Detecção de Objetos

Prof. Jefersson A. dos Santos
jefersson@dcc.ufmg.br

Prof. Fabrício Murai
murai@dcc.ufmg.br

DCC
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

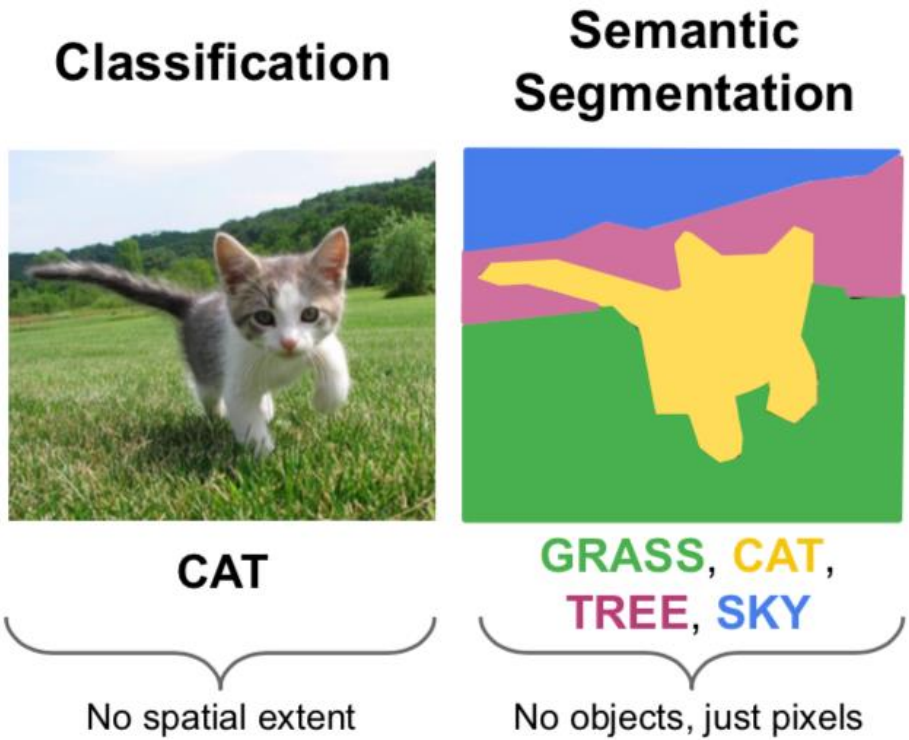
UFMG

Roteiro da aula de hoje

- Tarefas de visão computacional
  - Classificação, segmentação semântica, **detecção de objetos e segmentação de instâncias**
- Bounding boxes
- Detecção com janelas deslizantes
- Alternativas:
  - CNNs baseadas em proposta de regiões
    - R-CNN, Fast R-CNN, Faster R-CNN
  - Implementação convolucional de janelas deslizantes
- Segmentação de Instâncias: Mask R-CNN

Tarefas de visão computacional

Tarefas de visão computacional



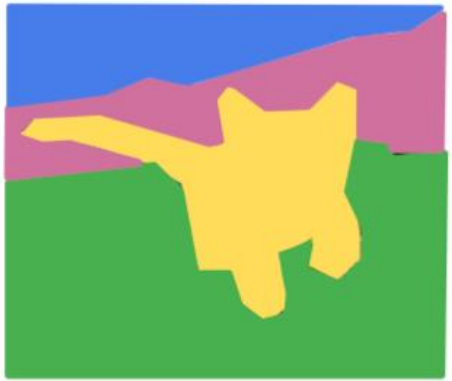
Tarefas de visão computacional

Classification

CAT

No spatial extent

Semantic Segmentation



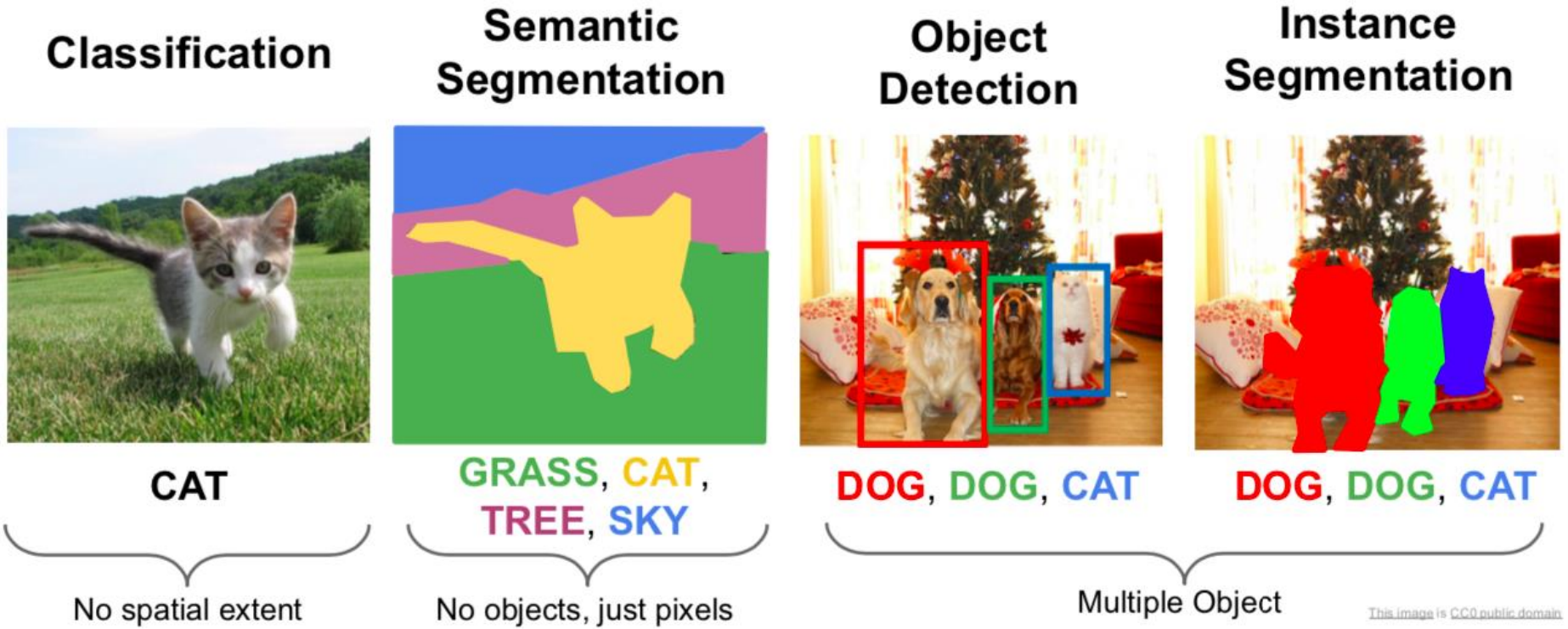
GRASS, CAT, TREE, SKY

No objects, just pixels

Object Detection

DOG, DOG, CAT

Tarefas de visão computacional



Tarefas de visão computacional

- **Classificação**: retorna classe associada a imagem
- **Segmentação semântica**: associa cada pixel a uma classe; não distingue entre instâncias da mesma classe
- **Detecção de objetos**: retorna classe e localização de cada objeto presente na imagem
- **Segmentação de instância**: atribui cada pixel a uma instância e classifica a instância

Antes de abordar detecção de objetos, vamos considerar apenas UM objeto:

- **Classificação com localização**: retorna classe associada a imagem e a localização do (único) objeto

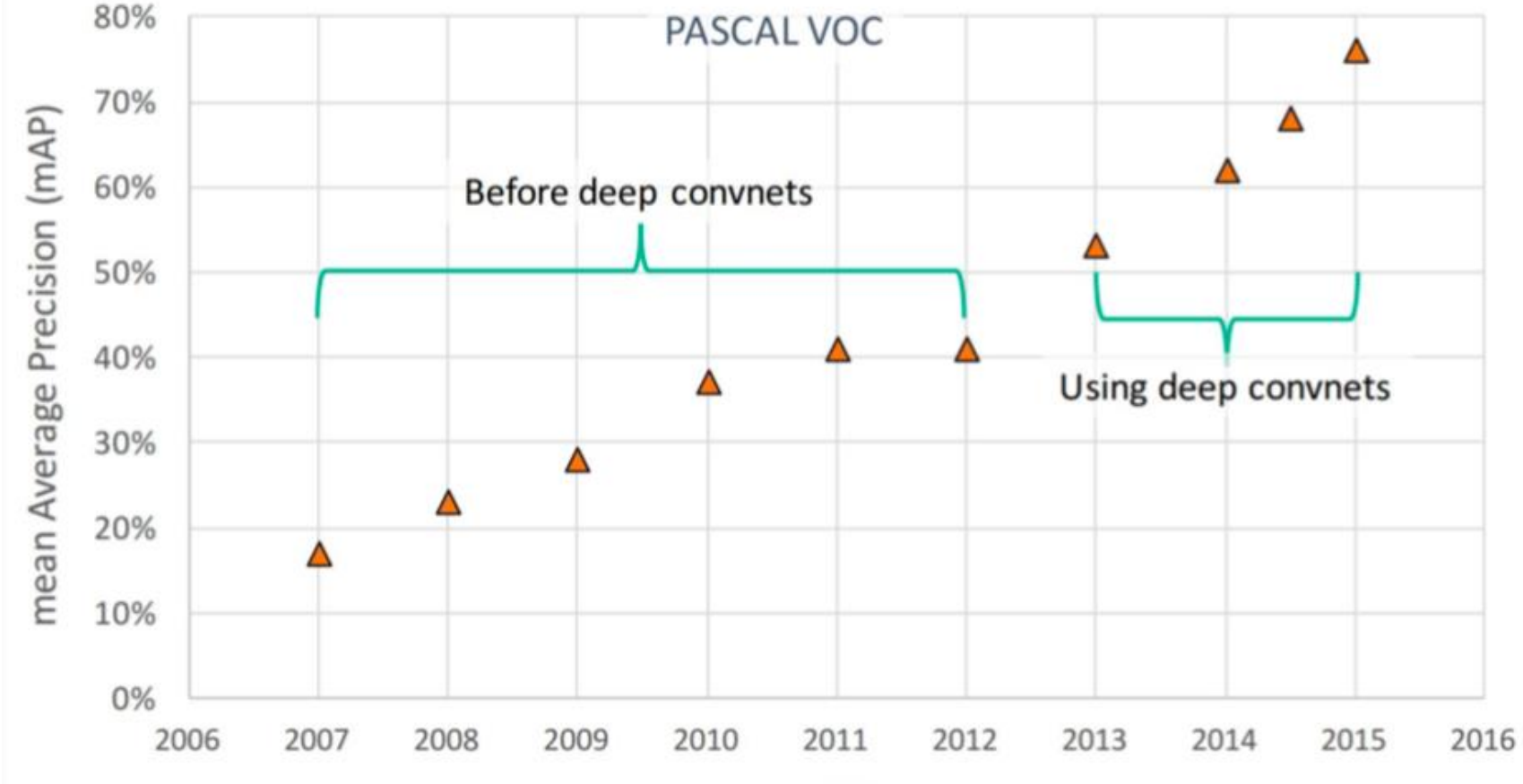
Ideias para uma tarefa mais simples são relevantes para uma tarefa mais complexa

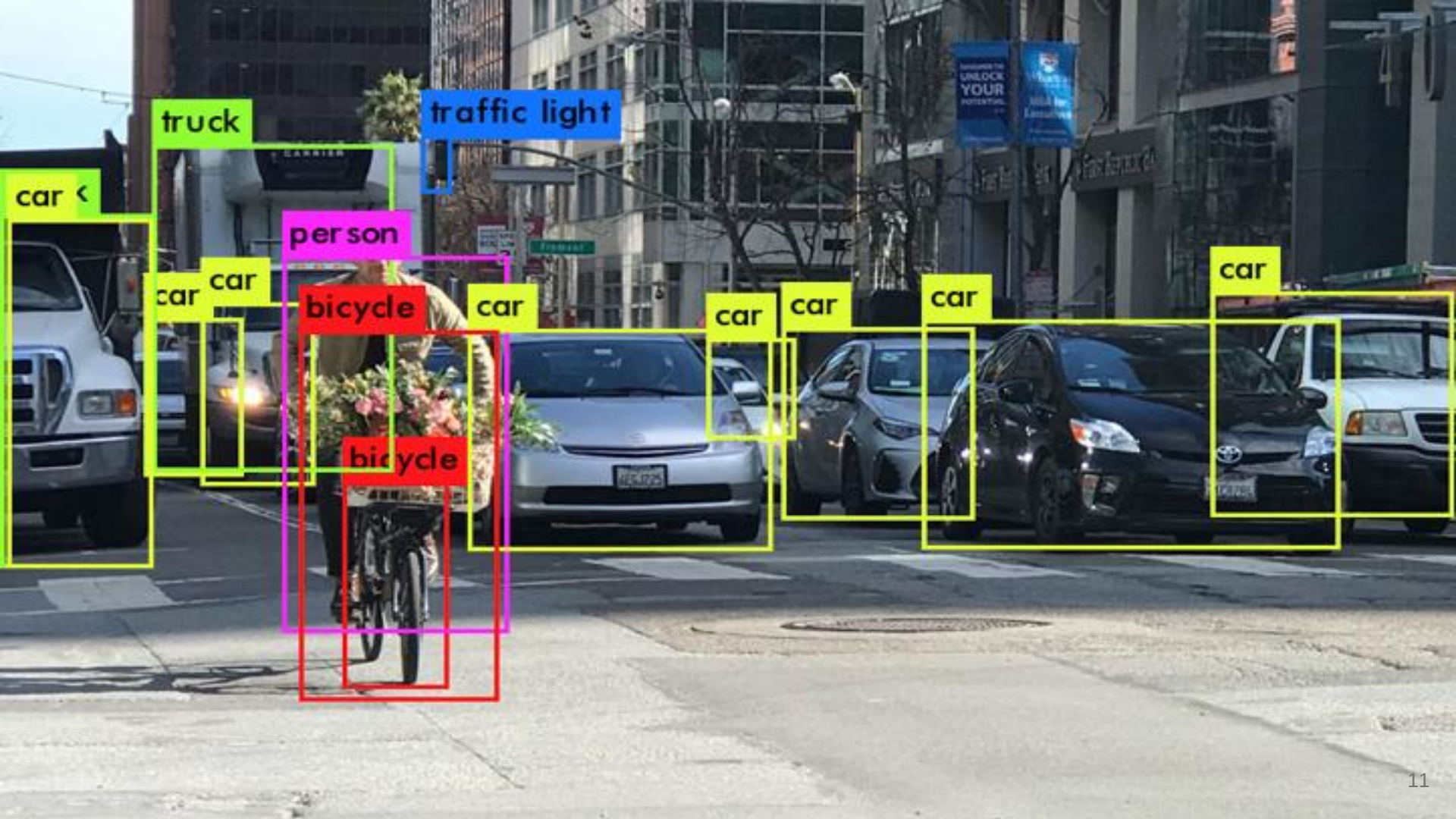
Detecção de objetos

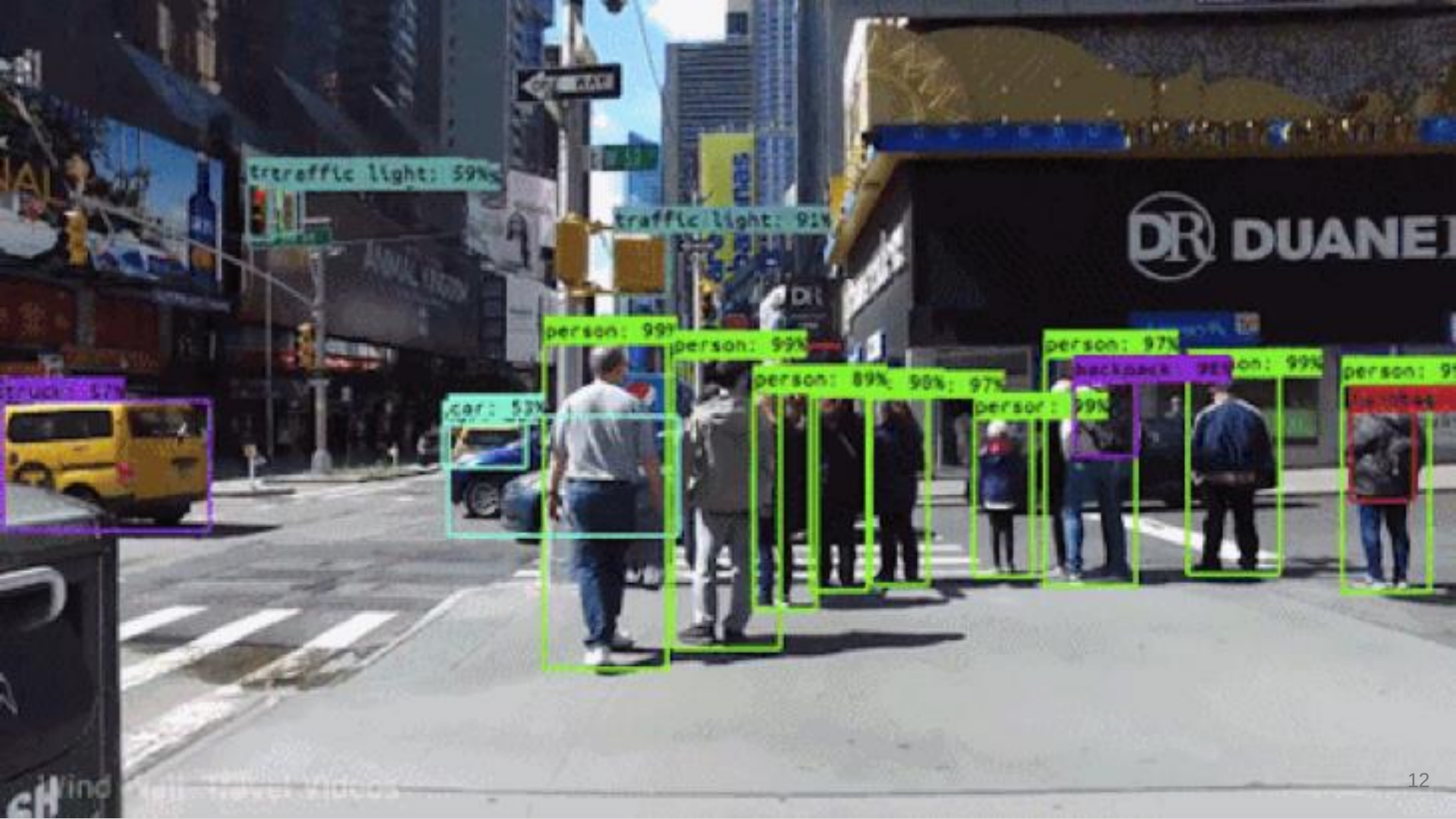
Detecção de objetos

- Área em visão computacional que vem se desenvolvendo muito
- Aplicações:
  - Carros autônomos: planejar rotas identificando a localização de pedestres, veículos, ruas e obstáculos nas imagens capturadas
  - Robôs (tarefas semelhantes)
  - Sistemas de segurança: identificar anomalias (e.g., intrusos ou bombas)
- Desempenho nessas tarefas é hoje MUITO melhor que há apenas alguns anos atrás

Detecção de objetos: impacto de Deep Learning




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


trraffic light: 59%
traffic light: 91%
DUANE
person: 99%
person: 99%
person: 89%
person: 97%
backpack
car: 53%

Detecção de objetos

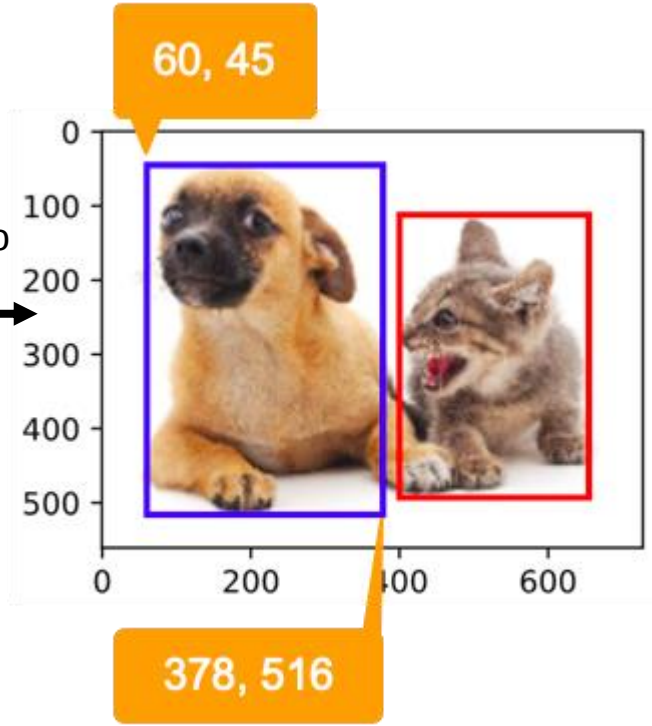
Classificação com localização

Bounding box

Uma bounding box é definida por 4 números:

