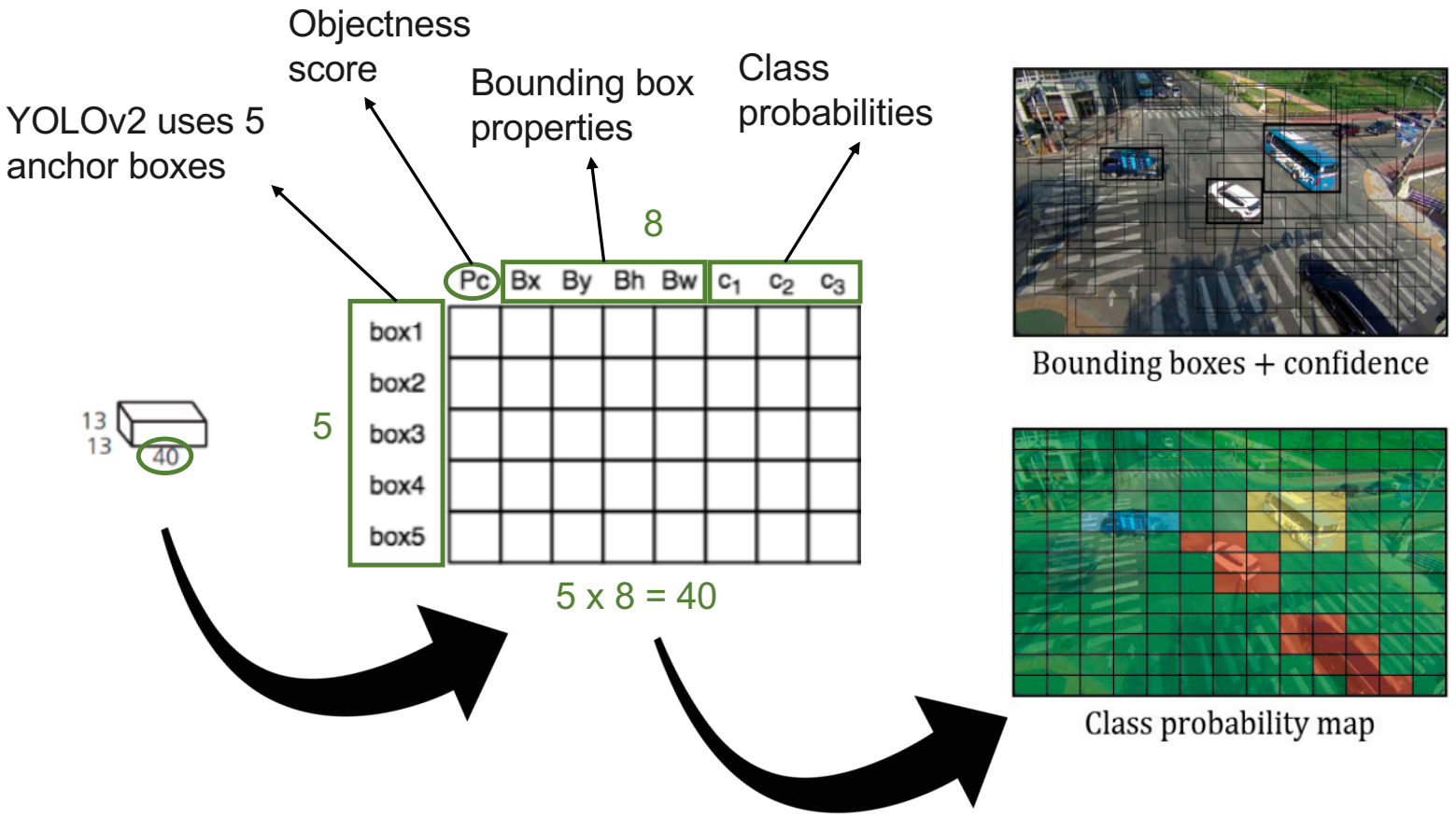


Object Detection with Anchor Boxes

YOLO v2 architecture



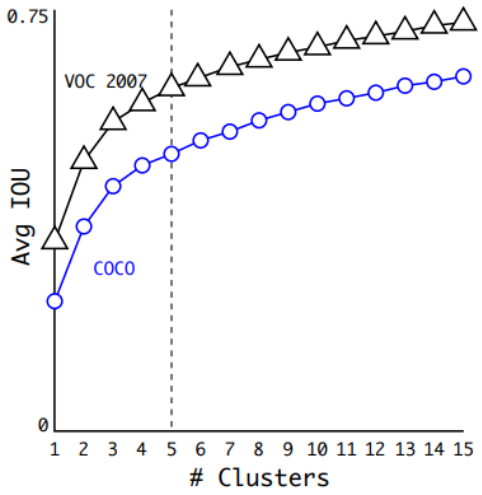
Object Detection with Anchor Boxes

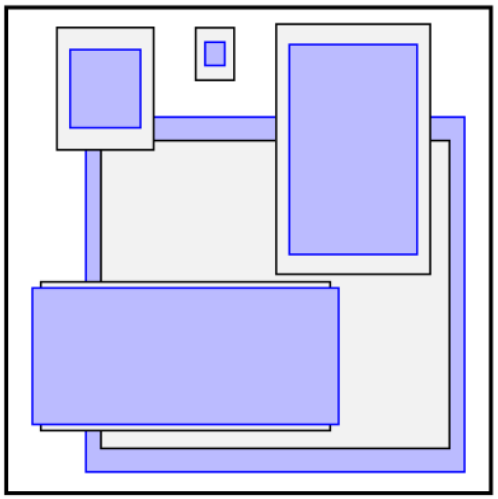


Object Detection with Anchor Boxes

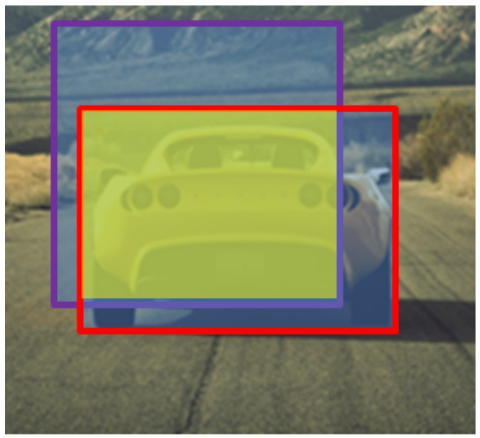
The optimal number of anchor boxes for YOLOv2 were determined by

- K-means clustering of bounding box dimensions on COCO and VOC datasets
- Considering computational efficiency, finding an optimum between # clusters & average detection accuracy (IoU)





Intersection over Union (IoU)



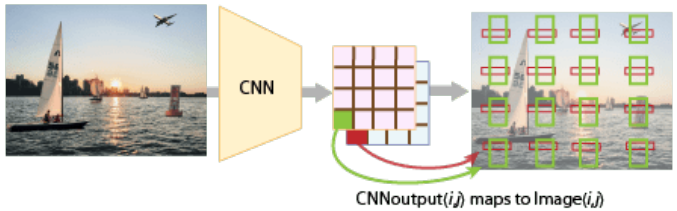
Intersection over Union (IoU):

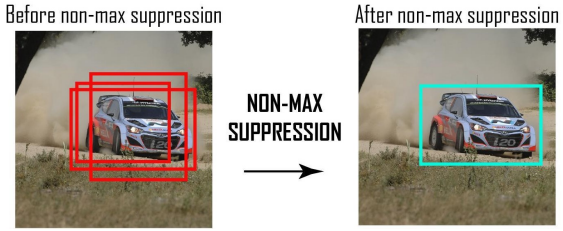
$$= \frac{\text{size of } \square}{\text{size of } \square}$$

Correct if IoU≥0.5 (loose fit)

How does detection with anchor boxes (single-stage detection) address the three issues identified for sliding windows methods?

1) Performance issue is addressed ***by using a fixed grid***
2) Overlapping objects are detected ***by using anchor boxes and non-max suppression***
3) Better bounding box fit is achieved ***by using anchor boxes***



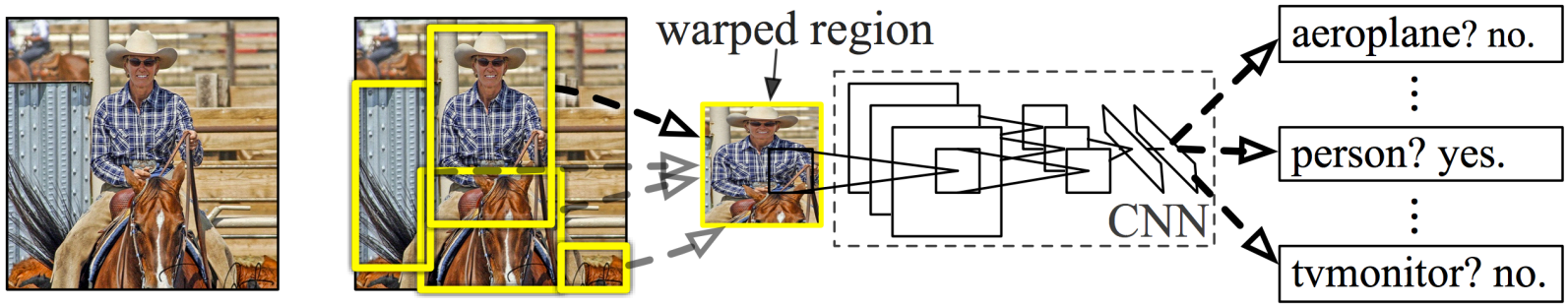


Object Detection with Region Proposals

Two-step process. First step is identifying image regions with similar attributes

Object Detection with Region Proposals

R-CNN: An early region-proposals-based detection algorithm



Object Detection with Region Proposals

R-CNN: Propose regions. Classify proposed regions one at a time.

