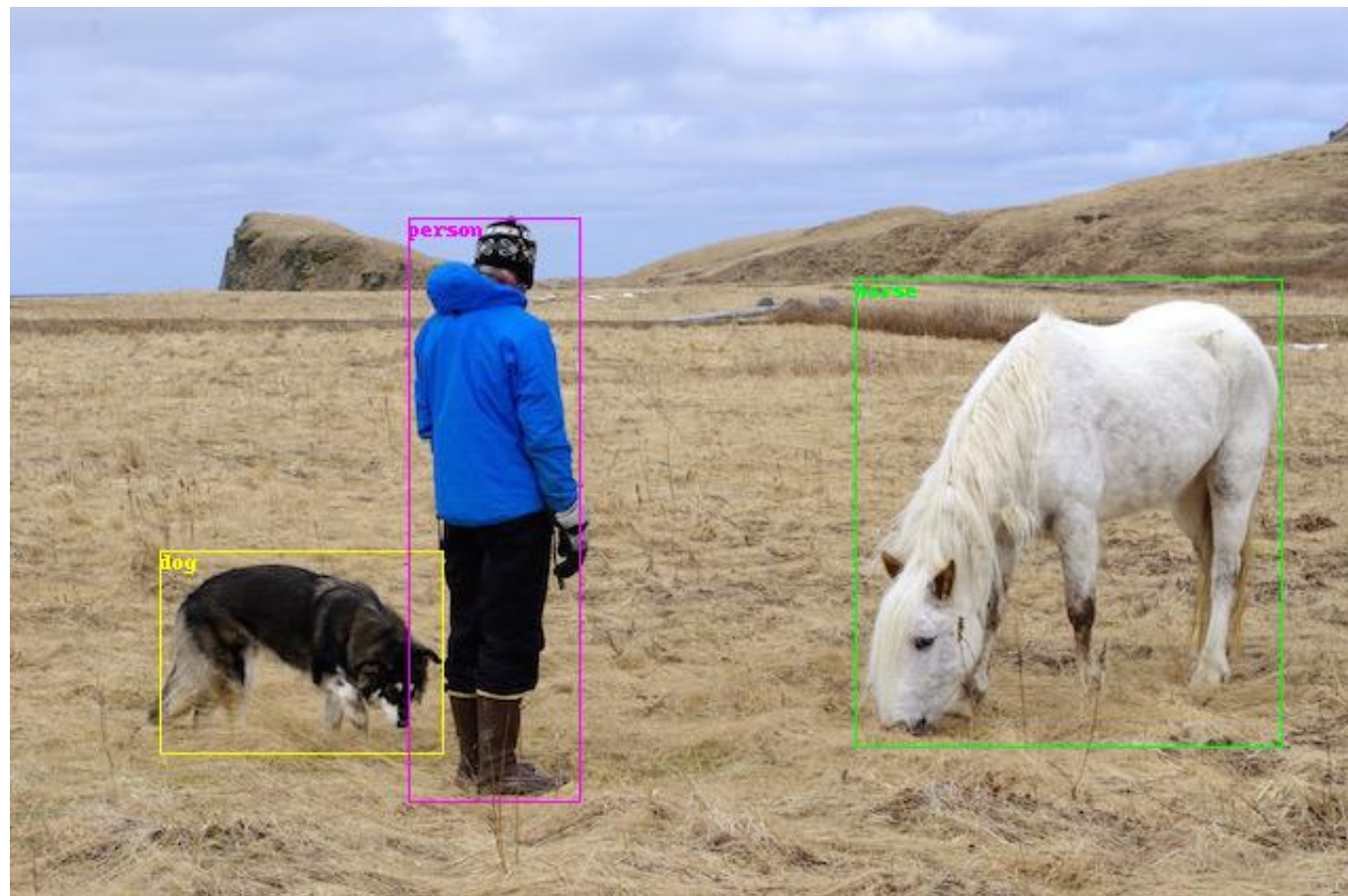


کارگاه یادگیری عمیق با پایتون: تشخیص اشیا

سید ناصر رضوی www.snrazavi.ir

۱۳۹۷



□ دسته‌بندی و مکان‌یابی.

- یک شی در تصویر
- شناسایی نقاط عطف
- تشخیص وضعیت قرارگیری

□ تشخیص اشیا.

- چند شی در تصویر
- پنجره لغزان
- شبکه‌های کاملاً کانولوشنی
- الگوریتم YOLO

تشخیص اشیا: اجرای نمایشی

۳



تشخیص اشیا

۴

□ دسته‌بندی تصویر و شناسایی اشیا.



دسته‌بندی تصویر



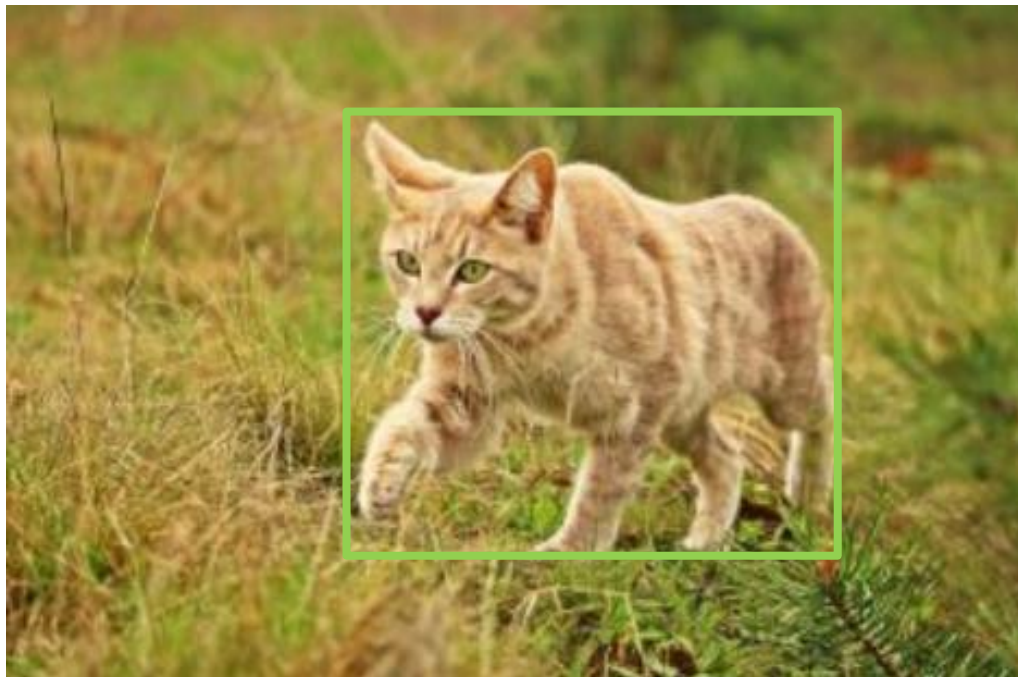
دسته‌بندی و مکان‌یابی



شناسایی اشیا

مکان‌یابی

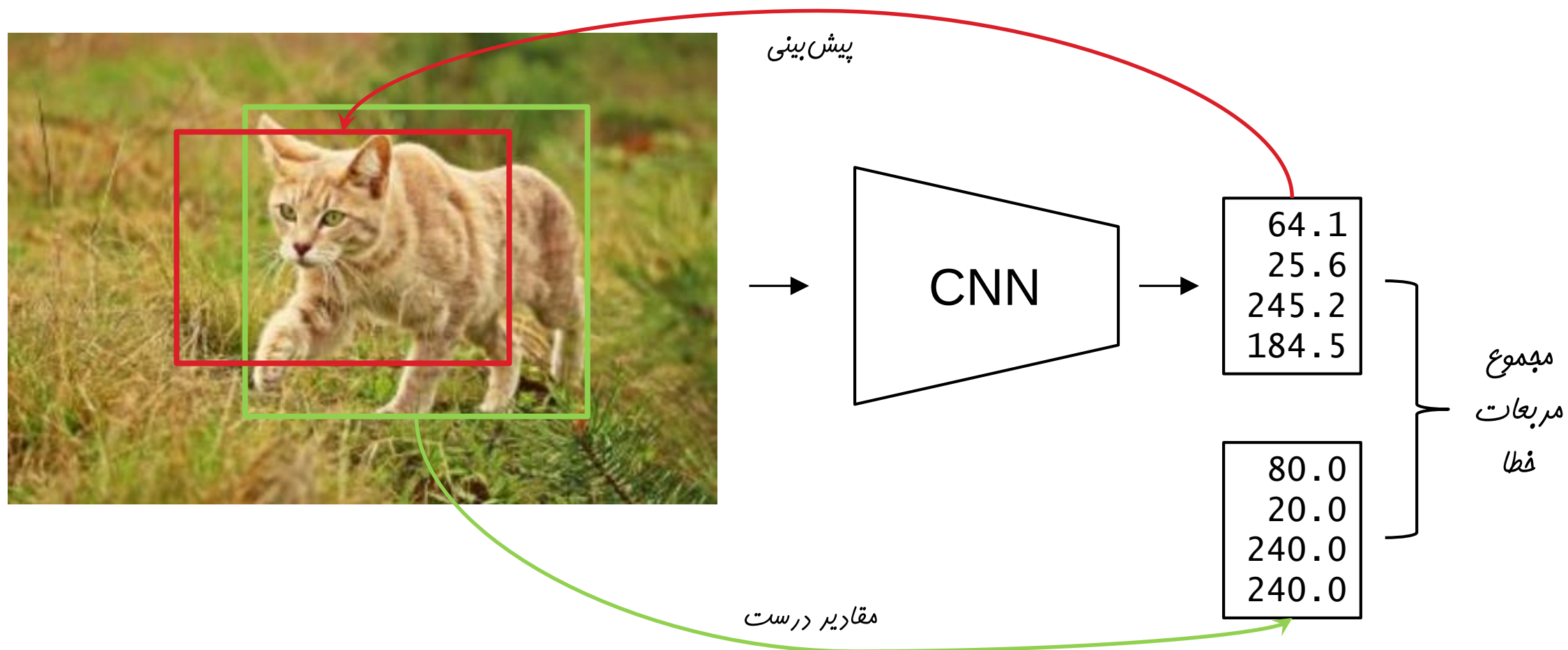
۵



- یک شی در هر تصویر
- تخمین مختصات جعبه محاطی
- ارزیابی از طریق نسبت اشتراک به اجتماع (IoU)

