Curso Inteligência Artificial: do Zero ao Infinito

Detecção de Objetos: Modelos R-CNN

Universidade Federal de Mato Grosso

Agenda

Detecção de Objetos

Introdução

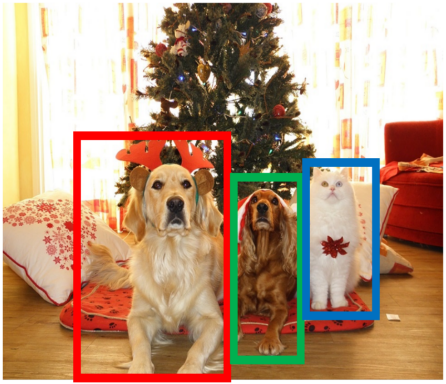
- **Image Classification** é uma tarefa de atribuir uma classe para uma imagem.
- **Object Localization** consiste em localizar um ou mais objetos em uma imagem.
- **Objection Detection** combina essas duas tarefas: localizar um ou mais objetos e classificar cada um deles.

Fonte: https://machinelearningmastery.com/object-recognition-with-deep-learning/

Detecção de Objetos

Definição da Tarefa

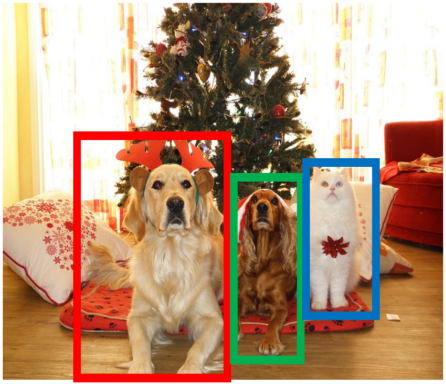
- Entrada: Imagem RGB
- Saída: Um conjunto de objetos detectados; para cada objeto, deve-se ter:
  1. *Label* (categoria) dentro de um conjunto fixo de categorias.
  2. *Bounding box* composto por quatro números: x, u, width, height.



Detecção de Objetos

Desafios

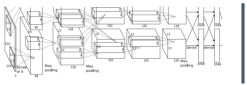
- **Múltiplas Saídas**: número variável de objetos detectados por imagem.
- **Múltiplos tipos de saída**: prediz "qual" (label da categoria) e "onde" (*bounding box*).
- **Tamanho** variável de objetos.
- **Dimensão**: classificação funciona em imagens de dimensão 224×224; enquanto que detecção deve funcionar em imagens de alta resolução ($\geq 800 \times 600$).