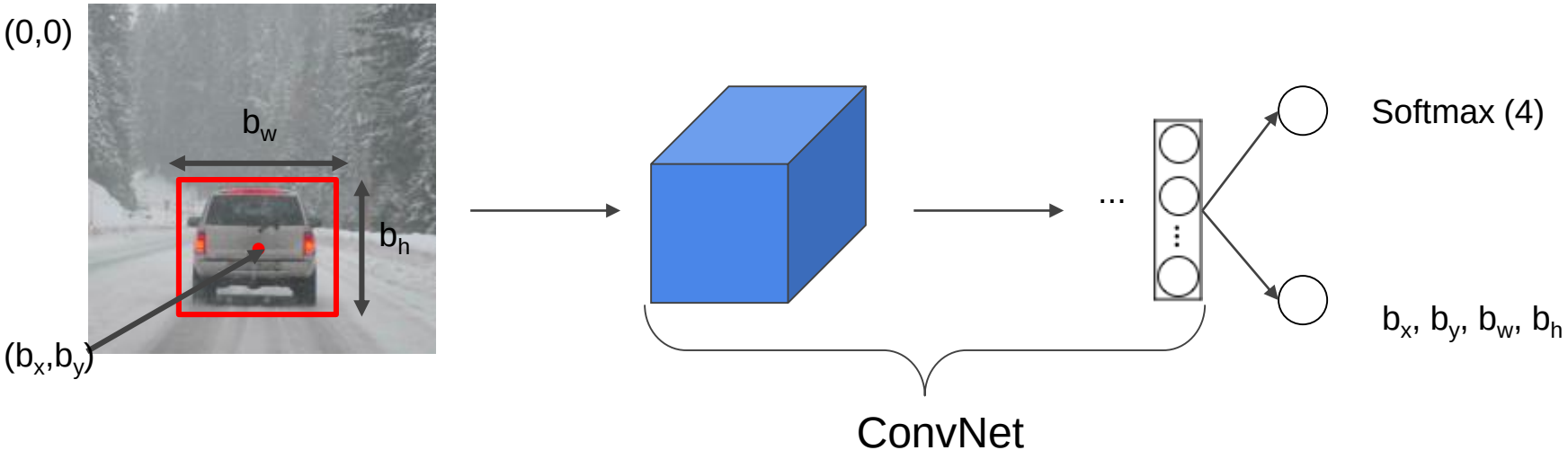
- superior esquerdo x, superior esquerdo y, inferior direito x, inferior direito y
- superior esquerdo x, superior esquerdo y, largura, altura
- centro x, centro y, largura altura

Exemplo c/ esta convenção



Nesses slides, vamos utilizar esta convenção, mas primeiro vamos normalizar a escala da imagem

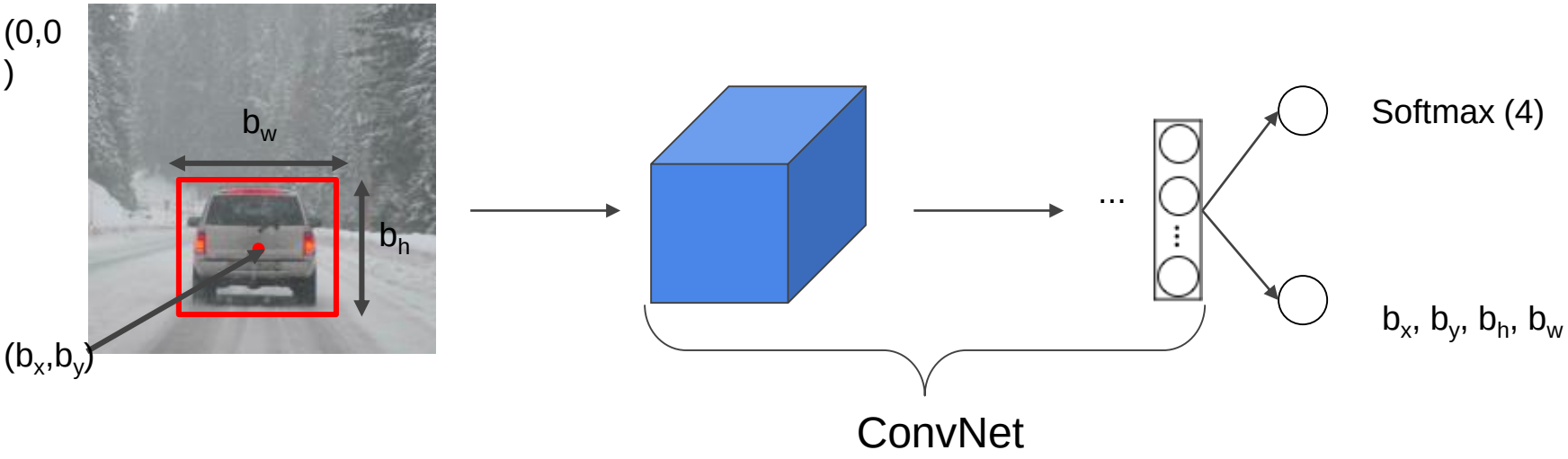
Exemplo: classificação com localização



Classes

1. Pedestre
2. Carro
3. Motocicleta
4. Fundo

Exemplo: classificação com localização



Classes

1. Pedestre
2. Carro
3. Motocicleta
4. Fundo

Bounding box

b_x= 0.5
b_y = 0.7
b_h = 0.3
b_w = 0.4

Classificação com Localização

- Convenção utilizada
  - Imagem: canto superior esquerdo: (0,0); canto inferior direito: (1,1)
- Classificação de imagem:
  - Rede convolucional, última camada é FC com saídas para softmax (4)
- Com localização:
  - Adiciona 4 novas saídas que localizam bounding box: b_x, b_y, b_h, b_w

    Ponto médio (b_x,b_y), largura b_w, altura b_h
  - Trata localização como problema de regressão

Definindo o rótulo y

Precisa retornar: b_x, b_y, b_h, b_w, label da classe (1-4)

x= 

Classes

1. Pedestre
2. Carro
3. Motocicleta
4. Fundo

Definindo o rótulo y

Precisa retornar: b_x, b_y, b_h, b_w, label da classe (1-4)

Classes

1. Pedestre
2. Carro
3. Motocicleta
4. Fundo

x=

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

Há um objeto?

Don't care

Possível função de perda (com erro quadrático):

$$L(\hat{y}, y) = \begin{cases} (\hat{y}_1 - y_1)^2 + (\hat{y}_2 - y_2)^2 + \ldots (\hat{y}_8 - y_8)^2 & \text{if } y_1 = 1 \\ (\hat{y}_1 - y_1)^2 & \text{if } y_1 = 0 \end{cases}$$

Definindo o target label y

- Vetor y tem 8 coordenadas:
  - p_c: há um objeto?
  - b_x, b_y, b_h, b_w: bounding box
  - c_1, c_2, c_3: classe
- Dois casos:
  - Há um objeto, p_c=1, contabilizar o erro em todas as coordenadas
  - Não há objeto, p_c=0, outras coordenadas são "don't cares" e não devem ser consideradas na função de perda
- Na prática, erro é medido com função diferente em cada coordenada: p_c (logistic regression loss), bounding box (squared error), classe (log-likelihood loss)

Detecção de objetos

Utilizando janelas deslizantes

Car detection example

Training set:

x	y
	1
	1
	1
	0
	0

Car detection example

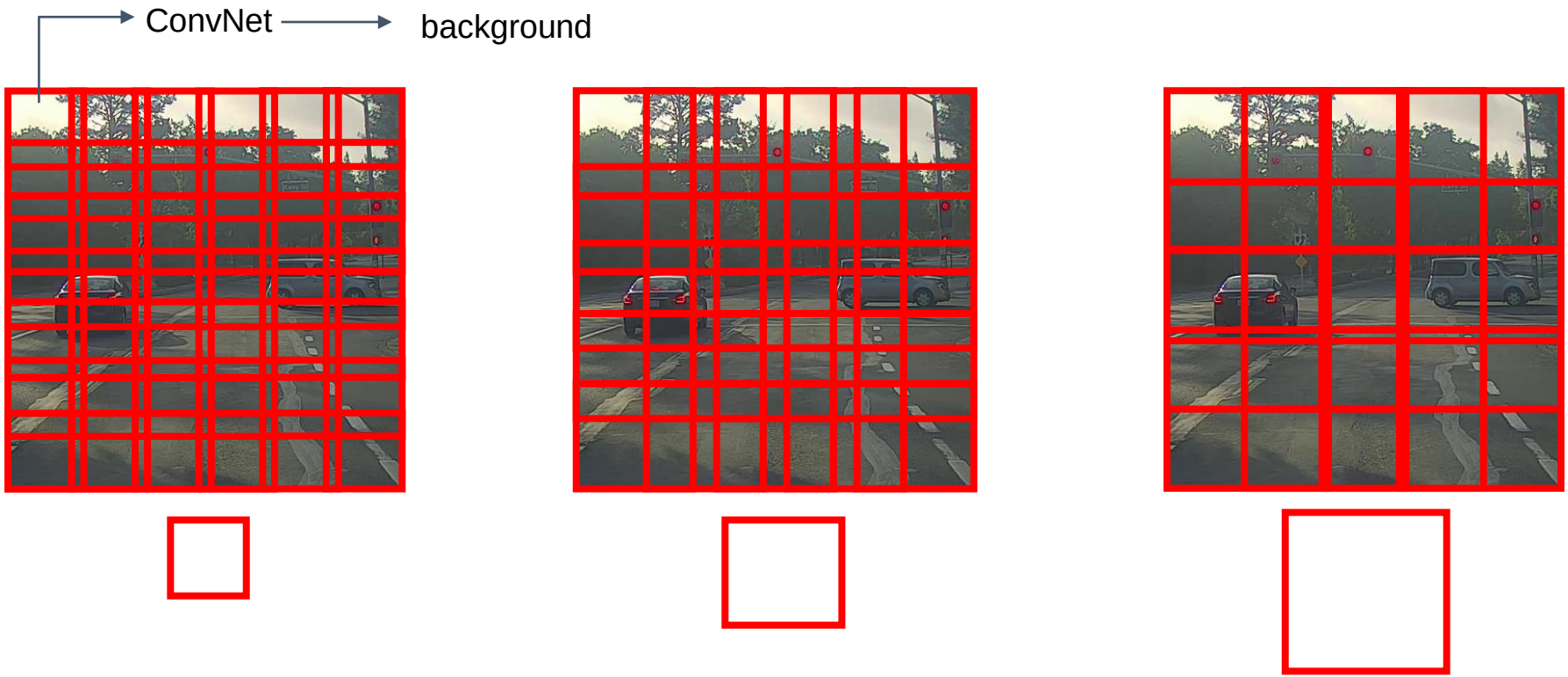
Training set:

x	y
	1
	1
	1
	0
	0

Suponha que tenhamos uma rede de classificação:

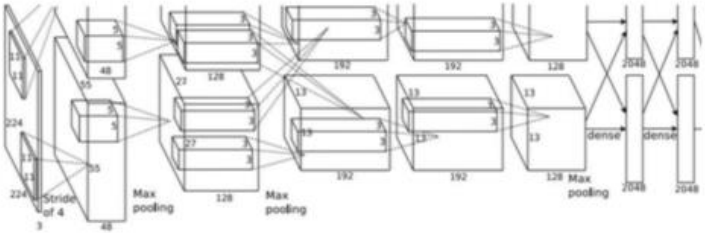
Agora vamos usar janelas deslizantes

Sliding windows detection



Detecção de objetos: múltiplos objetos

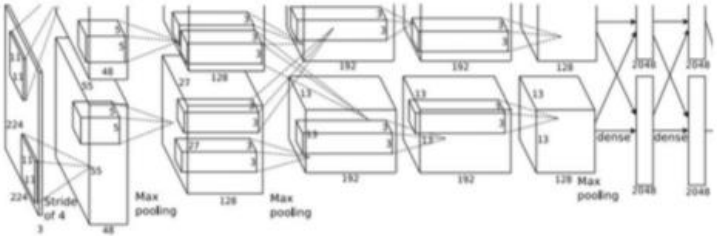
Aplica uma CNN a diferentes recortes da imagem, CNN classifica cada recorte como objeto ou background



Dog? NO
Cat? NO
Background? YES

Detecção de objetos: múltiplos objetos

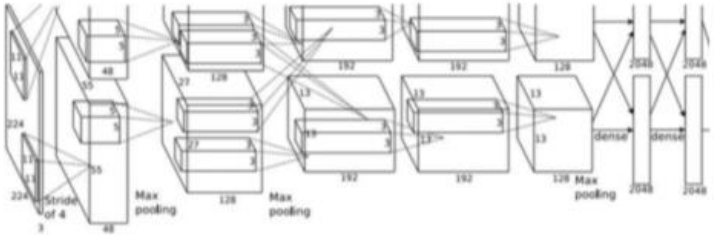
Aplica uma CNN a diferentes recortes da imagem, CNN classifica cada recorte como objeto ou background



Dog? YES
Cat? NO
Background? NO

Detecção de objetos: múltiplos objetos

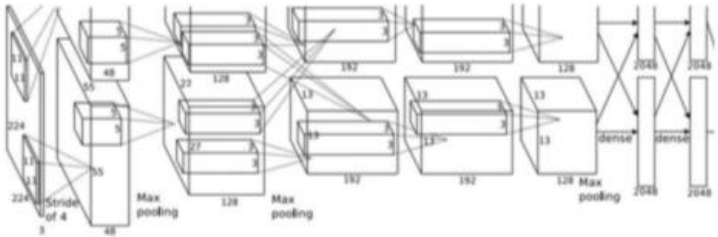
Aplica uma CNN a diferentes recortes da imagem, CNN classifica cada recorte como objeto ou background



Dog? YES
Cat? NO
Background? NO

Detecção de objetos: múltiplos objetos

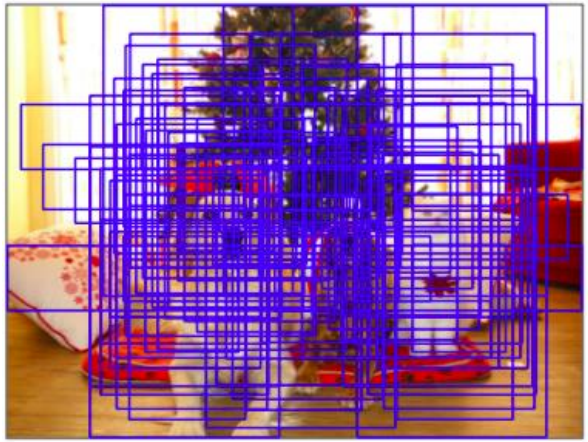
Aplica uma CNN a diferentes recortes da imagem, CNN classifica cada recorte como objeto ou background

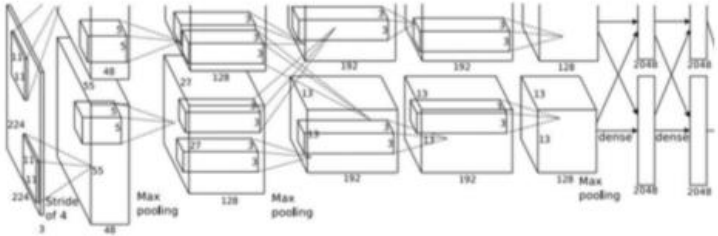


Dog? NO
Cat? YES
Background? NO

Detecção de objetos: múltiplos objetos

Aplica uma CNN a diferentes recortes da imagem, CNN classifica cada recorte como objeto ou background





Dog? NO
Cat? YES
Background? NO

Problem: Need to apply CNN to huge number of locations, scales, and aspect ratios, very computationally expensive!

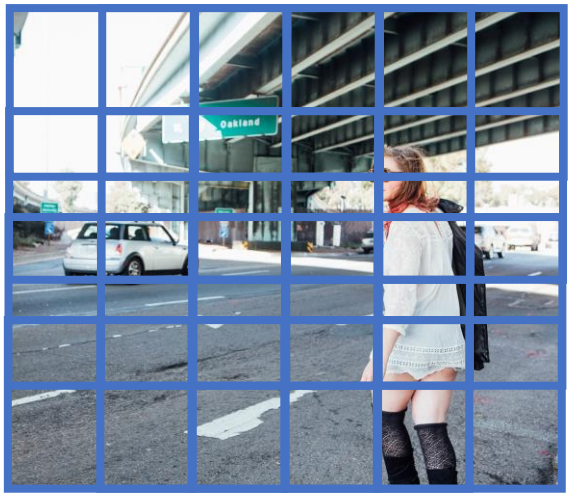
Detecção por janelas deslizantes

- 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

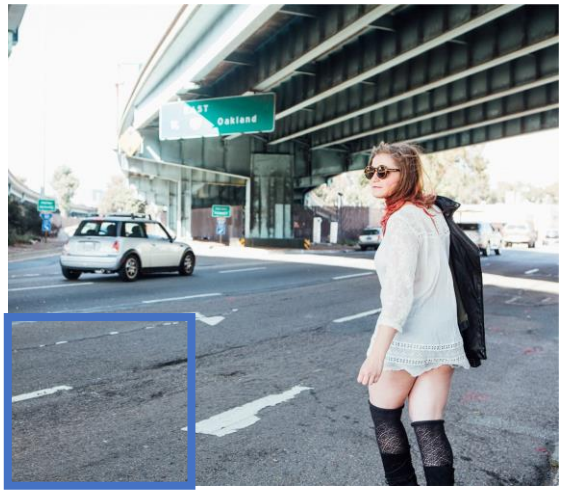
Detecção de objetos por region proposals

R-CNN: alternativa a janelas deslizantes

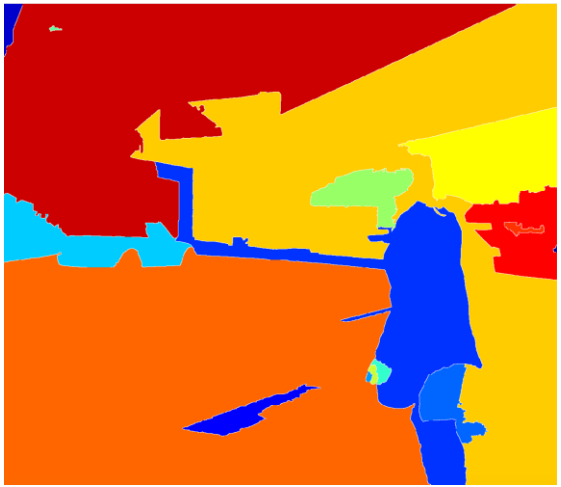
Region proposal: R-CNN



Deslizar a janela, classificando cada imagem



Ineficiente: tem que classificar muitas regiões "sem nada"



R-CNN: seleciona apenas algumas janelas

Algoritmo de segmentação: ~2000 *blobs*

[Girshik et. al, 2013, Rich feature hierarchies for accurate object detection and semantic segmentation]

Proposta de regiões: R-CNN

- De certo modo, detecção de objetos é ineficiente: maior parte das regiões não contêm objetos
- R-CNN: encontra "blobs" (regiões de interesse) e classifica apenas elas
  - Usa *selective search* (algoritmo de segmentação) para encontrar bordas que separam regiões da imagem (retorna ~2000 blobs)
  - Particularmente eficiente quando os objetos possuem muitos formatos/escalas diferentes

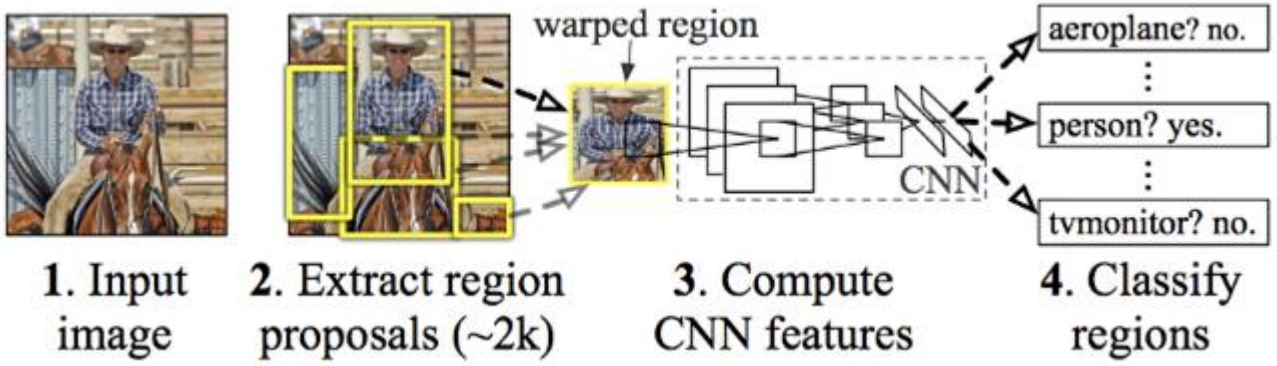
Algoritmo de segmentação (Selective Search)

