

Detecção de Objetos

(single-stage object detector)

Prof. Jefersson A. dos Santos

jefersson@dcc.ufmg.br

Roteiro

Aula anterior

- Bounding boxes
- Detecção de pontos de referências
- Detecção com janelas deslizantes
- CNNs baseadas em proposta de regiões
 - R-CNN
 - Fast R-CNN (com RoI pooling)
 - Faster R-CNN (com RPN)
- Instance Segmentation - Mask R-CNN

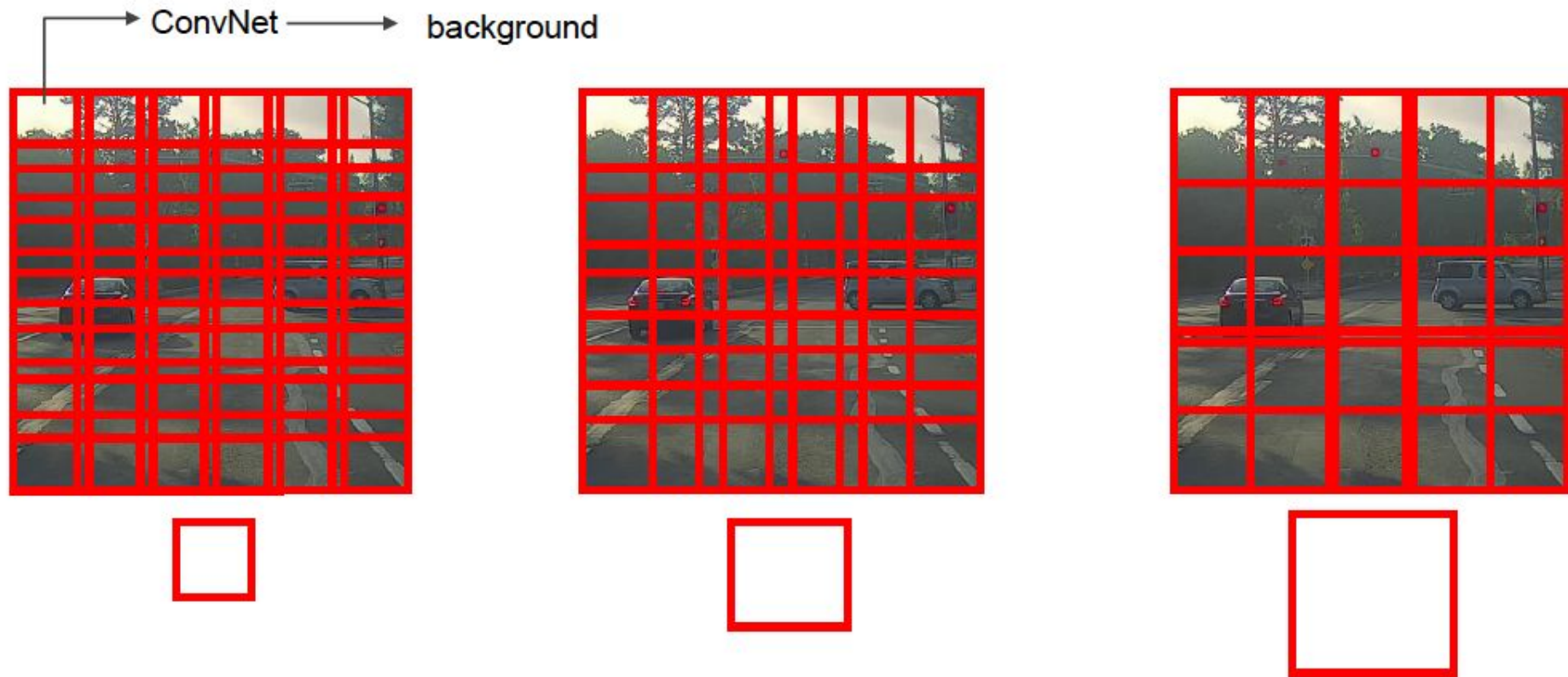
Aula de hoje

- Implementação convolucional de janelas deslizantes
- Razão entre interseção e união (IoU)
- Non-max suppression
- Anchor boxes
- Juntando tudo: YOLO
Algoritmo You Only Look Once

Implementação Convolutacional de Janelas Deslizantes

Uma alternativa a janelas deslizantes


Relembrando: Sliding windows detection



Detecção por janelas deslizantes

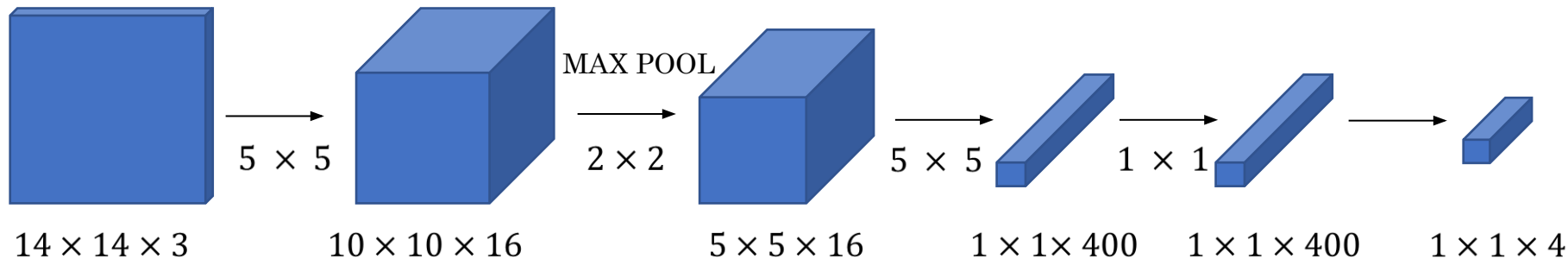
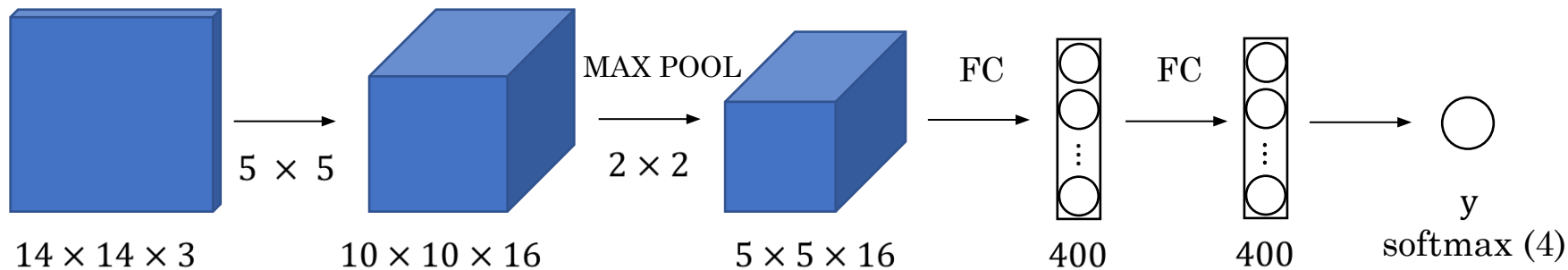
- Começa com uma janela pequena que desliza da esquerda para a direita, de cima para baixo, de acordo com o stride escolhido. Para cada janela, usamos a ConvNet para retornar uma previsão.
- Recomeça o mesmo algoritmo, mas com janelas maiores.
- Problema: custo computacional
 - Podemos aumentar o stride, mas desempenho pode cair
 - Antigamente, quando as features eram hand-engineered, isso não era um problema
- Soluções alternativas:
 - não classificar todas as regiões (proposta de regiões)
 - implementação convolucional das janelas deslizantes