Agenda

Detecting a single object

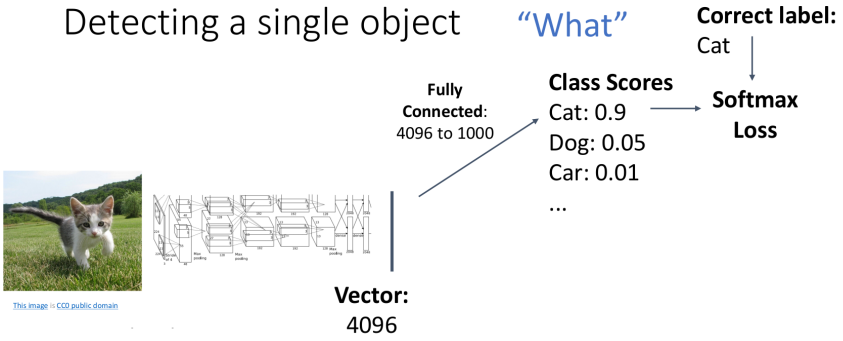
This image is CC0 public domain



Vector:
4096

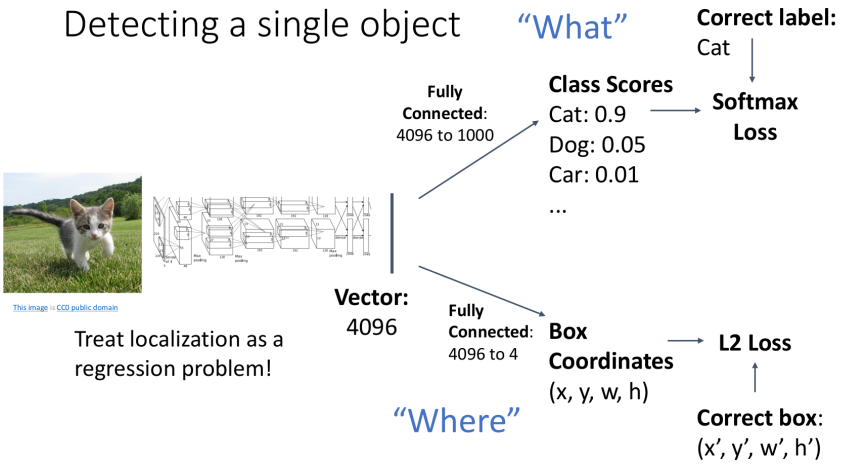
Detecção de Objetos

Detectando um objeto



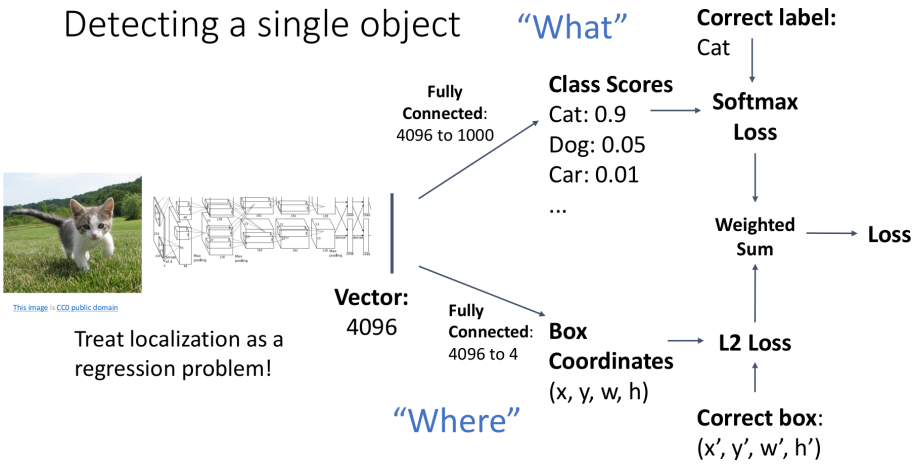
Detecção de Objetos

Detectando um objeto



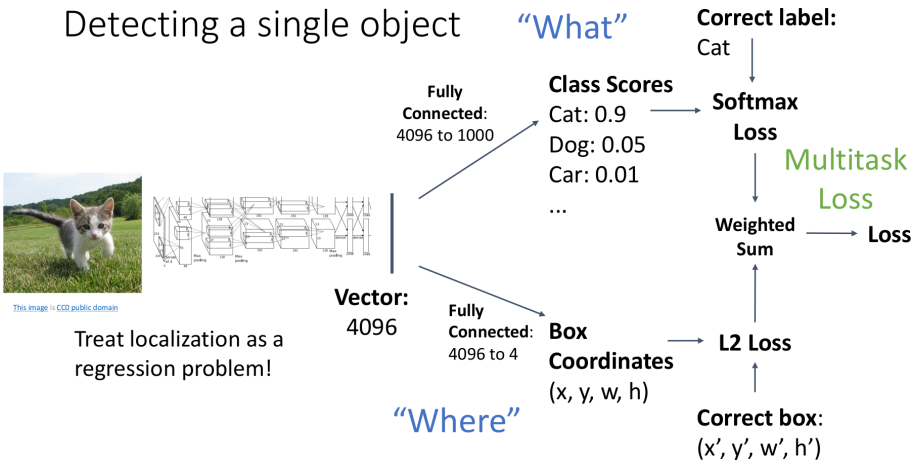
Detecção de Objetos

Detectando um objeto



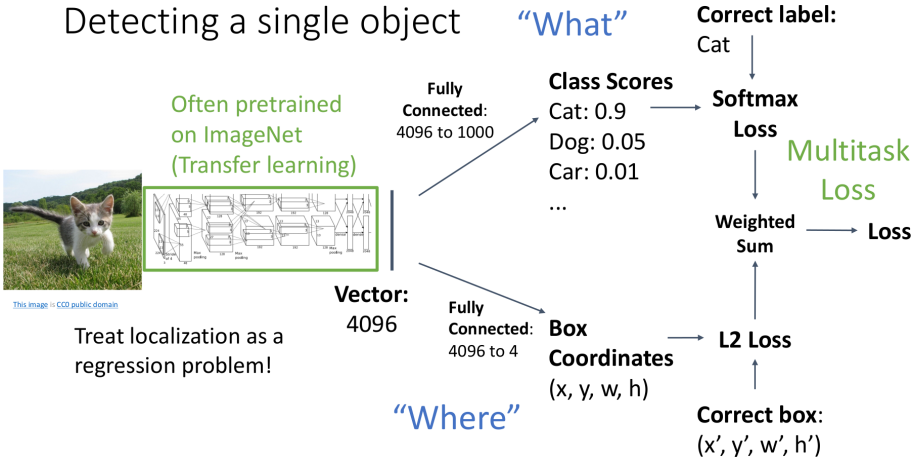
Detecção de Objetos

Detectando um objeto



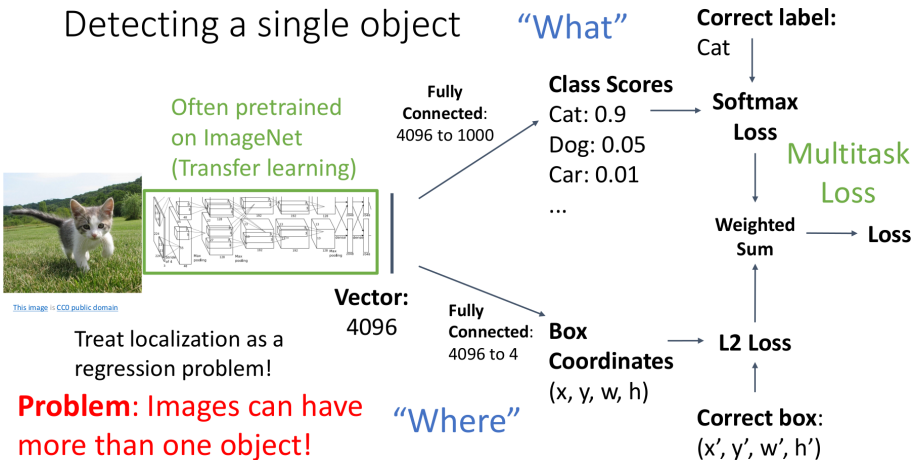
Detecção de Objetos

Detectando um objeto



Detecção de Objetos

Detectando um objeto



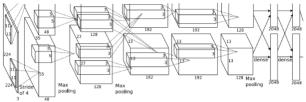
Agenda

Detecção de Objetos

Detectando múltiplos objetos

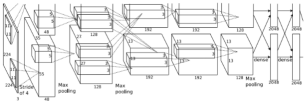
Detecting Multiple Objects

Need different numbers of outputs per image



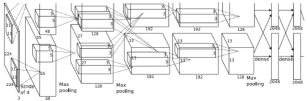
CAT: (x, y, w, h)

4 numbers



DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h)

12 numbers

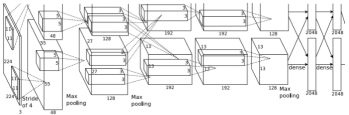


DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
....

Many numbers!

