YOLO – You Only Look Once

Bahan Kuliah IF4073 Pemrosesan Citra Digital

Oleh: Dr. Ir. Rinaldi, M.T

Teknik Informatika ITB

Referensi untuk PPT ini

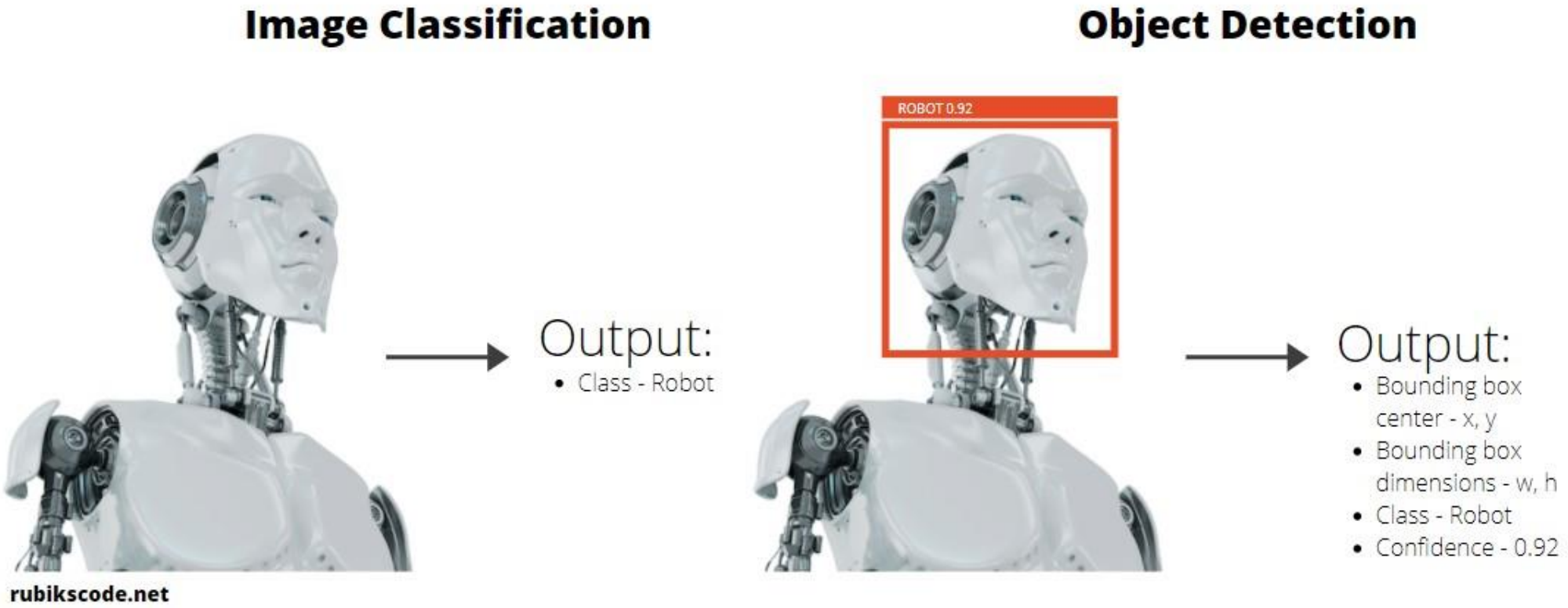
- Andrea Ferri, YOLO: *You Only Look Once, Unified Real-Time Object Detection*

- *YOLO: You Only Look Once, Unified Real-Time Object Detection*, Presenter: Liyang Zhong Quan Zou

Perbedaan *image classification* dengan *object detection*

Image classification → memetakan sebuah citra termasuk ke dalam kelas tertentu

Object detection → menentukan lokasi objek (di mana objek tersebut di dalam citra) dan membingkainya di dalam *bounding box*, memprediksi kelas objek, dan *confidence level* prediksi tersebut