Detecção por janelas deslizantes

- Começa com uma janela pequena que desliza da esquerda para a direita, de cima para baixo, de acordo com o stride escolhido. Para cada janela, usamos a ConvNet para retornar uma previsão.
- Recomeça o mesmo algoritmo, mas com janelas maiores.
- Problema: custo computacional
 - Podemos aumentar o stride, mas desempenho pode cair
 - Antigamente, quando as features eram hand-engineered, isso não era um problema
- Soluções alternativas:
 - não classificar todas as regiões (proposta de regiões) 
 - **implementação convolucional das janelas deslizantes**

Region-based
methods (última aula)

Turning FC layer into convolutional layers



Tornando camada FC em camada convolucional

1a. camada FC:

- Camada $5 \times 5 \times 16$ é achatada (flattened) resultando em 400 nós, que são ligados com outros 400 nós
- Cada um dos 400 nós da camada FC é uma combinação linear do volume $5 \times 5 \times 16$ seguido de ativação

2a. camada FC:

- Cada um dos 400 nós da 2a camada FC é uma combinação dos 400 nós da 1a FC seguido de ativação

Última camada:

- Combina as ativações da camada anterior em 4 nós + ativação softmax

1a. camada convolucional $1 \times 1 \times 400$:

- Cada um dos 400 filtros é $5 \times 5 \times 16$
- Cada nó da camada convolucional é uma combinação linear do volume $5 \times 5 \times 16$ seguido de ativação

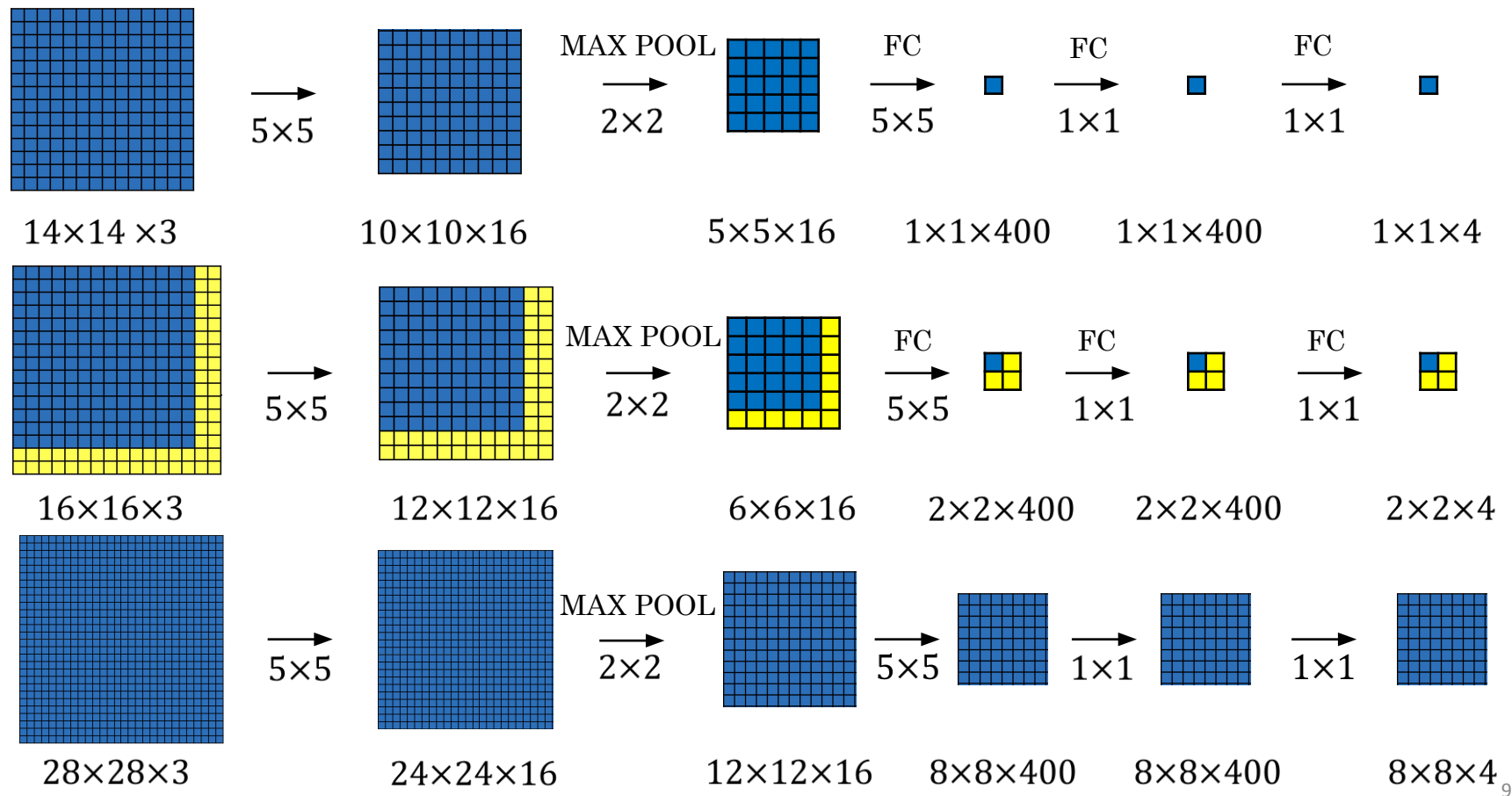
2a. camada convolucional $1 \times 1 \times 400$:

- Cada um dos 400 filtros é $1 \times 1 \times 400$
- Cada um dos 400 nós da 2a camada conv. $1 \times 1 \times 400$ é uma combinação dos 400 nós da 1a. camada conv. Seguido de ativação

Última camada:

- Cada um dos 4 filtros é $1 \times 1 \times 400$
- Ativação softmax

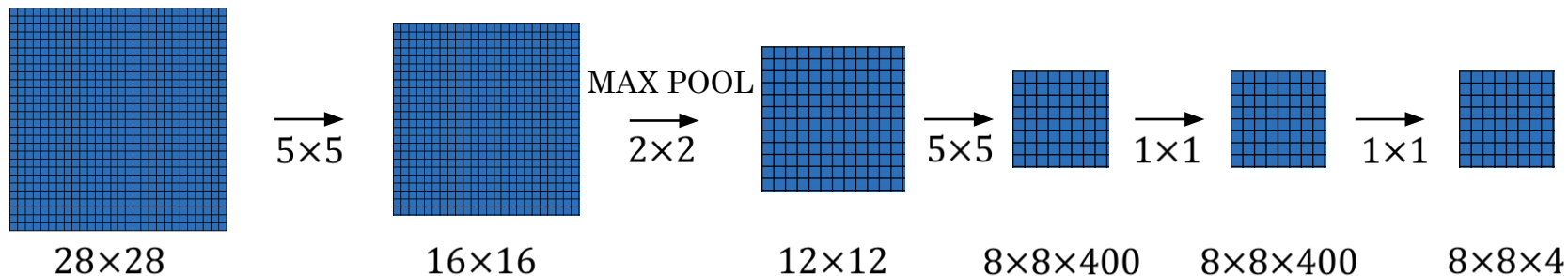
Convolution implementation of sliding windows