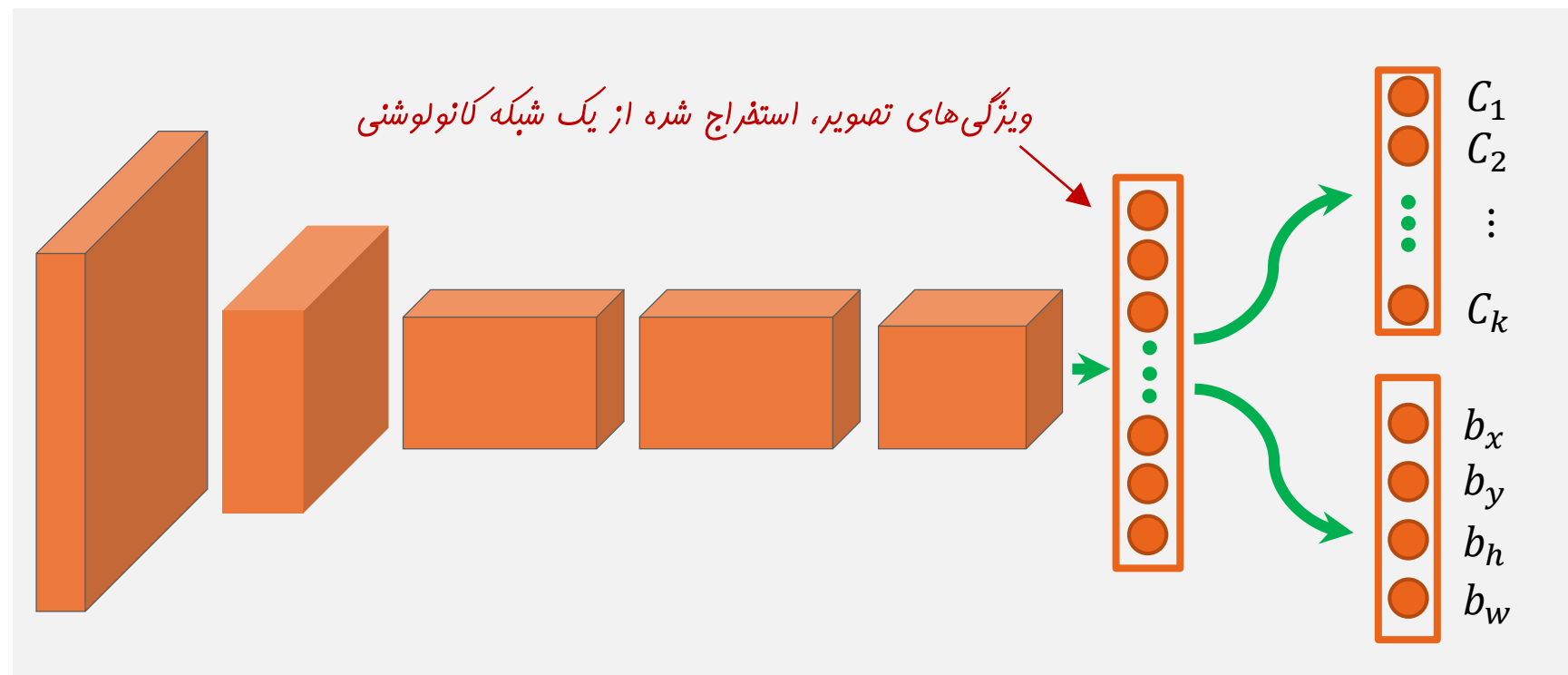
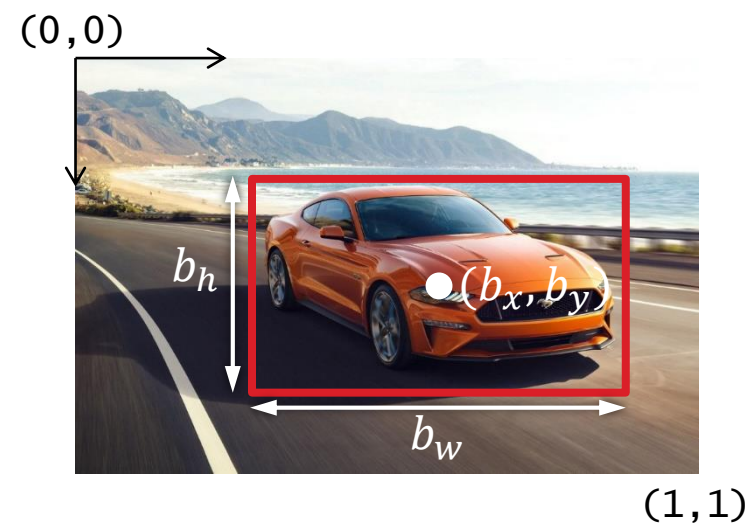
مکان‌یابی به عنوان رگرسیون

۶



دسته‌بندی و مکان‌یابی

۷



تابع هزینه: مجموع مربعات خطا

۸



$$\begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ C_1 \\ C_2 \\ C_3 \end{bmatrix}$$

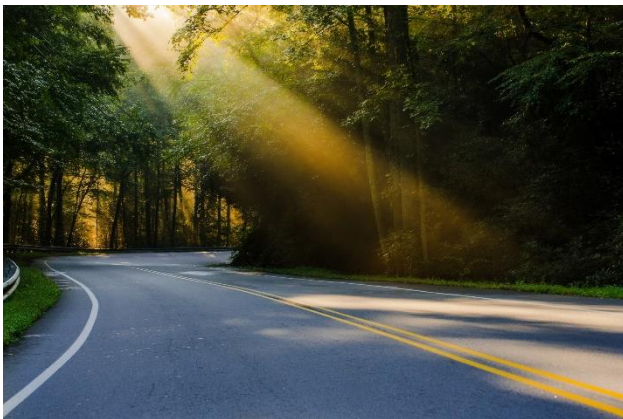
□ کلاس‌ها

۱ - رهگذر

۲ - خودرو

۳ - موتورسیکلت

۴ - پس‌زمینه



$$\begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix}$$

$$L(\hat{y}, y) = \begin{cases} \sum_{i=1}^f (\hat{y}_i - y_i)^2, & y_1 = 1 \\ (\hat{y}_1 - y_1)^2, & y_1 = 0 \end{cases}$$

تشخیص نقاط عطف: تشخیص احساسات

۹



Zhanpeng Zhang, et. al. Facial Landmark Detection by Deep Multi-task Learning, ECCV, 2014

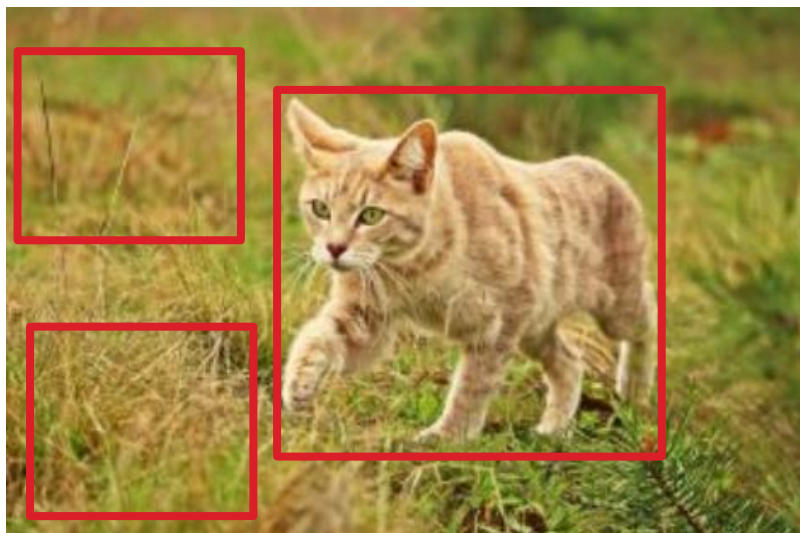
10



شناسایی اشیاء: پنجره لغزان

۱۱

ایجاد مجموعه آموزشی



1

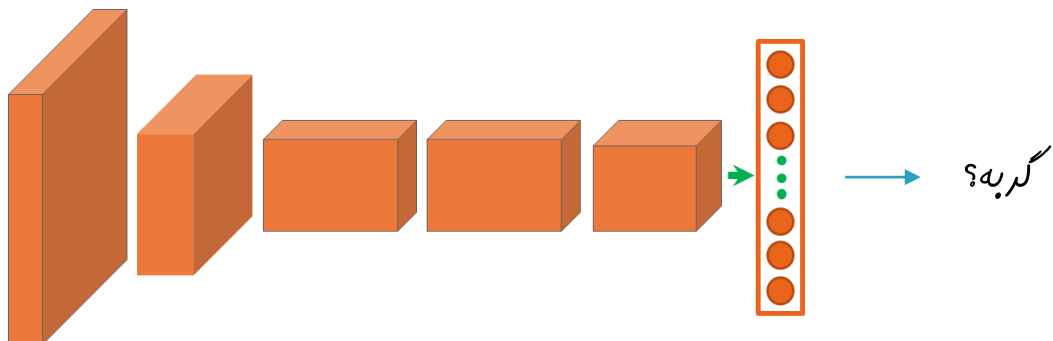


0



0

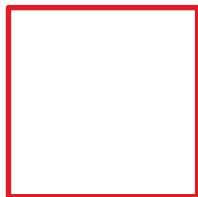
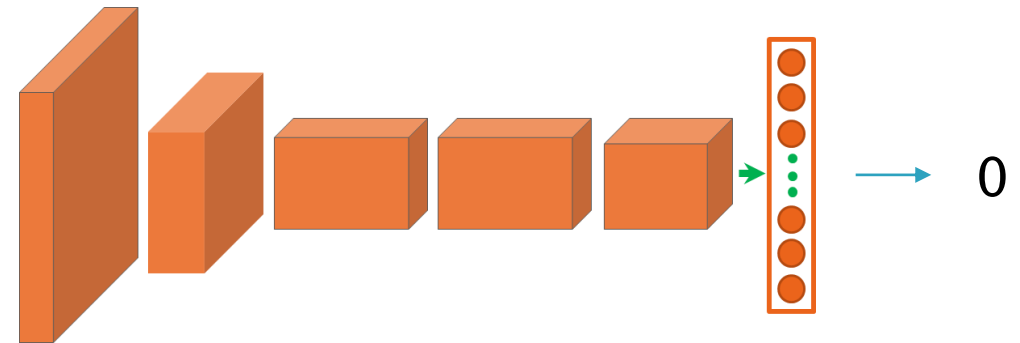
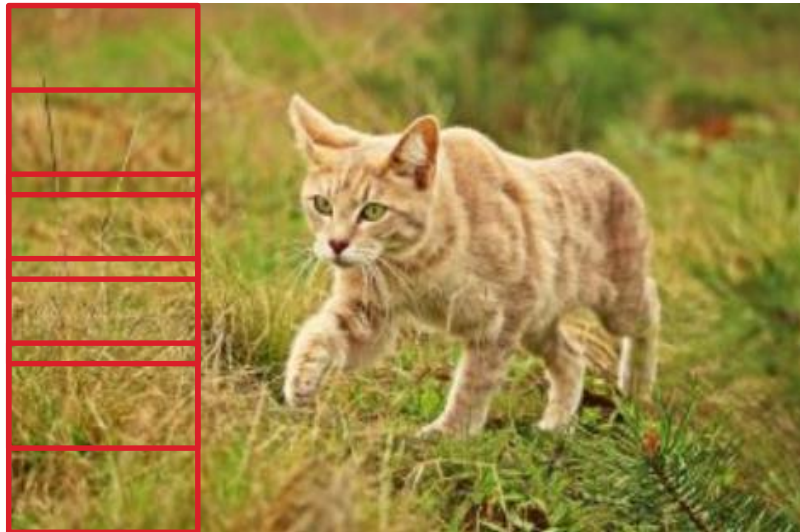
آموزش شبکه کانولوشنی