Computer Vision Tasks

Classification

CAT

No spatial extent

Semantic Segmentation

GRASS, CAT, TREE, SKY

No objects, just pixels

Object Detection

DOG, DOG, CAT

Instance Segmentation

DOG, DOG, CAT

Multiple Object

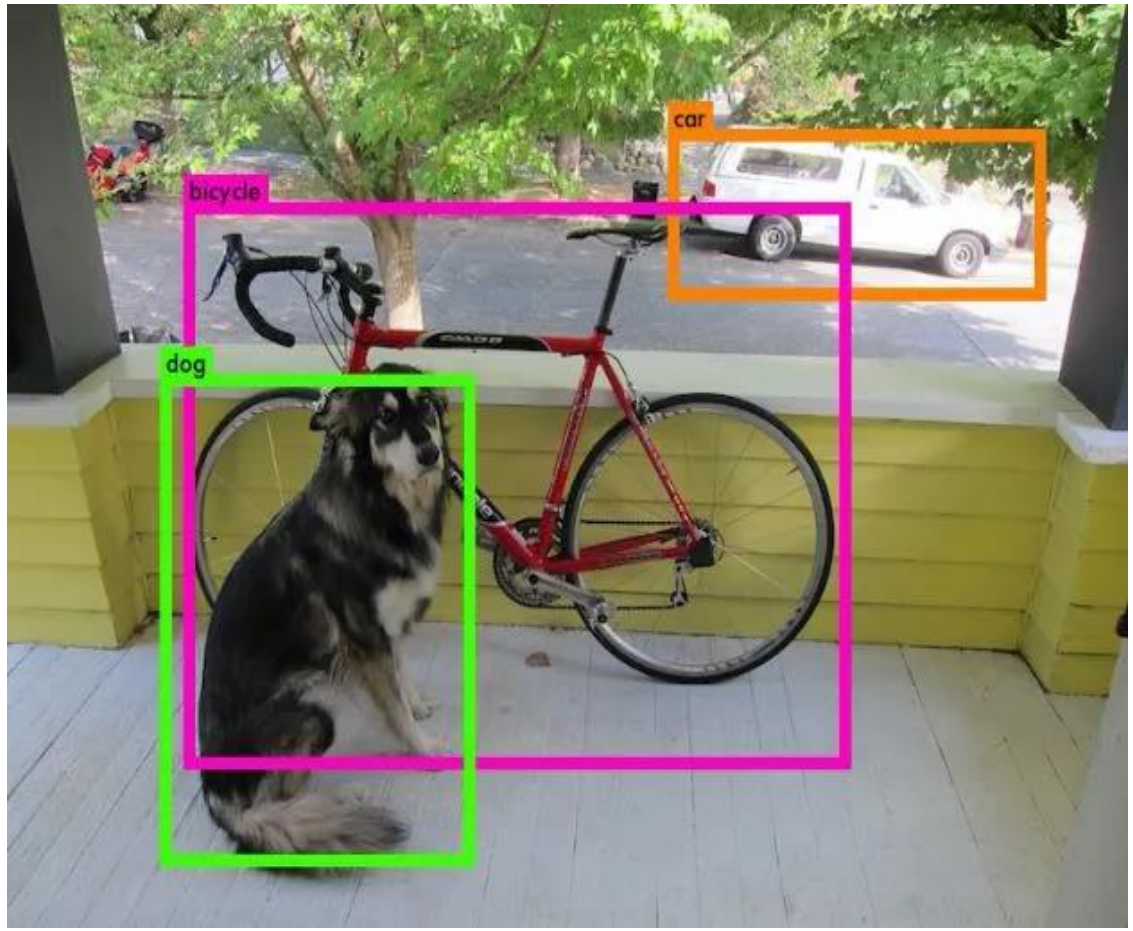
This image is CC0 public domain

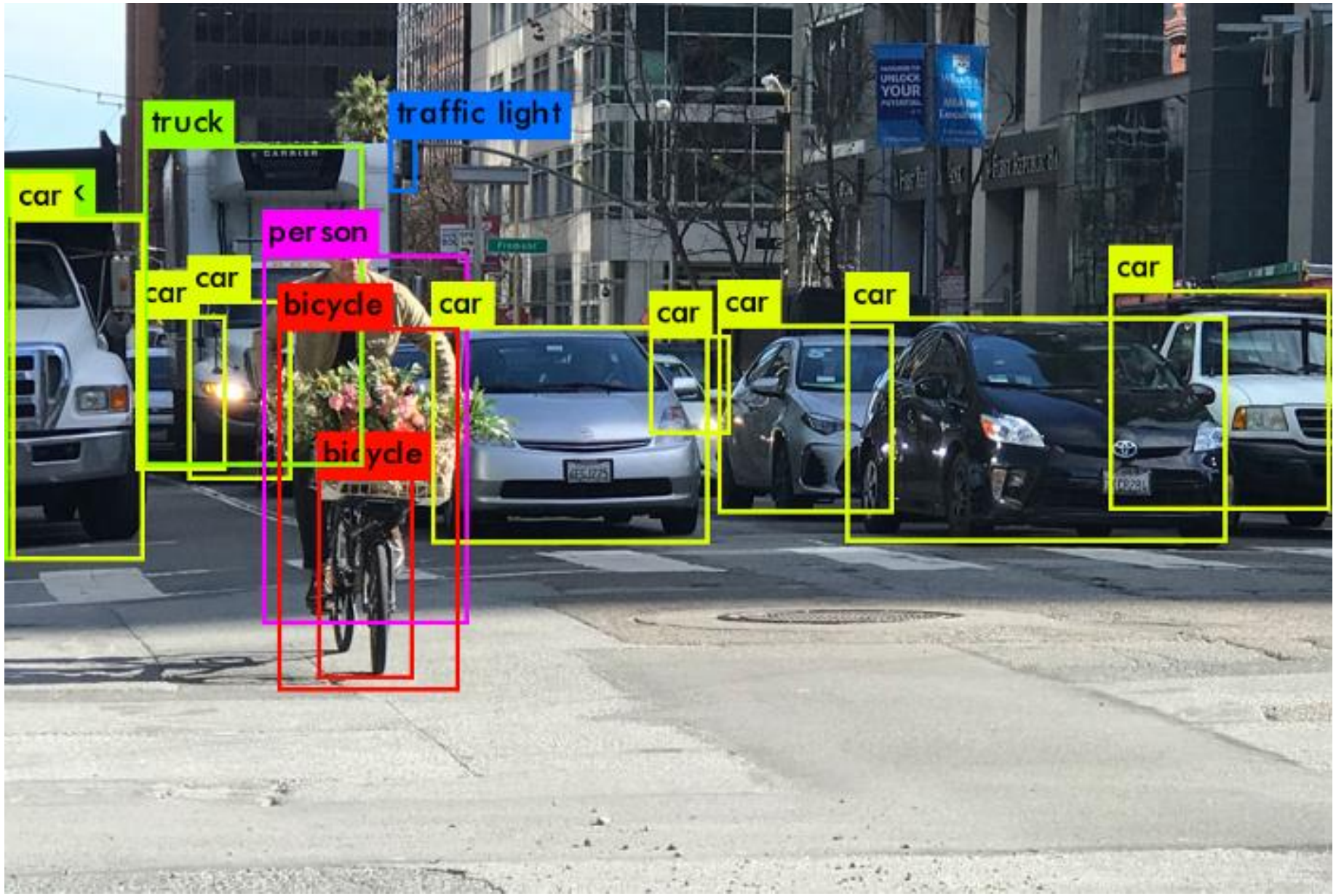
YOLO

- YOLO adalah algoritma untuk mendeteksi objek di dalam citra secara waktu nyata (*real time*)

- Objek yang dideteksi di dalam citra dibingkai di dalam *bounding box* beserta kelasnya (car, dog, bicyle, dsb)

- Disebut YOLO atau *You Only Look Once* karena YOLO melakukan hanya sekali pemindaian pada citra untuk mendeteksi dan mengidentifikasi objek yang terdapat di dalamnya
- Oleh karena itu YOLO diklasifikasikan sebagai model dengan pendekatan *single-stage* atau *one-shot*




truck
traffic light
car
person
car
car
bicycle
car
car
car
car
car
bicycle

- YOLO dikembangkan oleh Joseph Redmond, Santosh Divvala, Ross Girshick, dan Ali Farhadi pada tahun 2015.

- YOLO memiliki kecepatan prediksi yang tinggi sehingga cocok digunakan untuk keperluan pendeteksian objek secara *real-time*. YOLO dasar mampu memroses 45 *frame* per detik.

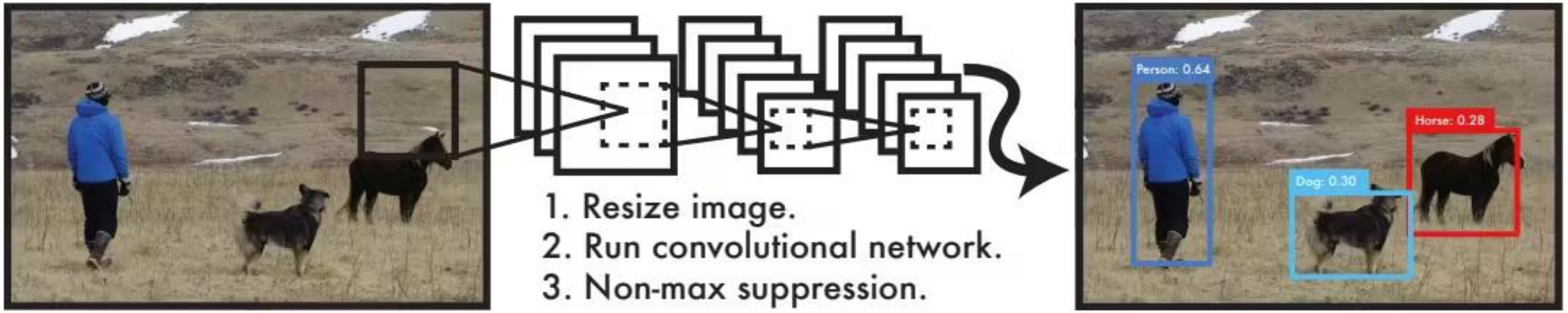


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence. (Redmond, 2016)

Algoritma YOLO

1. Citra masukan dilewatkan melalui CNN untuk mengekstraksi fitur-fitur citra tersebut.
2. Fitur-fitur tersebut kemudian dilewatkan melalui serangkaian *fully connected layer*, yang memprediksi probabilitas kelas dan koordinat kotak pembatas (*bounding box*).
3. Citra dibagi menjadi grid-grid, dan setiap pixel di dalam grid bertanggung jawab untuk memprediksi sekumpulan kotak pembatas (*bounding box*) dan probabilitas kelas.
4. Luaran jaringan adalah sekumpulan kotak pembatas dan probabilitas kelas untuk setiap sel.
5. Kotak pembatas kemudian difilter menggunakan algoritma pasca-pemrosesan yang disebut *non-max supression* untuk menghilangkan kotak yang tumpang tindih dan memilih kotak dengan probabilitas tertinggi.
6. Luaran akhir adalah sekumpulan kotak pembatas yang diprediksi dan label kelas untuk setiap objek dalam citra.