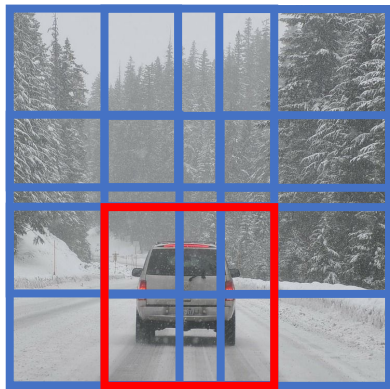
Implementação convolucional da janela deslizante

- Números pequenos e face frontal dos volumes para simplificar
- Tamanhos:
 - Imagem de treinamento: $14 \times 14 \times 3$
 - Imagem de teste: $16 \times 16 \times 3$
- Suponha uso de janelas deslizantes com stride 2 (4 janelas)
- Usar janelas deslizantes causa muita computação redundante
 - Resultado ao passar imagem de teste pela rede: $2 \times 2 \times 4$
 - Cada volume $1 \times 1 \times 4$ corresponde a uma janela na imagem original
- Quando teste é $28 \times 28 \times 3$, resultado é $8 \times 8 \times 4$
- O stride da janela na imagem original está relacionado ao max-pool 2×2

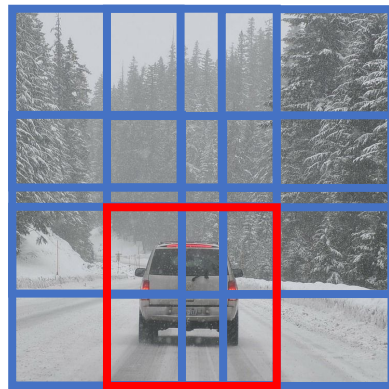
Convolution implementation of sliding windows



Janela por janela



Implementação
convolucional



Razão entre interseção e união *(Intersection over Union)*

Razão entre interseção e união

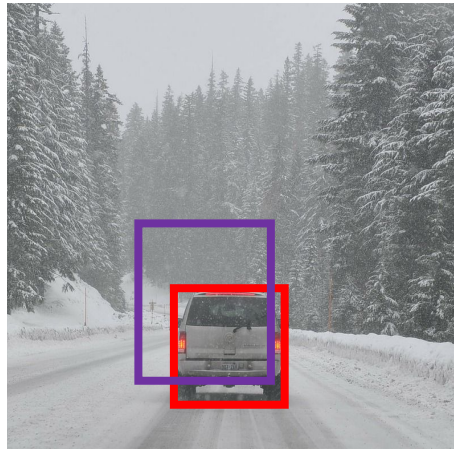
- Como avaliar se o algoritmo de detecção de objetos está funcionando bem?
- Métrica IoU (intersection over union): razão entre interseção e união
 - **0** indica sem sobreposição
 - **1** indica idênticos
 - É um caso especial do índice de **Jaccard**

Dados dois conjuntos A e B

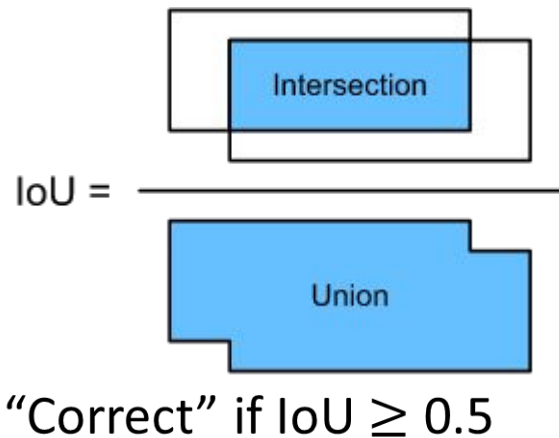
$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

- Também usado como componente de certos algoritmos de detecção

Avaliando a localização do objeto



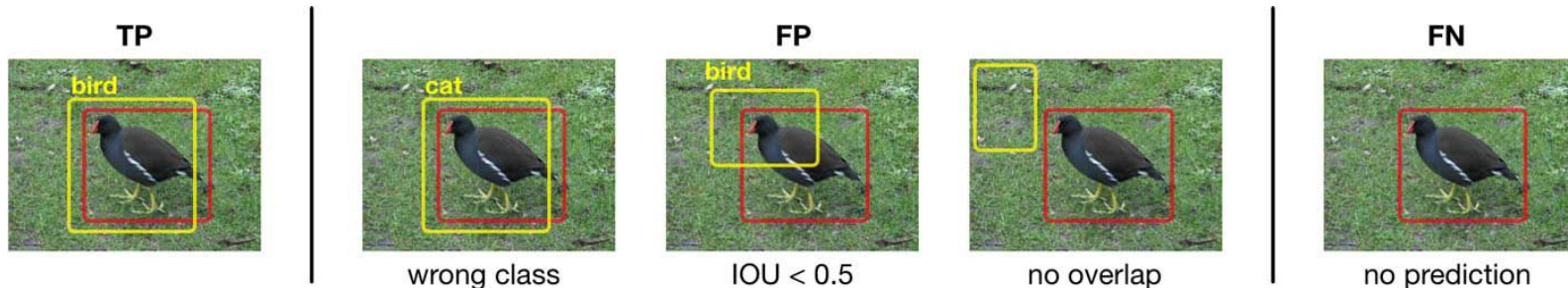
Intersection over Union (IoU)



Mais genericamente, IoU é uma medida de sobreposição entre duas bounding boxes.

Avaliando a localização do objeto

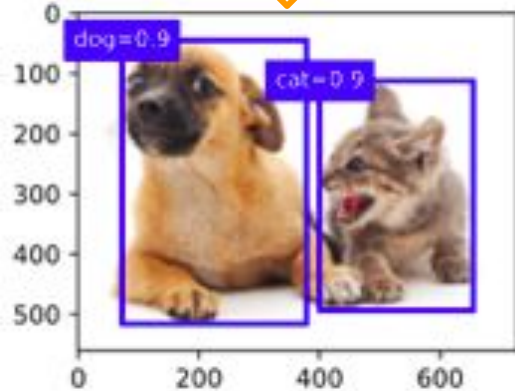
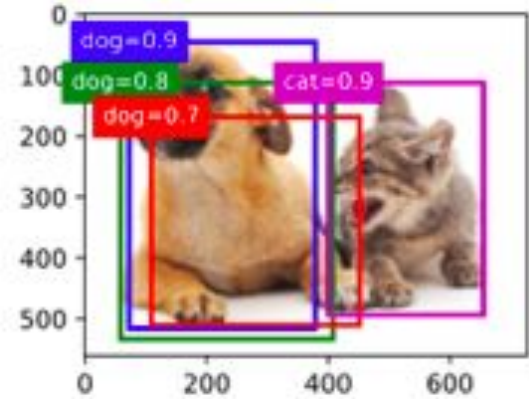
- Região retornada: contorno **roxo**
- Região correta: contorno **vermelho**
- Na maioria das tarefas região retornada é correta se $IoU \geq 0.5$
 - Em alguns casos o limiar é 0.6 ou 0.7
 - Raramente limiar fica abaixo de 0.5
- Pode ser usado para avaliar a qualidade do bounding box retornado
- De maneira mais geral, é uma medida de overlap entre regiões