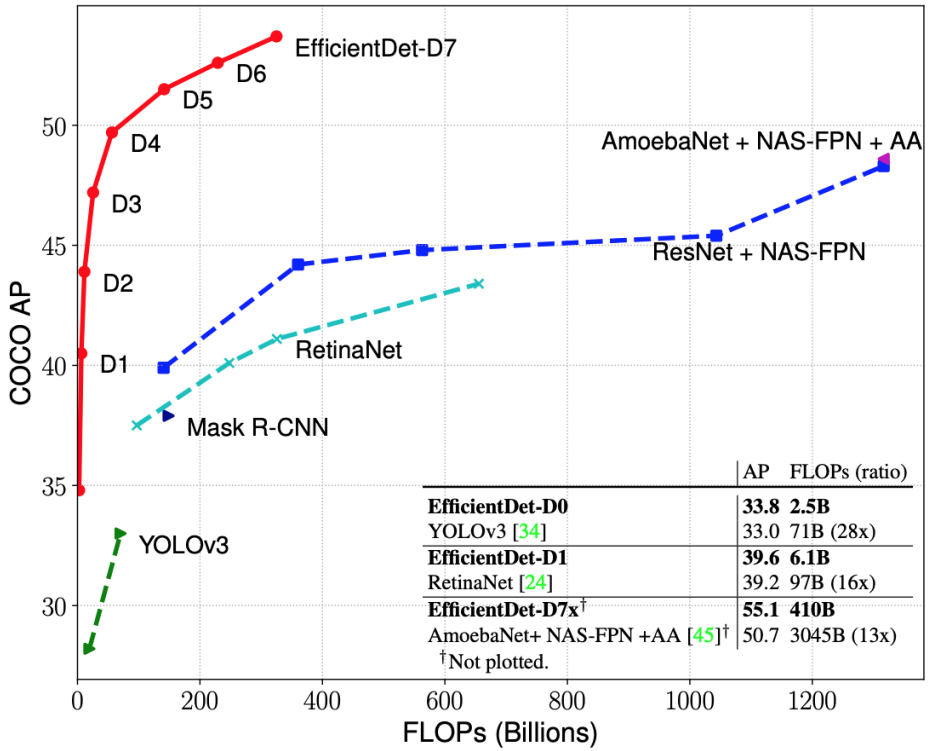
Fast R-CNN: Propose regions. Use convolution implementation of sliding windows to classify all the proposed regions

Faster R-CNN: Use convolutional network to propose regions.

Why are detection methods using region proposals less popular?

They are ***more computationally expensive.*** They are ***also less accurate than latest single-stage detection methods***

SOTA Object Detection: EfficientDet

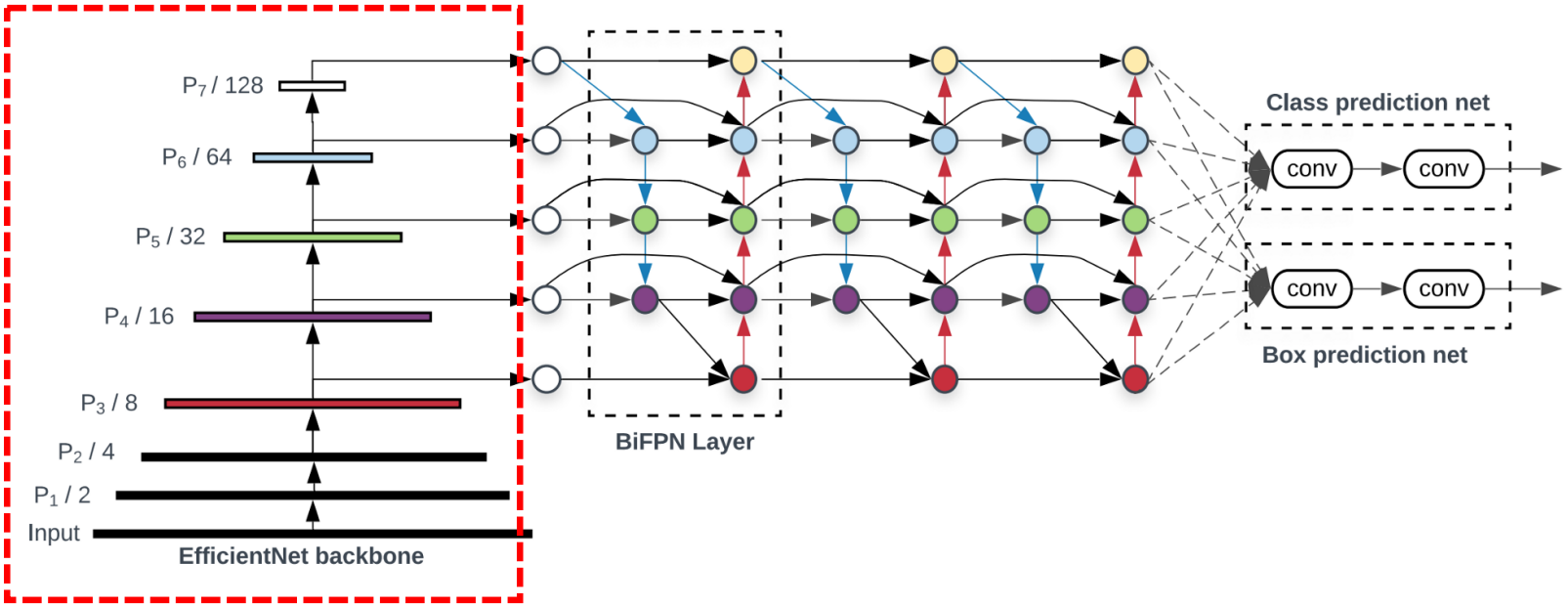


	AP	FLOPs (ratio)
EfficientDet-D0	**33.8**	**2.5B**
YOLOv3 [34]	33.0	71B (28x)
EfficientDet-D1	**39.6**	**6.1B**
RetinaNet [24]	39.2	97B (16x)
EfficientDet-D7x†	**55.1**	**410B**
AmoebaNet+ NAS-FPN +AA [45]†	50.7	3045B (13x)

†Not plotted.