شناسایی اشیاء: پنجره لغزان

۱۲

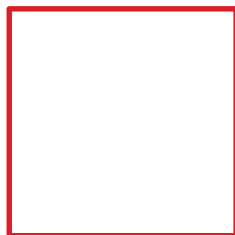
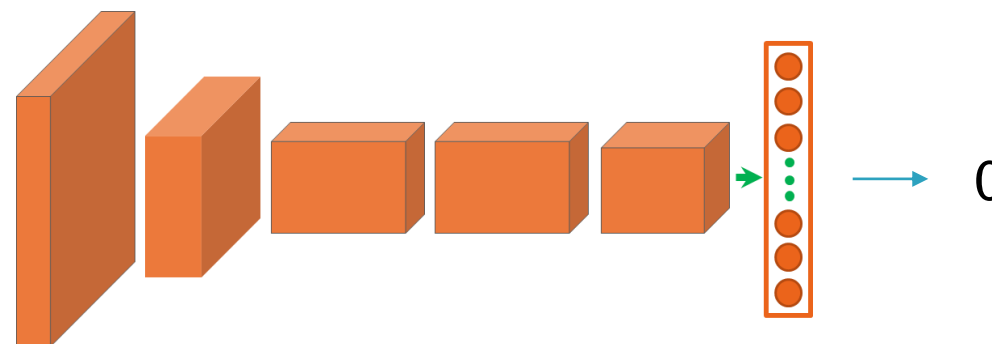
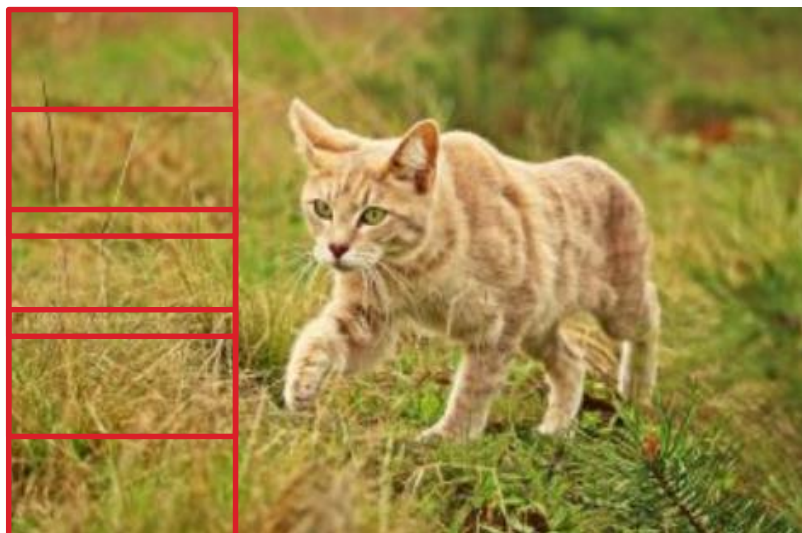
شناسایی اشیاء با پنجره لغزان



شناسایی اشیا: پنجره لغزان

۱۳

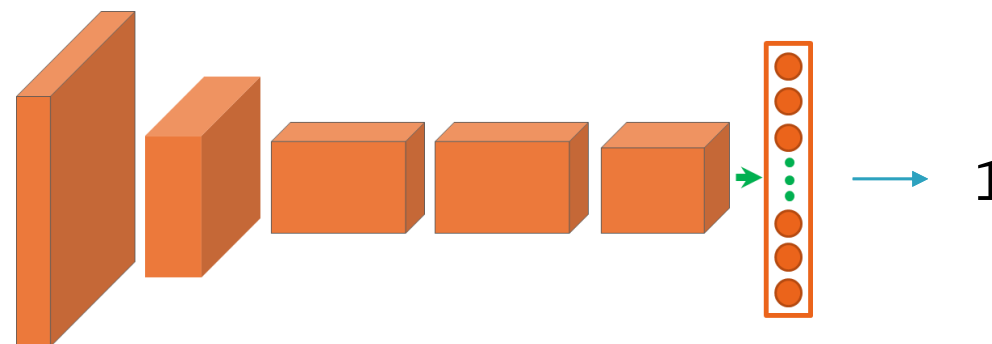
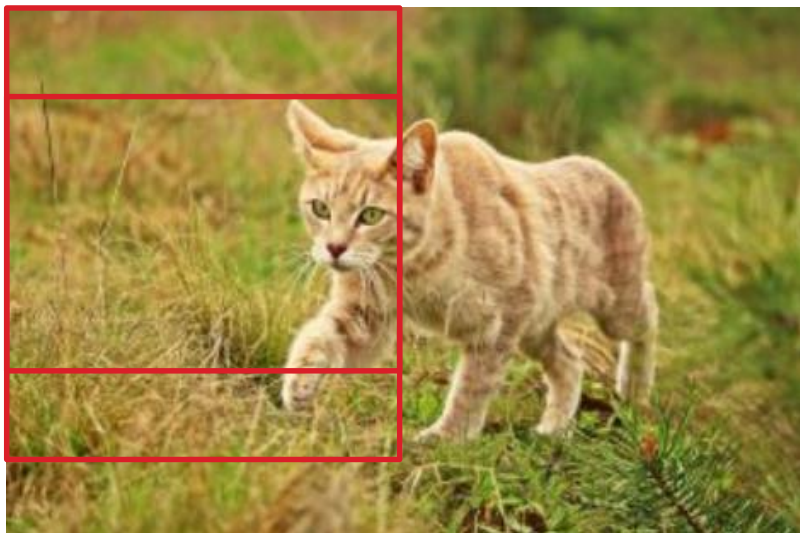
شناسایی اشیا با پنجره لغزان



شناسایی اشیاء: پنجره لغزان

۱۴

شناسایی اشیاء با پنجره لغزان

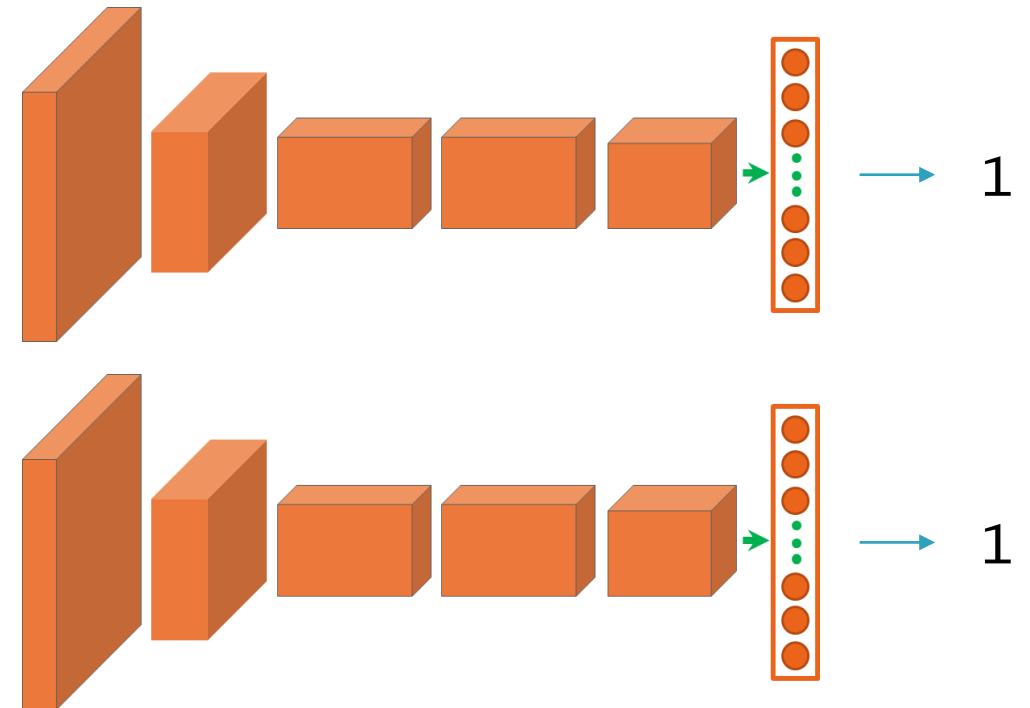
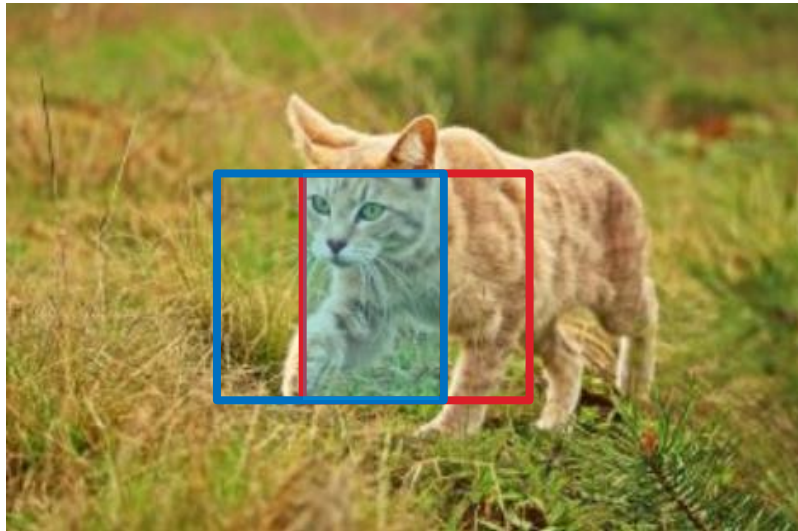


هزینه محاسباتی بسیار بالا!

پنجره لغزان: کاهش هزینه‌های محاسباتی

۱۵

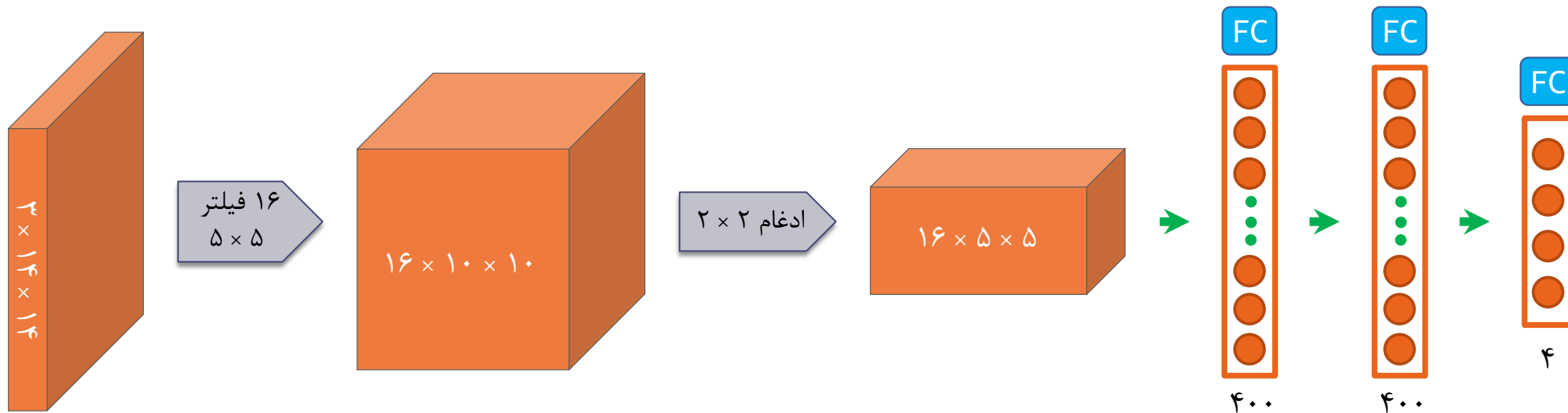
شناسایی اشیاء با پنجره لغزان



پنجره لغزان: پیاده‌سازی کانولوشنی

۱۶

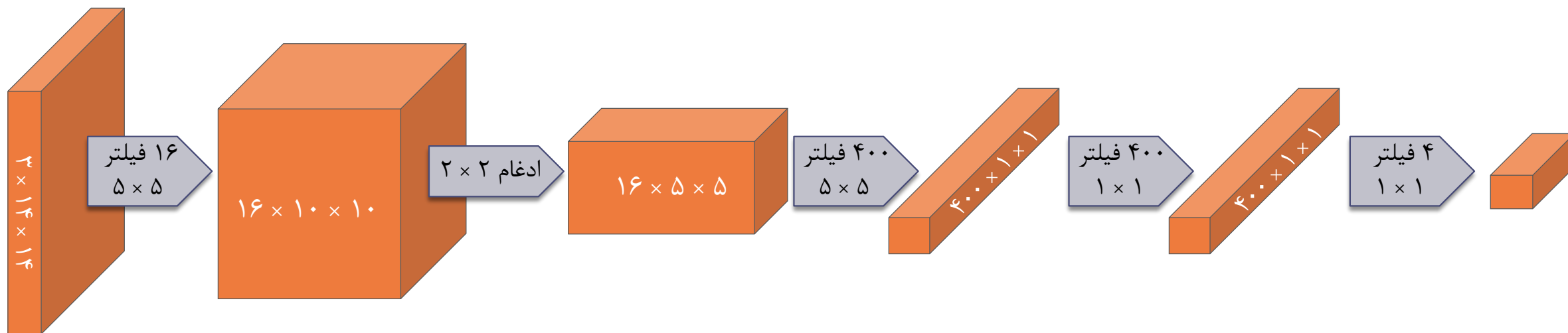
□ تبدیل لایه‌های کاملاً متصل به لایه‌های کانولوشنی.