1. Generate initial sub-segmentation, we generate many candidate regions

2. Use greedy algorithm to recursively combine similar regions into larger ones

3. Use the generated regions to produce the final candidate region proposals

R-CNN

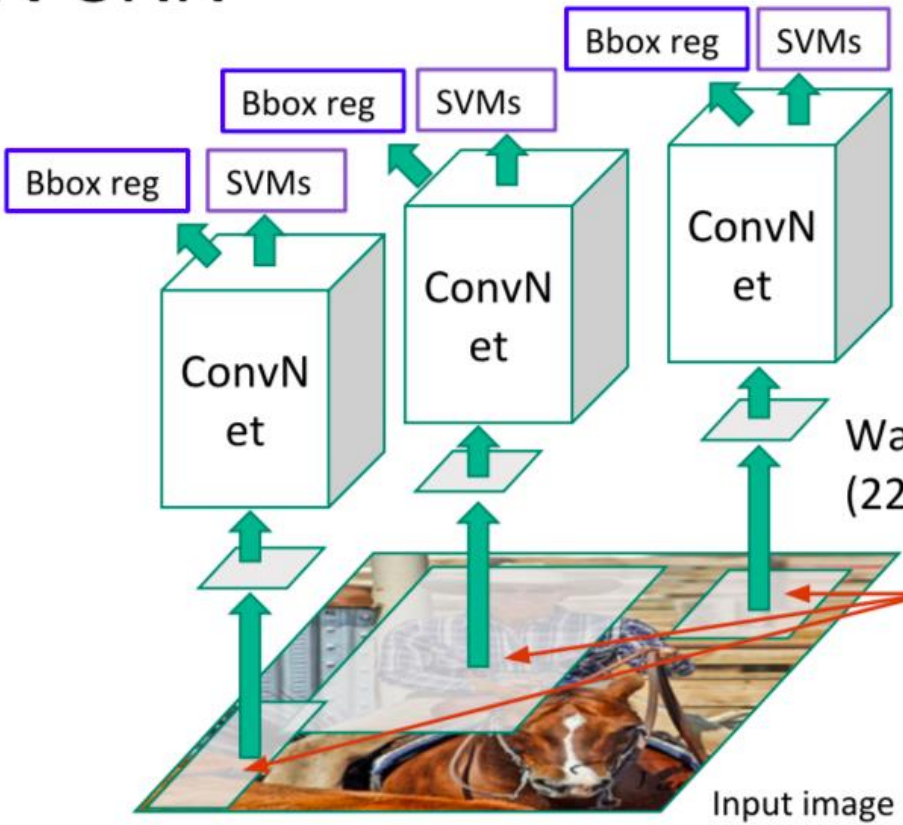


RoI (regions of interest) podem estar "erradas"; retorna bboxes também

- Seleciona Region of Interests (RoIs) com algoritmo baseado em heurística
- Usa redes CNN pré-treinadas para extrair features de cada RoI através do *forward computation*
- Treina um SVM para classificar a categoria
- Treina uma regressão linear para prever *bounding boxes*

R-CNN

Predict “corrections” to the RoI: 4 numbers: (dx, dy, dw, dh)



Classify regions with SVMs

Forward each region through ConvNet

Warped image regions (224x224 pixels)

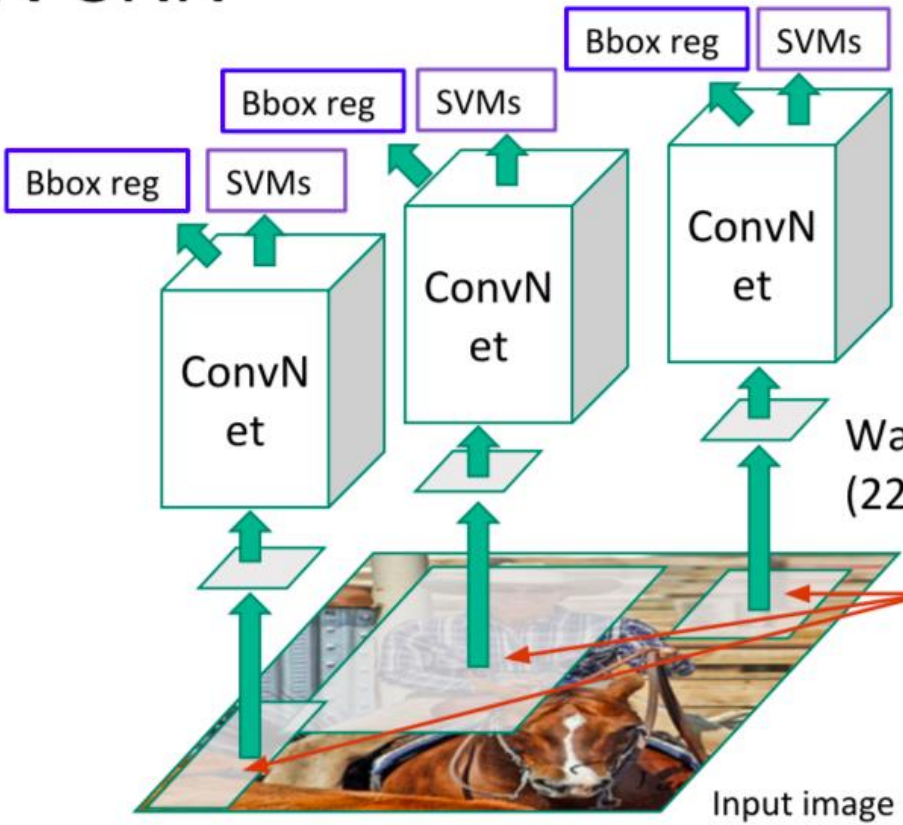
Regions of Interest (RoI) from a proposal method (~2k)

Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

"Slow"

R-CNN

Predict “corrections” to the RoI: 4 numbers: (dx, dy, dw, dh)



Classify regions with SVMs

Forward each region through ConvNet

Warped image regions (224x224 pixels)

Regions of Interest (RoI) from a proposal method (~2k)

Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Passos no treinamento da R-CNN

1. **Pré-treinamento supervisionado:** treina CNN (SGD λ=0.01) em um dataset auxiliar grande (ImageNet) anotadas apenas com 1k classes (sem bboxes)

2. **Fine-tuning para domínio específico**: nova tarefa (detecção) + novo domínio (warped windows).
   Substitui camada de classif (10k classes) por camada de classif (N+1) e treina (SGD λ=0.001).
   Datasets: VOC (N=20), ILSVRC2013 (N=200).
   Cada **region proposal** é mapeada para com ground truth bboxes com maior overlap.
   Se overlap > 0.5, exemplos **positivos**; c.c, exemplo **negativos**. Mini-batch = 32 (+) e 96 (-).

3. **Treinamento do classificador:** treina N classificadores SVM binários a partir de ConvNet features.
   Exemplos positivos e negativos são definidos de forma diferente agora:
   a. Ground-truth bounding boxes -> instâncias positivas
   b. Overlap < 0.3 -> instâncias negativas (usa apenas “hard negatives”)

4. **Bounding box regression**: localização retornada sem a regressão não era muito precisa. (continua próximo slide)

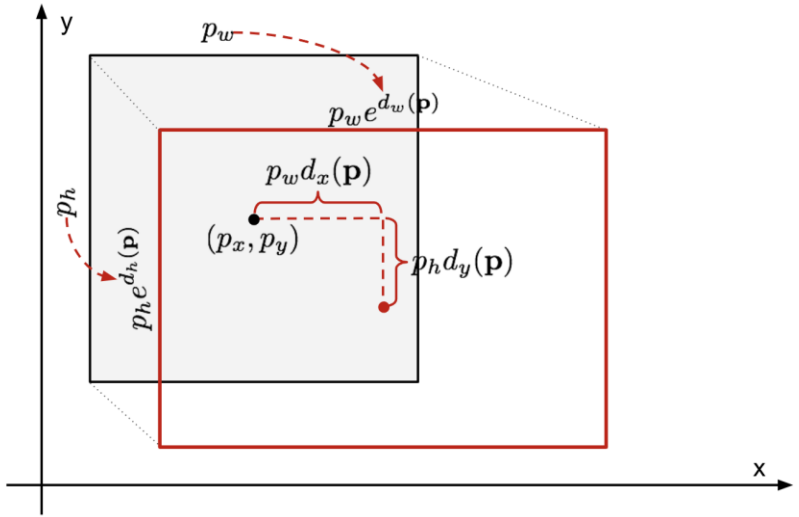
Treinamento da R-CNN: bounding box regression

Queremos treinar regressão linear usando saída da CNN a partir de região proposta

Entrada: N pares $\{(P^{(i)}, G^{(i)})\}_{i=1,...,N}$, onde $P^{(i)} = \{P_x^{(i)}, P_y^{(i)}, P_h^{(i)}, P_w^{(i)}\}$ são as coordenadas da proposta e $G^{(i)} = \{G_x^{(i)}, G_y^{(i)}, G_h^{(i)}, G_w^{(i)}\}$ são as coordenadas da ground truth bbox.

Regressão linear é uma transformação que mapeia P em G, que é parametrizada por

- $d_x(P)$ e $d_y(P)$: translação de escala invariante do centro da bbox P
- $d_h(P)$ e $d_w(P)$: transformações em escala log da altura e largura de P



Treinamento da R-CNN: bounding box regression

Relação entre P e G:

$$\hat{G}_x = P_w d_x(P) + P_x$$
$$\hat{G}_y = P_h d_y(P) + P_y$$
$$\hat{G}_w = P_w \exp(d_w(P))$$
$$\hat{G}_h = P_h \exp(d_h(P)).$$

Targets das funções $d_*(P)$:

$$t_x = (G_x - P_x)/P_w$$
$$t_y = (G_y - P_y)/P_h$$
$$t_w = \log(G_w/P_w)$$
$$t_h = \log(G_h/P_h).$$

Função a ser otimizada:

$$\mathcal{L}_{\text{reg}} = \sum_{i \in \{x,y,w,h\}} (t_i - d_i(\mathbf{p}))^2 + \lambda \|\mathbf{w}\|^2$$

R-CNN: limitações

- Tempo de treinamento é alto, pois precisa classificar ~2k RoIs por imagem
- O algoritmo selective search é fixo (não envolve aprendizado), podendo selecionar regiões ruins
- Não pode ser implementado em tempo real, pois leva 47s para cada imagem de teste (~2k *forward passes* **independentes** por imagem)

R-CNN: limitações

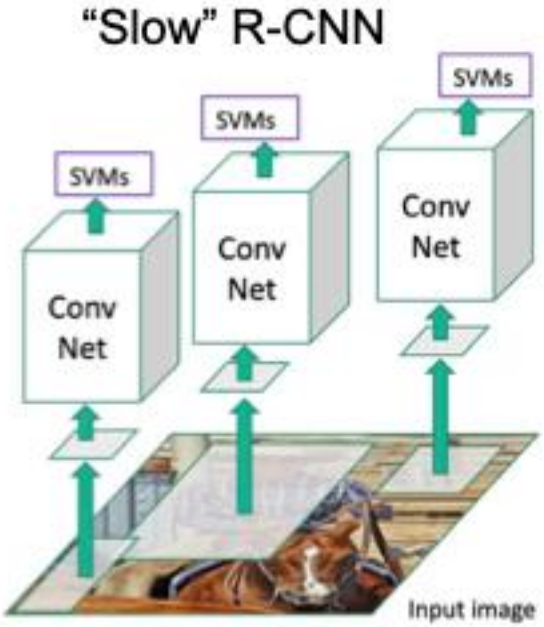
- Tempo de treinamento é alto, pois precisa classificar ~2k RoIs por imagem
- O algoritmo selective search é fixo (não envolve aprendizado), podendo selecionar regiões ruins
- Não pode ser implementado em tempo real, pois leva 47s para cada imagem de teste (~2k *forward passes* **independentes** por imagem)
    - Ideia: processar imagem antes de fazer o cropping! Trocar ordem entre ConvNet e cropping.