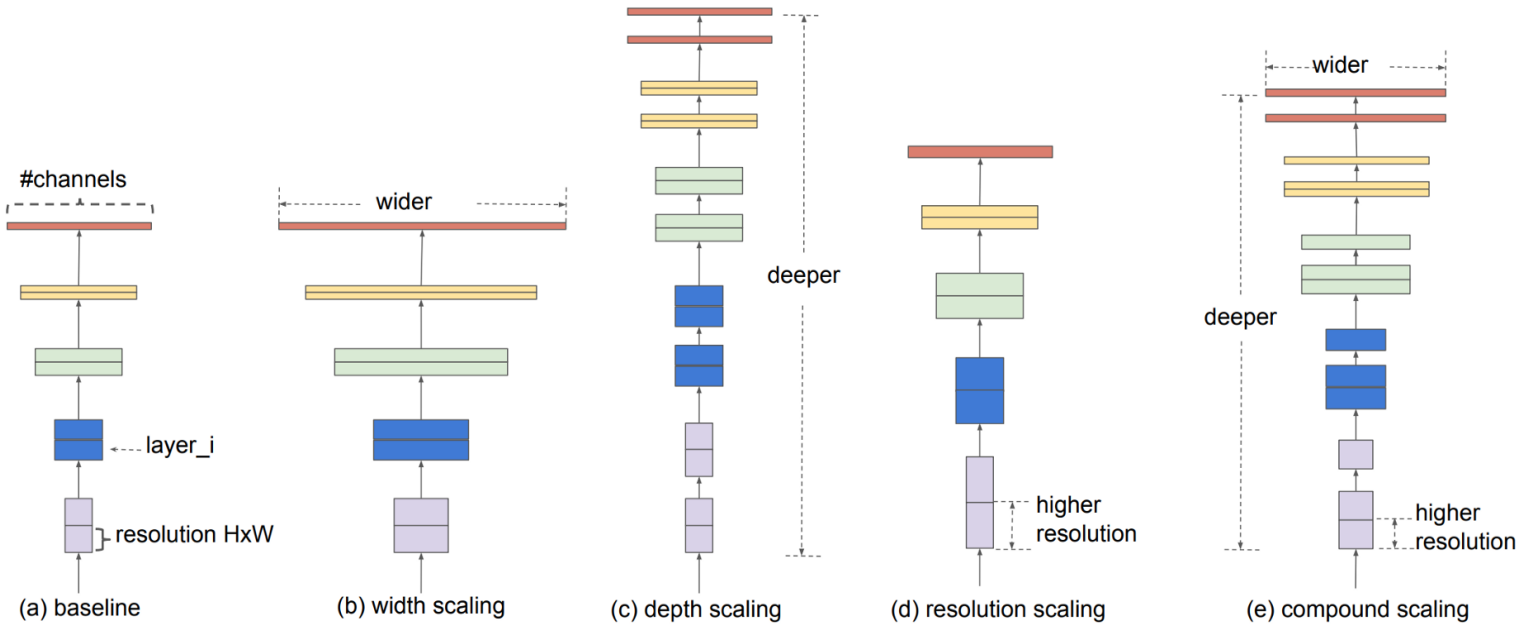
SOTA Object Detection: EfficientDet

Better backbone: In general, EfficientNet models achieve both higher accuracy and better efficiency over existing CNNs