Arsitektur utama YOLO

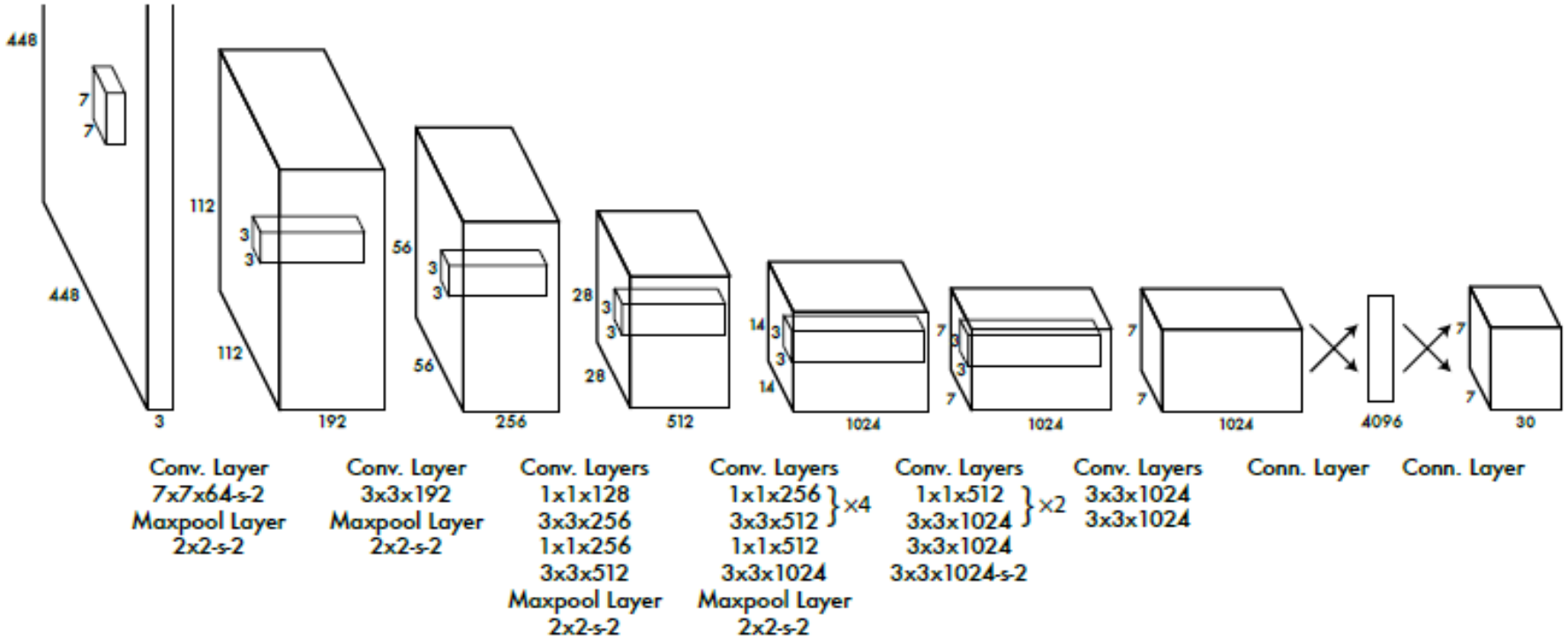
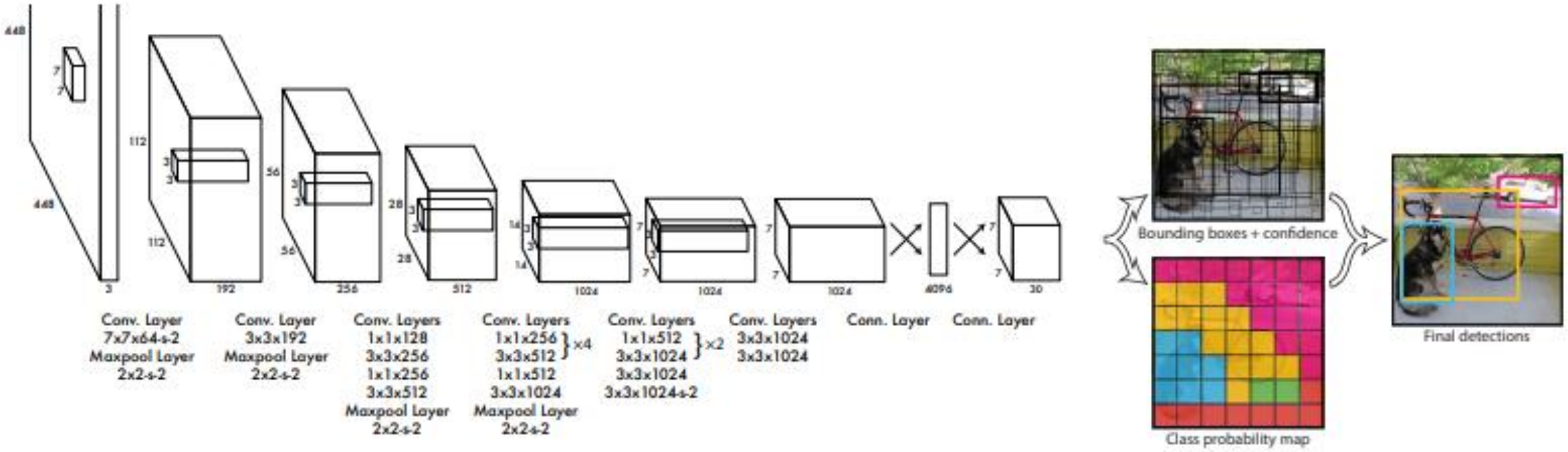


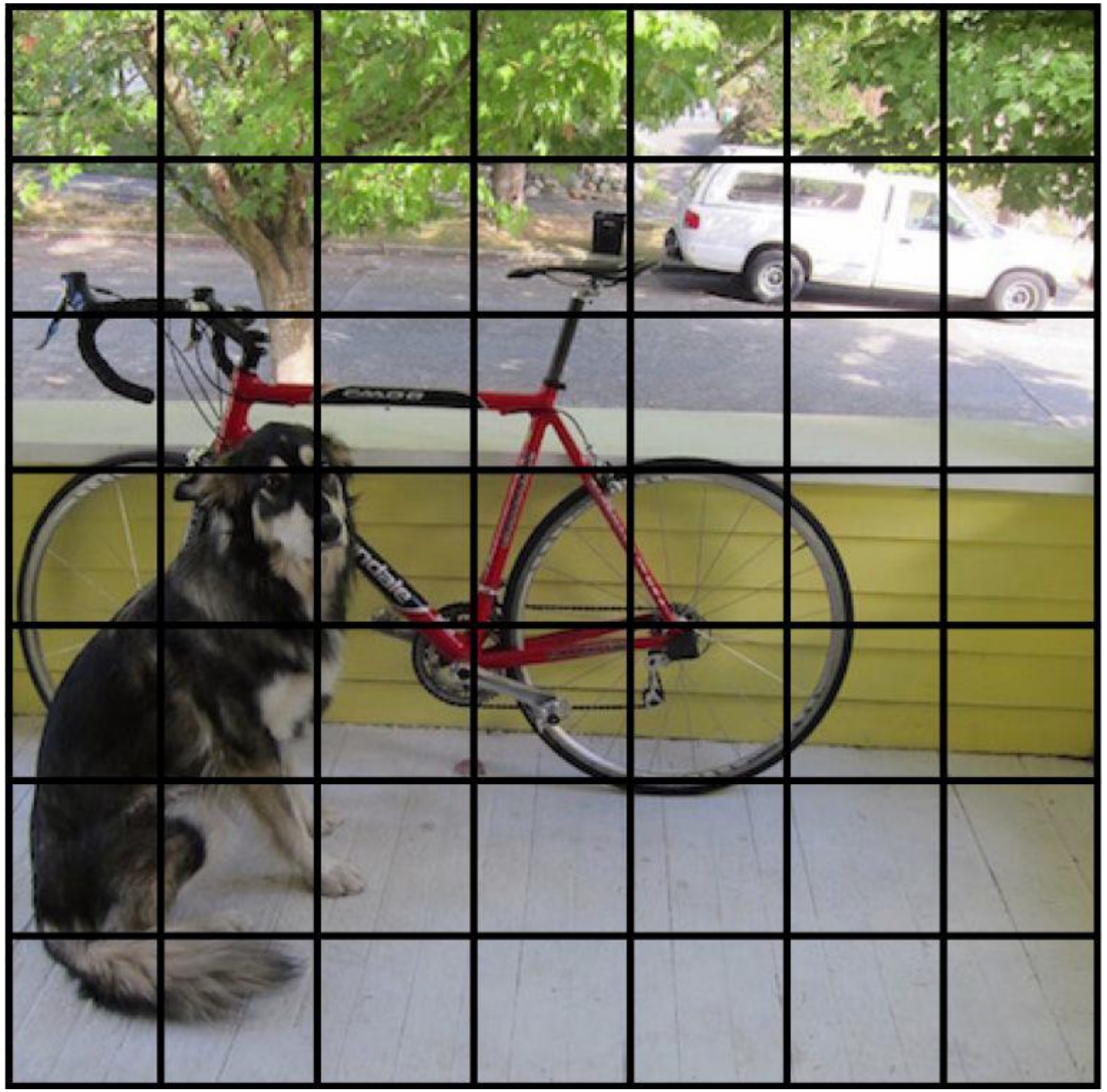
Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.