Detecção de objetos por region proposals

Fast R-CNN

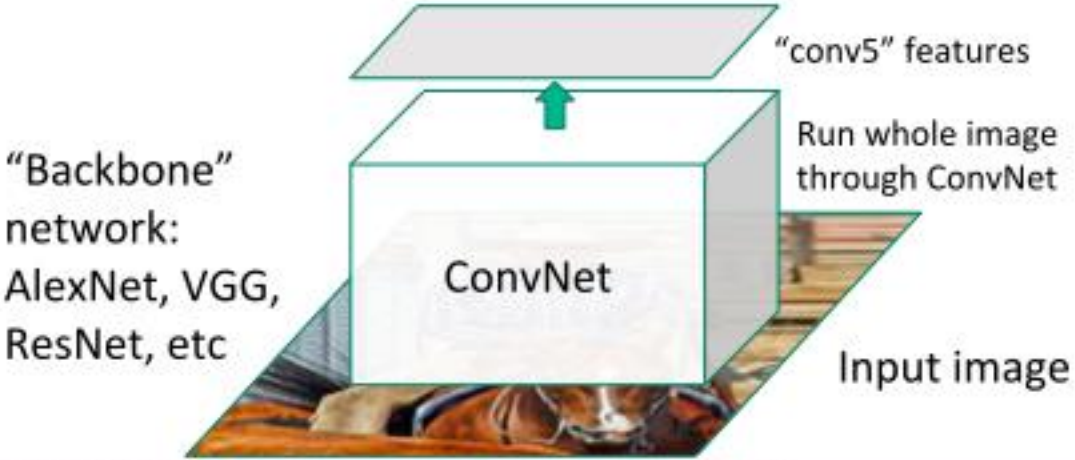
Fast R-CNN

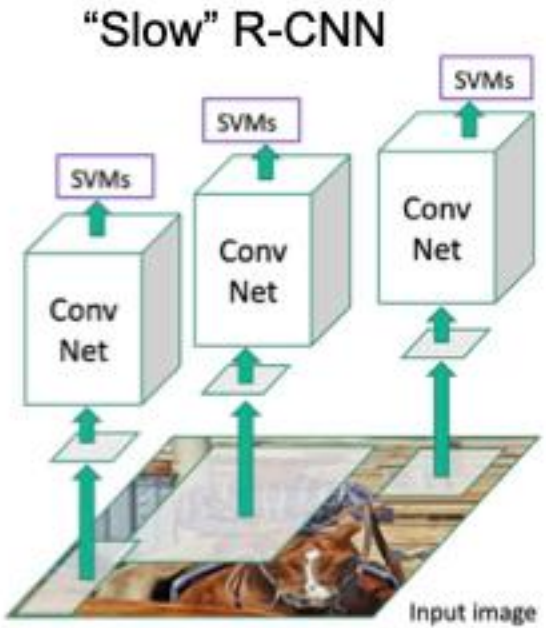
Input image



Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Fast R-CNN



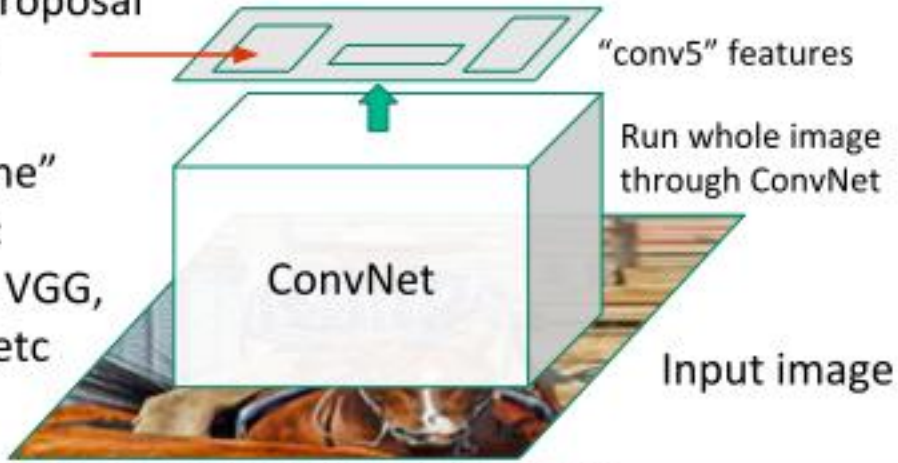


Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

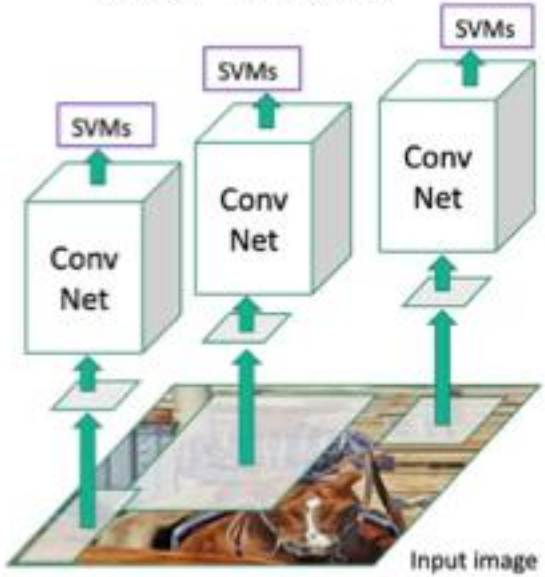
Fast R-CNN

Regions of Interest (RoIs) from a proposal method

"Backbone" network: AlexNet, VGG, ResNet, etc

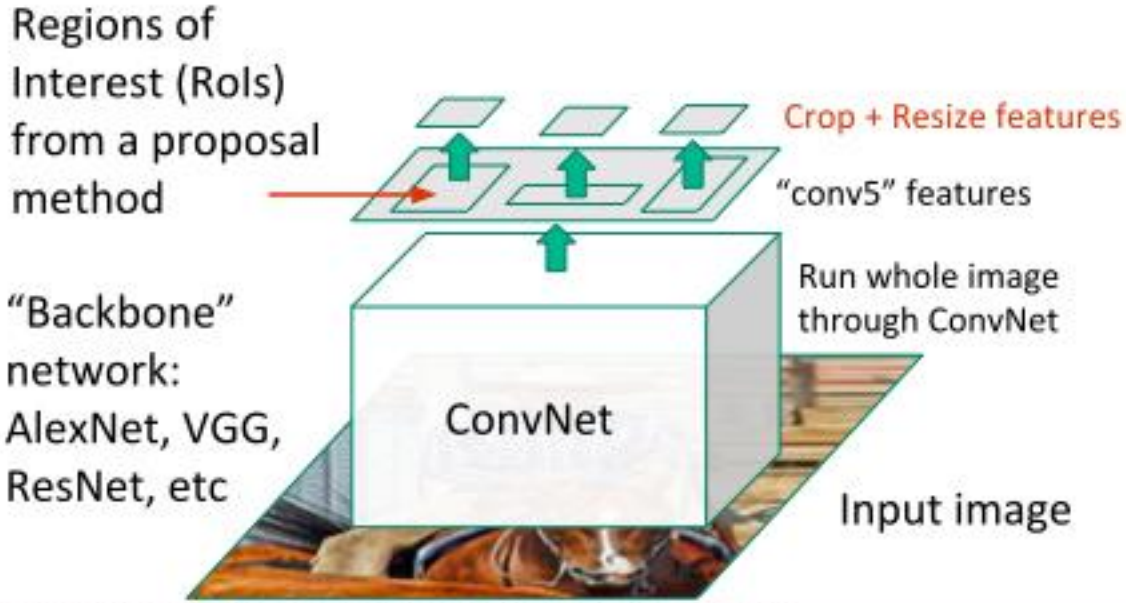


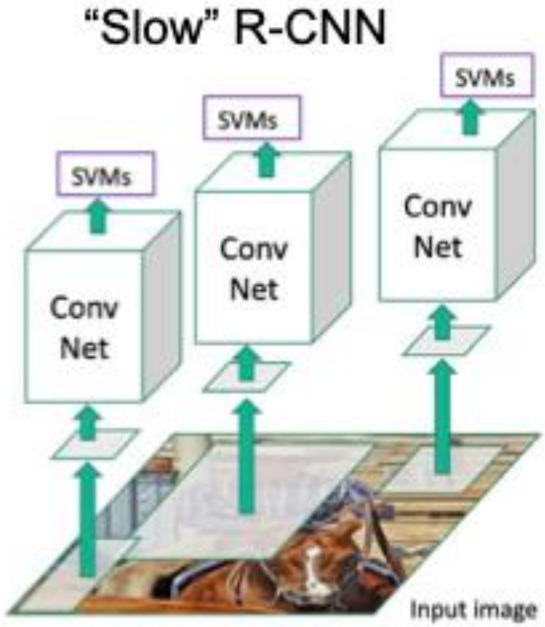
"Slow" R-CNN



Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

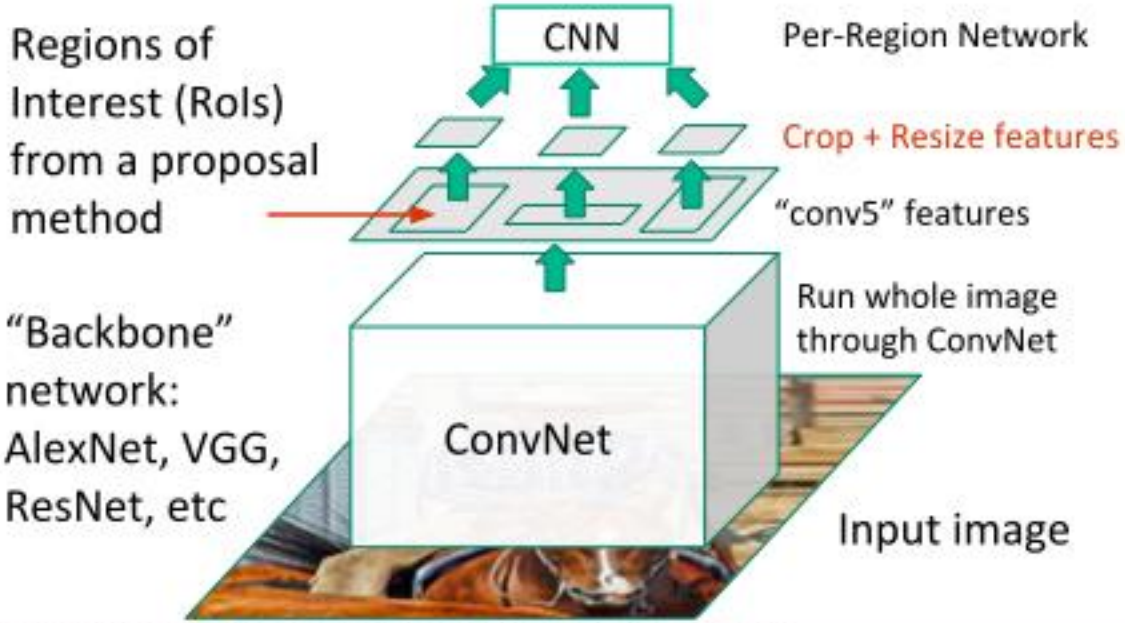
Fast R-CNN

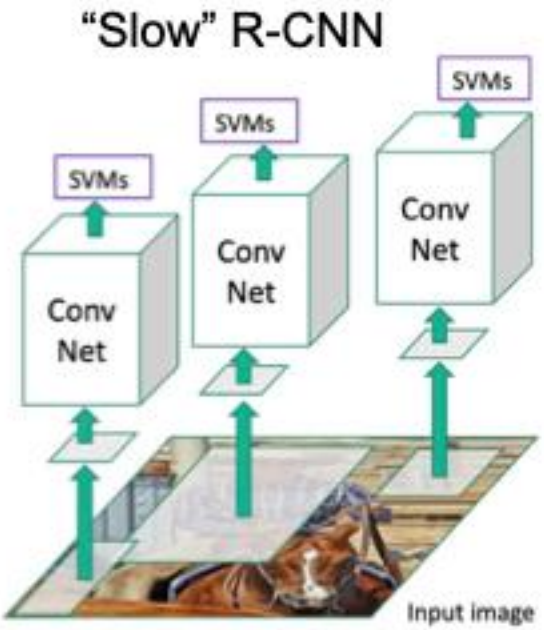




Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

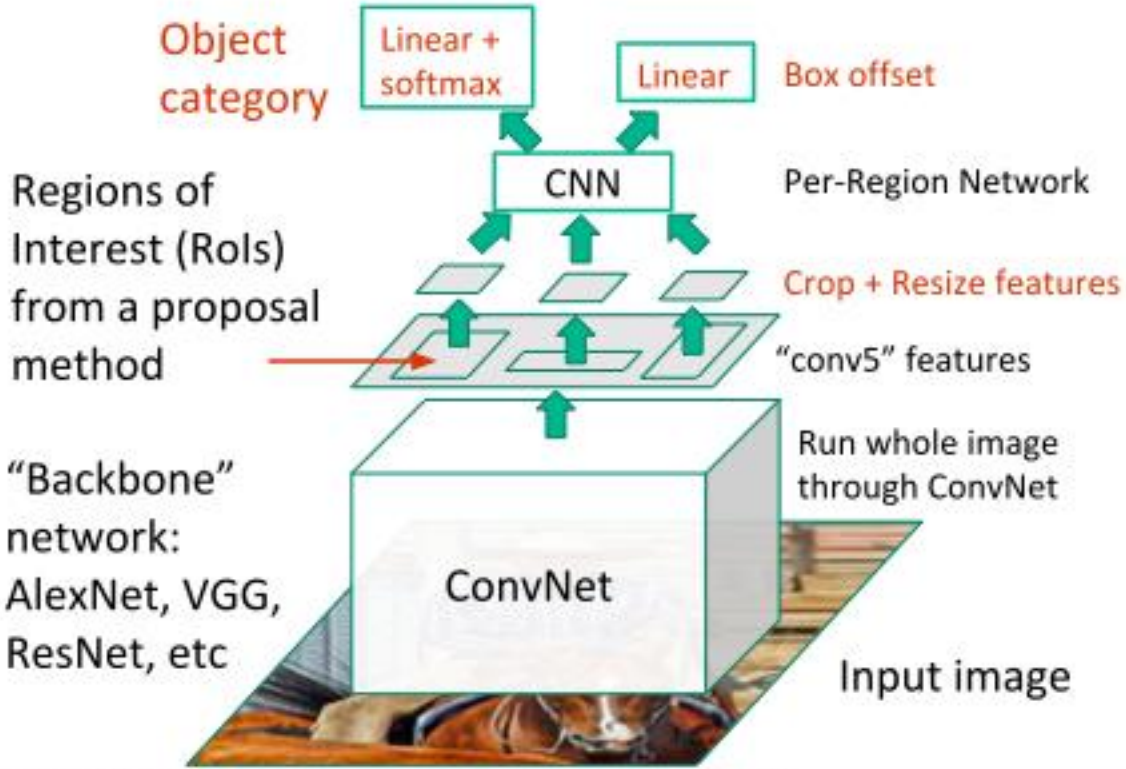
Fast R-CNN

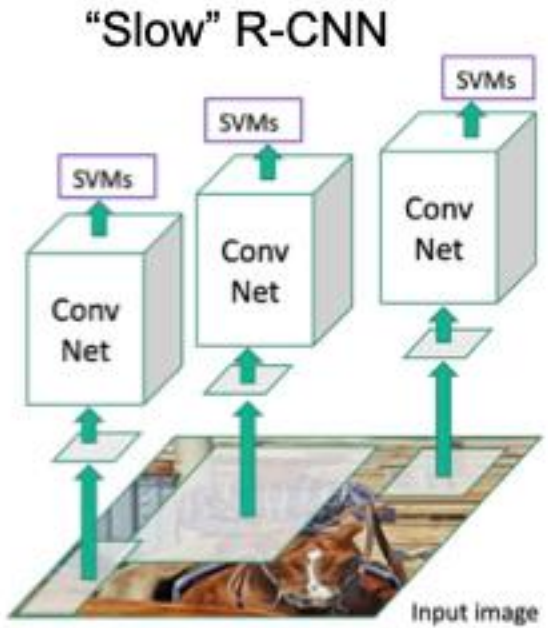




Girshick, 'Fast R-CNN', ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

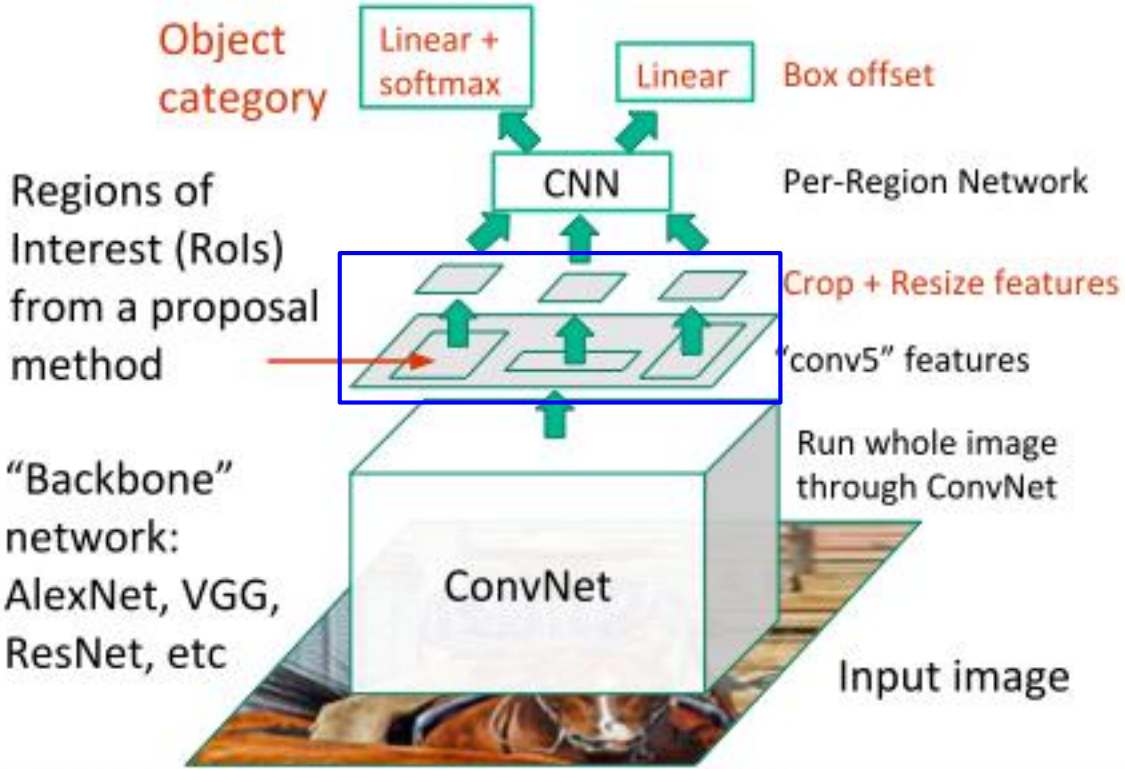
Fast R-CNN

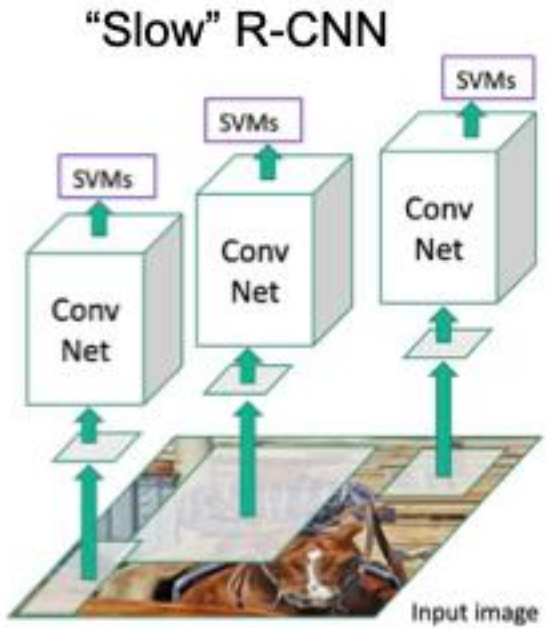




Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

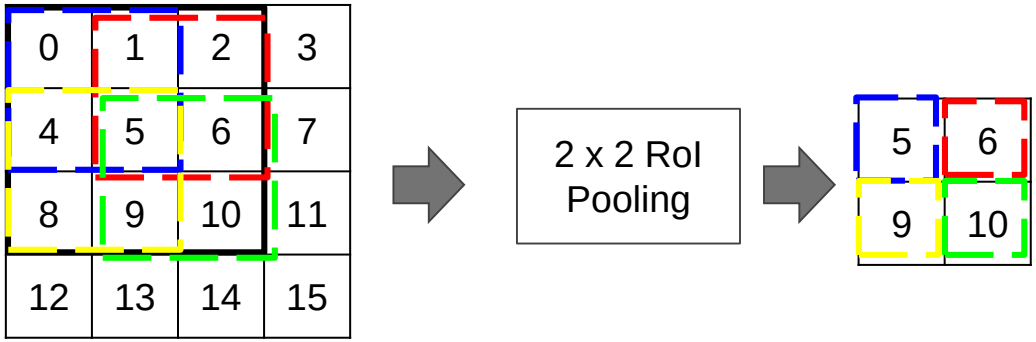
Fast R-CNN





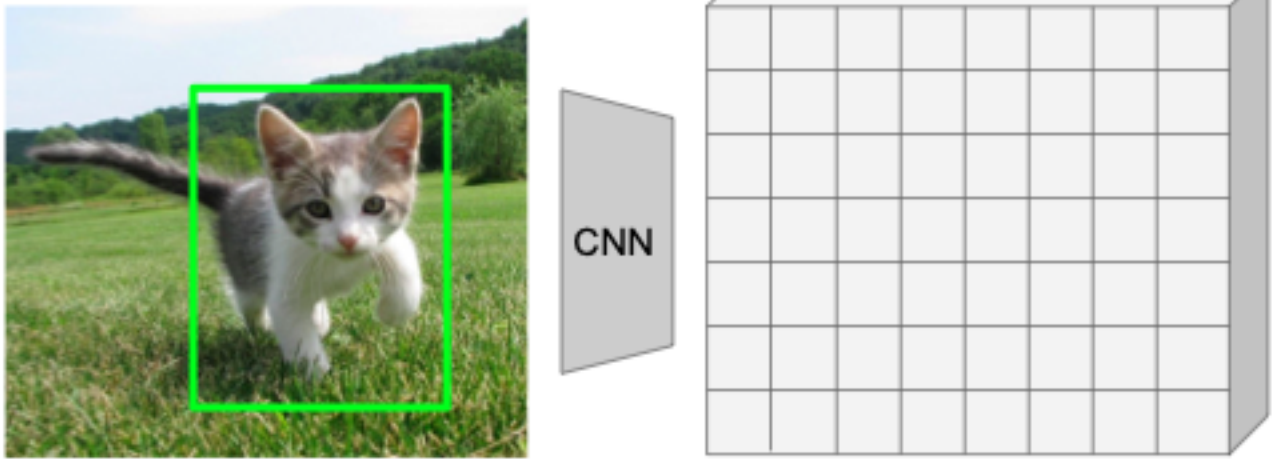
Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

RoI pooling



- Dada uma anchor box (na fig., região 3x3), corta a região uniformemente em *n* x *n* blocos (na fig., *n*=*m*=2) e retorna o valor máximo de cada região
- Se RoI tem tamanho a x a, janela tem tamanho $\lceil a/n \rceil$ e stride $\lfloor a/n \rfloor$
- Retorna n^2 valores para cada RoI

Cropping Features: RoI Pool



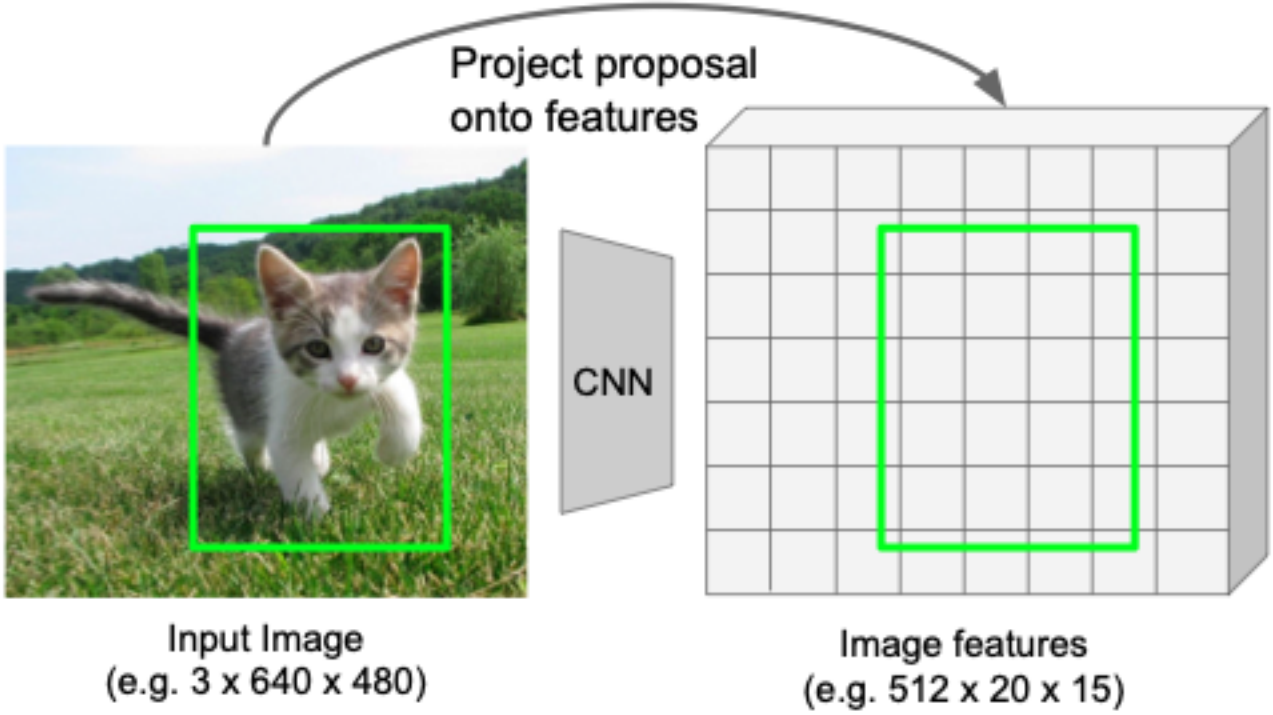
Input Image
(e.g. 3 x 640 x 480)

Image features
(e.g. 512 x 20 x 15)

Girshick, "Fast R-CNN", ICCV 2015.

Girshick, "Fast R-CNN", ICCV 2015.

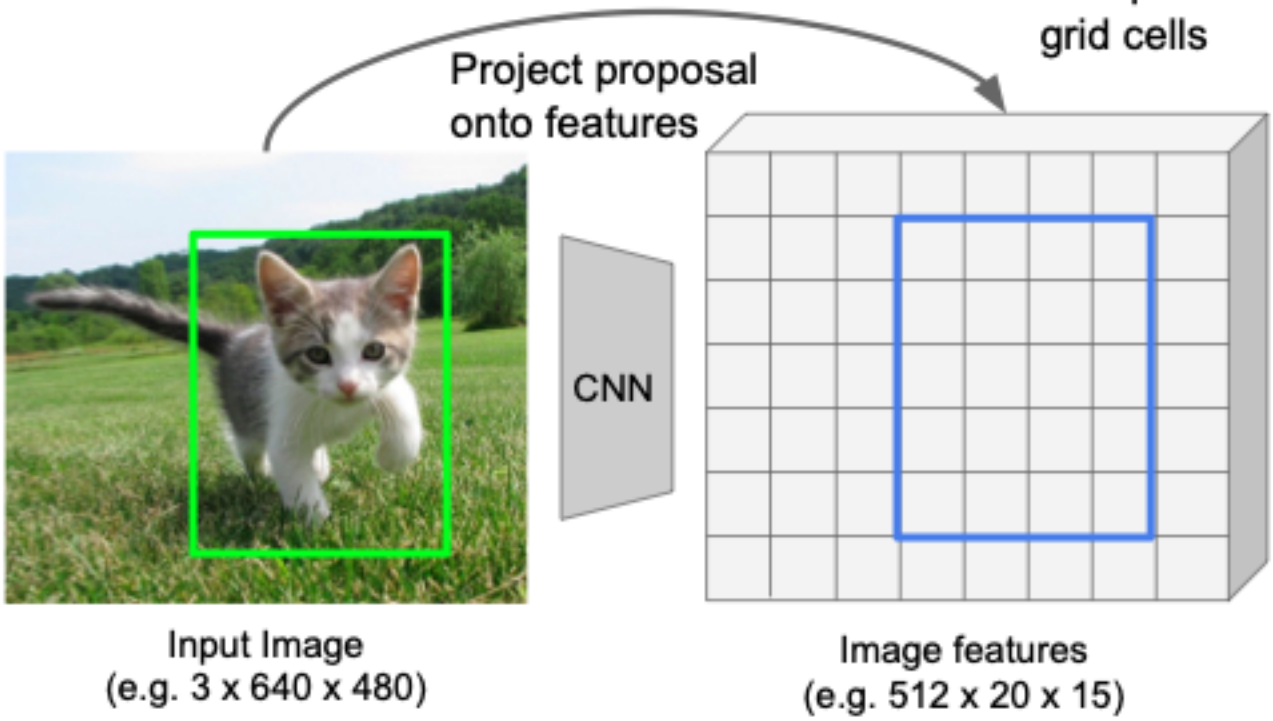
Cropping Features: RoI Pool



Girshick, "Fast R-CNN", ICCV 2015.