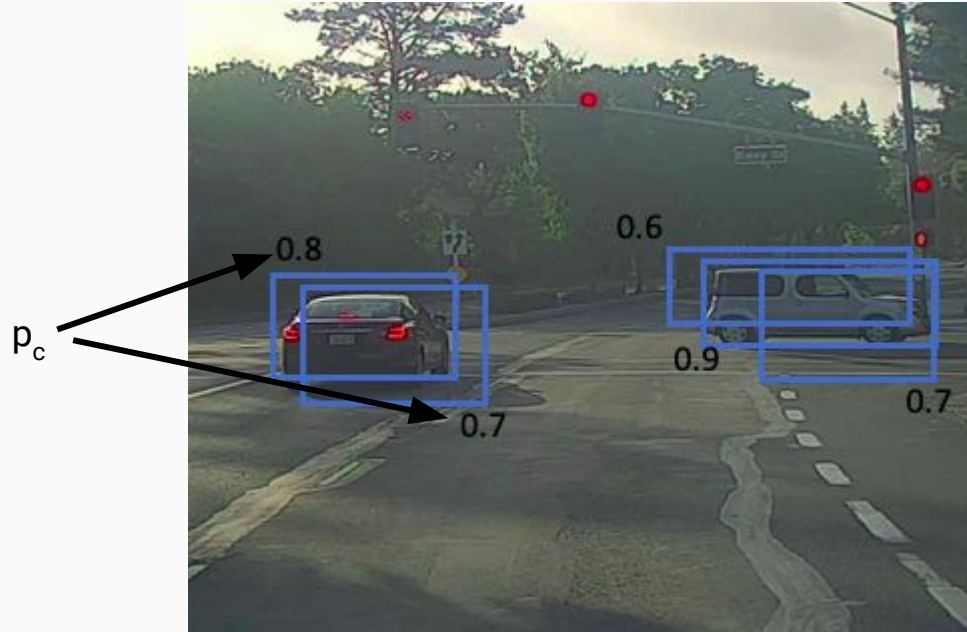
Non-max suppression

Problema com algoritmos vistos até agora

- Pode encontrar múltiplas bounding boxes se referindo ao mesmo objeto
- Non-max suppression permite detectar o objeto uma única vez



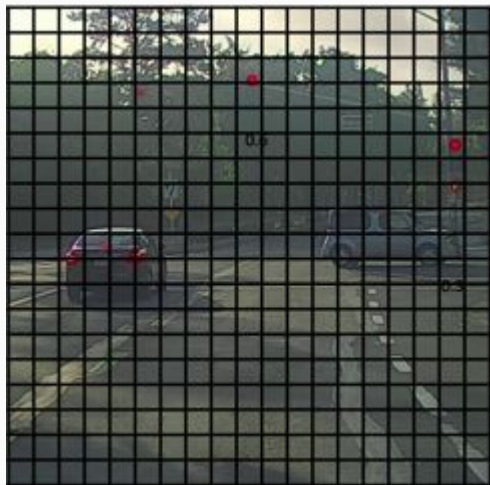
Exemplo de non-max suppression



Exemplo de non-max suppression

- Non-max suppression tem como objetivo retornar apenas uma caixa por objeto
- Primeiro, avalia p_c para cada uma das bounding boxes
 - Pega a caixa com maior p_c
 - Todas as caixas com alto IoU são descartadas
- Continua pegando a bounding box com maior p_c dentre as restantes, até que cada caixa tenha sido selecionada ou removida
- No exemplo, as duas predições correspondem as caixas com $p_c = 0.9$ e 0.8

Algoritmo de non-max suppression



Uma classe

A saída de cada previsão é:

$$\begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \end{bmatrix}$$

Descarte todas as caixas com $p_c \leq 0.6$

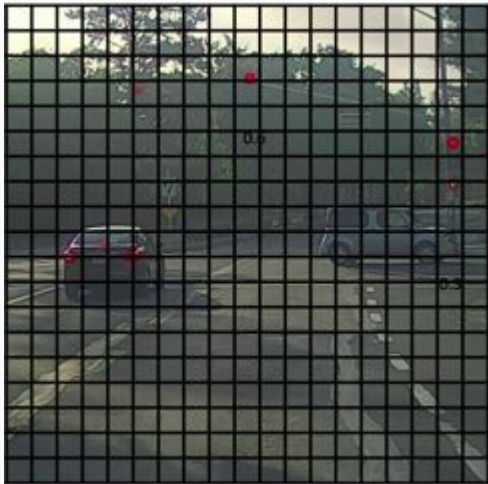
Enquanto existirem caixas restando:

- Pegue a caixa B com maior p_c
Retorne B como previsão
- Descarte qualquer caixa restante
com $\text{IoU} \geq 0.5$ com a caixa B
retornada no passo anterior

Algoritmo de non-max suppression

- No exemplo anterior, estamos assumindo apenas uma classe por simplicidade
- Caso haja mais de uma classe,
 - O vetor de saída y tem número de coordenadas igual a $5 + \text{no. de classes}$
 - Cada classe deve ser avaliada de forma independente

Algoritmo de non-max suppression



3 classes

A saída de cada previsão é:

$$\begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Prob. de existir um objeto da classe i é $p_c \cdot c_i$

Fazer para cada classe i de forma independente

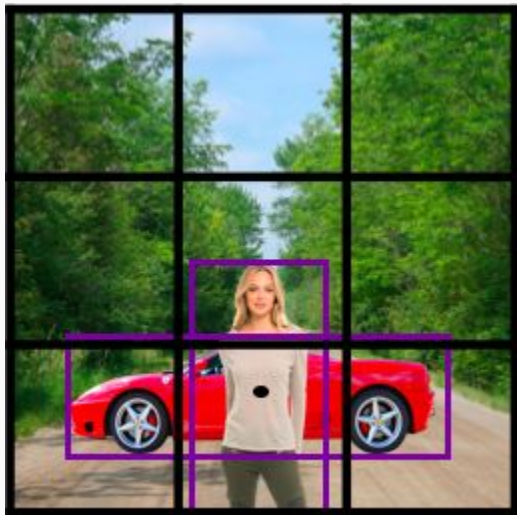
Anchor boxes



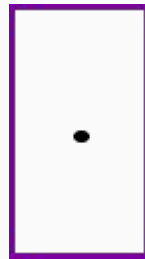
Como detectar múltiplos objetos na mesma janela?