448
7
7
448
3
112
3
3
112
192
56
3
3
56
256
28
3
3
28
512
14
3
3
14
1024
7
3
3
7
1024
7
7
1024
4096
7
7
30
Conv. Layer
7x7x64-s-2
Maxpool Layer
2x2-s-2
Conv. Layer
3x3x192
Maxpool Layer
2x2-s-2
Conv. Layers
1x1x128
3x3x256
1x1x256
3x3x512
Maxpool Layer
2x2-s-2
Conv. Layers
1x1x256
3x3x512
×4
1x1x512
3x3x1024
Maxpool Layer
2x2-s-2
Conv. Layers
1x1x512
3x3x1024
×2
3x3x1024
3x3x1024-s-2
Conv. Layers
3x3x1024
3x3x1024
Conn. Layer
Conn. Layer
Bounding boxes + confidence
Class probability map
Final detections

Detil Prosedur Deteksi

Input:

- Bagi citra input ke dalam grid-grid berukuran S x S

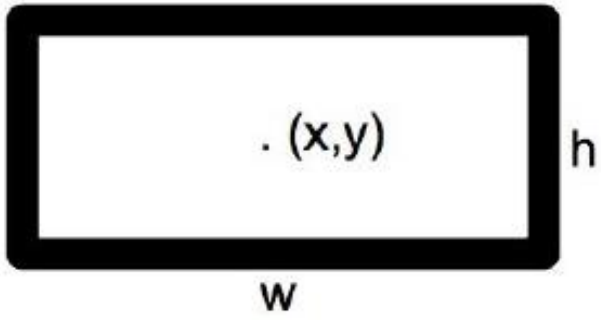


Pada contoh ini, grid 7 x 7

- Setiap sel memprediksi B buah *box* (x, y, w, h) dan *confidence* setiap box: P(Object)

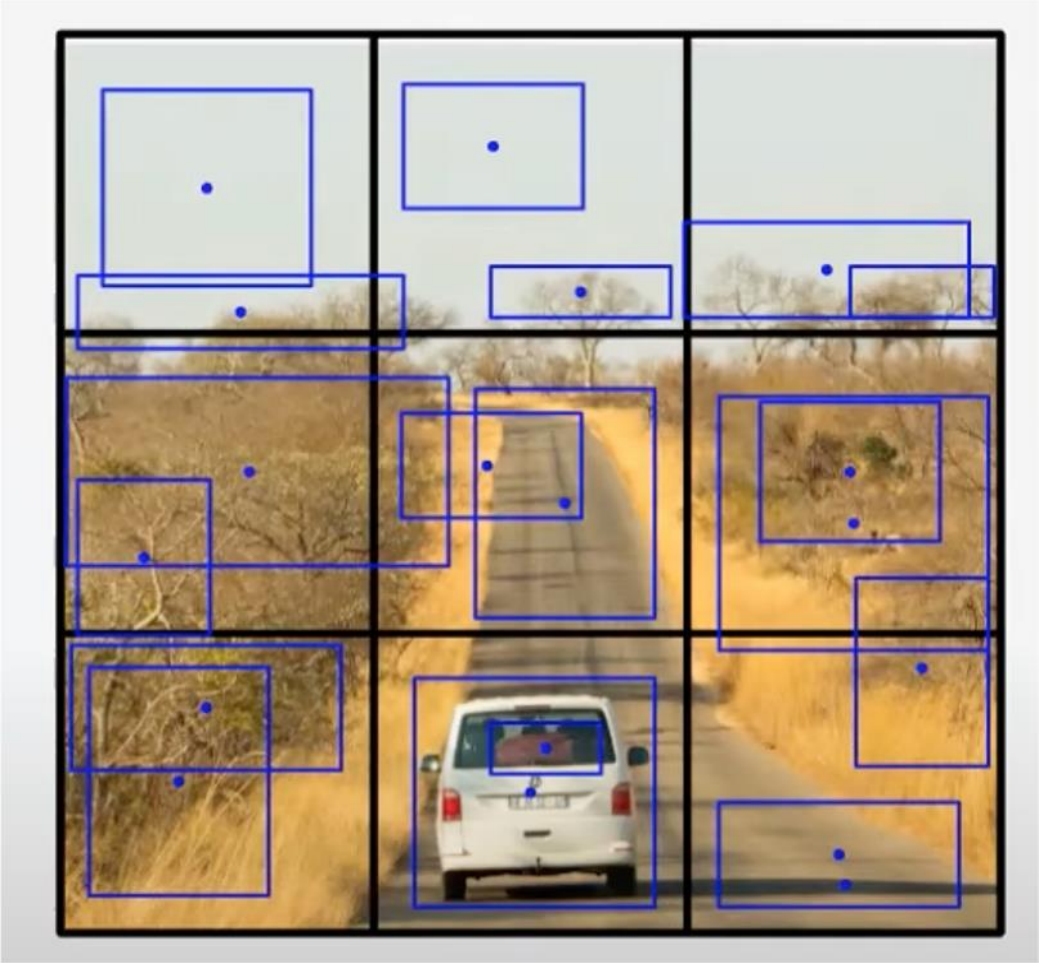
B = 2

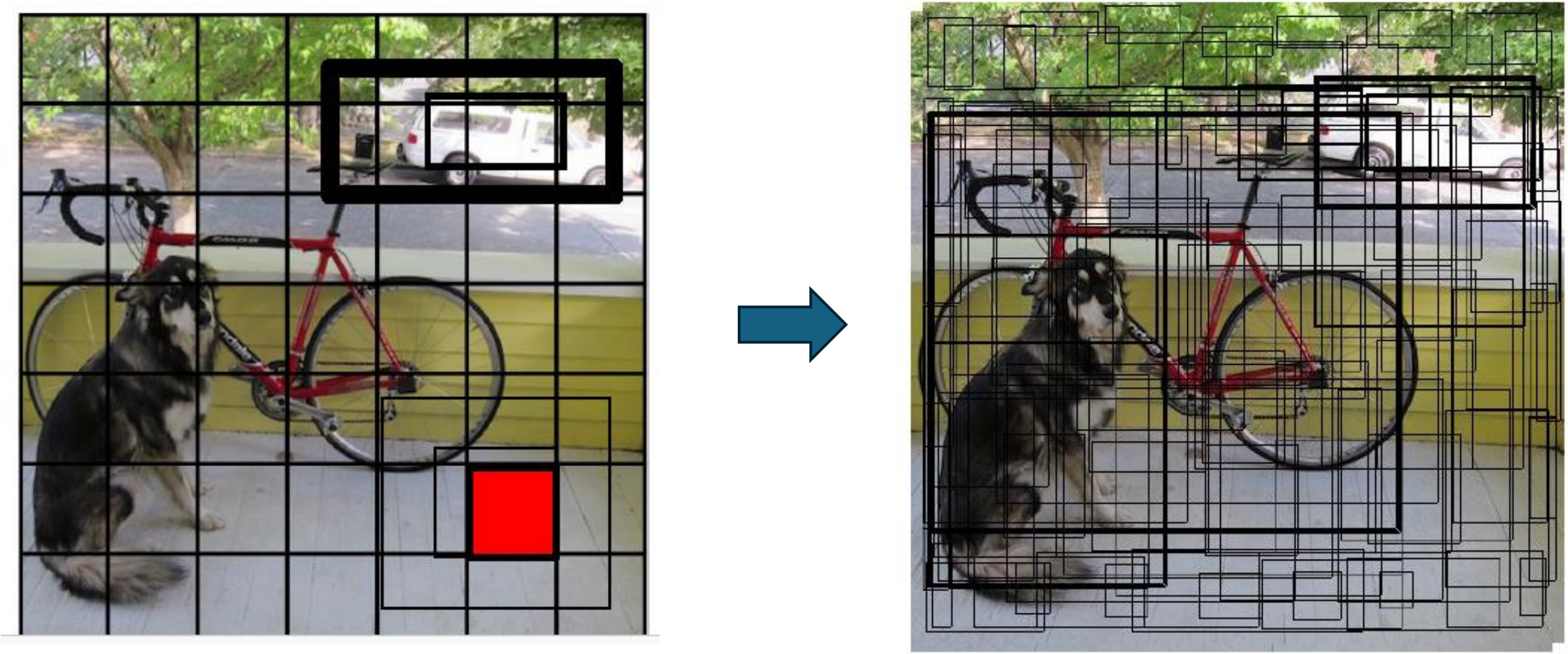
each box predict:



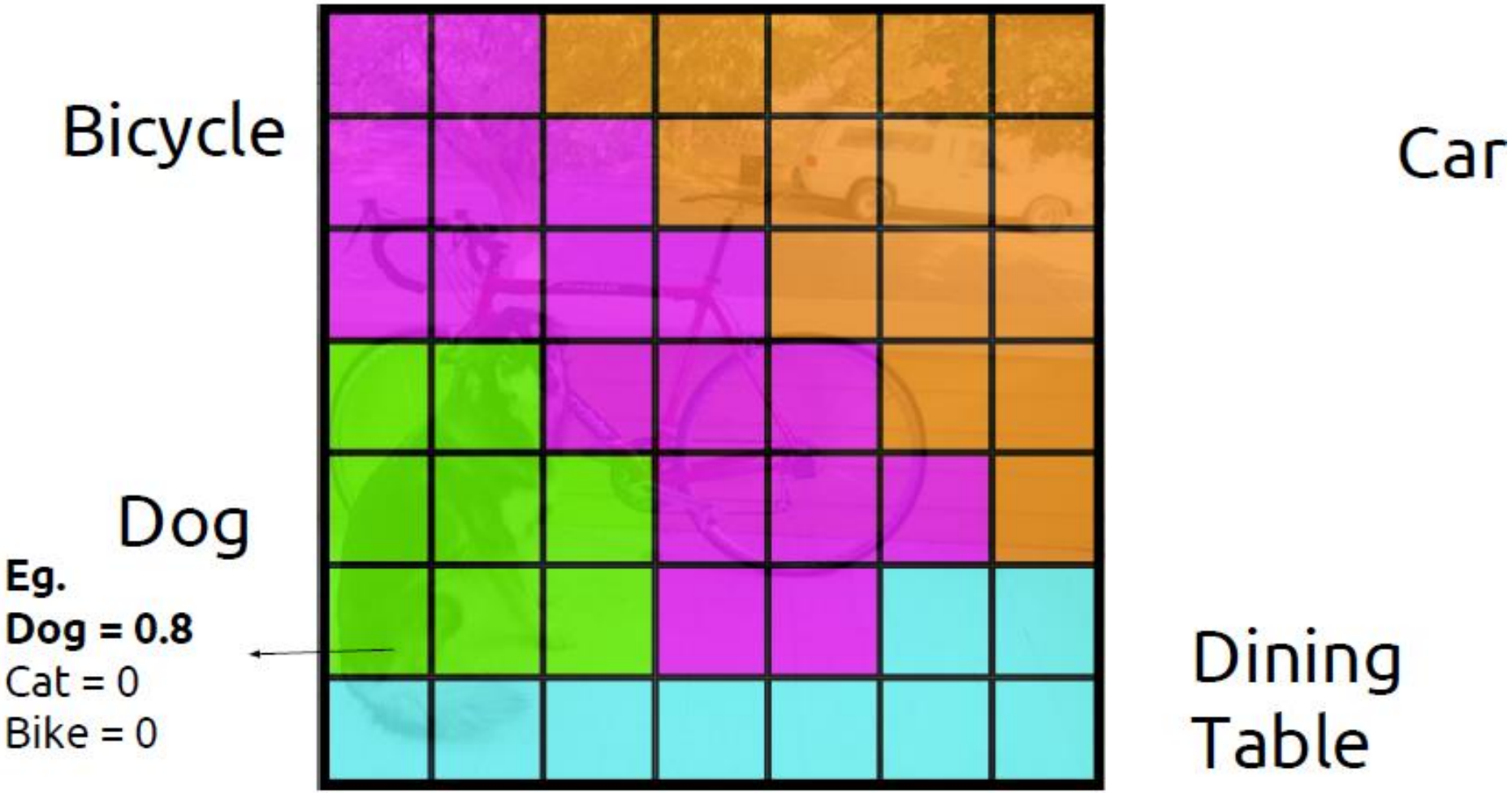
P(Object): probability that the box contains an object

- Contoh lain (S = 3, B = 2):

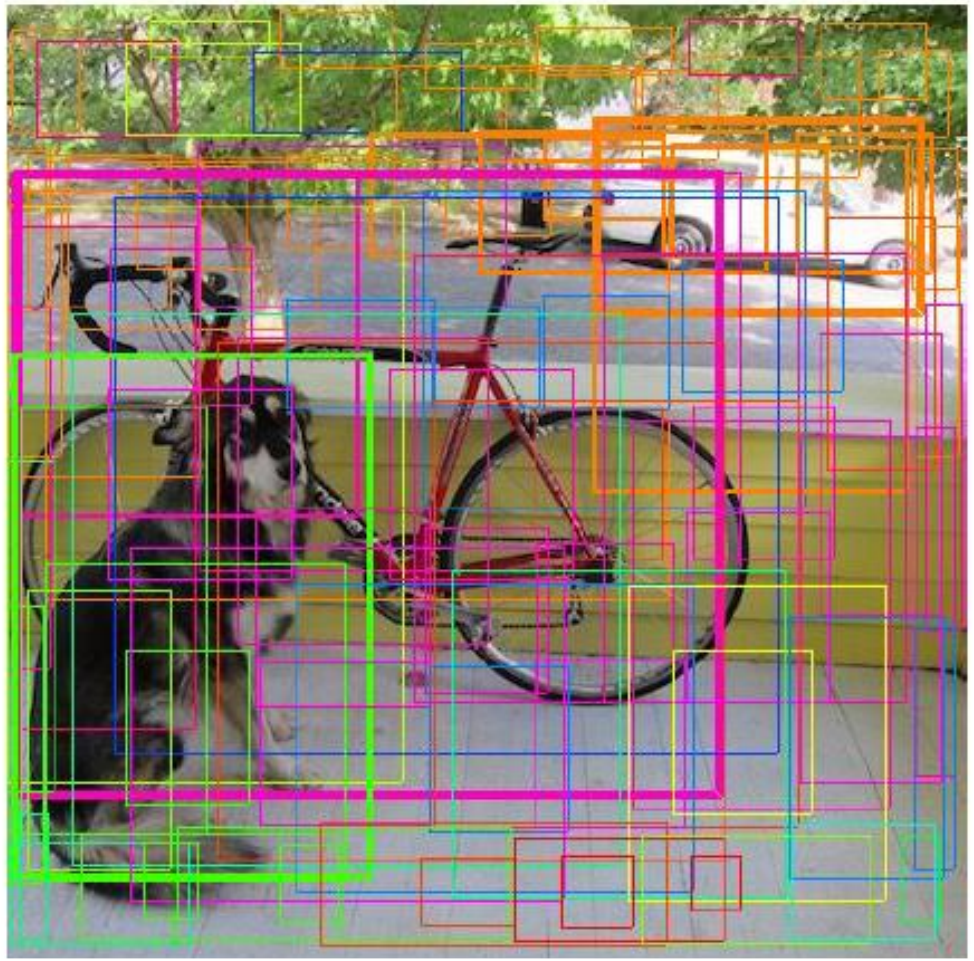




- Setiap sel juga memprediksi probabilitas kelas



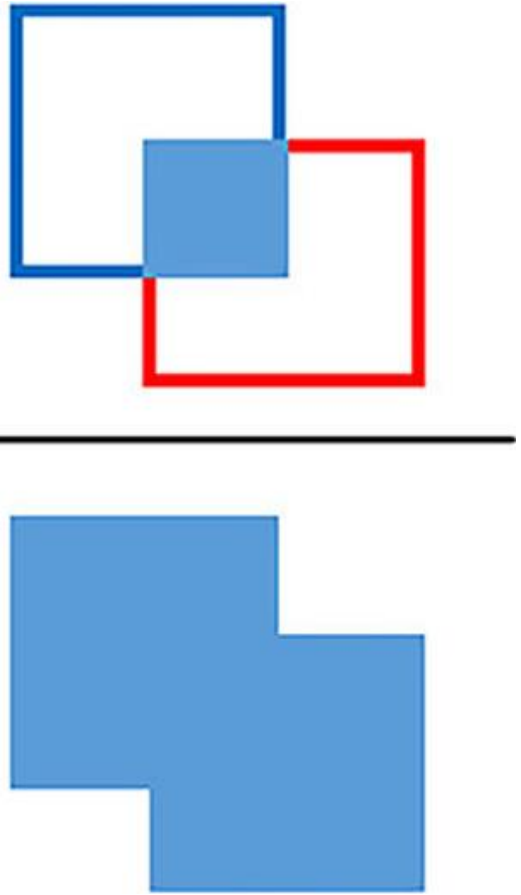
- Kemudian gabung box dan prediksi kelas