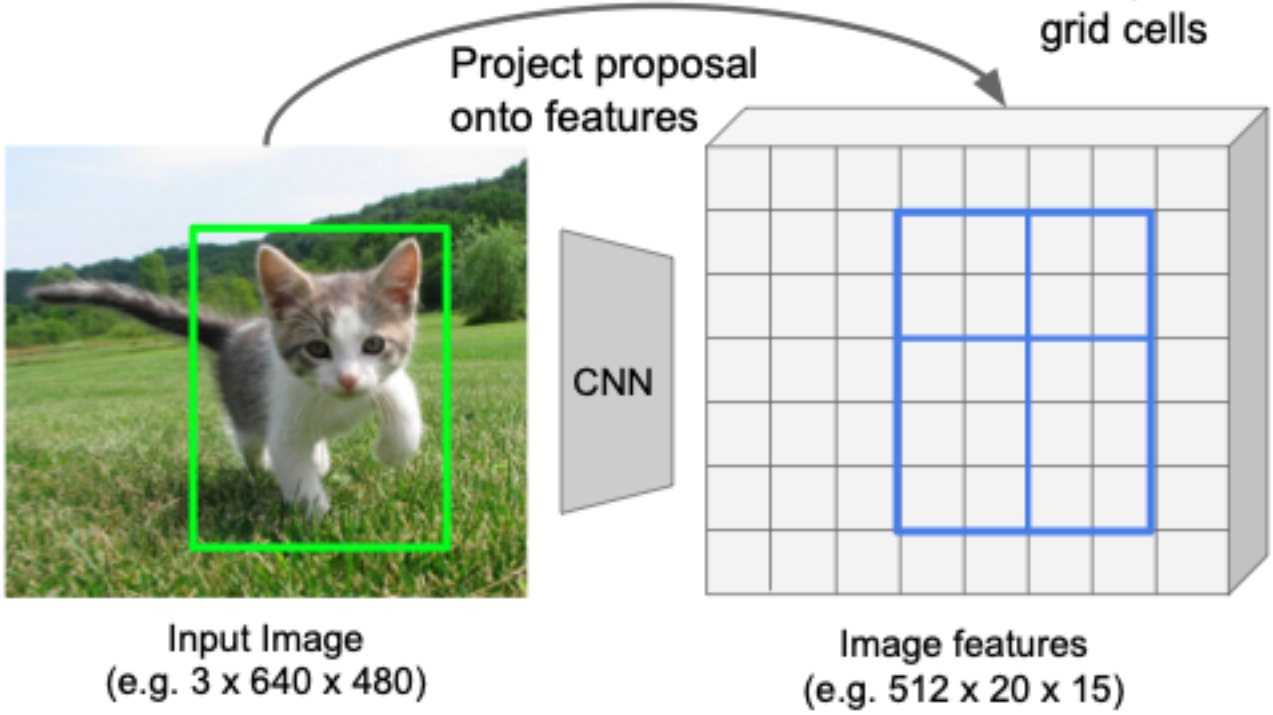
Girshick, "Fast R-CNN", ICCV 2015.

Cropping Features: RoI Pool



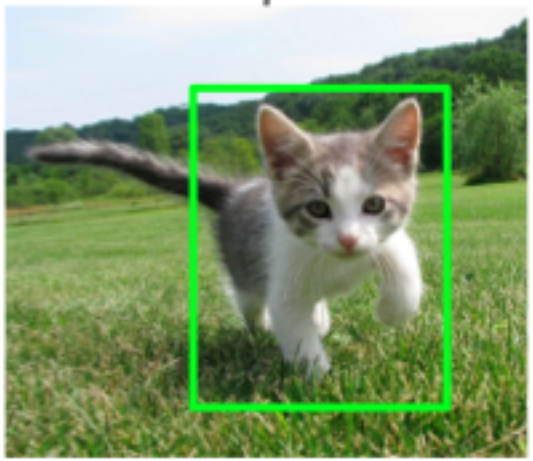
Girshick, "Fast R-CNN", ICCV 2015.

Cropping Features: RoI Pool



Girshick, "Fast R-CNN", ICCV 2015.

Cropping Features: RoI Pool



Input Image
(e.g. 3 x 640 x 480)

Project proposal onto features

"Snap" to grid cells

CNN

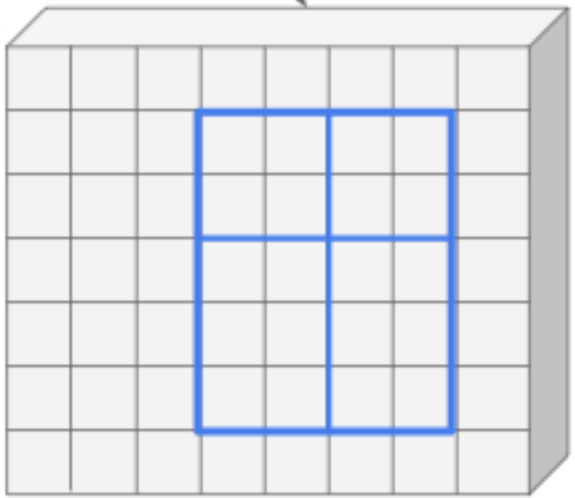
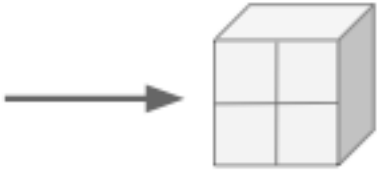


Image features
(e.g. 512 x 20 x 15)

Divide into 2x2 grid of (roughly) equal subregions

Max-pool within each subregion

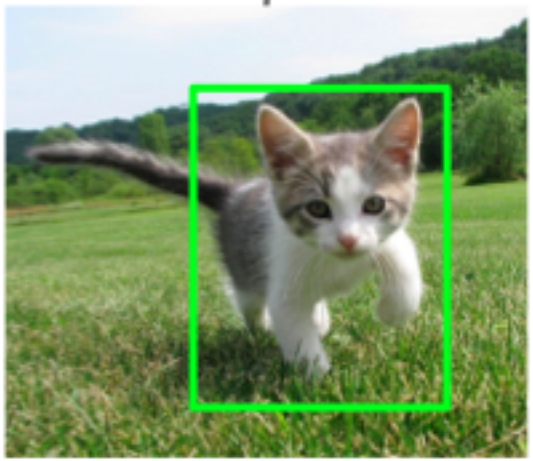


Region features
(here 512 x 2 x 2;
In practice e.g 512 x 7 x 7)

Region features always the same size even if input regions have different sizes!

Girshick, "Fast R-CNN", ICCV 2015.

Cropping Features: RoI Pool



Input Image
(e.g. 3 x 640 x 480)

Project proposal onto features

CNN

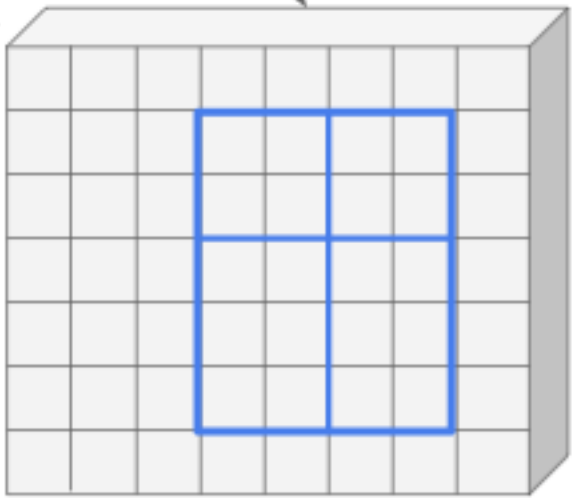
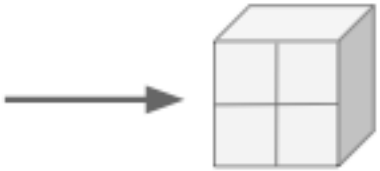


Image features
(e.g. 512 x 20 x 15)

"Snap" to grid cells

Divide into 2x2 grid of (roughly) equal subregions

Max-pool within each subregion



Region features
(here 512 x 2 x 2;
In practice e.g 512 x 7 x 7)

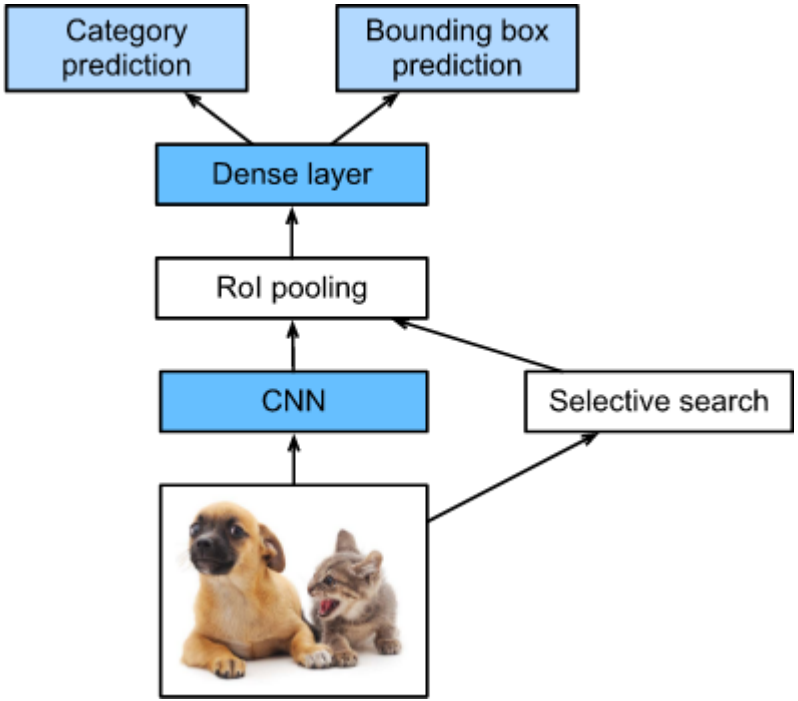
Region features always the same size even if input regions have different sizes!

Problem: Region features slightly misaligned

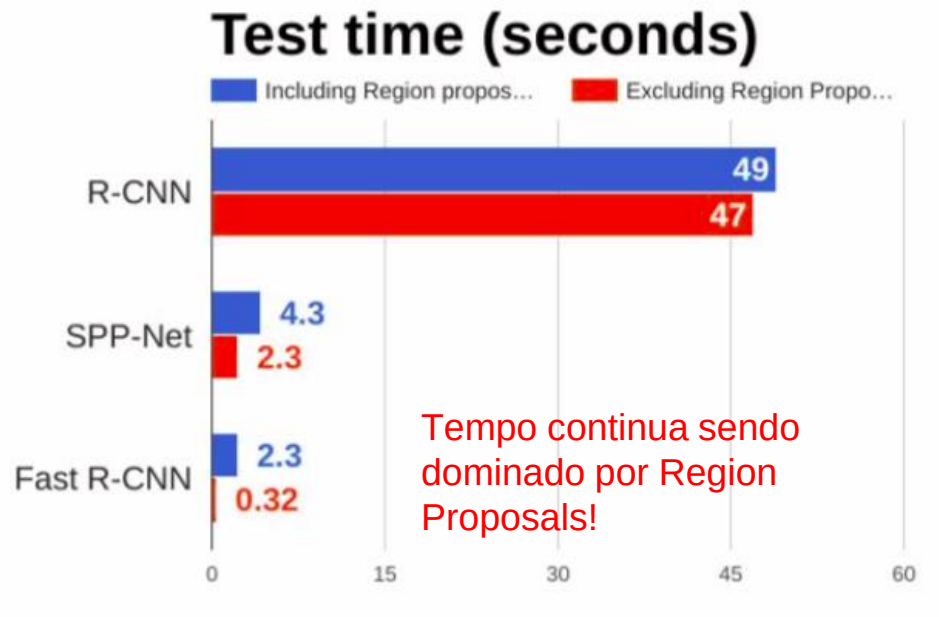
Girshick, "Fast R-CNN", ICCV 2015.

Fast R-CNN

- CNN para extrair features da imagem inteira de uma vez (rápido)
- Selective search seleciona regiões (RoIs) a partir da imagem original
- RoI (region of interest) pooling toma as features de cada região e retorna conjunto de features de tamanho fixo
- Há um problema de alinhamento com o RoI Pooling cuja solução só foi proposta em 2017 (Mask R-CNN)



R-CNN vs. Fast R-CNN



Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
He et al, “Spatial pyramid pooling in deep convolutional networks for visual recognition”, ECCV 2014
Girshick, “Fast R-CNN”, ICCV 2015

Detecção de objetos por region proposals

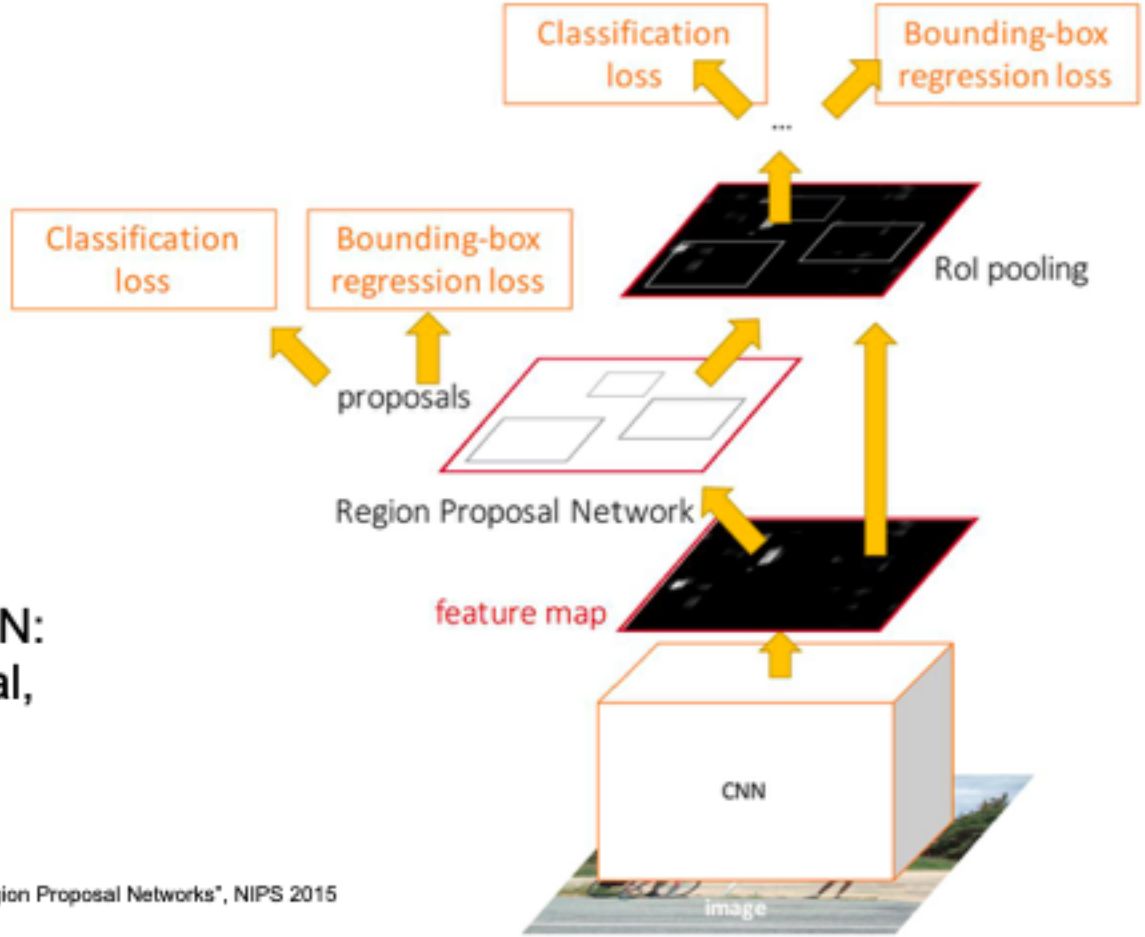
Fast**er** R-CNN

Faster R-CNN:

Make CNN do proposals!

Insert **Region Proposal Network (RPN)** to predict proposals from features

Otherwise same as Fast R-CNN: Crop features for each proposal, classify each one



Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015
Figure copyright 2015, Ross Girshick; reproduced with permission

Region Proposal Network

Input Image
(e.g. 3 x 640 x 480)

CNN

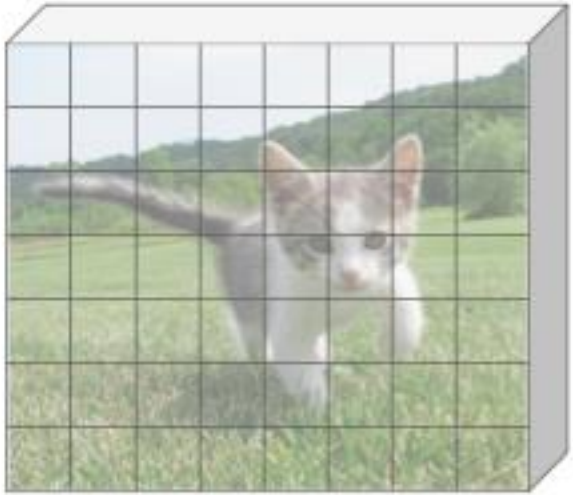


Image features
(e.g. 512 x 20 x 15)

Region Proposal Network

Imagine an **anchor box** of fixed size at each point in the feature map

Input Image
(e.g. 3 x 640 x 480)

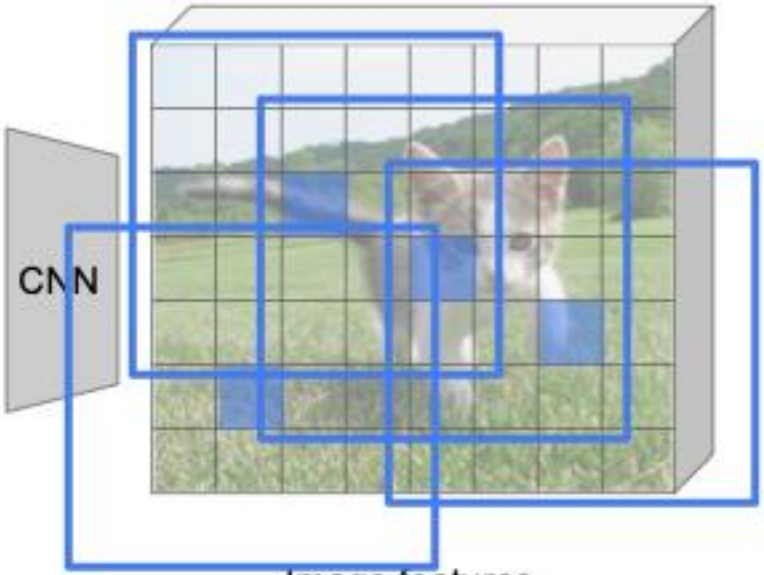


Image features
(e.g. 512 x 20 x 15)

Region Proposal Network

Imagine an **anchor box** of fixed size at each point in the feature map

Input Image
(e.g. 3 x 640 x 480)

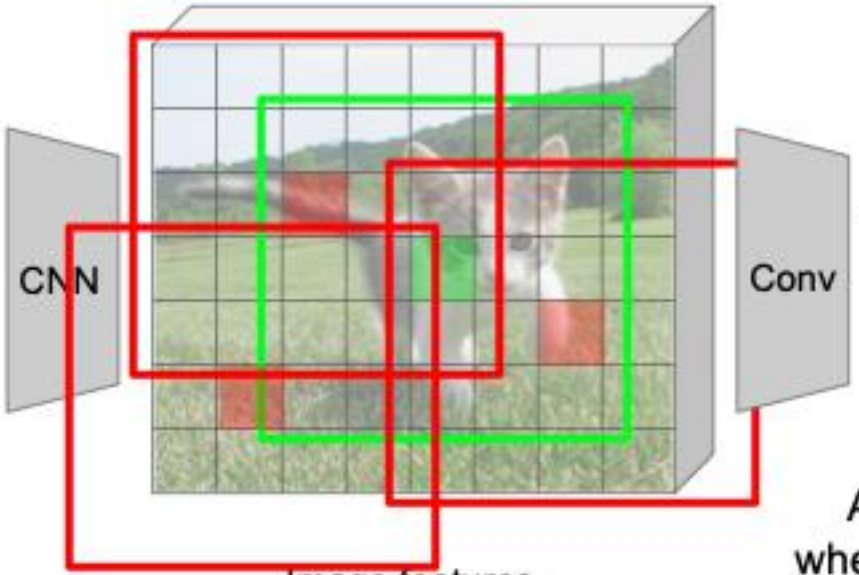


Image features
(e.g. 512 x 20 x 15)

At each point, predict whether the corresponding anchor contains an object (per-pixel logistic regression)

Region Proposal Network

Imagine an **anchor box** of fixed size at each point in the feature map

Input Image
(e.g. 3 x 640 x 480)

CNN

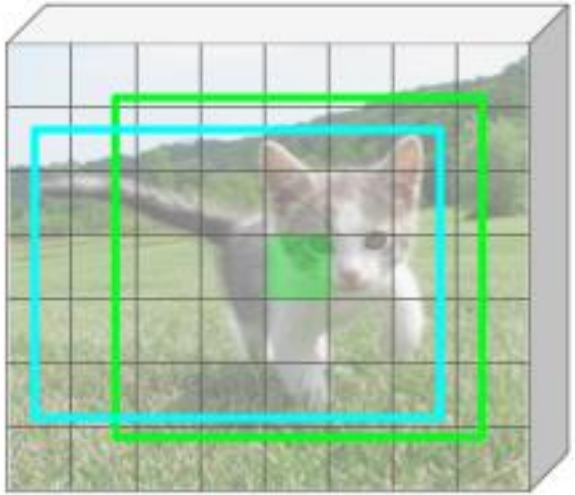


Image features
(e.g. 512 x 20 x 15)