- Algoritmos de detecção de objetos normalmente amostram muitas regiões da imagem de entrada e ajustam as bordas para retornar bounding boxes precisas
- Diferentes modelos podem usar diferentes métodos para amostrar regiões
- Veremos como usar *anchor boxes* (caixas âncoras/caixas candidatas) para isso
- Anchor boxes são bounding boxes de múltiplos tamanhos e *aspect ratios*

Objetos sobrepostos



Anchor box 1



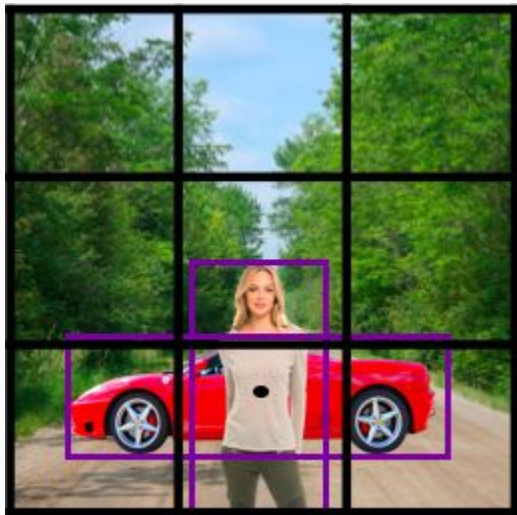
Anchor box 2



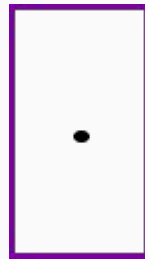
$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

[Redmon et al., 2015, You Only Look Once: Unified real-time object detection]

Objetos sobrepostos



Anchor box 1



Anchor box 2



$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

$$y = \left\{ \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \right\} \begin{matrix} \text{Anchor box 1} \\ \\ \end{matrix}$$
$$y = \left\{ \begin{bmatrix} p_c \\ b_x \\ \vdots \\ c_3 \end{bmatrix} \right\} \begin{matrix} \text{Anchor box 2} \end{matrix}$$

[Redmon et al., 2015, You Only Look Once: Unified real-time object detection]

Objetos sobrepostos

- Não é possível retornar dois objetos usando o vetor y definido anteriormente
- Saída: usar anchor boxes com formatos diferentes
- Vimos exemplo com 2 anchor boxes, mas na prática são usadas ~5 a 10.

No nosso exemplo:

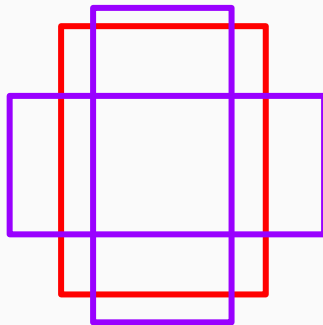
- Usamos a primeira para codificar o pedestre
- Usamos a segunda para codificar o carro

Algoritmo Anchor Box

Antes (sem anchor boxes):

- Cada objeto na imagem de treino era associado à célula da grade que contém o ponto médio do objeto

Saída y:
3 x 3 x 8

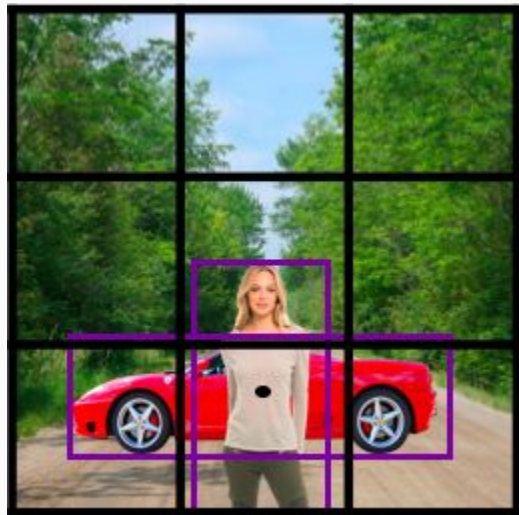


Agora (com 2 anchor boxes por região):

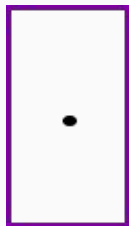
- Cada região contendo múltiplas anchor boxes é um exemplo de treino
- Cada objeto na imagem de treino, delimitado por uma bounding box, é associado à anchor box com maior IoU.

Saída y:
3 x 3 x 16 ou
3 x 3 x 2 x 8

Exemplo com anchor boxes



Anchor box 1



Anchor box 2

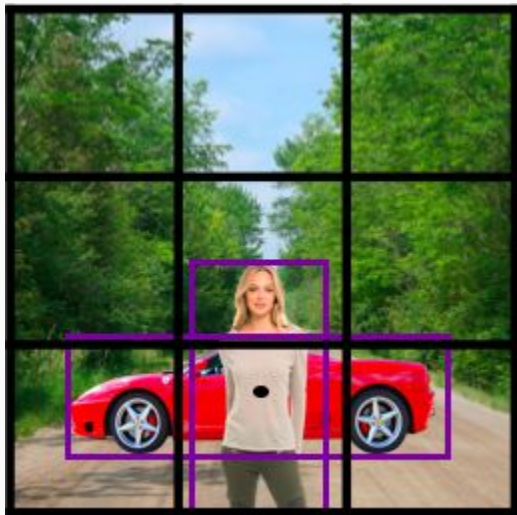


Esta imagem

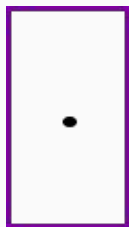
E se só houvesse carro?

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Exemplo com anchor boxes



Anchor box 1



Anchor box 2



Esta imagem

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

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

E se só houvesse carro?

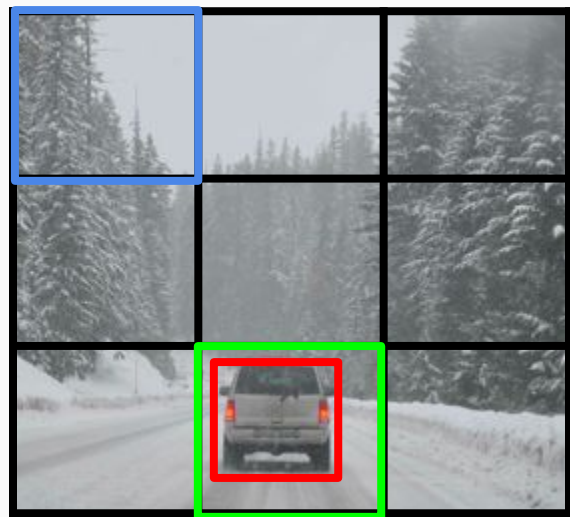
$$\begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Exemplo com anchor box

- A dimensão 8 em $3 \times 3 \times 2 \times 8$ tem a ver com o número de classes
 - E se fossem 6 classes?
- Na imagem anterior, tínhamos um carro e um pedestre
- Caso houvesse apenas um carro, apenas a 2a. metade do vetor seria preenchida
- E se tivéssemos 2 anchor boxes, mas 3 objetos?
 - Com sorte não acontece, caso contrário, regra para quebra de empates
- E se tivéssemos 2 objetos associados a mesma anchor box em uma célula?
 - Também não funcionaria perfeitamente
- Em geral esses problemas não acontecem com grids mais finos
- Como escolher anchor boxes?
 - Manualmente, de 5 a 10
 - Usando algoritmo k-means para agrupar objetos segundo forma e depois escolher anchor boxes representativos