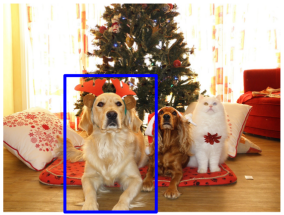
Detecting Multiple Objects: **Sliding Window**

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

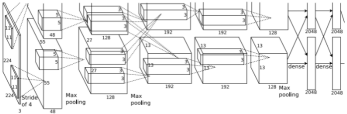


Dog? NO
Cat? NO
Background? YES

Detecting Multiple Objects: **Sliding Window**



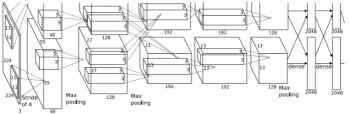
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? YES
Cat? NO
Background? NO

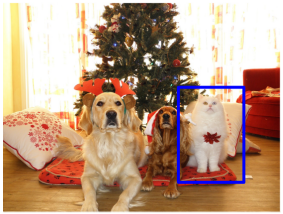
Detecting Multiple Objects: **Sliding Window**

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

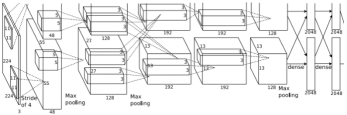


Dog? YES
Cat? NO
Background? NO

Detecting Multiple Objects: **Sliding Window**

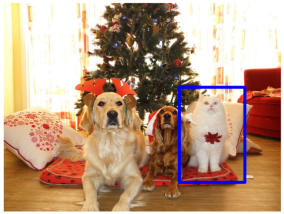


Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? NO
Cat? YES
Background? NO

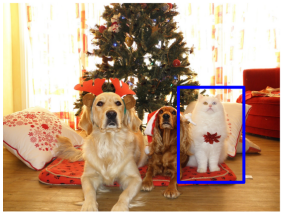
Detecting Multiple Objects: **Sliding Window**



Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

Question: How many possible boxes are there in an image of size H x W?

Detecting Multiple Objects: **Sliding Window**

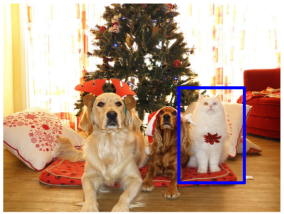


Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

Question: How many possible boxes are there in an image of size H x W?

Consider a box of size h x w:
Possible x positions: W – w + 1
Possible y positions: H – h + 1
Possible positions:
(W – w + 1) * (H – h + 1)

Detecting Multiple Objects: **Sliding Window**



Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

Question: How many possible boxes are there in an image of size H x W?

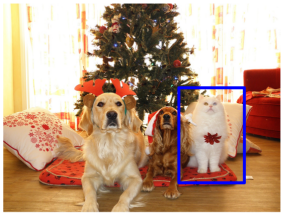
Consider a box of size h x w:
Possible x positions: W – w + 1
Possible y positions: H – h + 1
Possible positions:
(W – w + 1) * (H – h + 1)

Total possible boxes:

$$\sum_{h=1}^{H}\sum_{w=1}^{W}(W-w+1)(H-h+1)$$

$$=\frac{H(H+1)}{2}\frac{W(W+1)}{2}$$

Detecting Multiple Objects: **Sliding Window**



Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

800 x 600 image has ~58M boxes! No way we can evaluate them all

Question: How many possible boxes are there in an image of size H x W?

Consider a box of size h x w:
Possible x positions: W – w + 1
Possible y positions: H – h + 1
Possible positions:
(W – w + 1) * (H – h + 1)

Total possible boxes:

$$\sum_{h=1}^{H}\sum_{w=1}^{W}(W-w+1)(H-h+1)$$

$$=\frac{H(H+1)}{2}\frac{W(W+1)}{2}$$

Agenda

R-CNN

- Em 2014, no modelo **R-CNN** foi utilizada uma estratégia para definição das regiões de busca e classificação dessas regiões.
- Essa estratégia consiste em utilizar heurísticas para encontrar *Region Proposals*, que são regiões mais prováveis de conterem um objeto.
- Para seleção das *Region Proposals*, não é definido um método, mas são apresentados diversas propostas.