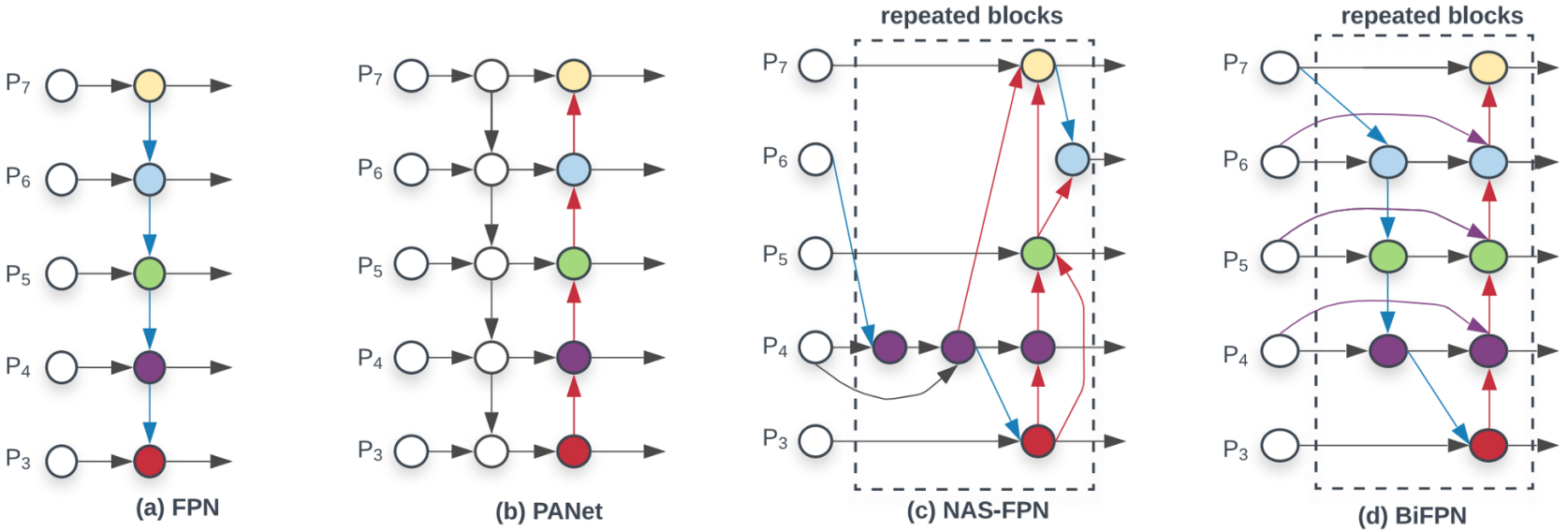
SOTA Object Detection: EfficientDet

Compound Scaling: Balancing all dimensions of the network—width, depth, and image resolution—against the available resources best improves overall performance