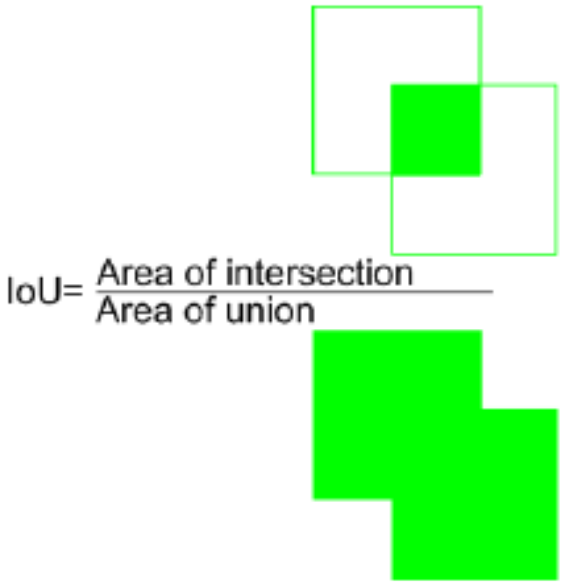


P(class|Object) * P(Object)
=P(class)

$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

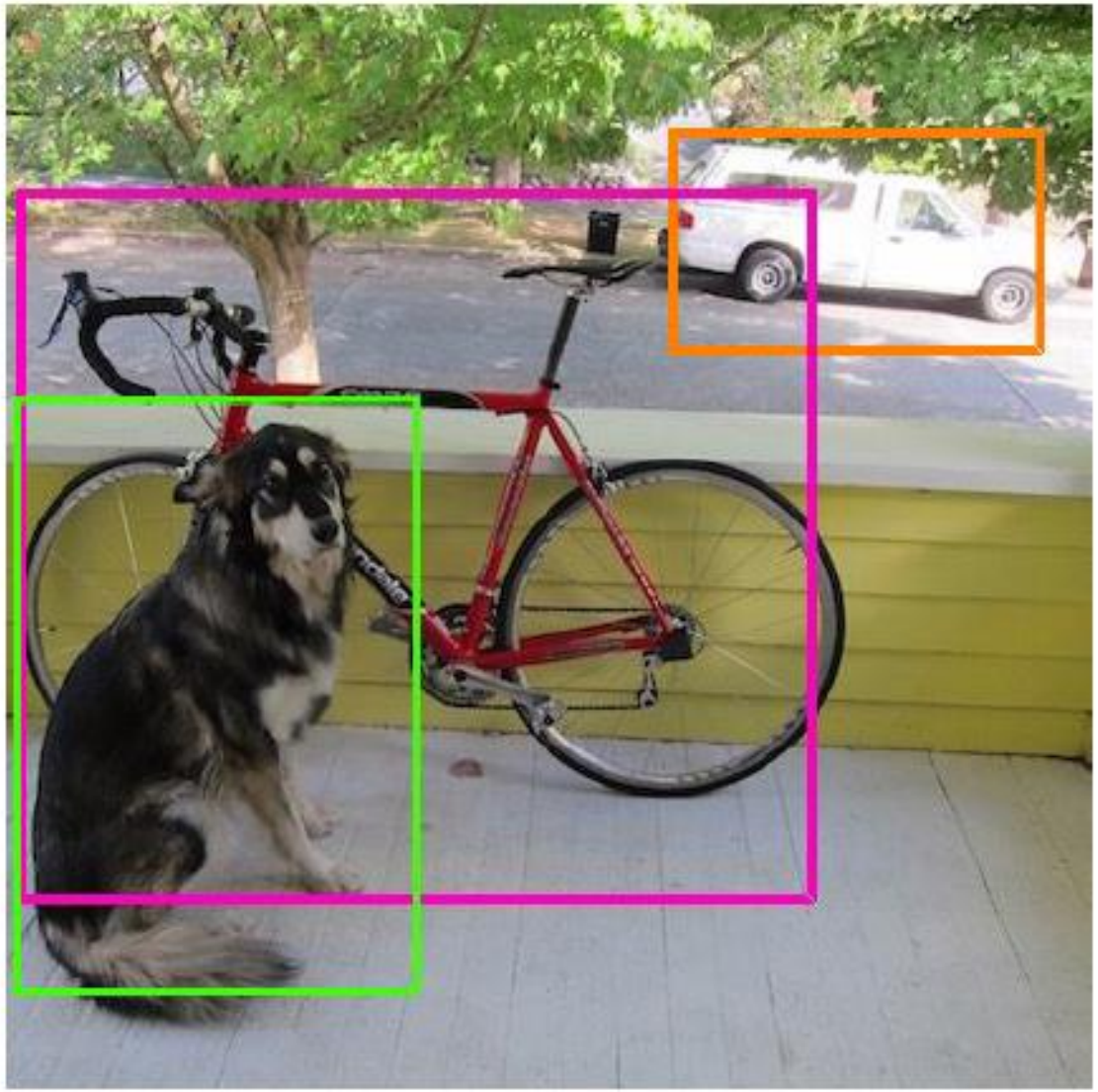
$$IoU = \frac{area\ of\ intersection}{are\ of\ union} =$$





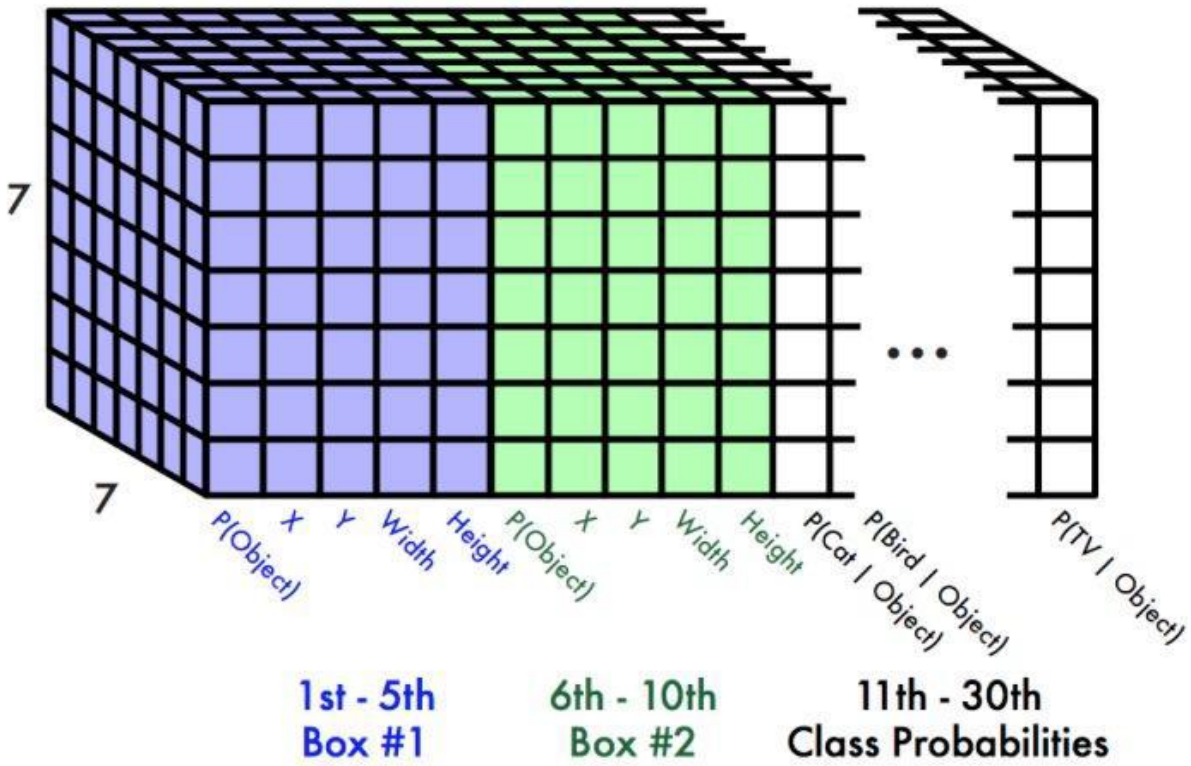
Ilustrasi *Intersection over Union* (IoU) (Adrian Rosebrock, 2016)