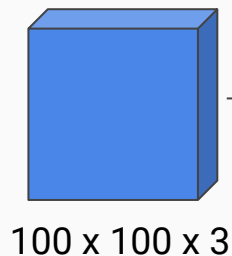
Juntando tudo: algoritmo YOLO

Treinamento



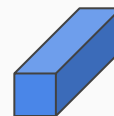
y é $3 \times 3 \times 2 \times 8$

- 1 - pedestre
- 2 - carro
- 3 - moto



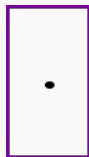
100 x 100 x 3

→ ConvNet →



3 x 3 x 16

Anchor box 1



Anchor box 2

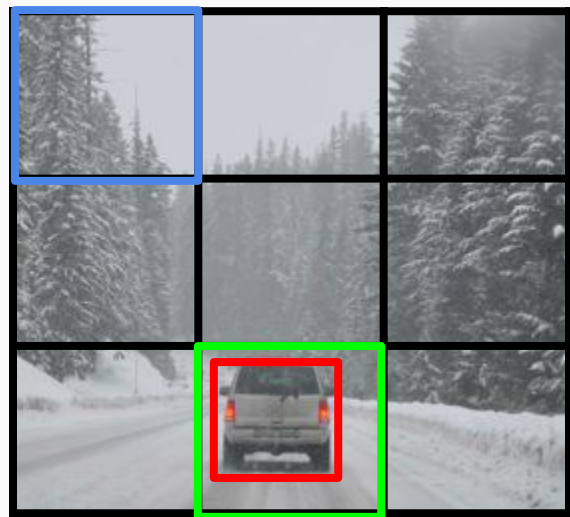


Célula 1 Célula 8

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

[Redmon et al., 2015, You Only Look Once: Unified real-time object detection]

Treinamento



y é $3 \times 3 \times 2 \times 8$

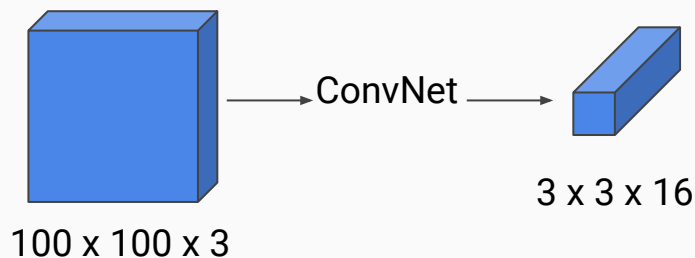
E se

... grid mais fino? $19 \times 19 \times 16$

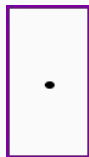
... 5 anchor boxes? $19 \times 19 \times 40$

[Redmon et al., 2015, You Only Look Once: Unified real-time object detection]

- 1 - pedestre
- 2 - carro
- 3 - moto



Anchor box 1

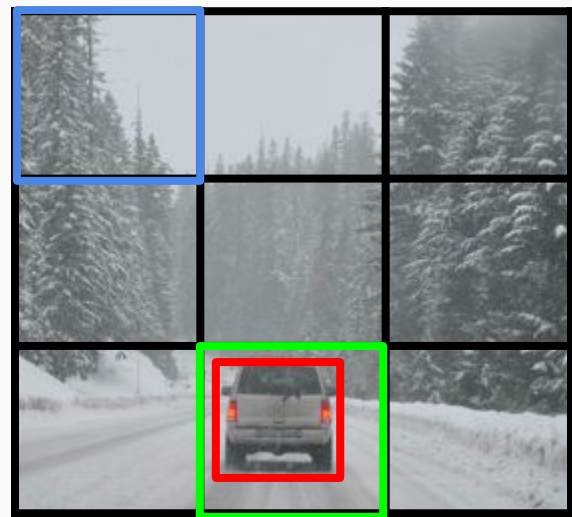


Anchor box 2



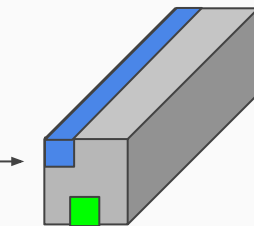
	Célula 1	Célula 8
$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$	$\begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix}$	$\begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$

Fazendo previsões



100 x 100 x 3

→ ConvNet →



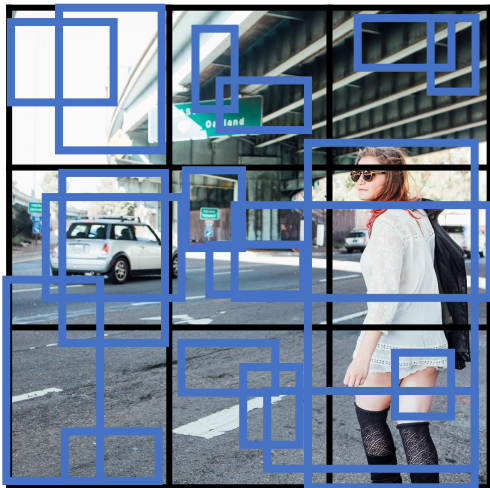
3 x 3 x 16

	previsão 1	previsão 8
$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$	$\begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix}$	$\begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ 1 \\ \textcolor{red}{b_x} \\ \textcolor{red}{b_y} \\ \textcolor{red}{b_h} \\ \textcolor{red}{b_w} \\ 0 \\ 1 \\ 0 \end{bmatrix}$

YOLO

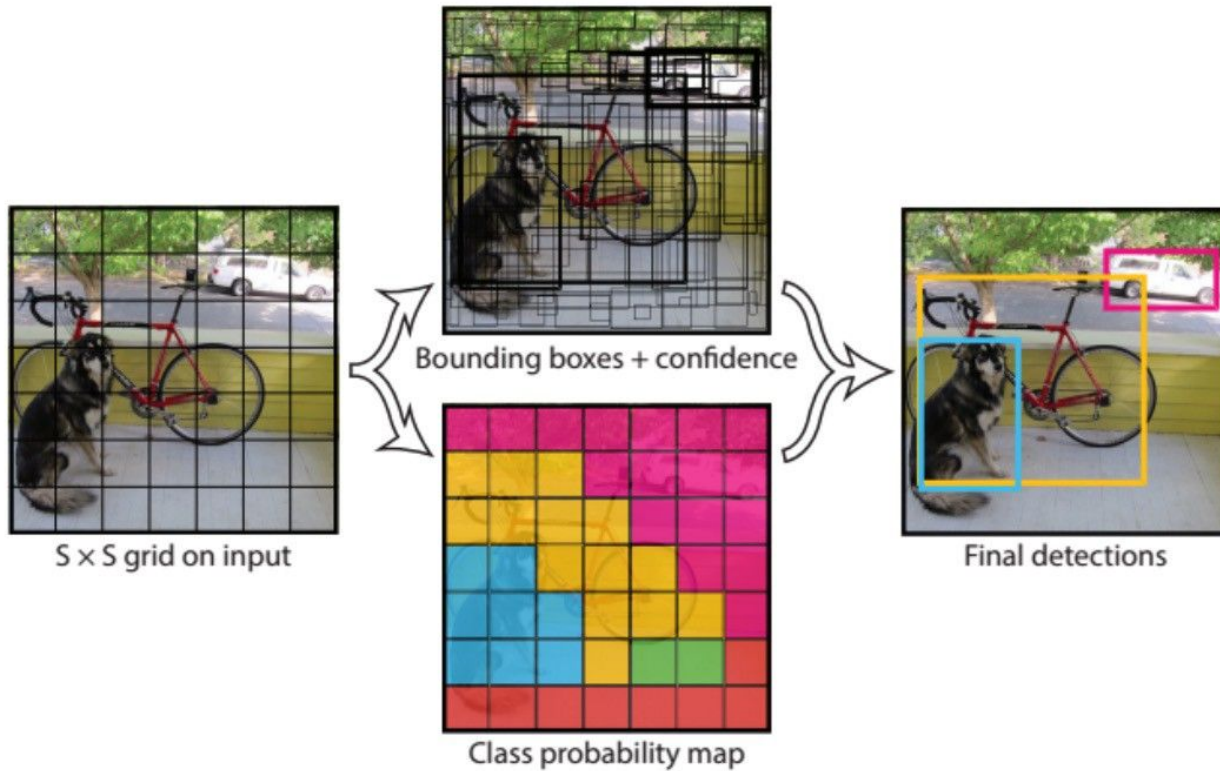
- No exemplo anterior
 - Treinamento:
 - Célula 1 não possui objeto
 - Célula 8 possui um carro, com IoU maior com anchor box 1
 - Previsão:
 - Célula 1: espera-se que $p_c = 0$, outras entradas não importam (don't care)
 - Célula 8: espera-se que anchor box 2 corresponda ao carro