Conv

Anchor is an object?
1 x 20 x 15

Box transforms
4 x 20 x 15

For positive boxes, also predict a transformation from the anchor to the ground-truth box (regress 4 numbers per pixel)

Region Proposal Network

In practice use K different anchor boxes of different size / scale at each point

Input Image
(e.g. 3 x 640 x 480)

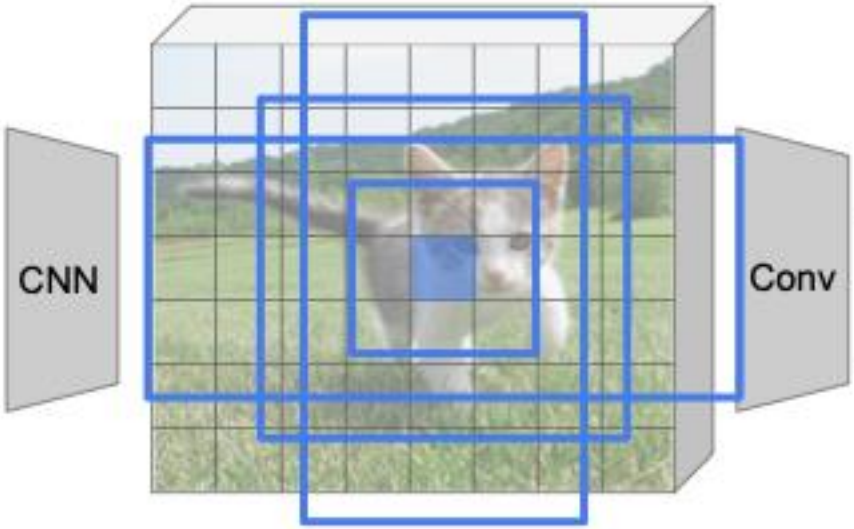


Image features
(e.g. 512 x 20 x 15)

Region Proposal Network

In practice use K different anchor boxes of different size / scale at each point

Input Image
(e.g. 3 x 640 x 480)

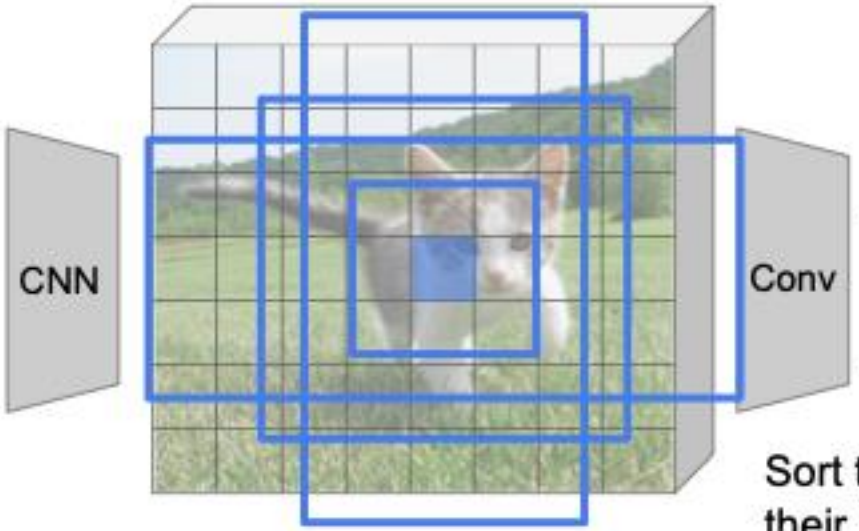


Image features
(e.g. 512 x 20 x 15)

Anchor is an object?
K x 20 x 15

Box transforms
4K x 20 x 15

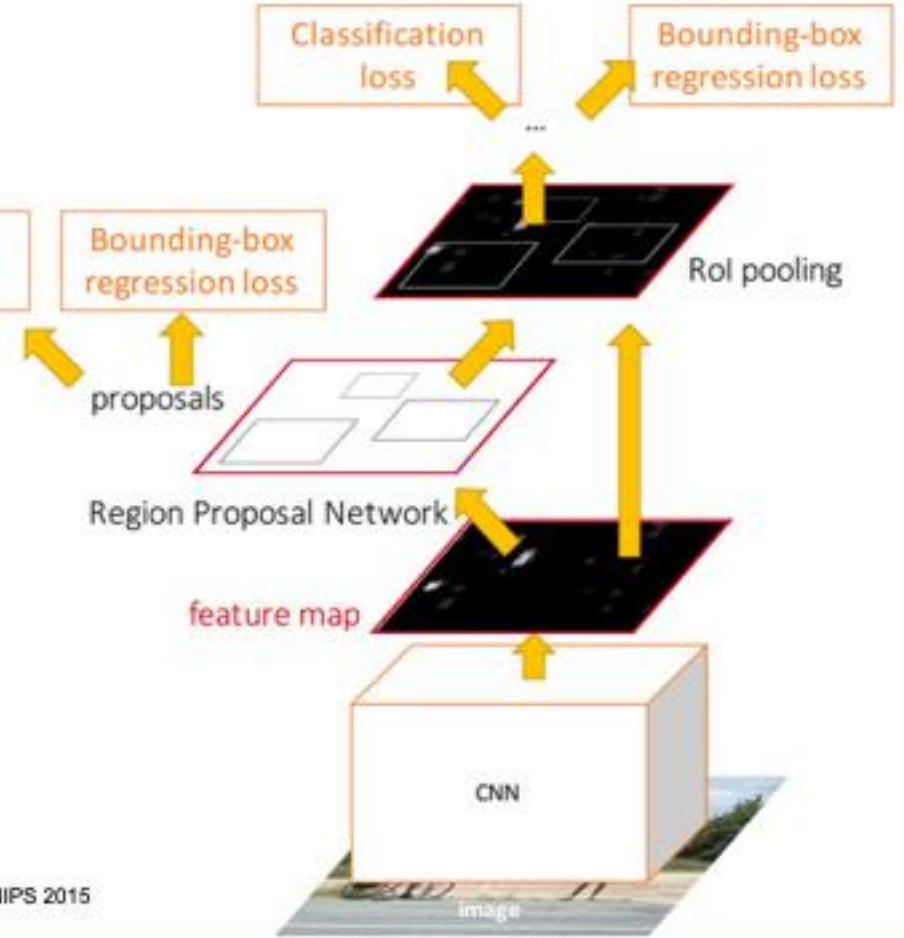
Sort the K*20*15 boxes by their "object" score, take top ~300 as our proposals

Faster R-CNN:

Make CNN do proposals!

Jointly train with 4 losses:

1. RPN classify object / not object
2. RPN regress box coordinates
3. Final classification score (object classes)
4. Final box coordinates



Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015
Figure copyright 2015, Ross Girshick; reproduced with permission

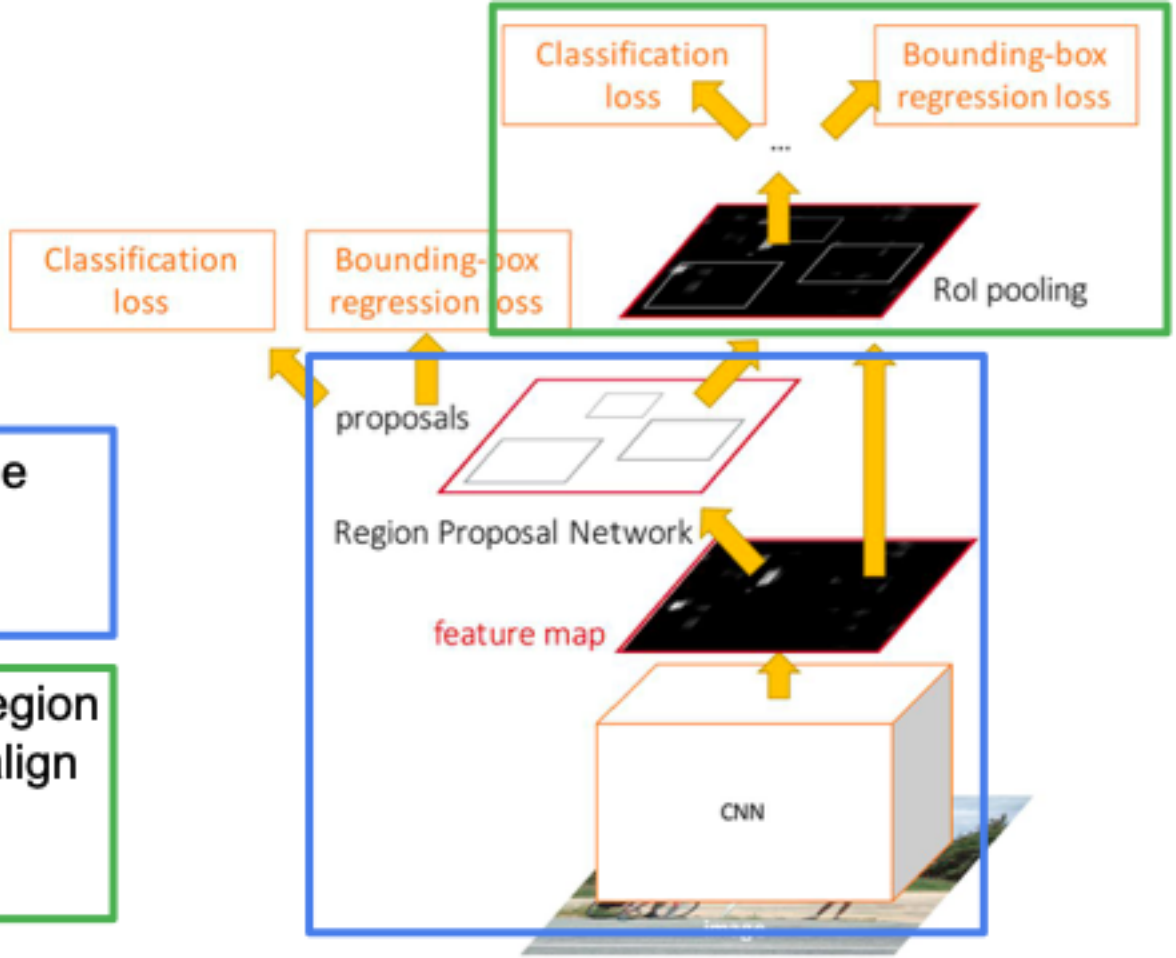
Faster R-CNN:

Make CNN do proposals!

Faster R-CNN is a
Two-stage object detector

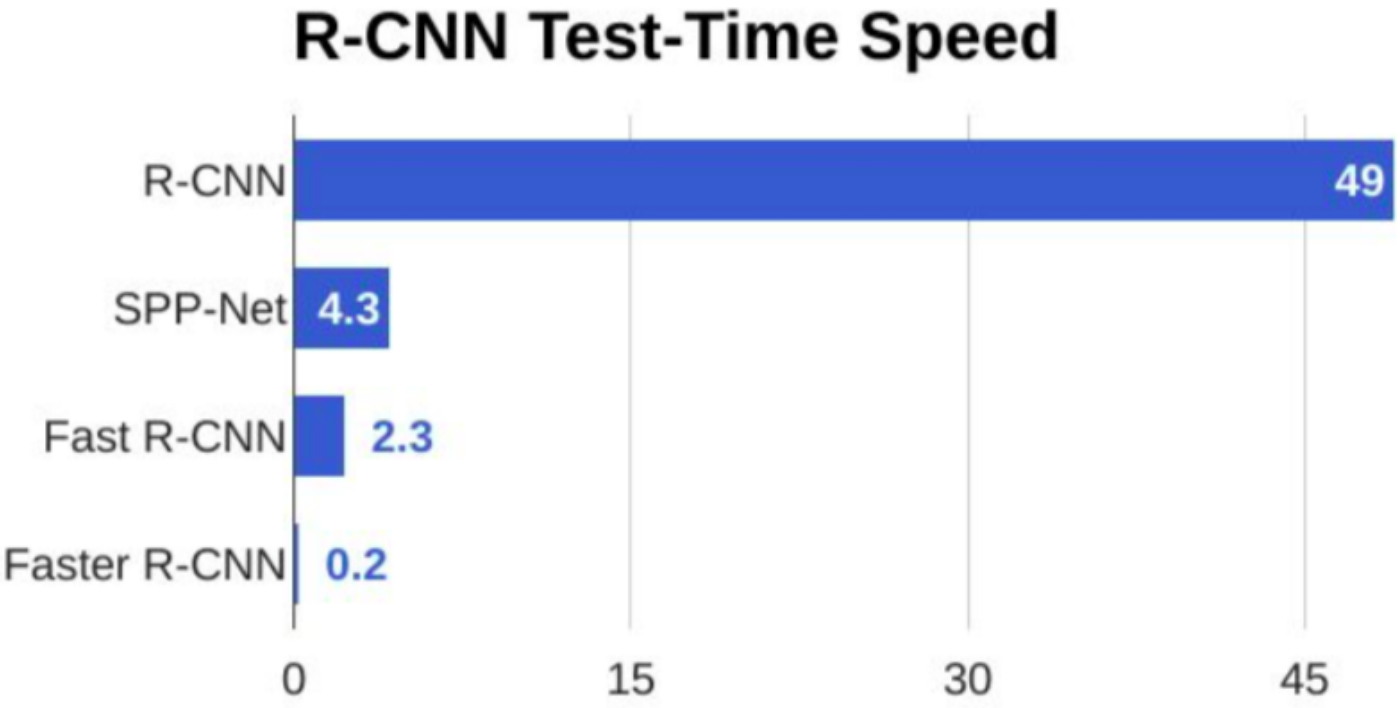
First stage: Run once per image
- Backbone network
- Region proposal network

Second stage: Run once per region
- Crop features: RoI pool / align
- Predict object class
- Prediction bbox offset



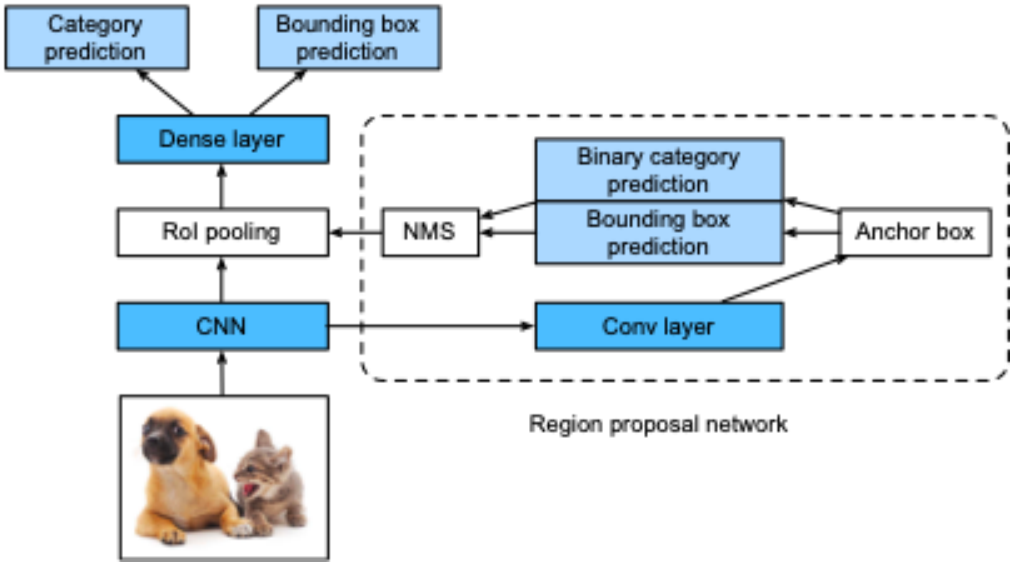
Faster R-CNNs

Fazer com que CNN proponha as regiões



Faster R-CNN

- Usa uma rede para propor regiões no lugar do selective search, encontrando anchor boxes de alta qualidade



Faster algorithms

R-CNN: Propõe regiões. Classifica as regiões propostas uma de cada vez. Retorna rótulo + bounding box

Fast R-CNN: Propõe regiões. Usar a implementação convolucional de janelas deslizantes para classificar todas as regiões propostas.

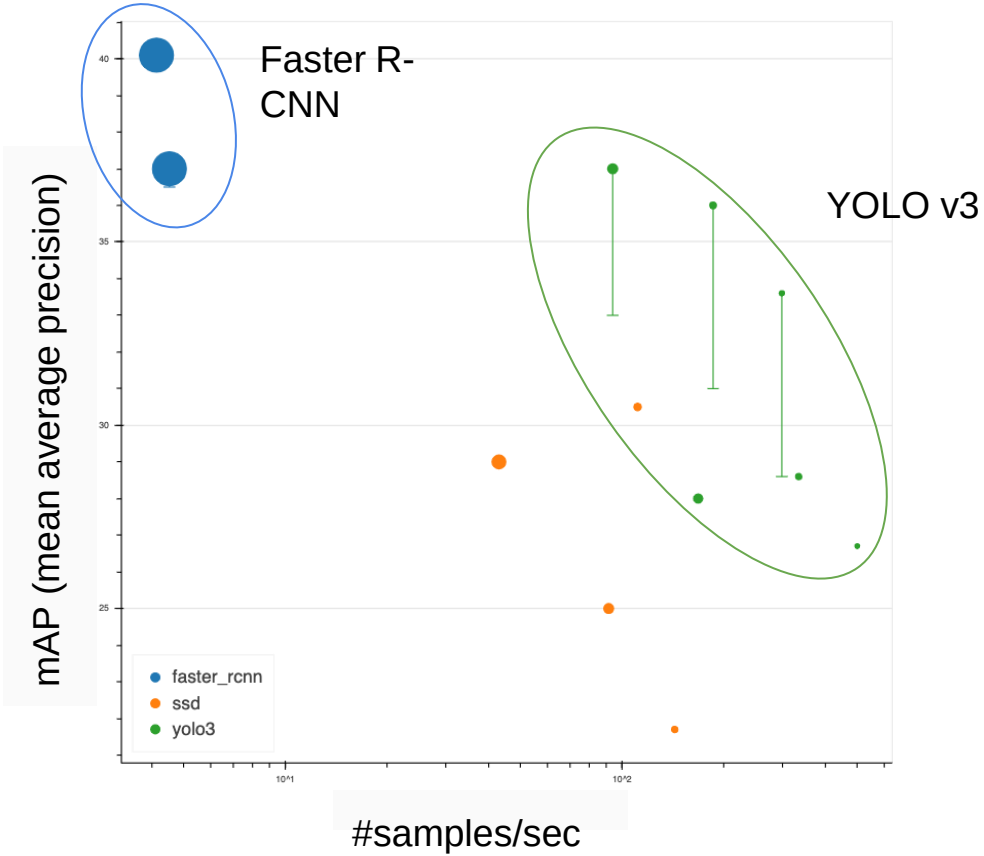
Faster R-CNN: Usa rede convolucional para propor regiões.