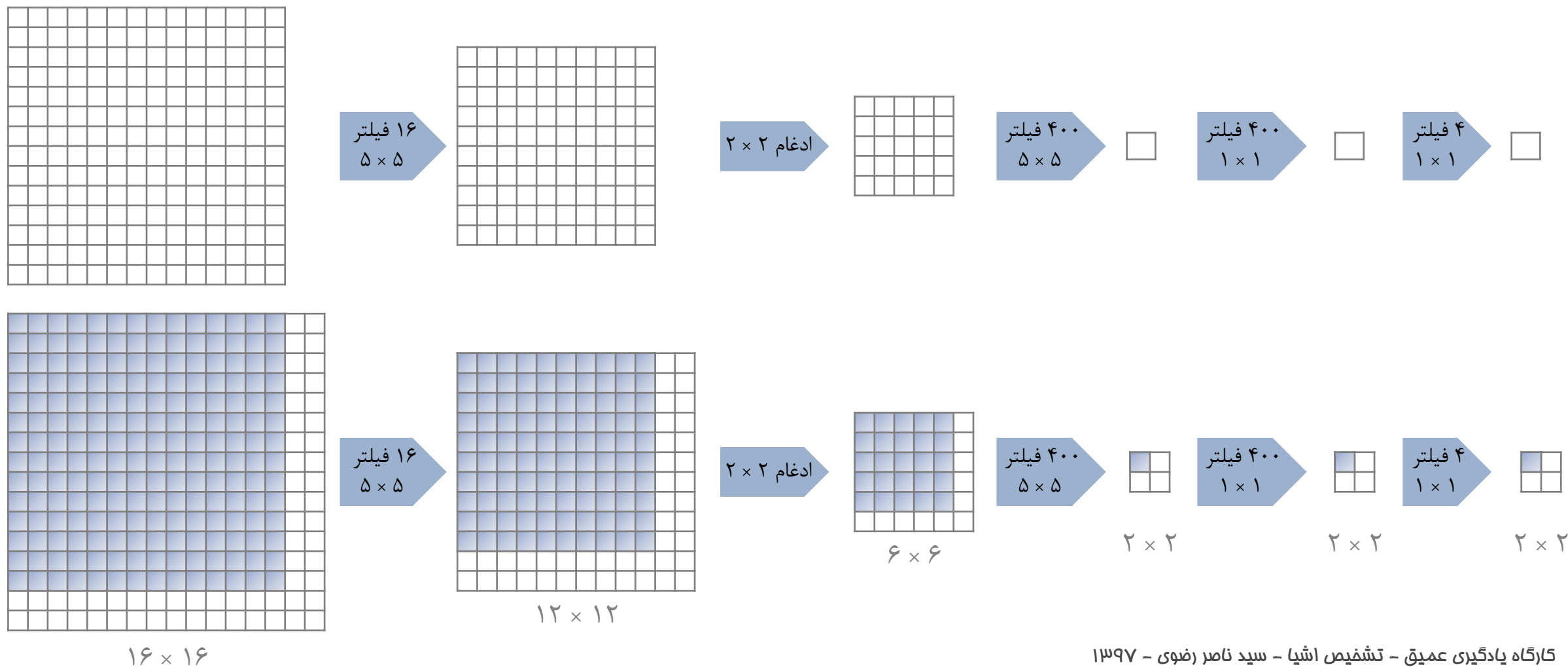
پنجره لغزان: پیاده‌سازی کانولوشنی

۱۷

□ تبدیل لایه‌های کاملاً متصل به لایه‌های کانولوشنی.