- Terakhir, lakukan *thresholding*



Each cell predicts:

- For each bounding box:
  - 4 coordinates (x, y, w, h)
  - 1 confidence value
- Some number of class probabilities



For Pascal VOC:

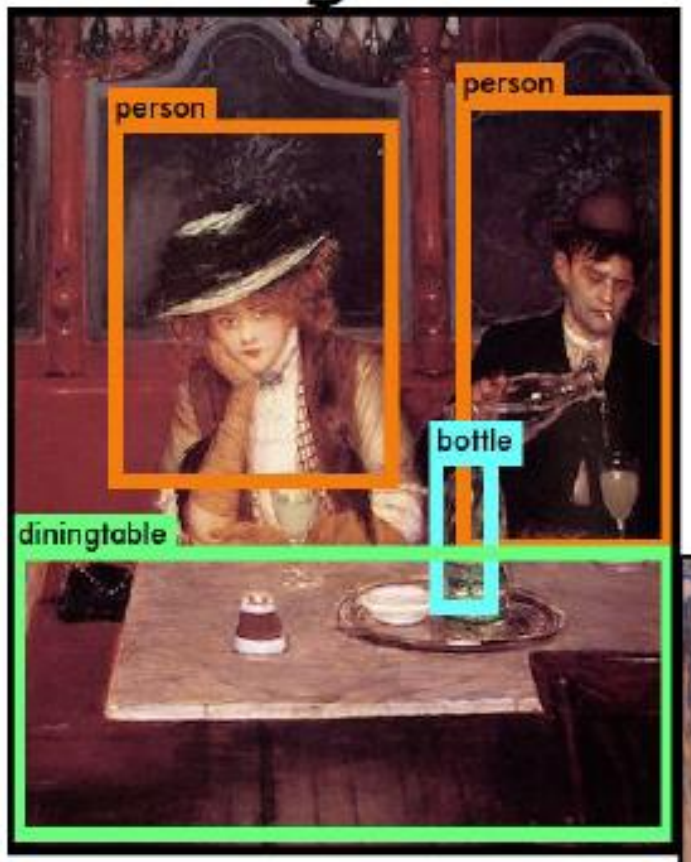
- 7x7 grid
- 2 bounding boxes / cell
- 20 classes

7 x 7 x (2 x 5 + 20) = 7 x 7 x 30 tensor = **1470 outputs**

- Loss function

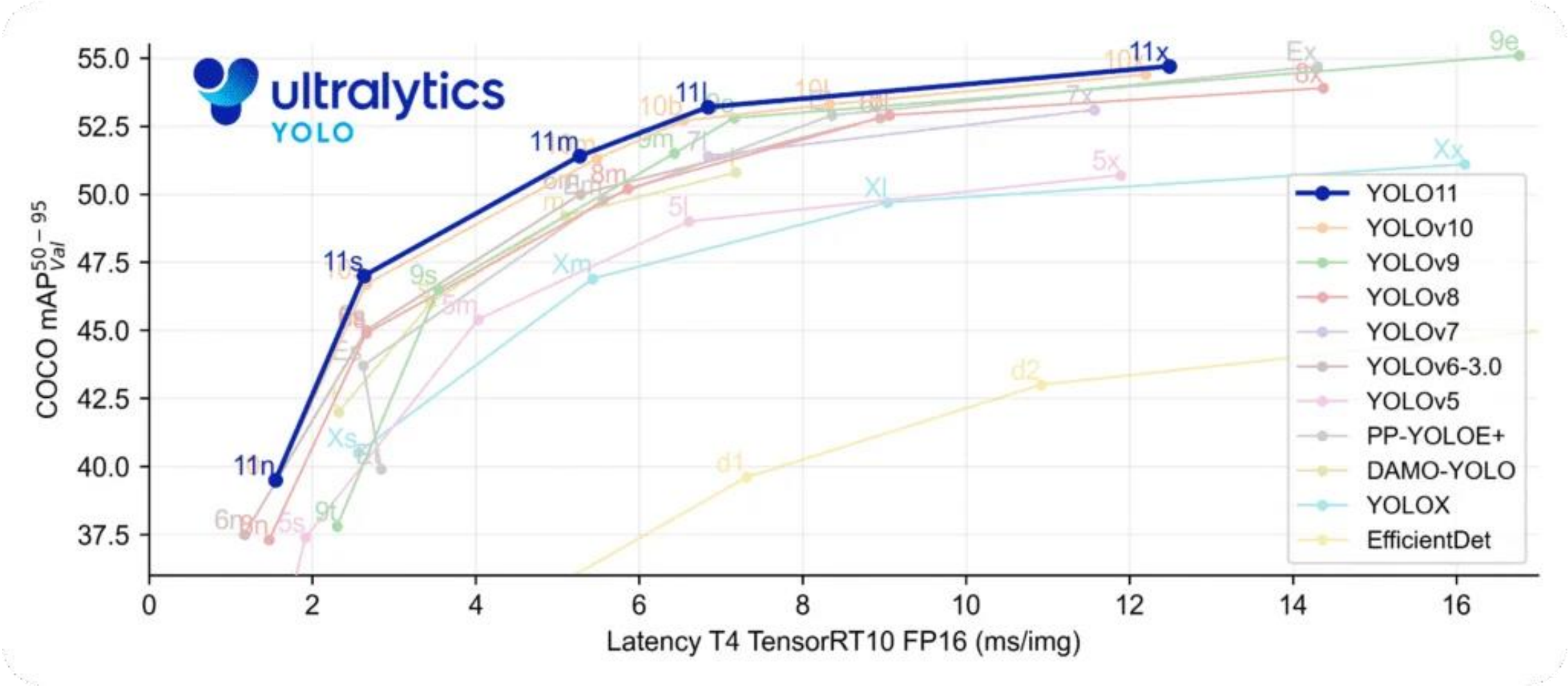
$$
\begin{aligned}
&\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(x_i - \hat{x}_i\right)^2 + \left(y_i - \hat{y}_i\right)^2 \\
&+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2 \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i\right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i\right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c)\right)^2
\end{aligned}
$$

YOLO generalizes well to new domains (like art)



Ref: https://pjreddie.com/publications/

Pustaka YOLO di dalam Bahasa Python: *Ultralytics*



mAP = mean Average Precision

Simbol	Arti
$P = \frac{TP}{TP+FP}$	Precision
$R = \frac{TP}{TP+FN}$	Recall
AP	Area under Precision–Recall curve untuk 1 kelas
mAP	Rata-rata dari semua AP kelas