Retornando as saídas após non-max suppression



- Para cada célula, retorne 2 bounding boxes da previsão
- Elimine as previsões com baixa probabilidade
- Para cada classe (pedestre, carro, moto) use non-max suppression para gerar as previsões finais

YOLO - Resumo



Yolo - Arquitetura (Pascal VOC)

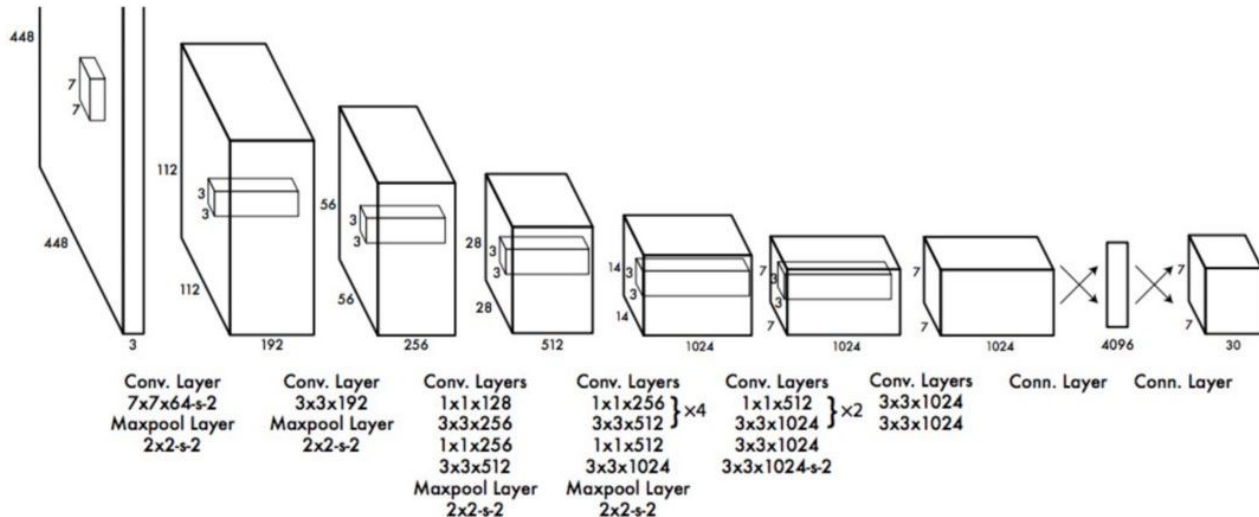
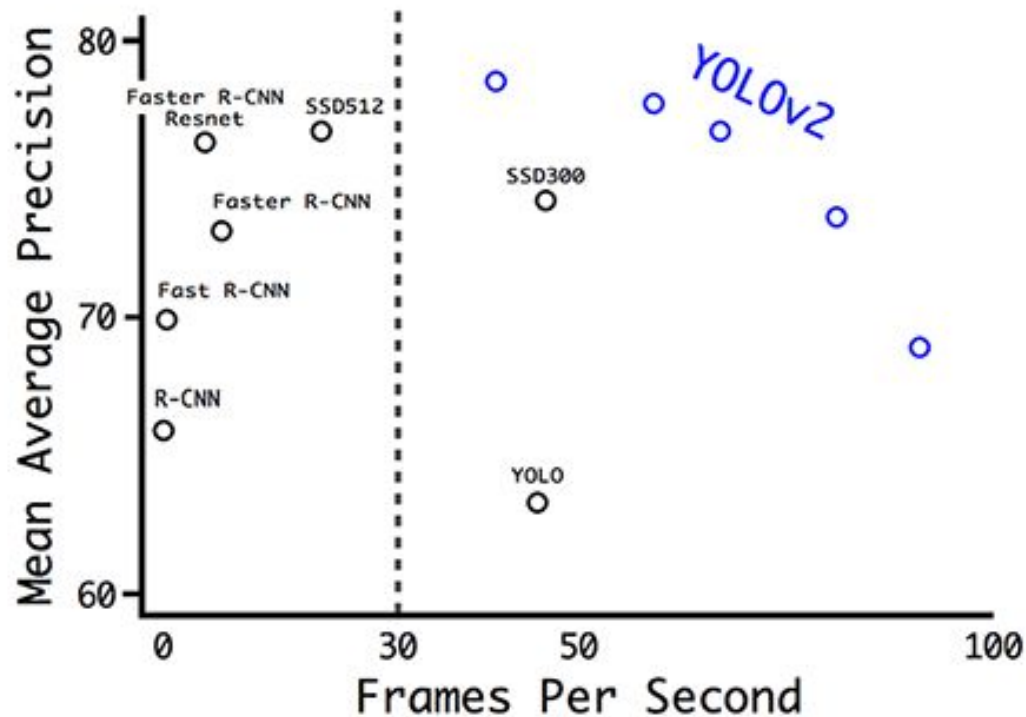