[Girshik et. al, 2013, Rich feature hierarchies for accurate object detection and semantic segmentation]
[Girshik, 2015. Fast R-CNN]
[Ren et. al, 2016. Faster R-CNN: Towards real-time object detection with region proposal networks]

Faster algorithms

- R-CNN não retorna apenas o blob quando um objeto é encontrada, mas também um bounding box
  - Ainda é muito ineficiente
- Fast R-CNN é mais rápida: usa implementação convolucional de janelas deslizantes
- Faster R-CNN é bem mais rápida que Fast R-CNN: usa implementação convolucional para segmentação (em vez de algoritmo clássico)
  - YOLO ainda é mais rápido (não possui dois passos: propor regiões e depois classificar), mas os resultados às vezes são inferiores

Comparação no dataset COCO com modelos pré-treinados



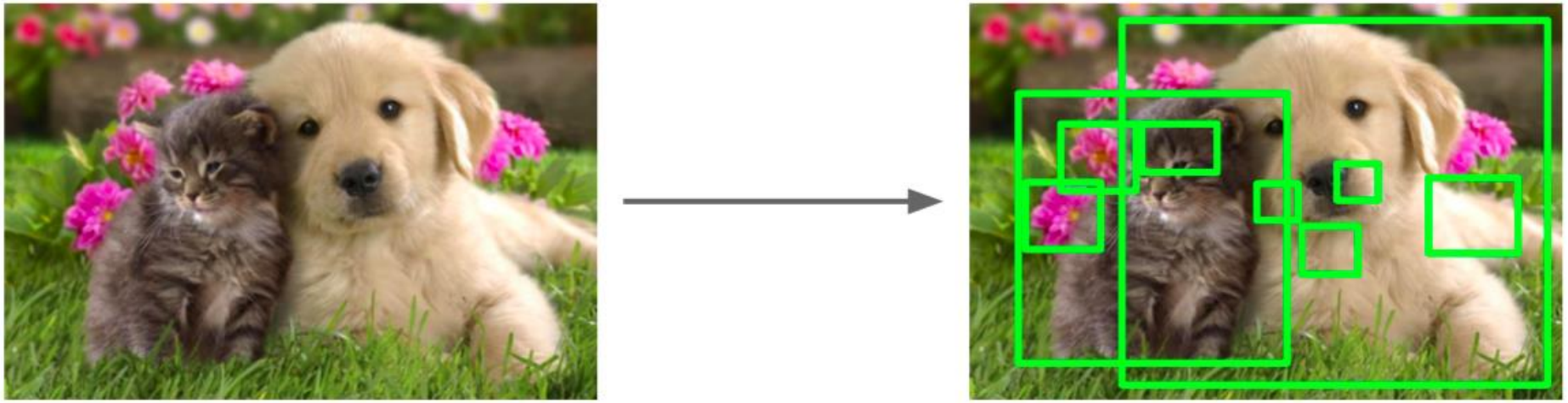
[https://gluon-cv.mxnet.io/model_zoo/detection.html]

Segmentação de instâncias

Mask R-CNNs

Relembrando: proposta de regiões

- Encontrar "blobs" (regiões da imagem) que provavelmente contém objetos
- R-CNN e Fast R-CNN usam Selective Search p/ retornar ~2000 RoIs
- Faster R-CNN usa region proposal network (RPN)



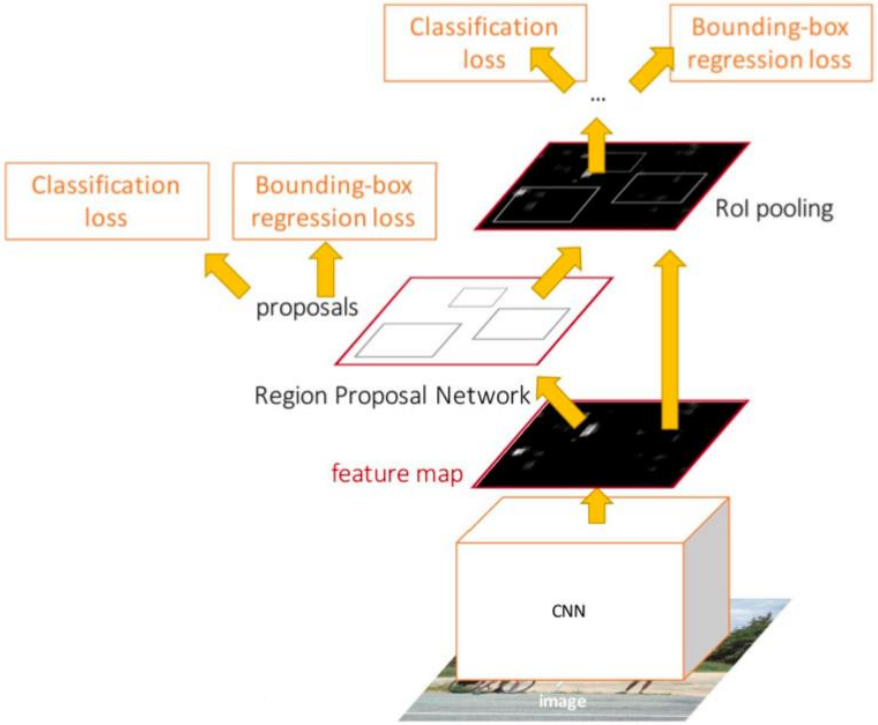
Faster R-CNN

Propostas feitas a partir do feature map

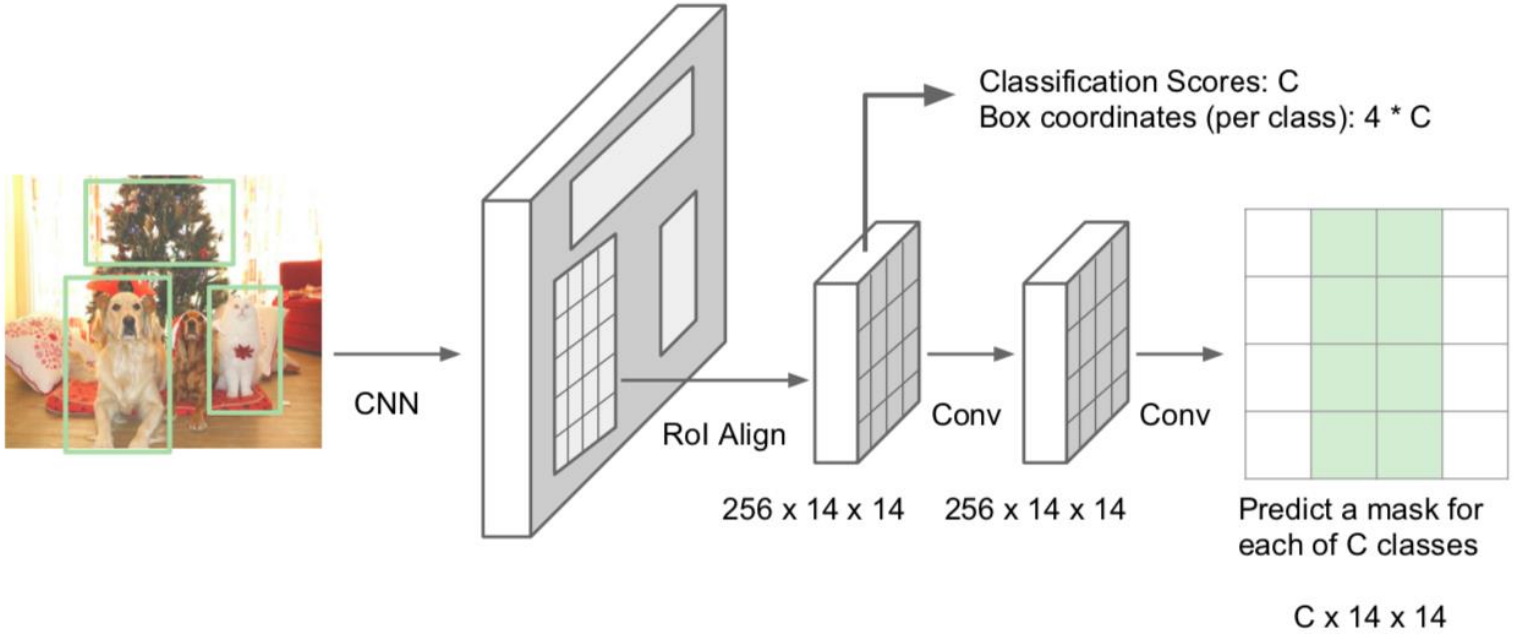
Inserir Region Proposal Network (RPN) para retornar proposals diretamente a partir das features

Treinar conjuntamente com 4 perdas:

1. RPN objeto/não-objeto
2. RPN coordenadas bounding box
3. Score final de classificação (classes de objetos)
4. Coordenadas finais da bbox



Mask R-CNN



Mask R-CNN **estende** Faster R-CNN para segmentação de instâncias **adicionando um branch** para prever **máscaras de segmentação** paralelo ao branch existente para classificação e bbox regression.

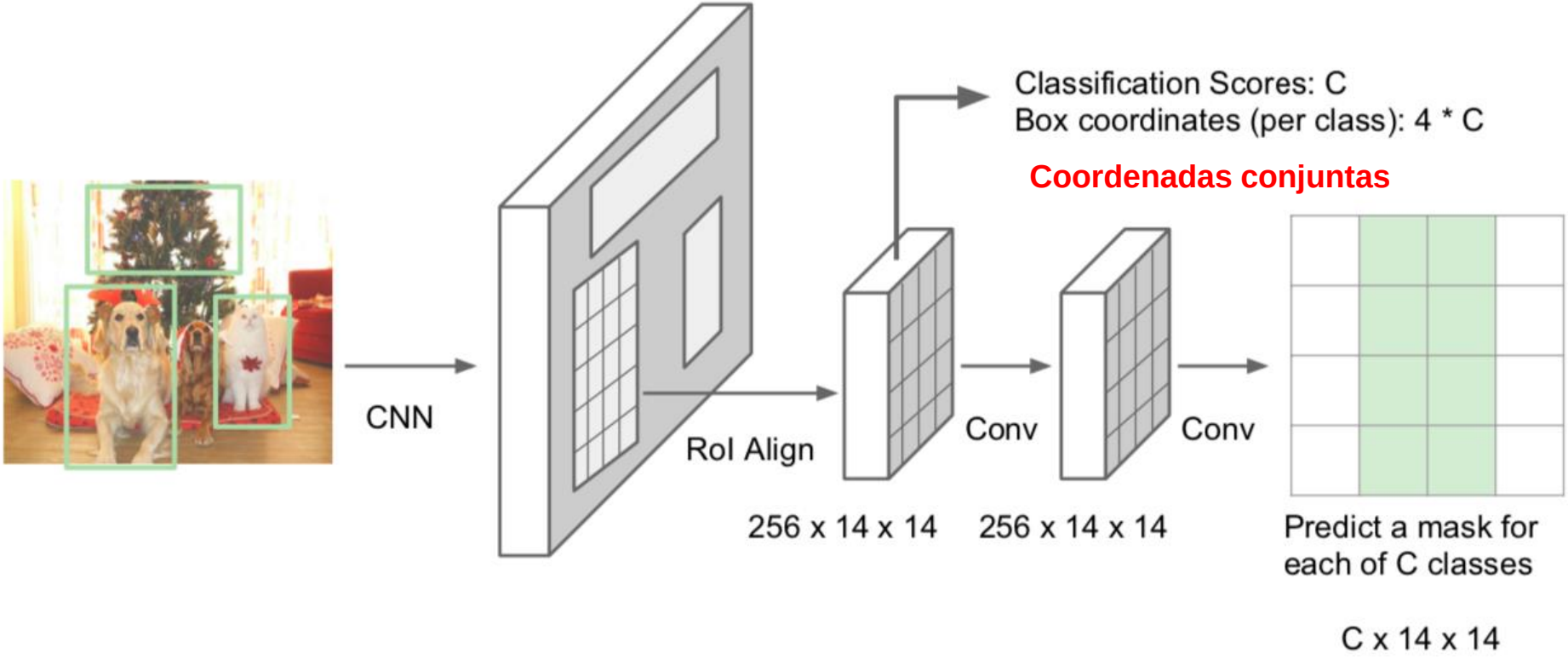
Maior dificuldade em se usar Faster R-CNN para segmentação é o **problema de alinhamento** causado pelo **RoIPool**. Artigo propõe **RoIAlign** para resolver o problema.

Mask R-CNN: resultados muito bons!

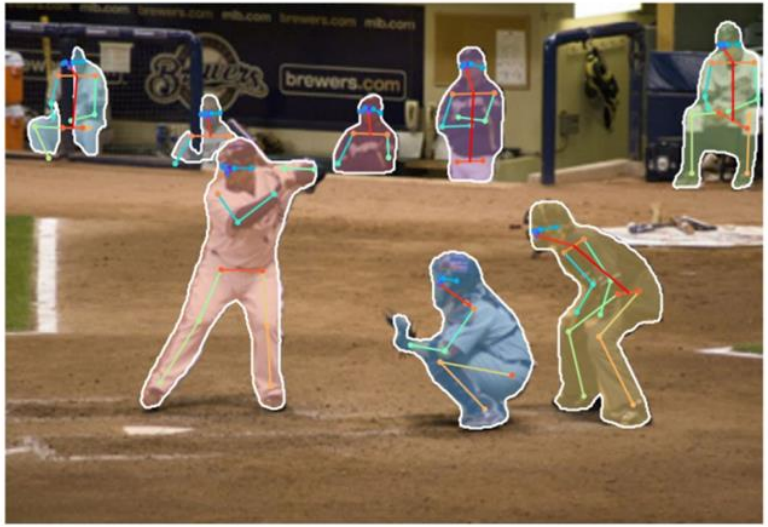
Segmentação de instâncias mostrada no exemplo acima.

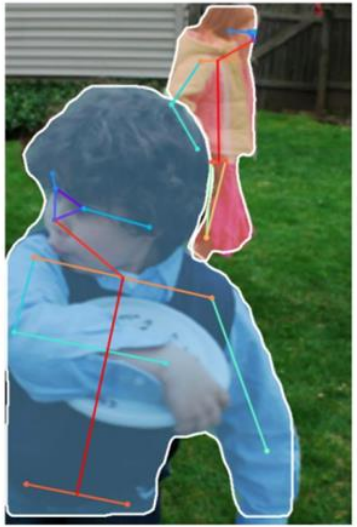
Mask R-CNN também supera Faster R-CNN + {RoIPool, RoIAlign} na detecção de objetos.

Mask R-CNN também pode ser usado para estimar pose

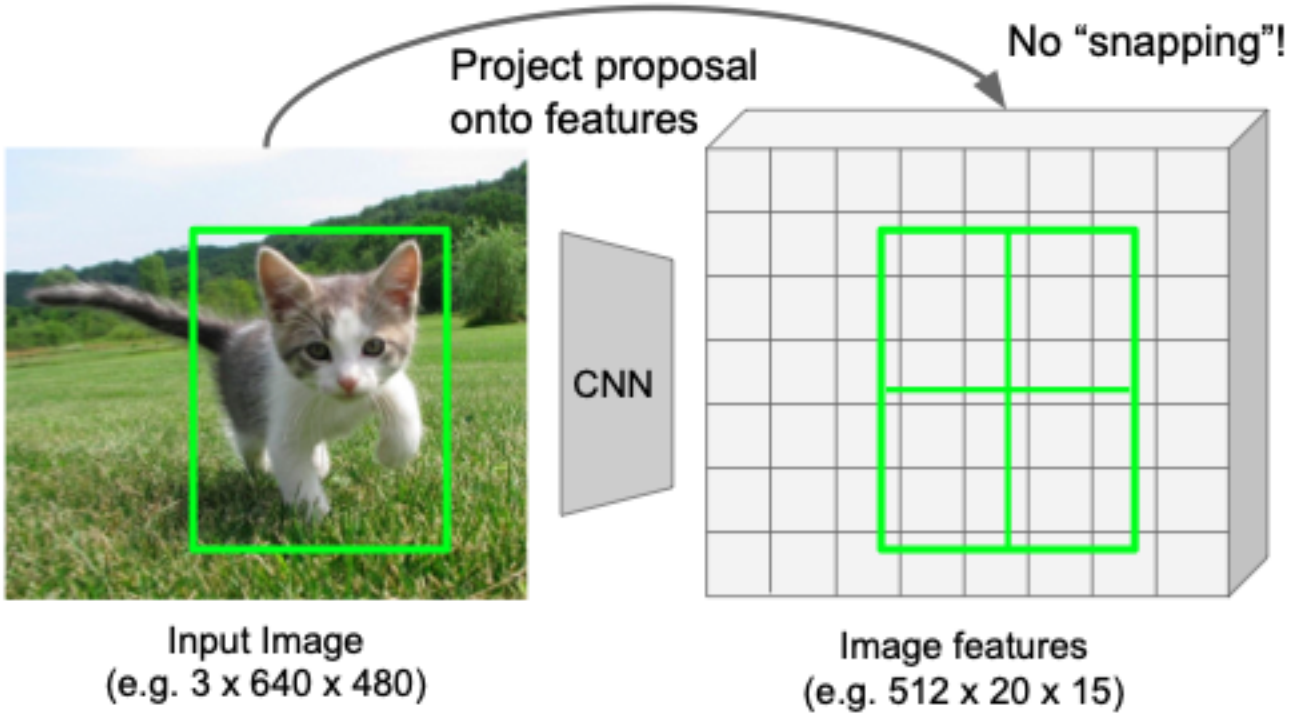


Resultados de estimação de pose





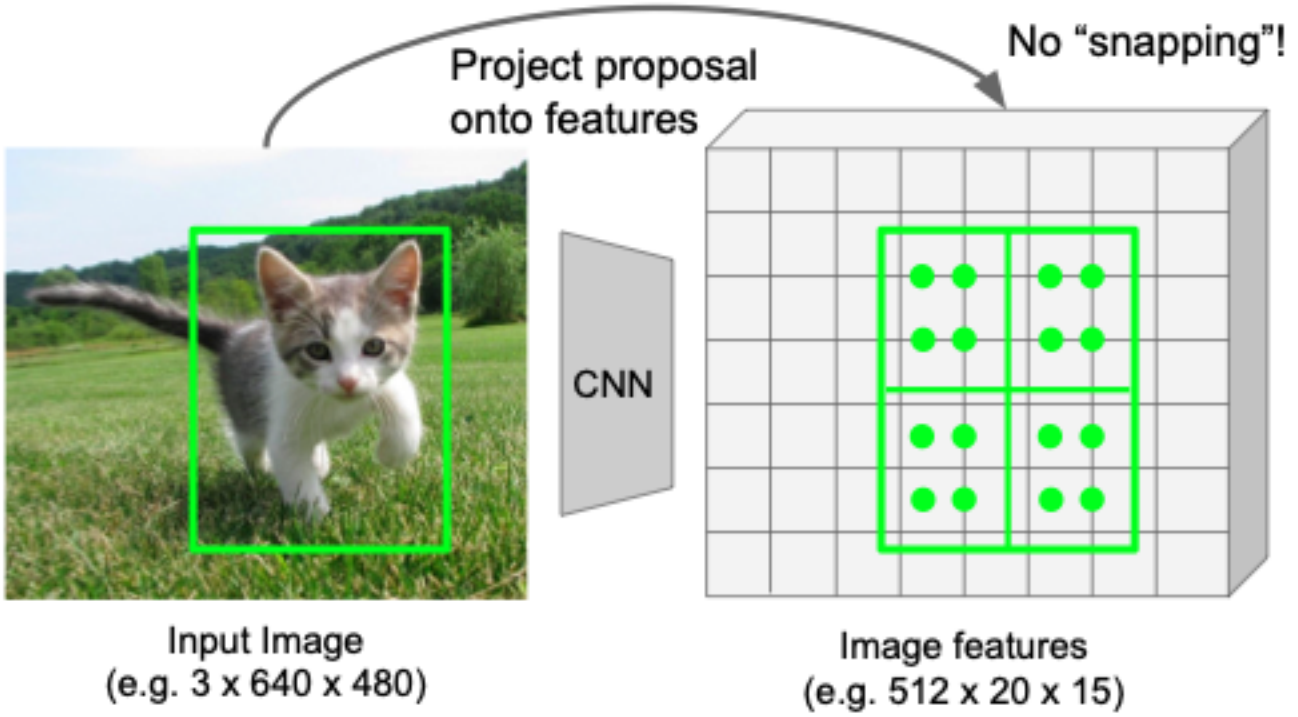
Cropping Features: RoI Align



He et al, "Mask R-CNN", ICCV 2017

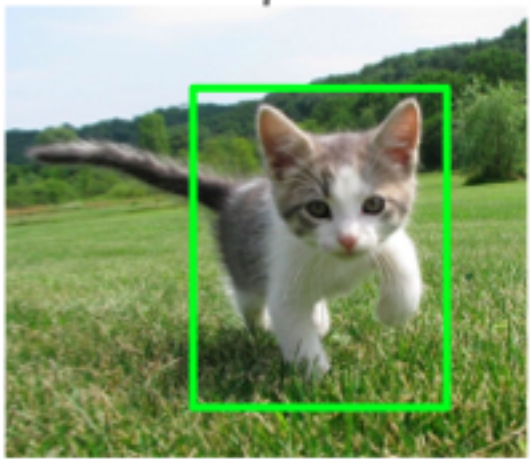
Cropping Features: RoI Align

Sample at regular points in each subregion using bilinear interpolation



He et al, "Mask R-CNN", ICCV 2017

Cropping Features: RoI Align



Input Image
(e.g. 3 x 640 x 480)

Project proposal onto features

No "snapping"!

CNN

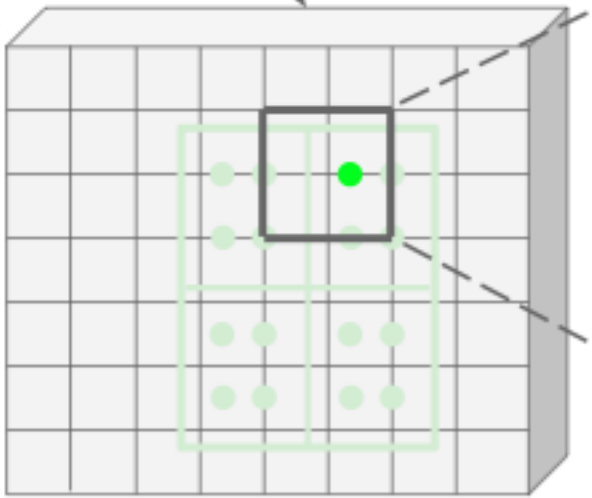
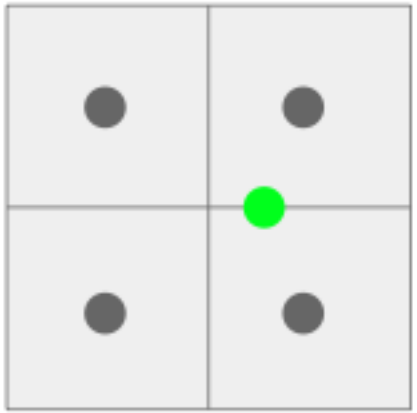


Image features
(e.g. 512 x 20 x 15)

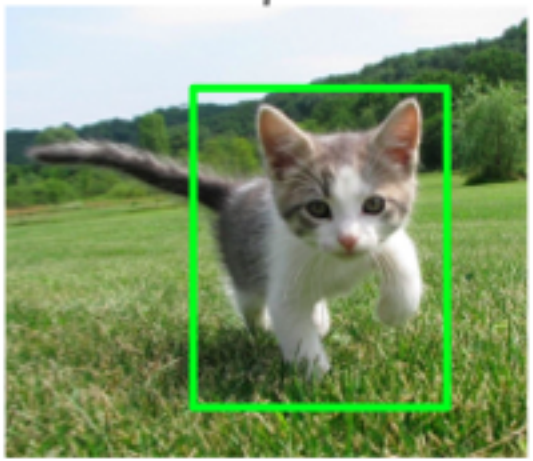
Sample at regular points in each subregion using bilinear interpolation



Feature f_{xy} for point (x, y) is a linear combination of features at its four neighboring grid cells:

He et al, "Mask R-CNN", ICCV 2017

Cropping Features: RoI Align



Input Image
(e.g. 3 x 640 x 480)

Project proposal onto features

No "snapping"!