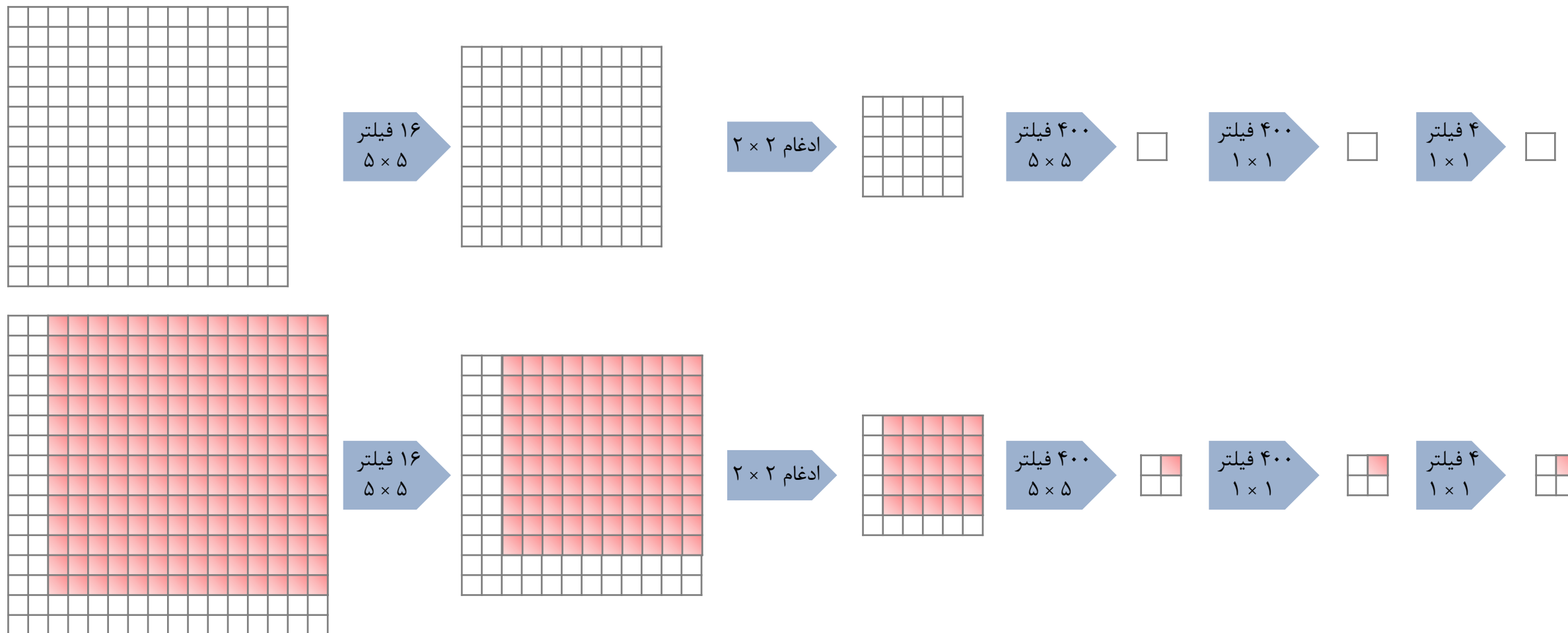


پنجره لځزان: پیاډه‌سازی کانونلوشنی



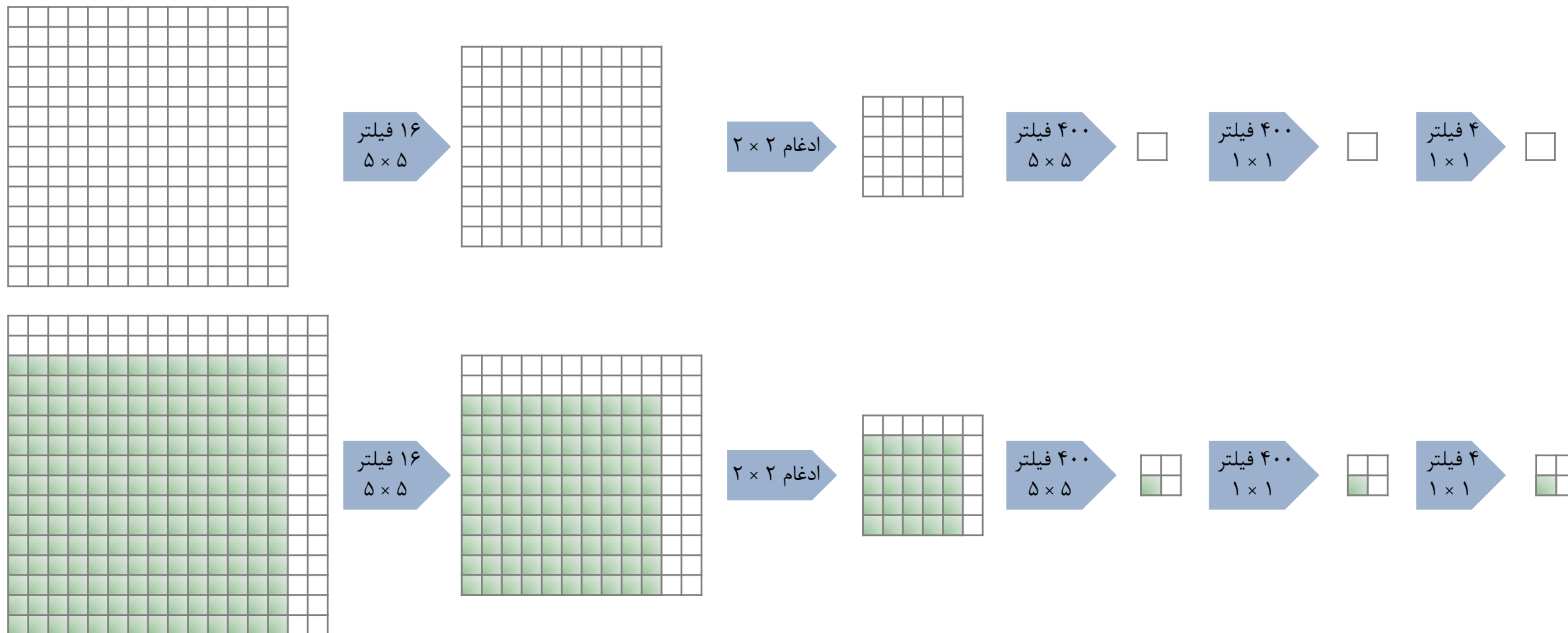
پنجره لغزان: پیاده‌سازی کانولوشنی

۱۹



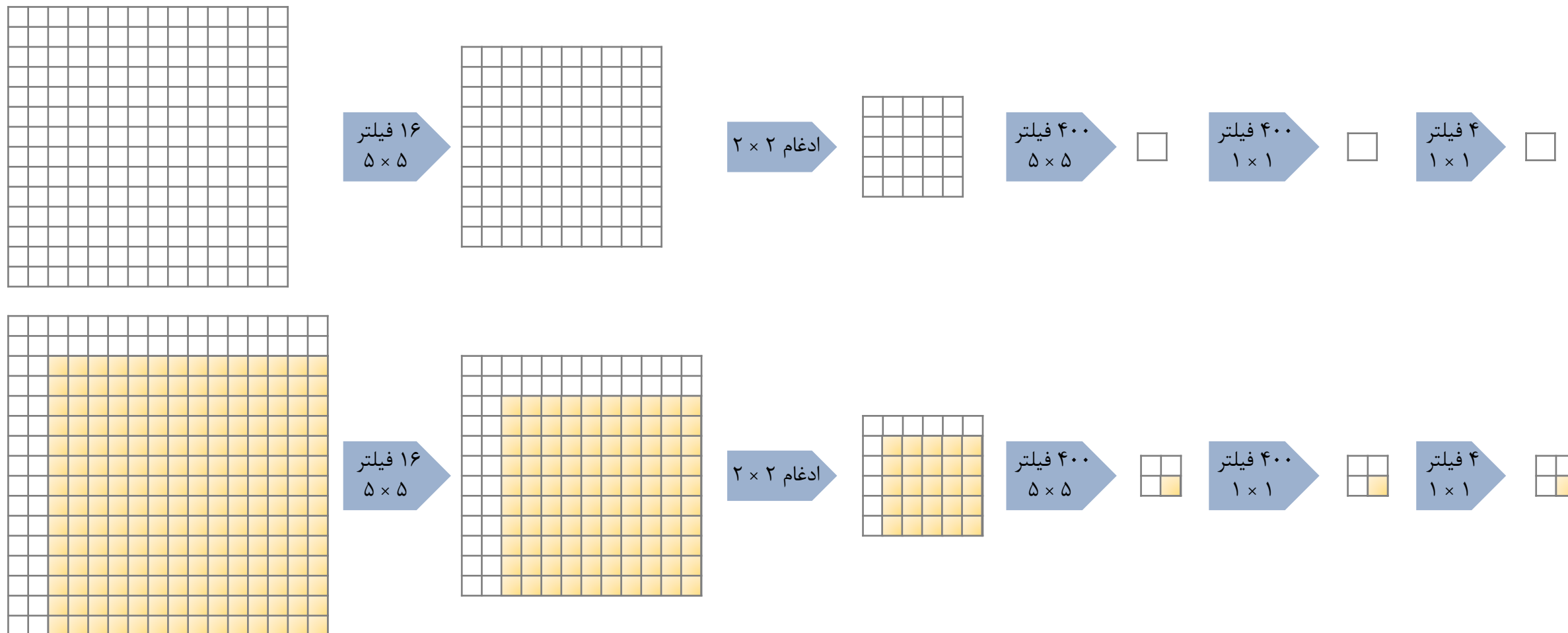
پنجره لغزان: پیاده‌سازی کانولوشنی

۲۰



پنجره لغزان: پیاده‌سازی کانولوشنی

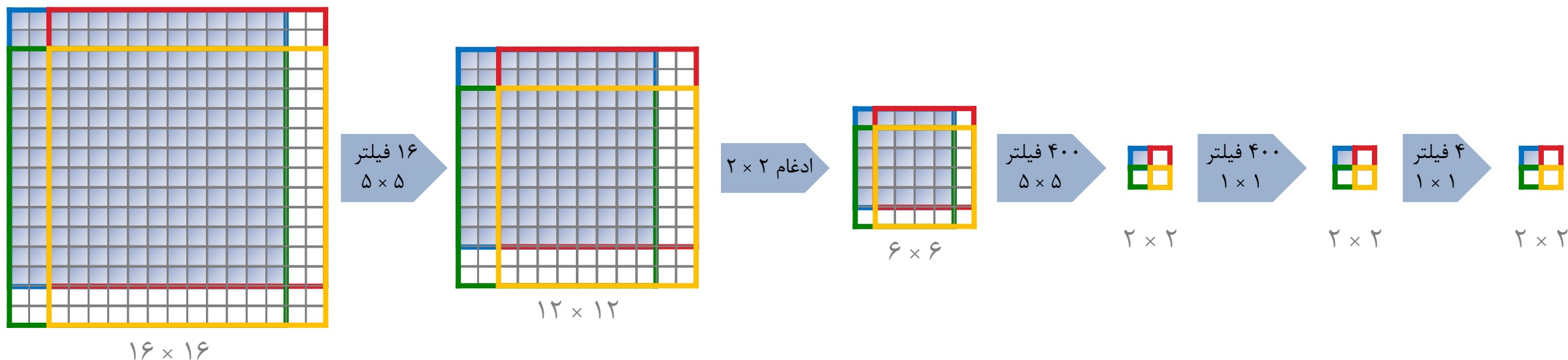
۲۱



پنجره لغزان: پیاده‌سازی کانولوشنی

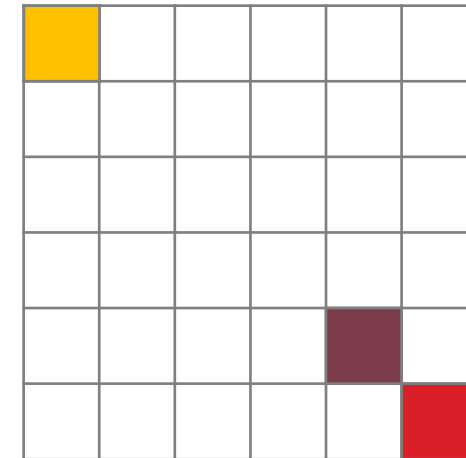
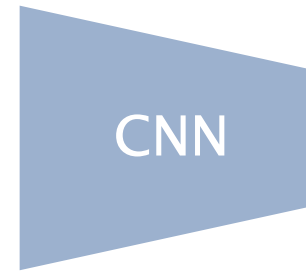
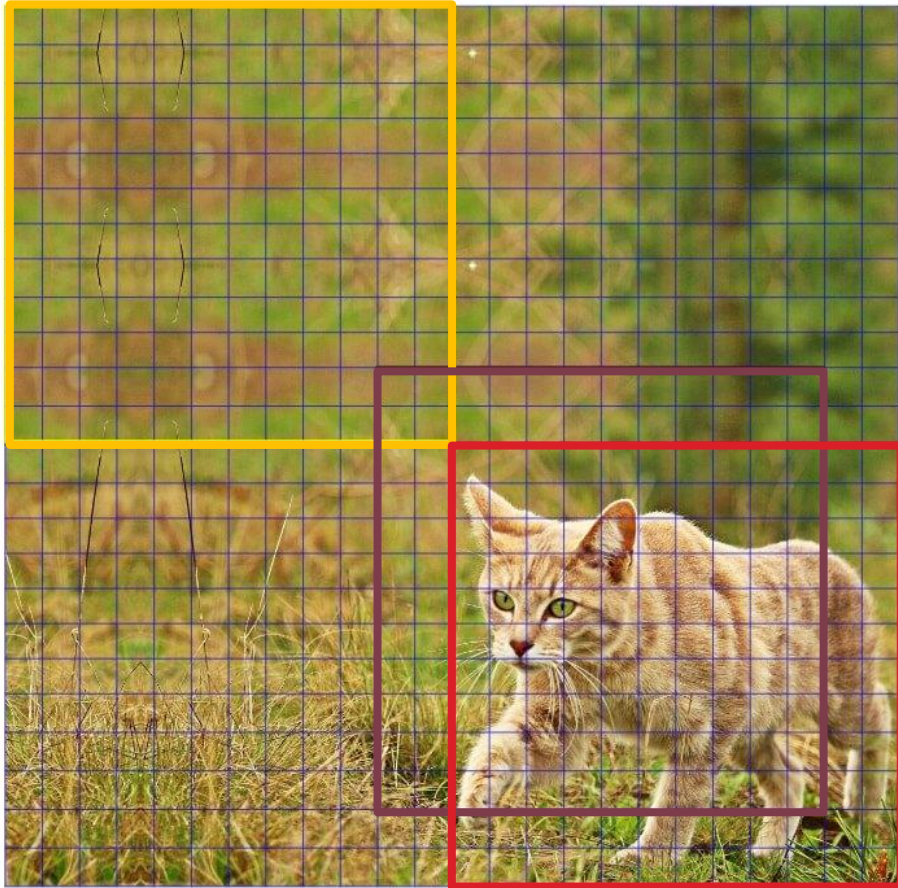
۲۲

انجام محاسبات مربوط به پنجره لغزان به صورت موازی و همزمان!



پنجره لغزان: پیاده‌سازی کانولوشنی

۲۳



احتمال وجود اشیا
در هر یک از پنجره‌ها

اشتراک بر روی اجتماع

۲۴

□ معیاری به منظور محاسبه میزان هم‌پوشانی دو جعبه محاطی.

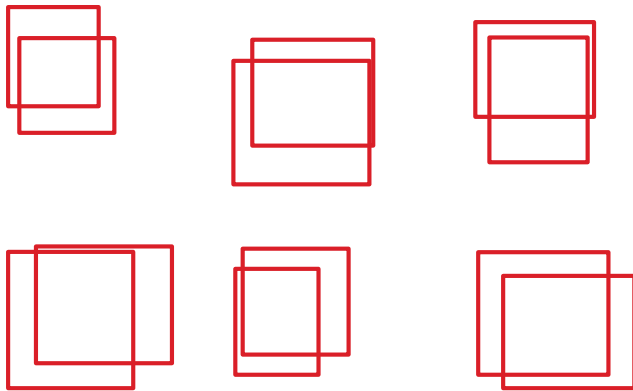


$$IoU = \frac{|B_1 \cap B_2|}{|B_1 \cup B_2|} \geq 0.5$$

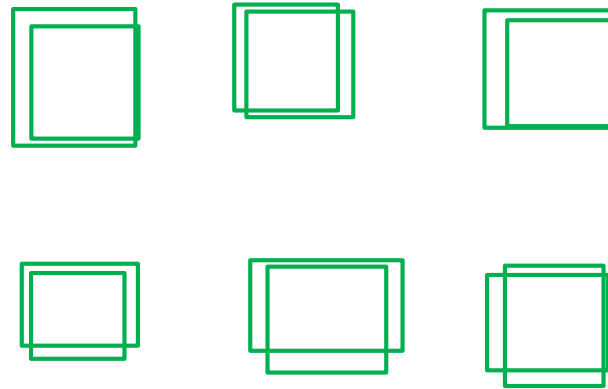
اشتراک بر روی اجتماع

۲۵

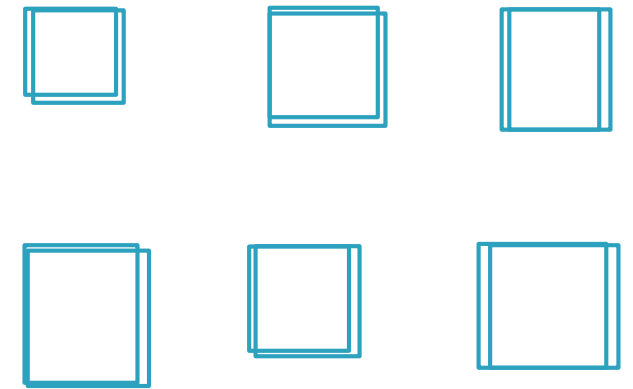
□ معیاری به منظور محاسبه میزان هم‌پوشانی دو جعبه محاطی.