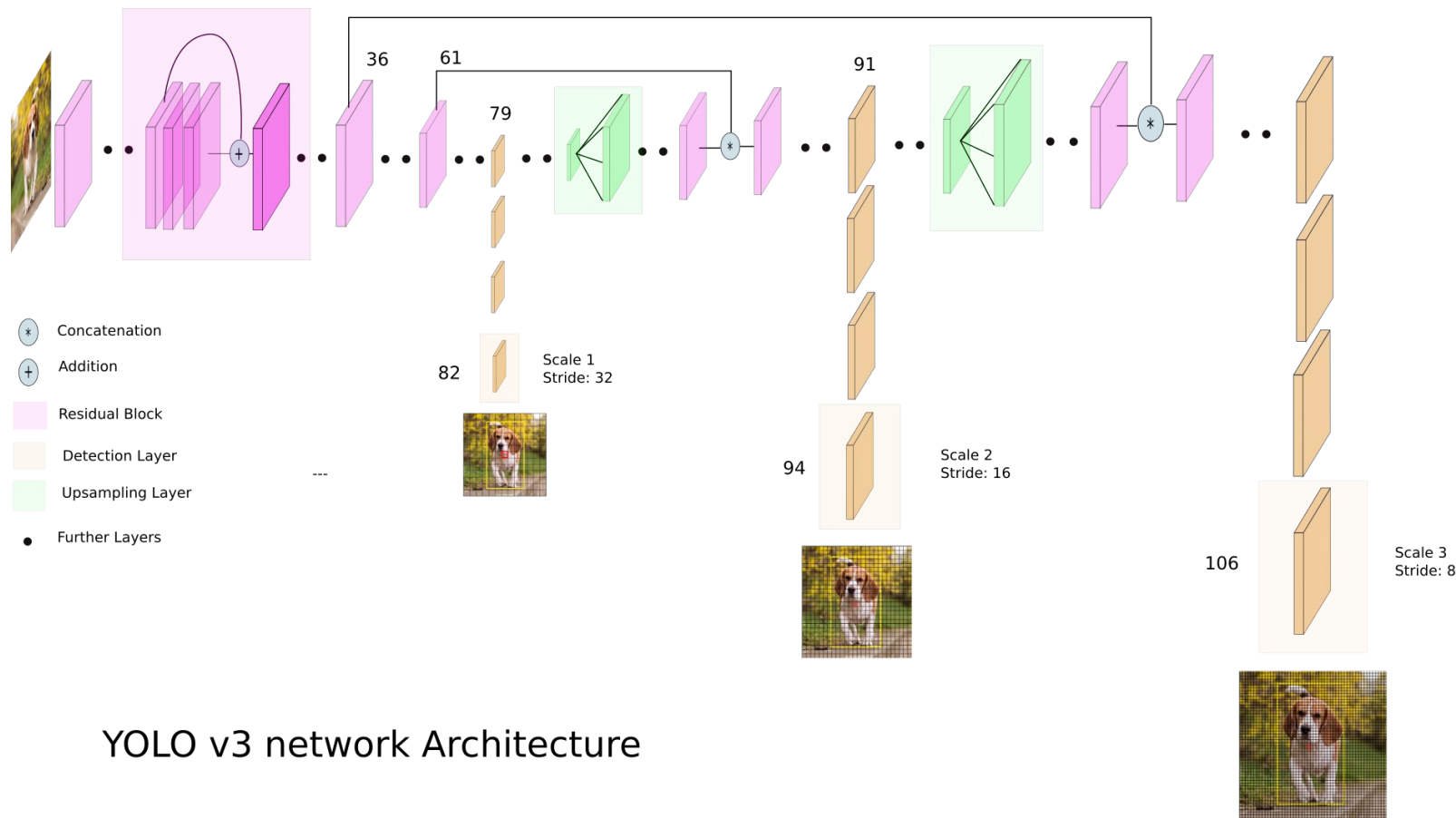
Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224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

YOLO vs Other methods



Redmon, J., & Farhadi, A. (2017). YOLO9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 7263-7271).

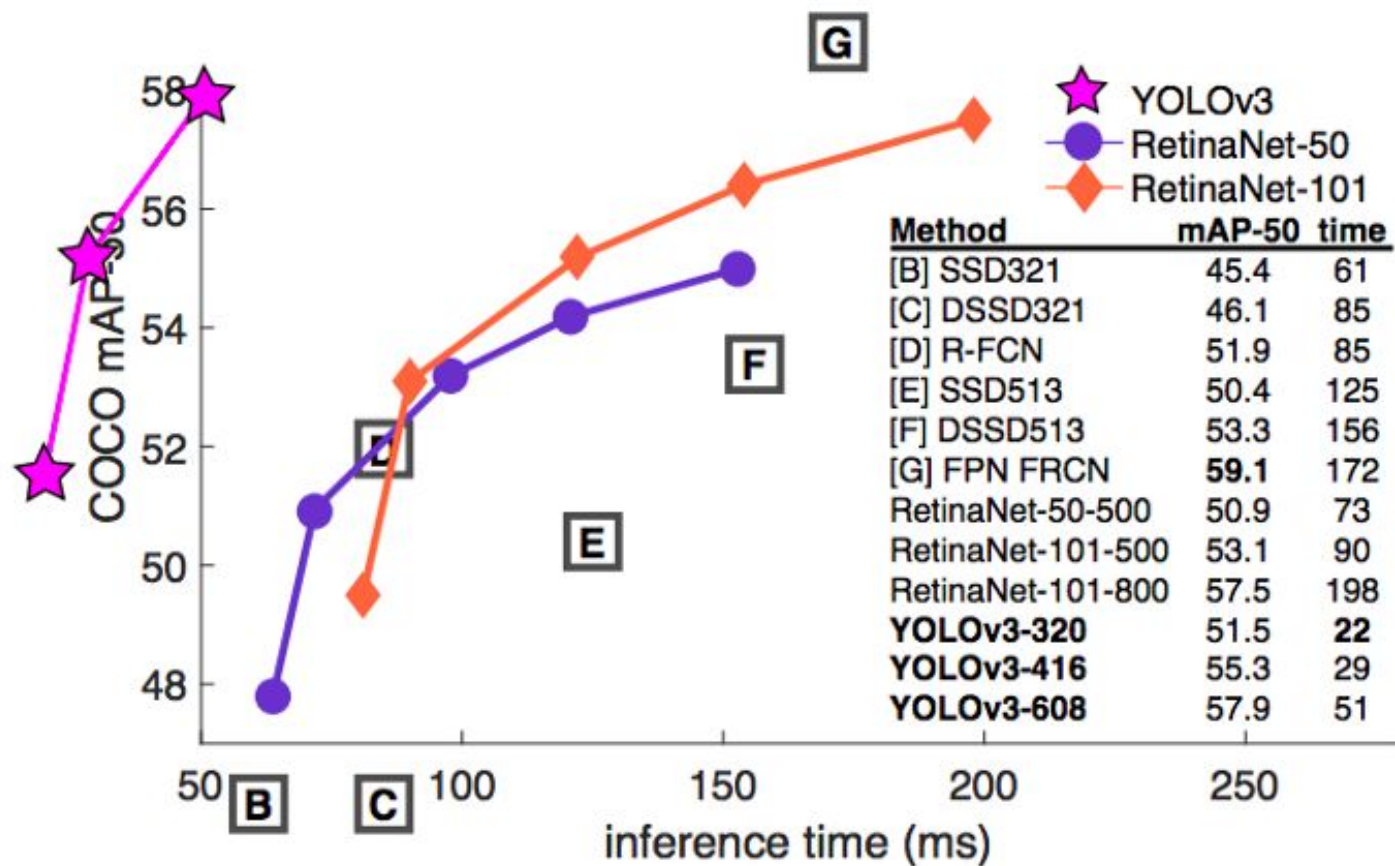
YOLO Architecture - V3



YOLO v3 network Architecture

YOLO vs Other methods

	backbone	AP	AP ₅₀	AP ₇₅	AP _S	AP _M	AP _L
<i>Two-stage methods</i>							
Faster R-CNN+++ [5]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [8]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [6]	Inception-ResNet-v2 [21]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [20]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	52.1
<i>One-stage methods</i>							
YOLOv2 [15]	DarkNet-19 [15]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [11, 3]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [3]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet [9]	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet [9]	ResNeXt-101-FPN	40.8	61.1	44.1	24.1	44.2	51.2
YOLOv3 608 × 608	Darknet-53	33.0	57.9	34.4	18.3	35.4	41.9



YOLO

Vantagens:

- Velocidade (45 quadros por segundo)
- A rede entende a representação generalizada de objetos (isso permitiu que eles treinassem a rede em imagens do mundo real e as previsões sobre obras de arte ainda eram bastante precisas).
- Versão mais rápida - 155 quadros por segundo, mas é menos precisa.

Desvantagem:

- Dificuldade de aproveitar features pré-treinadas de outras tarefas (backbone)
- Processa muitos bounding boxes desnecessários (background)
- Dificuldade de detectar objetos pequenos