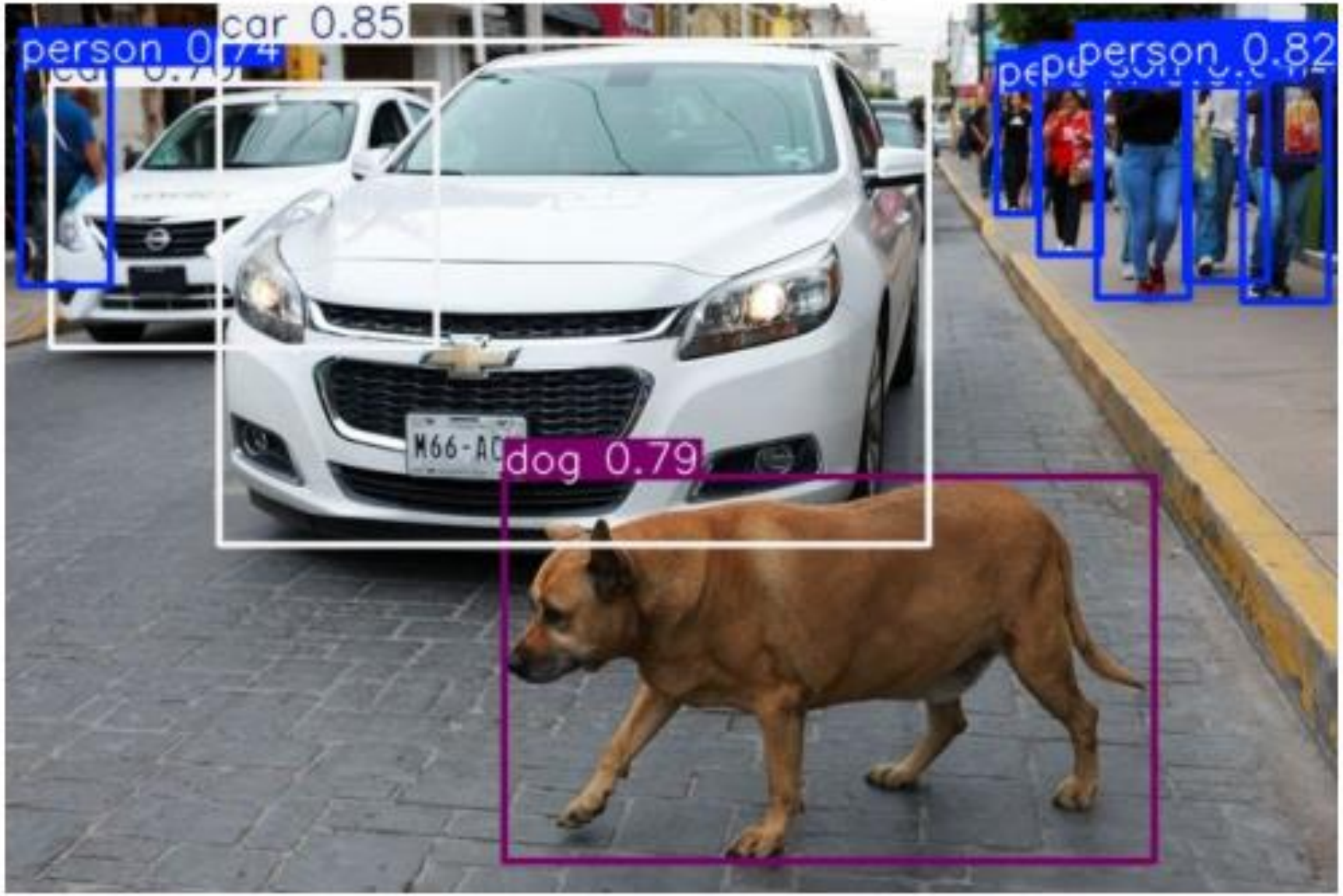
Dalam konteks deteksi objek, AP dihitung berdasarkan *IoU threshold* (*Intersection over Union*) antara prediksi dan *ground truth*.

Contoh kode Python untuk deteksi objek di dalam citra

```python
# Instal terlebih dahulu: pip install ultralytics opencv-python torch

from ultralytics import YOLO
import cv2
import matplotlib.pyplot as plt

# Load model YOLO11 (pretrained COCO)
model = YOLO("yolo11n.pt")   # n = nano, cepat. Bisa ganti: yolo11s.pt, yolo11m.pt, yolo11l.pt, yolo11x.pt

# Gambar input
img_path = "E://IF4073 Interpretasi dan Pengolahan Citra/man-and-dog.jpeg"

# Tampilkan gambar input
img_input = cv2.imread(img_path)

# Konversi BGR → RGB, konversi ruang warna dari BGR ke RGB agar warna  tampil benar
img_input_rgb = cv2.cvtColor(img_input, cv2.COLOR_BGR2RGB)

plt.figure(figsize=(8, 6))  # Ukuran canvas: Lebar = 8 inchi, Tinggi = 6 inch
plt.title("Gambar Input")
plt.imshow(img_input_rgb)
plt.axis('off')
plt.show()

# Prediksi objek di dalam gambar
results = model(img_path)
```

```python
for result in results:
    print(result.names)            # daftar kelas
    print(result.boxes.data)       # bounding boxes (x,y x,y, conf, class)

# Simpan gambar hasil deteksi
output_path = "E://IF4073 Interpretasi dan Pengolahan Citra/hasil_deteksi.jpg"
results[0].save(filename=output_path)

# Tampilkan gambar hasil deteksi
img = cv2.imread("E://IF4073 Interpretasi dan Pengolahan Citra/hasil_deteksi.jpg")

# Konversi BGR → RGB, konversi ruang warna dari BGR ke RGB agar warna  tampil benar
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

# Tampilkan gambar hasil deteksi
plt.title("Hasil deteksi objek")
plt.imshow(img_rgb)
plt.axis('off')
plt.show()
```

Gambar Input

```
image 1/1 E:\IF4073 Interpretasi dan Pengolahan Citra\man-and-dog.jpeg: 448x640 6 persons, 2 cars, 1 dog, 1 backpack, 100.2ms
Speed: 2.6ms preprocess, 100.2ms inference, 2.7ms postprocess per image at shape (1, 3, 448, 640)
{0: 'person', 1: 'bicycle', 2: 'car', 3: 'motorcycle', 4: 'airplane', 5: 'bus', 6: 'train', 7: 'truck', 8: 'boat', 9: 'traffic light', 10: 'fire hydrant
', 11: 'stop sign', 12: 'parking meter', 13: 'bench', 14: 'bird', 15: 'cat', 16: 'dog', 17: 'horse', 18: 'sheep', 19: 'cow', 20: 'elephant', 21: 'bear',
22: 'zebra', 23: 'giraffe', 24: 'backpack', 25: 'umbrella', 26: 'handbag', 27: 'tie', 28: 'suitcase', 29: 'frisbee', 30: 'skis', 31: 'snowboard', 32: 's
ports ball', 33: 'kite', 34: 'baseball bat', 35: 'baseball glove', 36: 'skateboard', 37: 'surfboard', 38: 'tennis racket', 39: 'bottle', 40: 'wine glass
', 41: 'cup', 42: 'fork', 43: 'knife', 44: 'spoon', 45: 'bowl', 46: 'banana', 47: 'apple', 48: 'sandwich', 49: 'orange', 50: 'broccoli', 51: 'carrot', 5
2: 'hot dog', 53: 'pizza', 54: 'donut', 55: 'cake', 56: 'chair', 57: 'couch', 58: 'potted plant', 59: 'bed', 60: 'dining table', 61: 'toilet', 62: 'tv',
63: 'laptop', 64: 'mouse', 65: 'remote', 66: 'keyboard', 67: 'cell phone', 68: 'microwave', 69: 'oven', 70: 'toaster', 71: 'sink', 72: 'refrigerator', 7
3: 'book', 74: 'clock', 75: 'vase', 76: 'scissors', 77: 'teddy bear', 78: 'hair drier', 79: 'toothbrush'}
tensor([[1.0427e+02, 1.9327e+01, 4.5039e+02, 2.6432e+02, 8.4974e-01, 2.0000e+00],
        [5.3281e+02, 1.3066e+01, 5.7776e+02, 1.4469e+02, 8.2382e-01, 0.0000e+00],
        [2.4316e+02, 2.3172e+02, 5.6090e+02, 4.1840e+02, 7.9066e-01, 1.6000e+01],
        [5.7585e+02, 1.0112e+01, 6.1526e+02, 1.3658e+02, 7.4116e-01, 0.0000e+00],
        [7.7803e+00, 3.1454e+01, 5.1482e+01, 1.3818e+02, 7.3852e-01, 0.0000e+00],
        [2.2199e+01, 4.0356e+01, 2.1010e+02, 1.6870e+02, 7.0184e-01, 2.0000e+00],
        [5.0436e+02, 3.8907e+01, 5.3438e+02, 1.2352e+02, 6.3710e-01, 0.0000e+00],
        [6.0361e+02, 1.3843e+01, 6.4533e+02, 1.4620e+02, 6.3233e-01, 0.0000e+00],
        [4.8372e+02, 4.2303e+01, 5.0297e+02, 1.0350e+02, 6.0364e-01, 0.0000e+00],
        [6.2201e+02, 4.0577e+01, 6.4384e+02, 7.7730e+01, 3.4214e-01, 2.4000e+01]])
```

Hasil deteksi objek