$\text{IoU} = 0.5$



$\text{IoU} = 0.7$

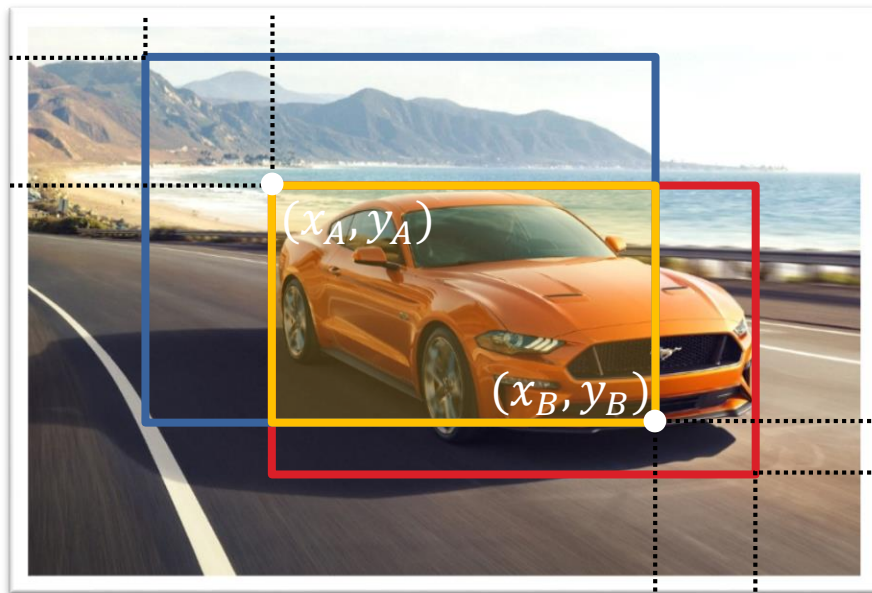


$\text{IoU} = 0.9$

اشتراک بر روی اجتماع

۲۶

□ معیاری به منظور محاسبه میزان هم‌پوشانی دو جعبه محاطی.

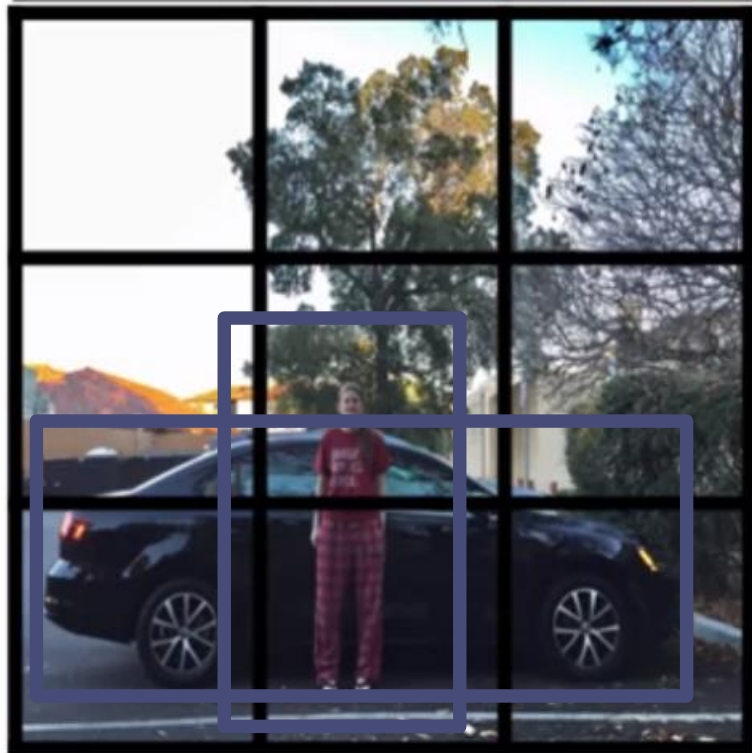


$$IoU = \frac{|B_1 \cap B_2|}{|B_1 \cup B_2|}$$

```
def iou(b1, b2):  
    # determine the coordinates of the intersection rectangle  
    # each box is a list of four numbers like [x1, y1, x2, y2]  
    xA = max(b1[0], b2[0])  
    yA = max(b1[1], b2[1])  
    xB = min(b1[2], b2[2])  
    yB = min(b1[3], b2[3])  
  
    # compute the area of intersection  
    area_intersect = (xB - xA + 1) * (yB - yA + 1)  
  
    # Calculate area of the two boxes  
    area_b1 = (b1[2] - b1[0] + 1) * (b1[3] - b1[1] + 1)  
    area_b2 = (b2[2] - b2[0] + 1) * (b2[3] - b2[1] + 1)  
  
    # compute and return the intersection over union  
    return area_intersect / float(area_b1 + area_b2 - area_intersect)
```

همپوشانی اشیا و جعبه‌های لنگر

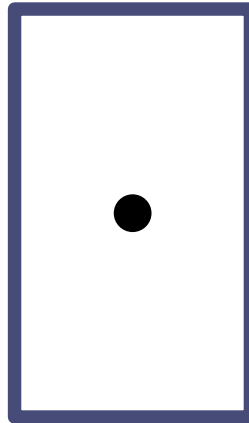
۲۷



$$y = \begin{bmatrix} y_{a1} \\ y_{a2} \end{bmatrix}$$

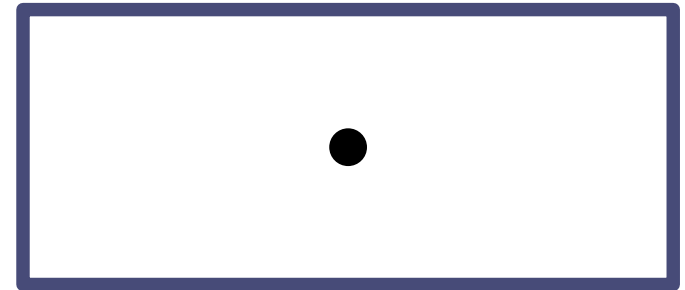
به ازای هر خانه

جعبه لنگر ۱



$$y_{a1} = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ C_1 \\ C_2 \\ C_3 \end{bmatrix}$$

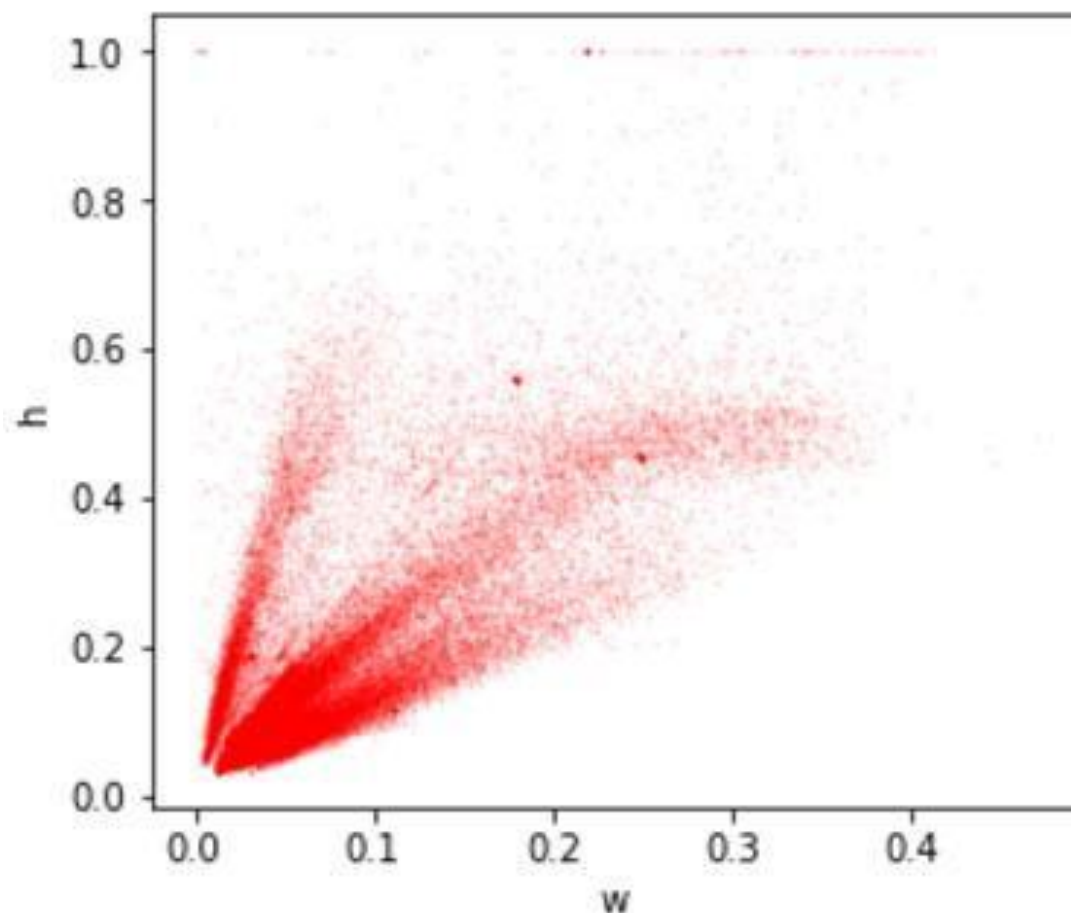
جعبه لنگر ۲



$$y_{a2} = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ C_1 \\ C_2 \\ C_3 \end{bmatrix}$$

جعبه‌های لنگر

۲۸



□ انتخاب جعبه‌های لنگر

□ طراحی جعبه‌های لنگر به صورت دستی

□ استفاده از الگوریتم خوشه‌بندی [۵ خوشه]

$$D(B_1, B_2) = 1 - IoU(B_1, B_2)$$

معیار فاصله

الگوریتم YOLO

۲۹

□ ایجاد مجموعه آموزشی.

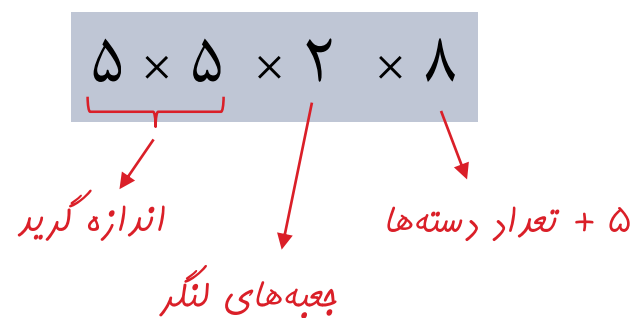
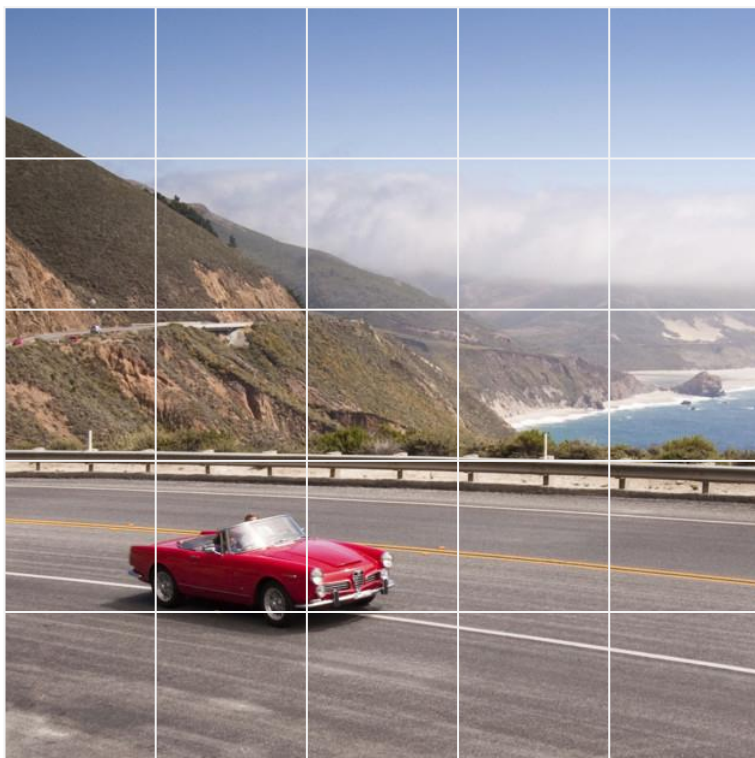
□ دسته‌ها.