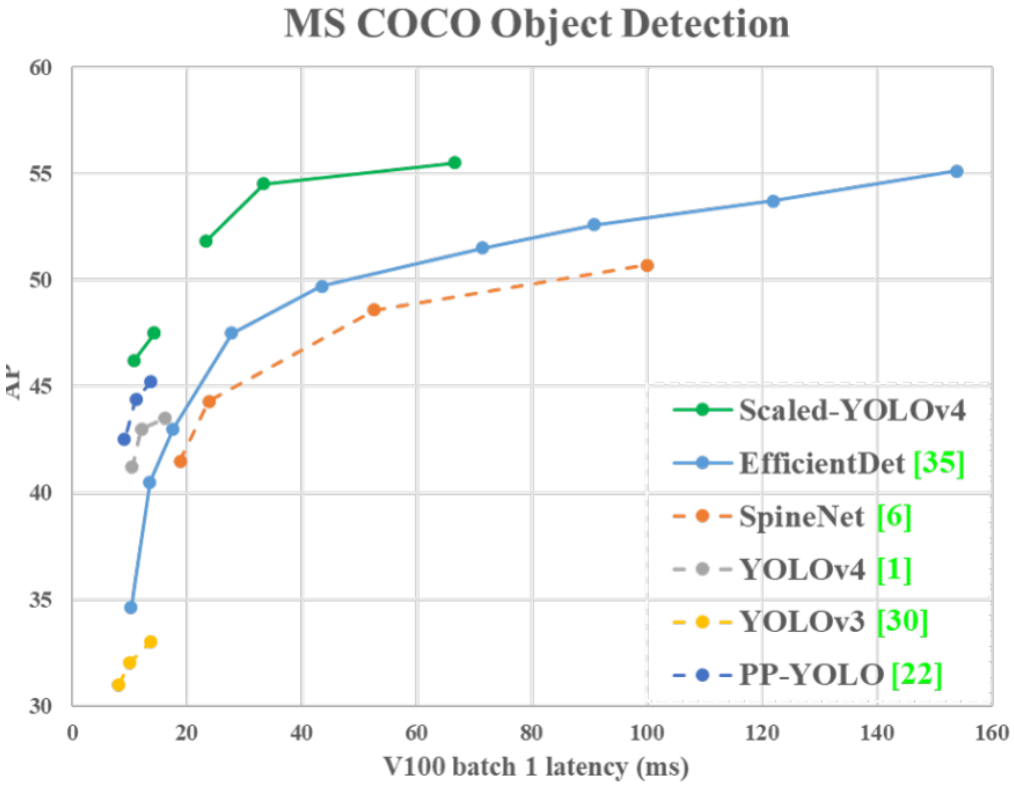


SOTA Object Detection: EfficientDet

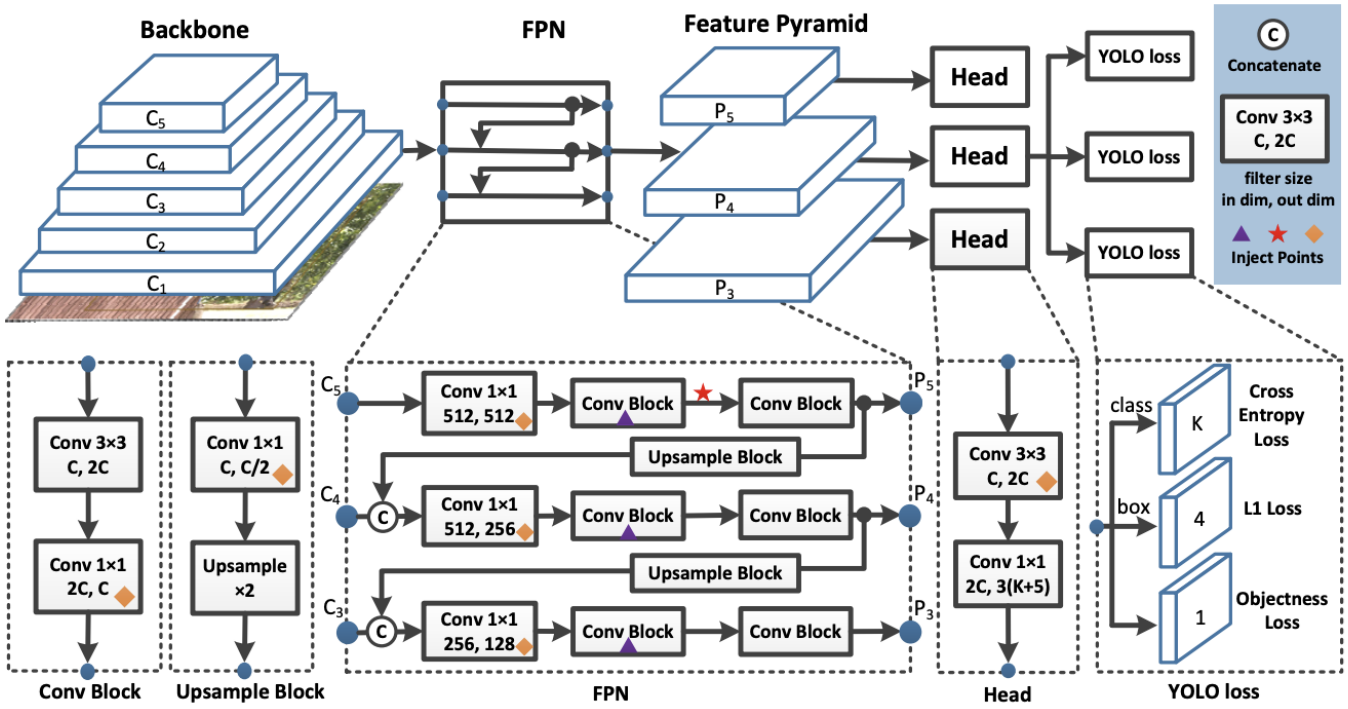
BiFPN Layers: 1) enable bottom-up and both top-down information flow while using regular and efficient connections, 2) add an additional weight for each input feature allowing the