CNN

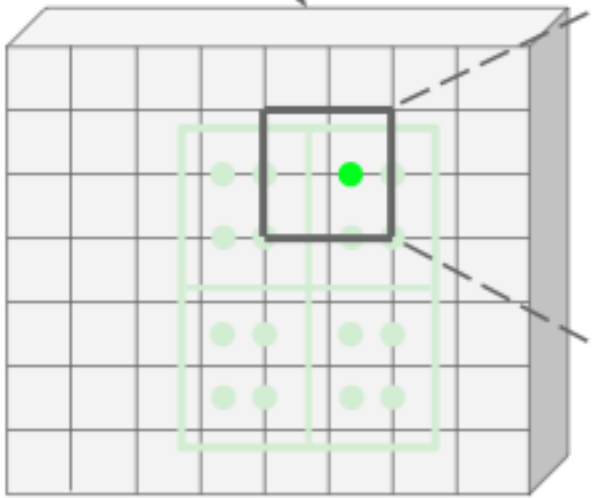
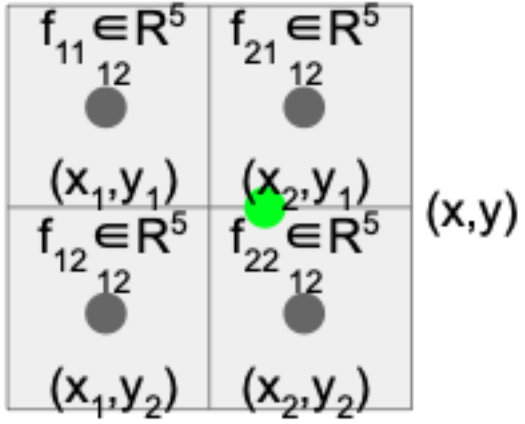


Image features
(e.g. 512 x 20 x 15)

Sample at regular points in each subregion using bilinear interpolation



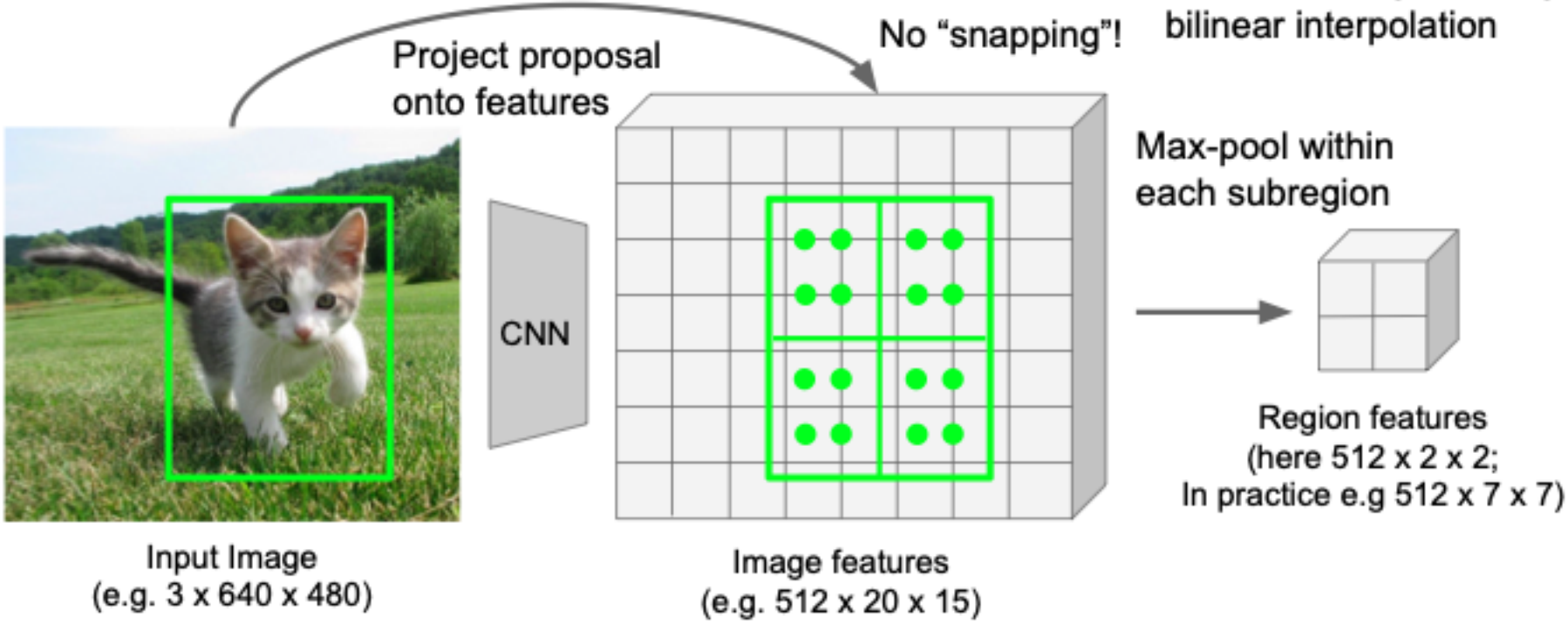
Feature f_{xy} for point (x, y) is a linear combination of features at its four neighboring grid cells:

$$f_{xy} = \sum_{i,j=1}^{2} f_{i,j} \max(0, 1 - |x - x_i|) \max(0, 1 - |y - y_j|)$$

He et al, "Mask R-CNN", ICCV 2017

Cropping Features: RoI Align

Sample at regular points in each subregion using bilinear interpolation



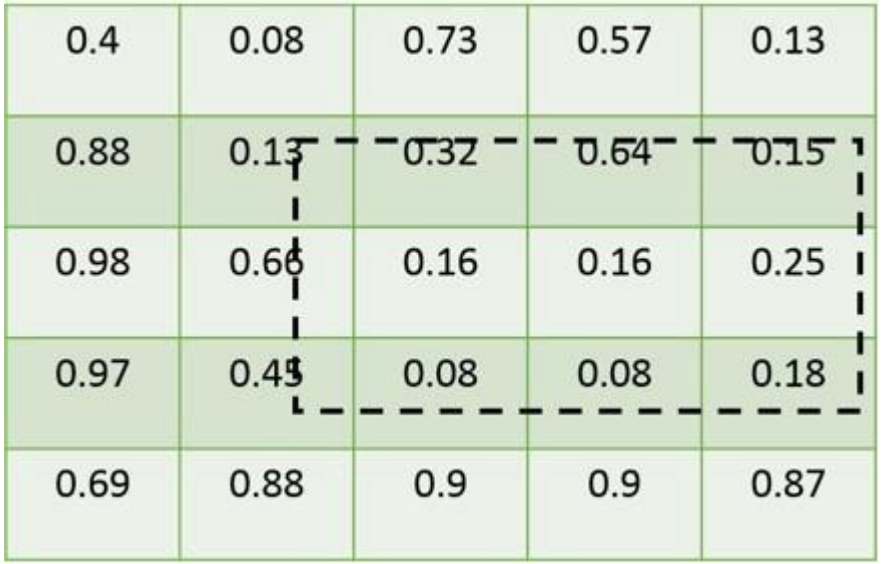
He et al, "Mask R-CNN", ICCV 2017

RoIPool vs. RoIAlign

RoIPool

- Suponha um feature map 5 x 5 com um RoI mostrado no retângulo pontilhado.

- Limites do RoI não coincidem com granularidade do feature map.



https://medium.com/@fractaldle/mask-r-cnn-unmasked-c029aa2f1296

RoIPool

- Limites do RoI são então quantizados (arredondados) para casar com feature map (snap to grid).

- Output size m x m. Suponha m=2. RoI é dividido em 4 bins.

0.4	0.08	0.73	0.57	0.13
0.88	0.13	0.32	0.64	0.15
0.98	0.66	0.16	0.16	0.25
0.97	0.45	0.08	0.08	0.18
0.69	0.88	0.9	0.9	0.87

RoIPool

- De novo, limites dos bins não se alinham com feature map.

- Faz-se uma nova quantização.

0.4	0.08	0.73	0.57	0.13
0.88	0.13	0.32	0.64	0.15
0.98	0.66	0.16	0.16	0.25
0.97	0.45	0.08	0.08	0.18
0.69	0.88	0.9	0.9	0.87

RoIPool

- MaxPool (ou AveragePool) sobre cada bin

- As 2 quantizações não são problema para detecção de objetos, mas sim para segmentação

0.4	0.08	0.73	0.57	0.13
0.88	0.13	0.32	0.64	0.15
0.98	0.66	0.16	0.16	0.25
0.97	0.45	0.08	0.08	0.18
0.69	0.88	0.9	0.9	0.87

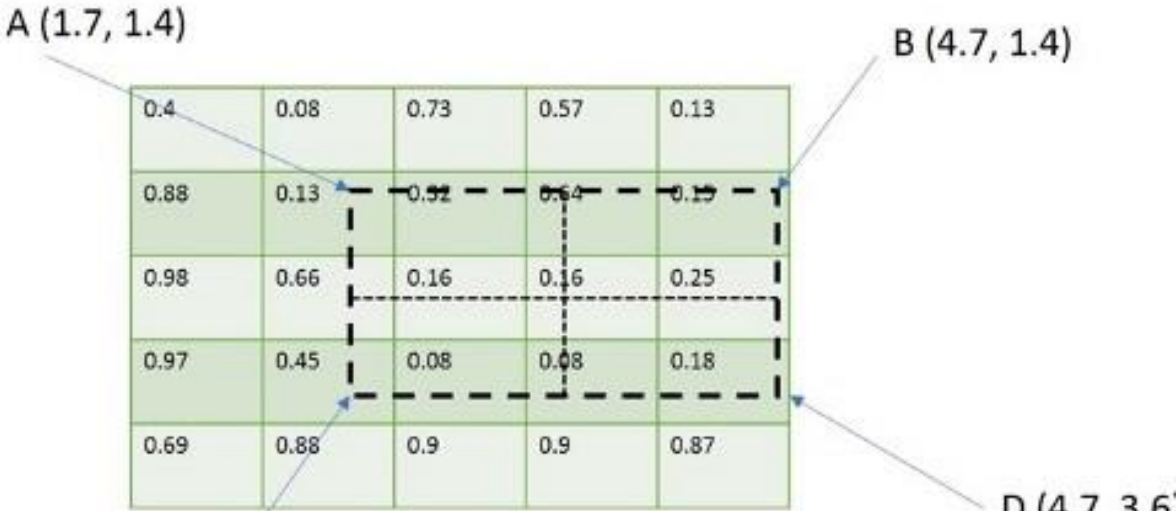
Resultado final com MaxPool

0.32	**0.64**
0.16	**0.25**

RoIAlign

- Mesmo RoI inicial. Desta vez, não há quantização. Saída deve ser matriz m x m, com m=2.

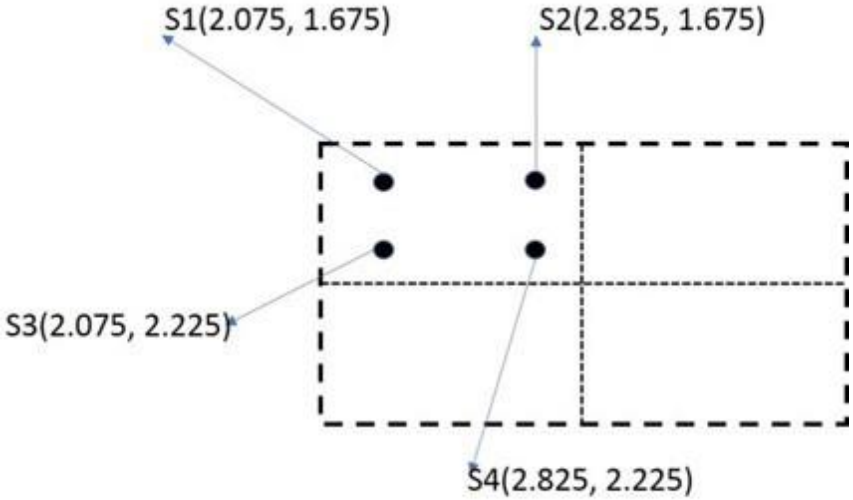
- Dividimos o RoI em m^2=4 bins.



RoIAlign

- Dentro de cada bin, selecionamos 4 pontos (ok selecionar mais pontos, mas não melhora muito).

- Pontos são escolhidos conforme equações ao lado.



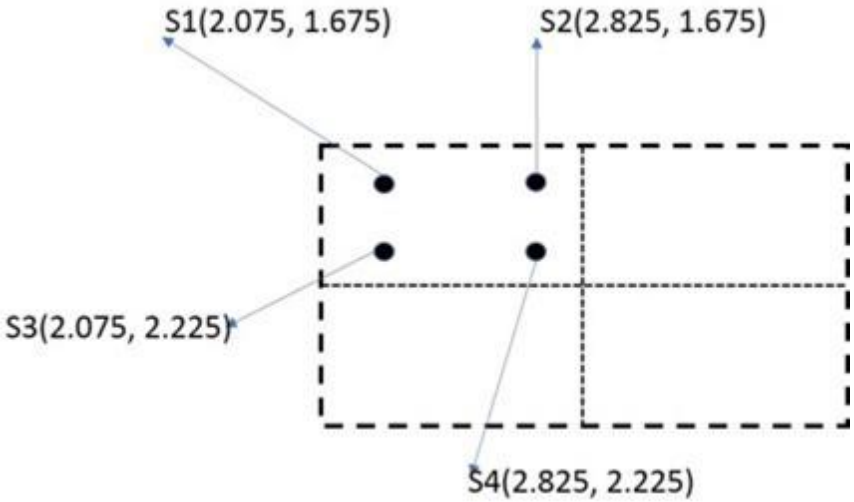
$$x = X_{low} + ((i + 0.5) * \frac{X_{high} - X_{low}}{numsamples}$$

$$y = Y_{low} + ((j + 0.5) * \frac{Y_{high} - Y_{low}}{numsamples}$$

$$i, j \in [0,1 \ldots, num\ of\ samples)$$

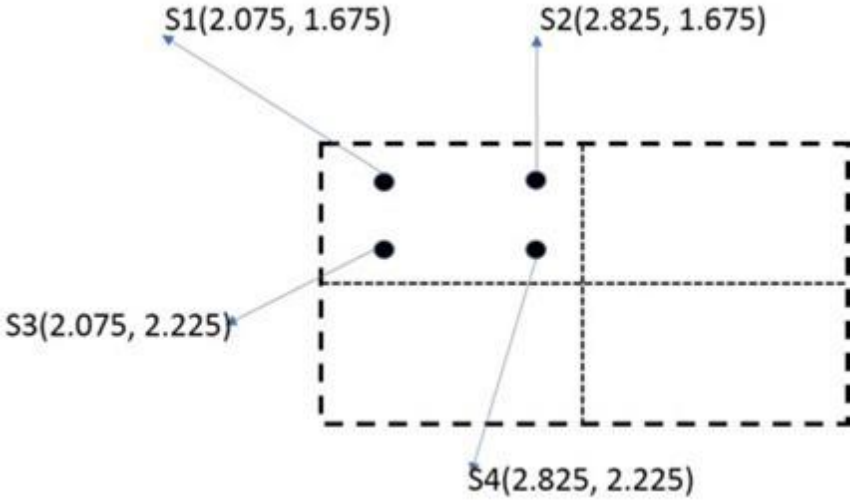
RoIAlign

- Feature em cada ponto é obtida a partir da interpolação dos pontos mais próximos (neste caso, P, Q, R, S).



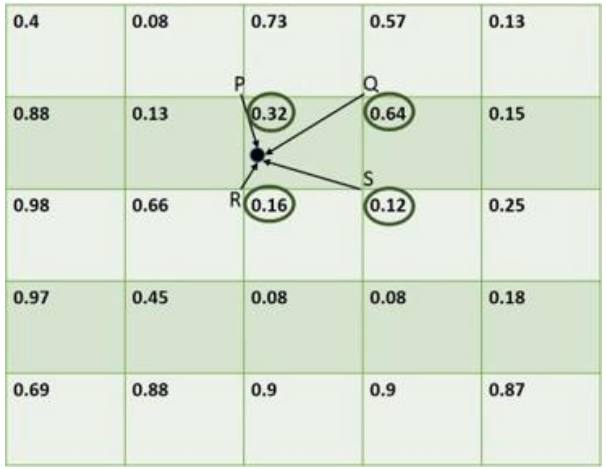
RoIAlign

- Cálculo "vetorizado" da interpolação no ponto S1



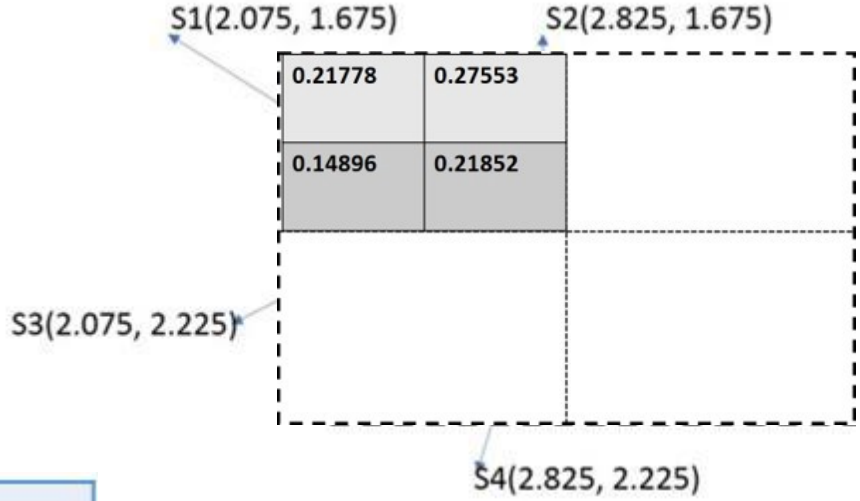
$$f(x,y) = \frac{1}{(x2-x1)(y2-y1)} \begin{bmatrix} x2-x & x-x1 \end{bmatrix} \begin{bmatrix} f(x1,y1) & f(x1,y2) \\ f(x2,y1) & f(x2,y2) \end{bmatrix} \begin{bmatrix} y2-y \\ y-y1 \end{bmatrix}$$

$$f(S1) = 0.21778 = \frac{1}{(1)(1)} \begin{bmatrix} 9.25 & 0.75 \end{bmatrix} \begin{bmatrix} 0.32 & 0.16 \\ 0.64 & 0.12 \end{bmatrix} \begin{bmatrix} 3.25 \\ 6.75 \end{bmatrix}$$



RoIAlign

- Dados os valores de cada bin, usamos average pooling para retornar cada entrada da matriz m x m

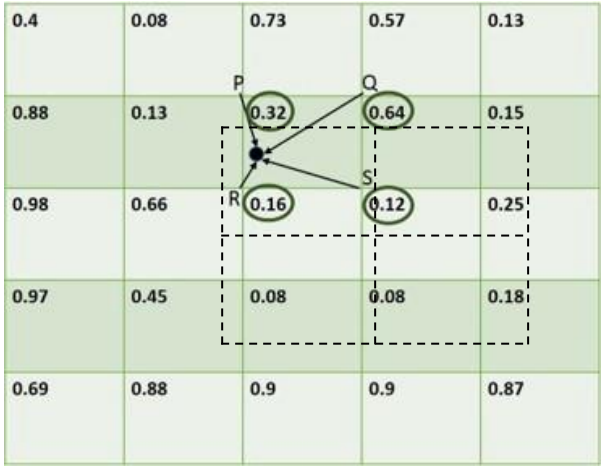


Resultado RoIAlign

0.2152	0.2335
0.3763	0.3562

Resultado RoIPool

0.32	0.64
0.16	0.25



Mask R-CNN também pode ser usado para estimar pose