۱ - رهگذر

۲ - خودرو

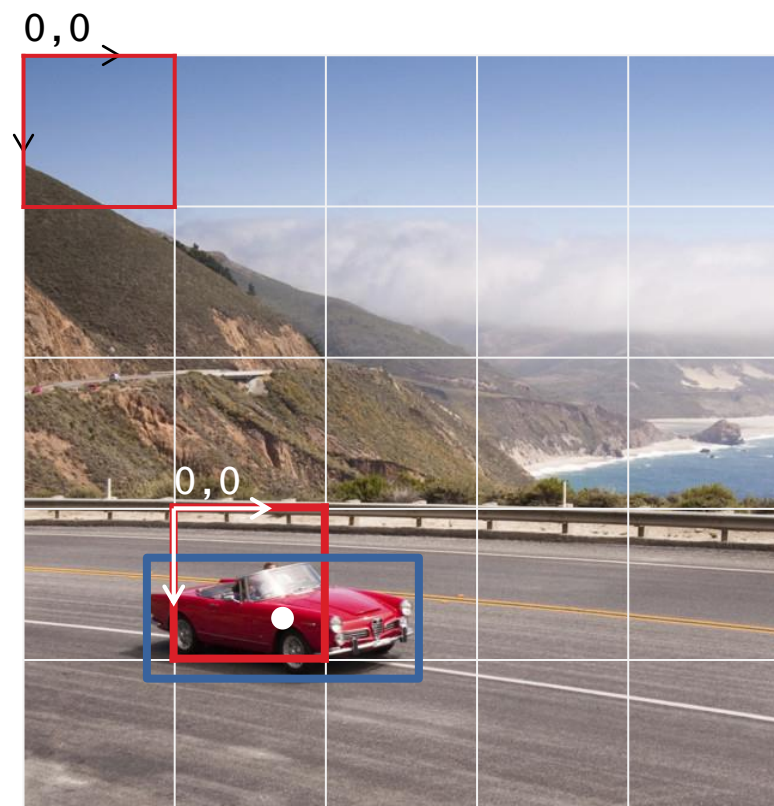
۳ - موتورسیکلت

□ خروجی.



الگوریتم YOLO

۳۰



خانه ۱

$p_c = 0$
 $b_x = 0$
 $b_y = 0$
 $b_h = 0$
 $b_w = 0$
 $C_1 = 0$
 $C_2 = 0$
 $C_3 = 0$
 $p_c = 0$
 $b_x = 0$
 $b_y = 0$
 $b_h = 0$
 $b_w = 0$
 $C_1 = 0$
 $C_2 = 0$
 $C_3 = 0$

خانه ۱۷

$p_c = 0$
 $b_x = 0$
 $b_y = 0$
 $b_h = 0$
 $b_w = 0$
 $C_1 = 0$
 $C_2 = 0$
 $C_3 = 0$
 $p_c = 1$
 $b_x = 0.7$
 $b_y = 0.7$
 $b_h = 0.8$
 $b_w = 1.9$
 $C_1 = 0$
 $C_2 = 1$
 $C_3 = 0$

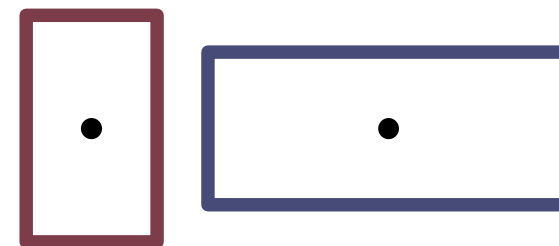
□ ایجاد مجموعه آموزشی.

□ دسته‌ها.

۱ - رهگذر

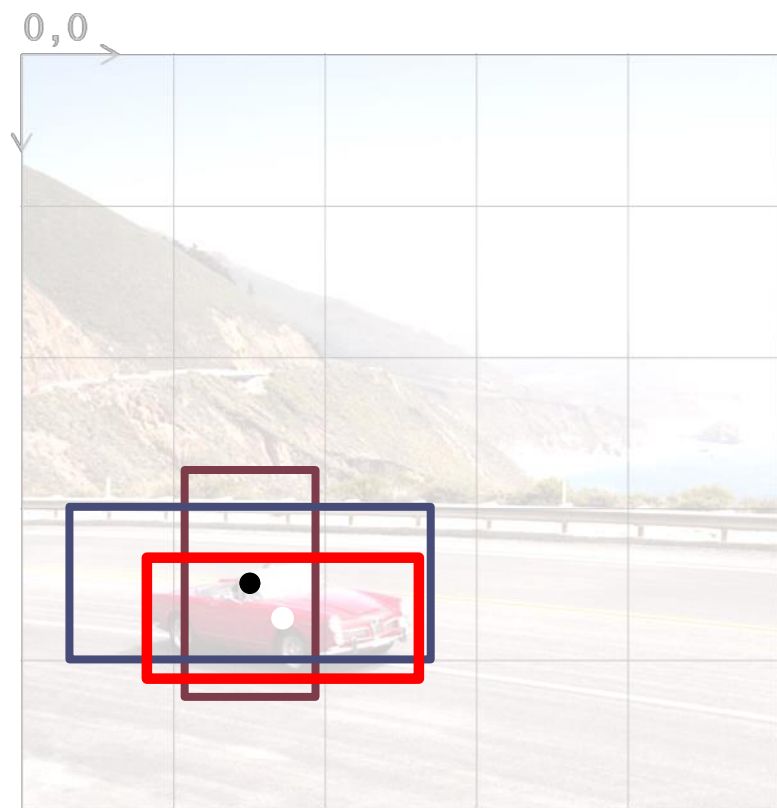
۲ - خودرو

۳ - موتورسیکلت



الگوریتم YOLO

۳۱



$19 \times 19 \times 5 \times 1$

خانه ۱

$p_c = 0$
 $b_x = 0$
 $b_y = 0$
 $b_h = 0$
 $b_w = 0$
 $C_1 = 0$
 $C_2 = 0$
 $C_3 = 0$
 $p_c = 0$
 $b_x = 0$
 $b_y = 0$
 $b_h = 0$
 $b_w = 0$
 $C_1 = 0$
 $C_2 = 0$
 $C_3 = 0$

خانه ۱۷

$p_c = 0$
 $b_x = 0$
 $b_y = 0$
 $b_h = 0$
 $b_w = 0$
 $C_1 = 0$
 $C_2 = 0$
 $C_3 = 0$
 $p_c = 1$
 $b_x = 0.7$
 $b_y = 0.7$
 $b_h = 0.8$
 $b_w = 1.9$
 $C_1 = 0$
 $C_2 = 1$
 $C_3 = 0$

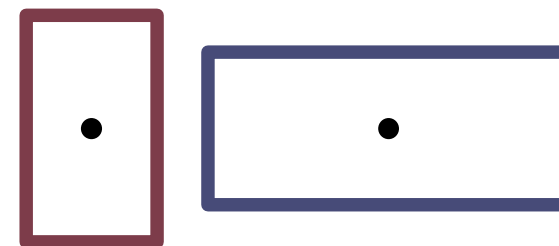
□ ایجاد مجموعه آموزشی.

□ دسته‌ها.

۱ - رهگذر

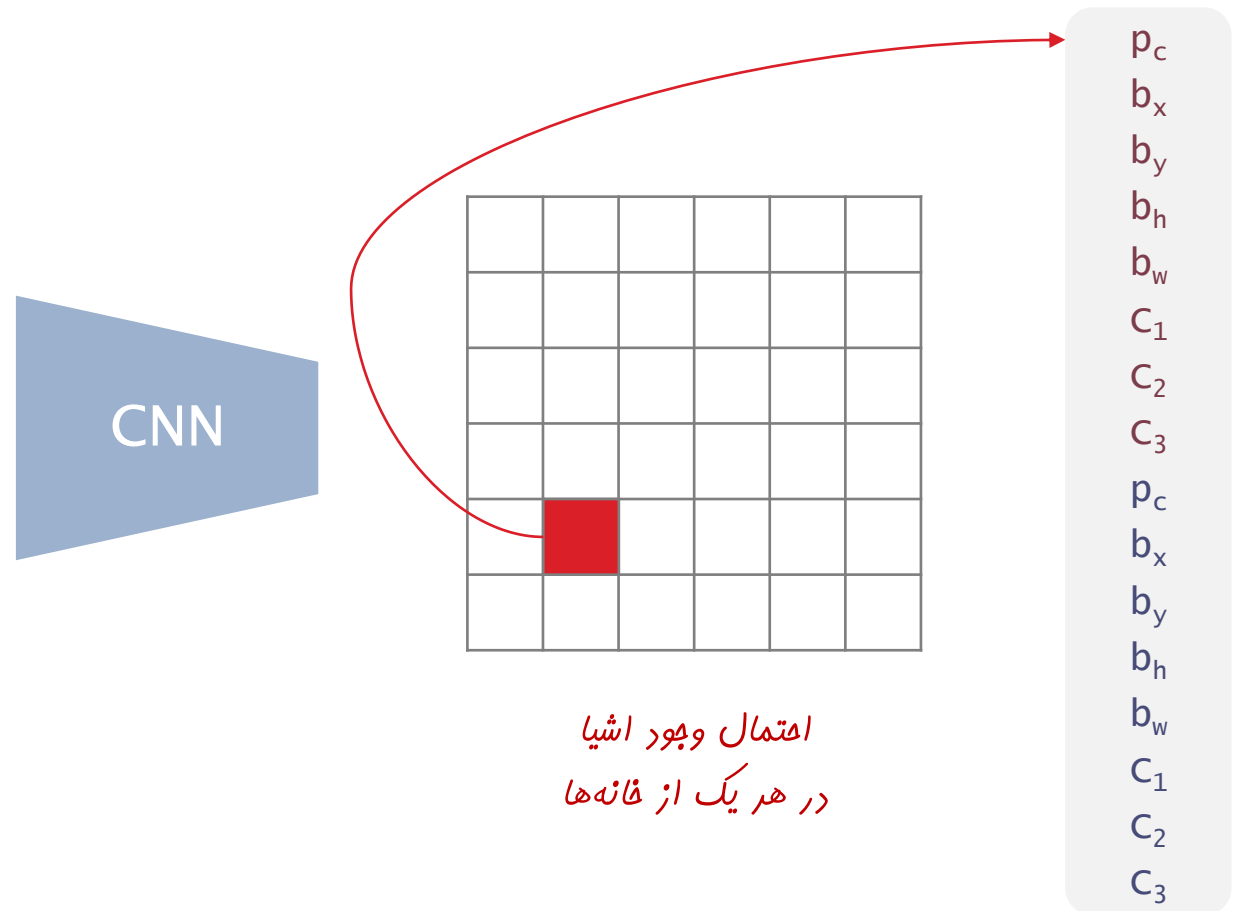
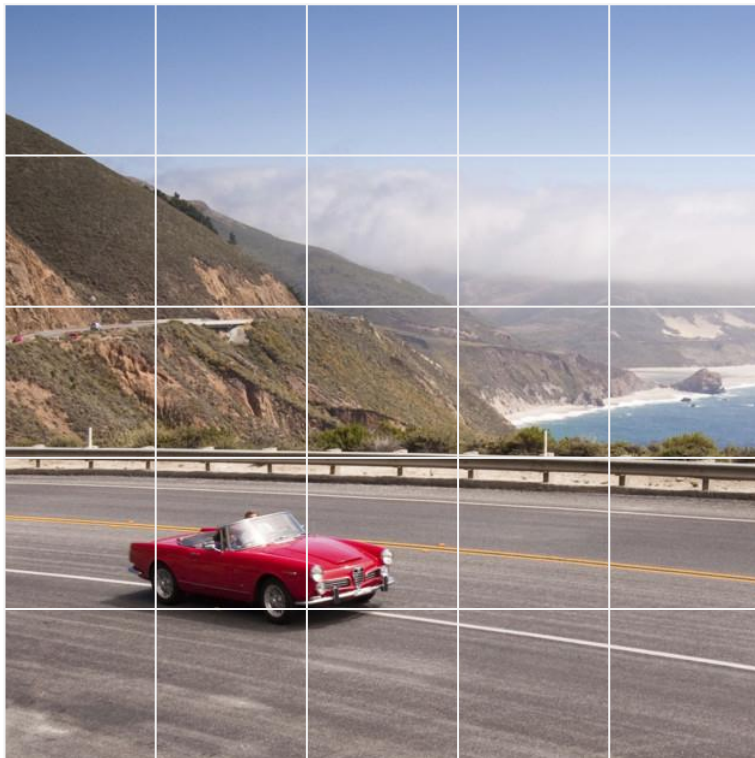
۲ - خودرو