network to learn the importance of features at different resolutions



SOTA Object Detection: Scaled YOLOv4



SOTA Object Detection: Scaled YOLOv4



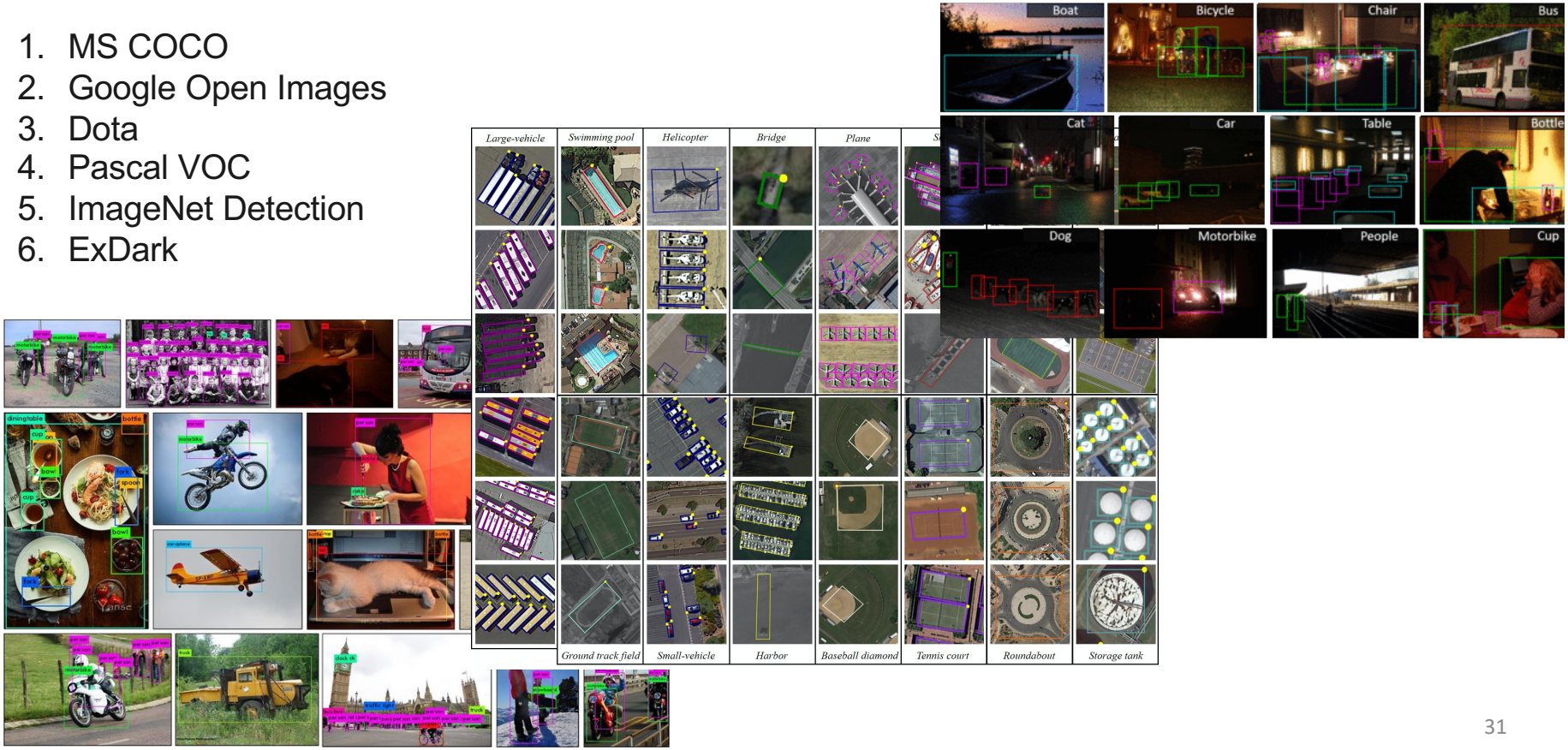
On a limited GPU memory (e.g., 8 GB), would you prefer to use Scaled YOLOv4 or EfficientDet for object detection?