۳ - موتورسیکلت



الگوریتم YOLO: پیش‌بینی

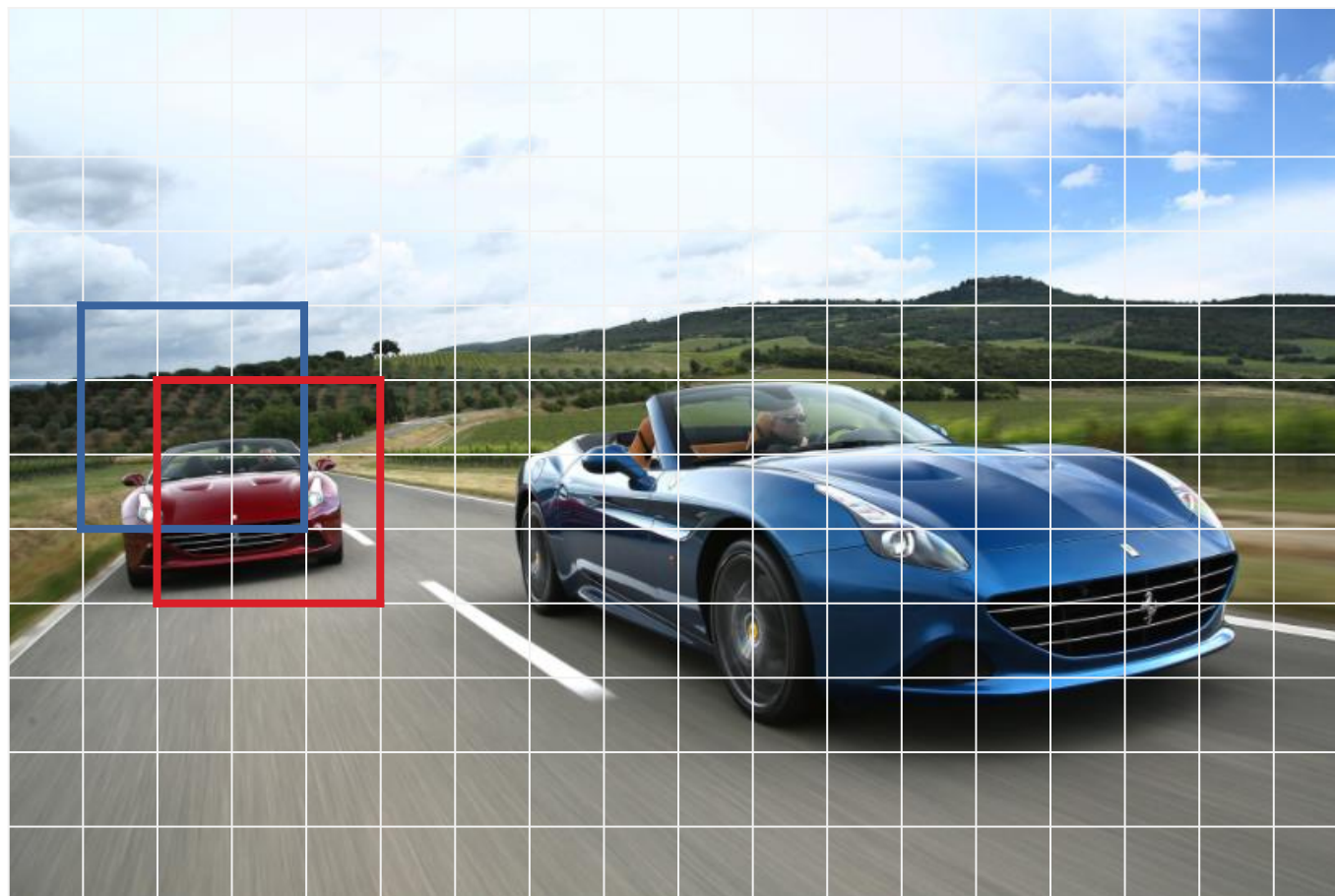
۳۲



سرکوب غیر بیشینه

۳۳

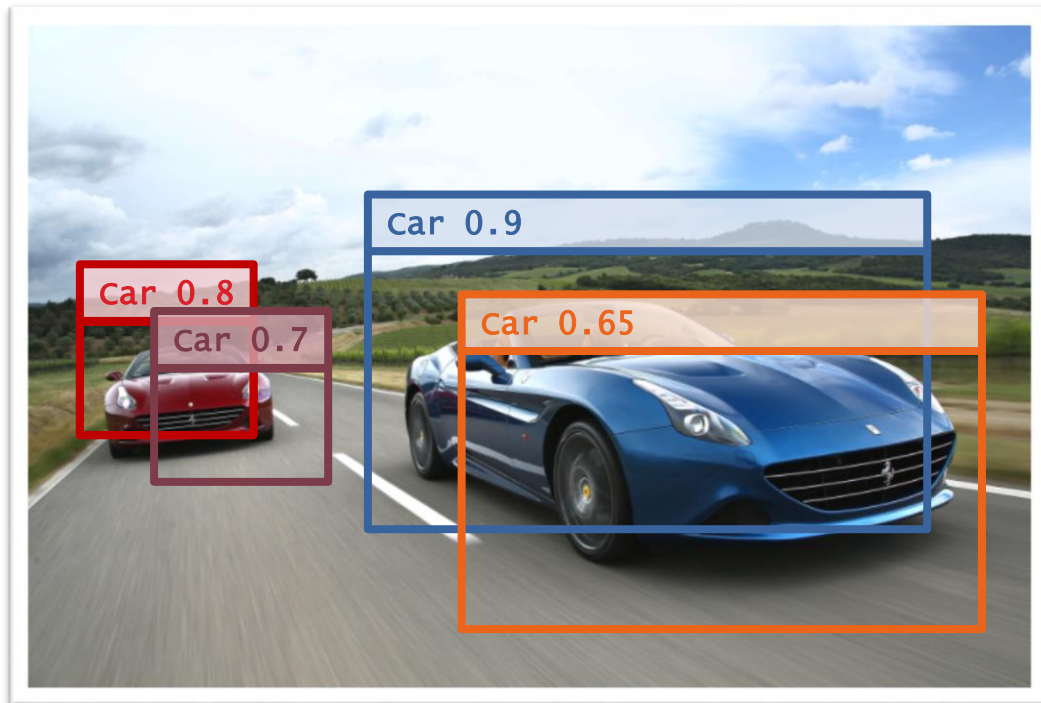
□ اطمینان از این که هر شی، بیش از یک بار شناسایی نمی شود.



الگوریتم YOLO: سرکوب غیر بیشینه

۳۴

□ پیش‌بینی‌هایی را که احتمال کمی دارند، حذف کن. [کمتر از ۰/۶]



□ برای هر دسته [ره‌گذر، خودرو، موتورسیکلت]:

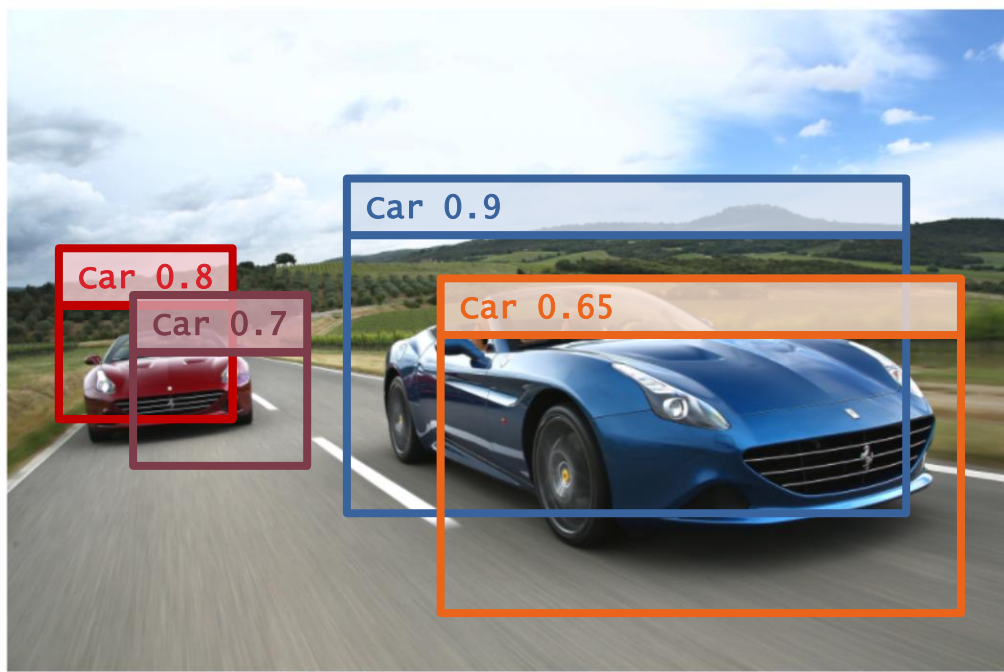
□ تا زمانی که پیش‌بینی دیگری وجود دارد:

- پیش‌بینی با بیشترین احتمال را انتخاب کن.
- این پیش‌بینی را به عنوان خروجی در نظر بگیر.
- تمام پیش‌بینی‌هایی را که هم‌پوشانی بالایی با پیش‌بینی انتخاب شده دارند، حذف کن.

$IoU \geq 0.5$

الگوریتم YOLO: سرکوب غیر بیشینه

۳۵



```
# Iterate each class
for c in range(classes):
    # Sort boxes according to their prob for class c
    boxes.sort(key=lambda box: box[0][c], reverse=True)
    # Iterate each box
    for i in range(len(boxes) - 1):
        box_i = boxes[i]
        if box_i[0][c] == 0:
            continue
        for box_j in boxes[i + 1:]:
            # Take iou threshold into account
            if iou(box_i[1], box_i[2], box_j[1], box_j[2]) >= threshold:
                box_j[0][c] = 0
    return boxes
```

معماری Yolo2

۳۶

Darknet