If computational resources are limited, you will likely attain better performance from Scaled YOLOv4

Datasets for Object Detection

Popular Object Detection Datasets

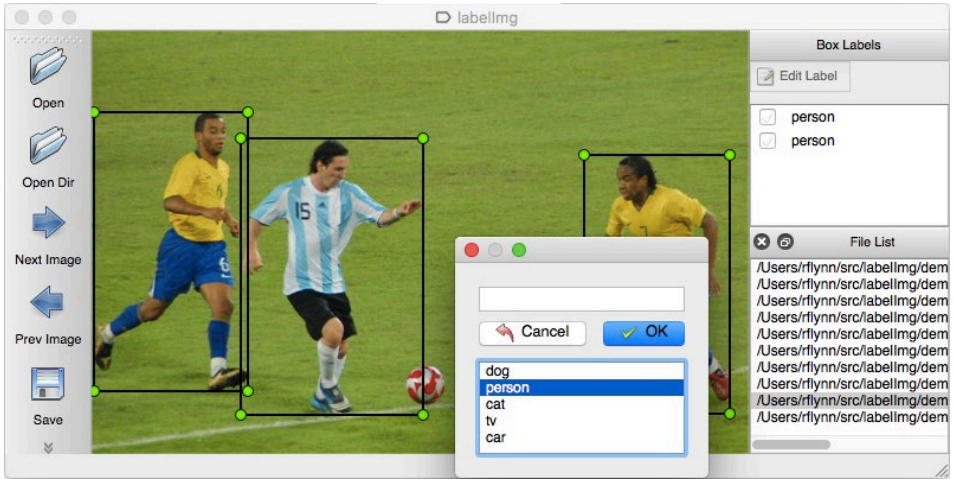
1. MS COCO
2. Google Open Images
3. Dota
4. Pascal VOC
5. ImageNet Detection
6. ExDark



Custom Datasets

Many tools available. I prefer the open-source tools LabelImg and CVAT

LabelImg



CVAT

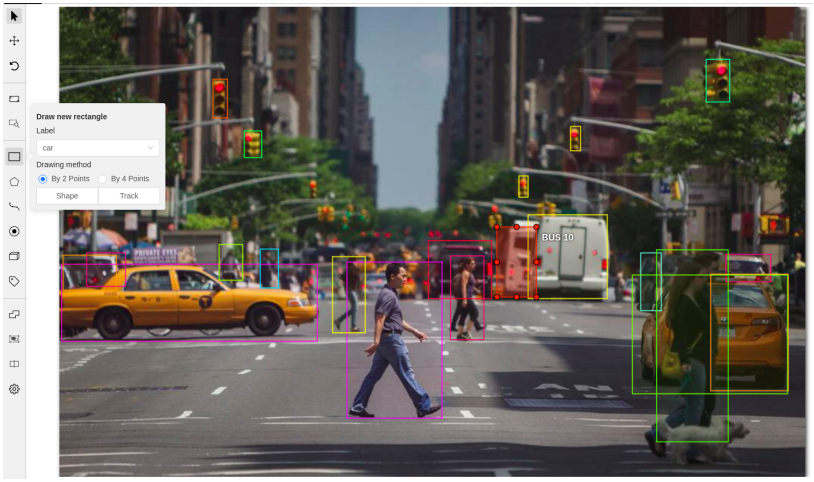


Image Augmentation

1. Augmentation is essential to introducing variability in the dataset, which helps models generalize better.
2. Also, the size of a dataset can be increased with augmentation (important if dataset size is small)

Popular techniques:

- Contrast/brightness/saturation adjustment
- Lighting noise
- Flip
- Rotate
- Random crop
- Cutout
- Mixup

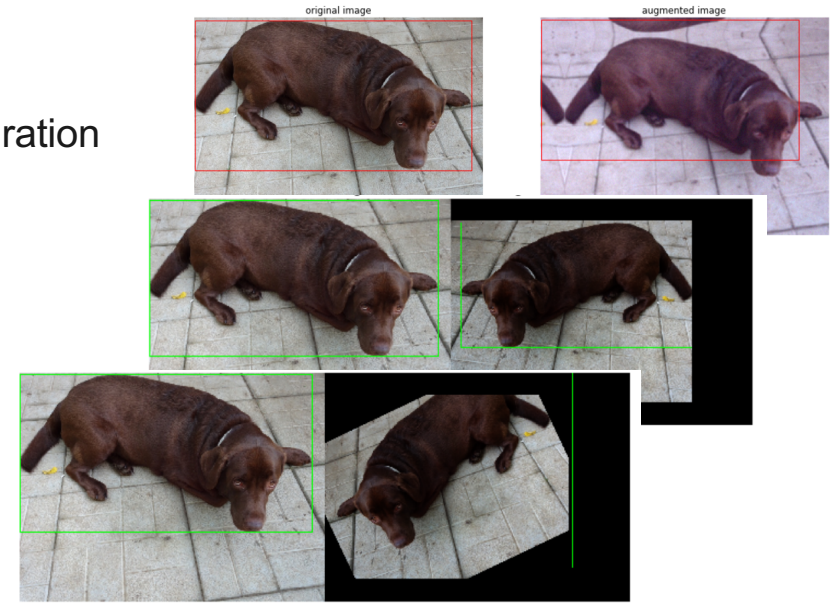


Image Augmentation

1. Augmentation is essential to introducing variability in the dataset, which helps models generalize better.
2. Also, the size of a dataset can be increased with augmentation (important if dataset size is small)

Popular techniques:

- Contrast/brightness/saturation adjustment
- Lighting noise
- Flip
- Rotate
- Random crop
- Cutout
- Mixup

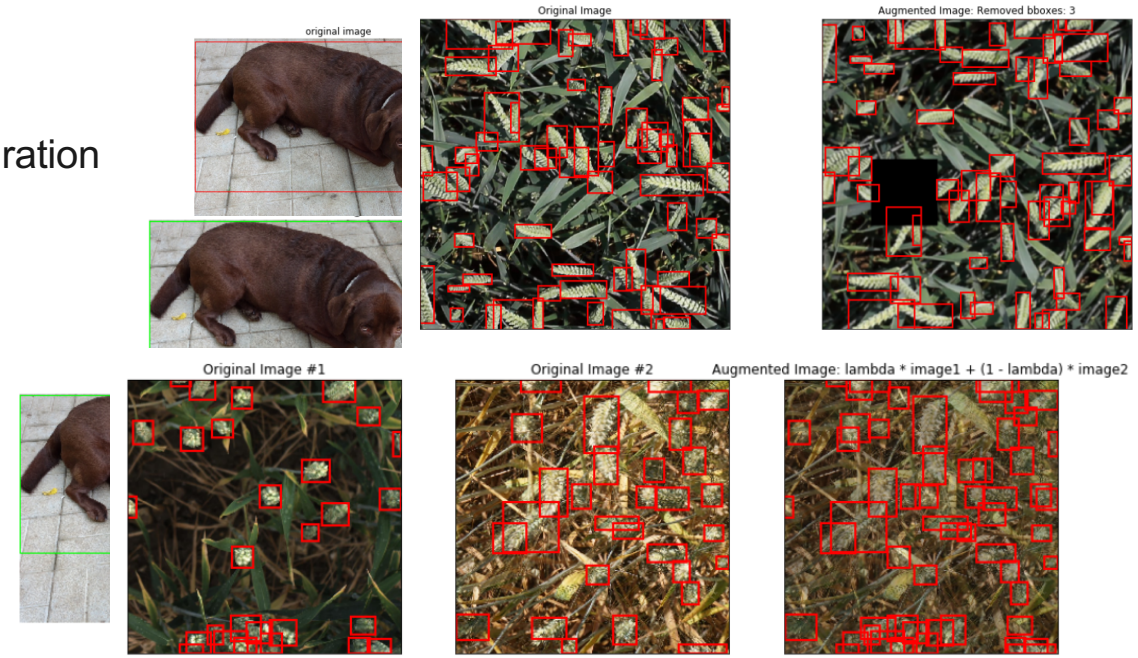


Image Augmentation using Python

Available augmentation libraries:

- Pillow
- Albumentations
- Augmentor
- imgaug

A Simple Framework for Model Development

Simple Framework for Model Development

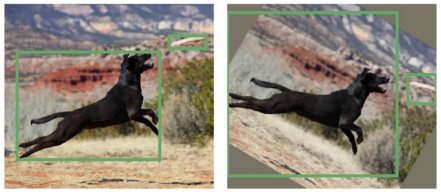


Image labeling → Image Preprocessing → Label preprocessing → Model training → Post-processing