Paper: Rich feature hierarchies for accurate object detection and semantic segmentation
Sourcecode: Official Github

R-CNN

- O método mais utilizado para seleção das *Region Proposals* é denominado *Selective Search*.
- Em cada uma dessas regiões aplica-se uma CNN, a AlexNet, para extração das *features*, treinada no ImageNet.
- Por fim, utiliza-se o SVM como classificador e um regressor neural para prever as *bounding boxes*.

Paper: Rich feature hierarchies for accurate object detection and semantic segmentation
Sourcecode: Official Github

R-CNN

Arquitetura

Fonte: R-CNN (Object Detection)

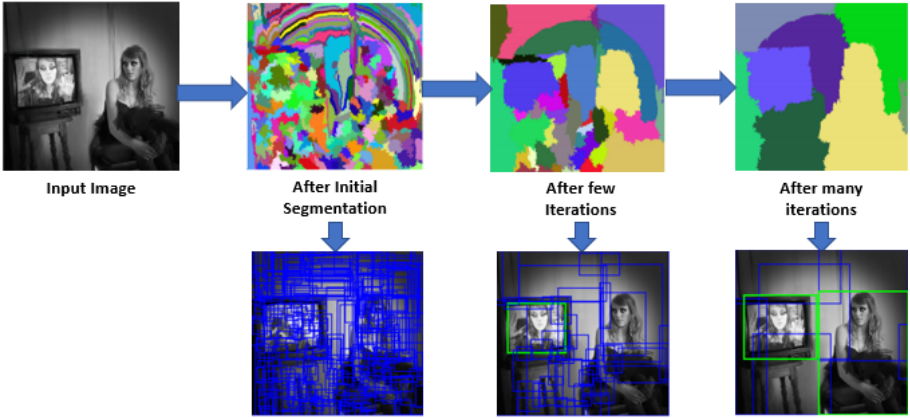
Algoritmo *Selective Search*:

1. Gere sub-segmentos da imagem de entrada.
2. Recursivamente combine as regiões similares em regiões maiores.
   - Dados os conjuntos de regiões, escolha os dois mais similares.
   - Combine-os em um único conjunto, ou seja, em uma região maior.
   - Repita os passos anteriores por múltiplas iterações.

Fonte: Selective Search for Object Detection | R-CNN

R-CNN

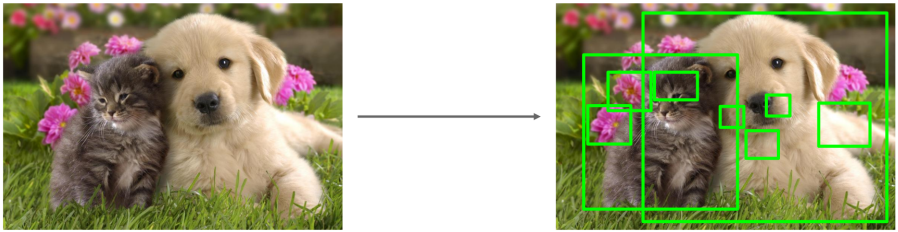
Selective Search



Fonte: Selective Search for Object Detection | R-CNN

Region Proposals

- Defina um pequeno conjunto de *bboxes* que provavelmente cubra todos os objetos.
- Utilize heurísticas para procurar por regiões mais prováveis de conter uma imagem.
- Execução relativamente rápida: *Selective Search* oferece 2k *Region Proposals* em poucos segundos na CPU.



R-CNN: Region-Based CNN

Input image

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

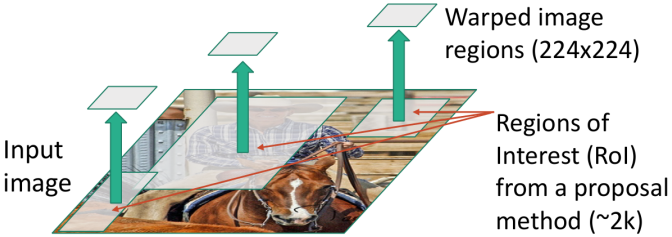
R-CNN: Region-Based CNN

Input image

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.

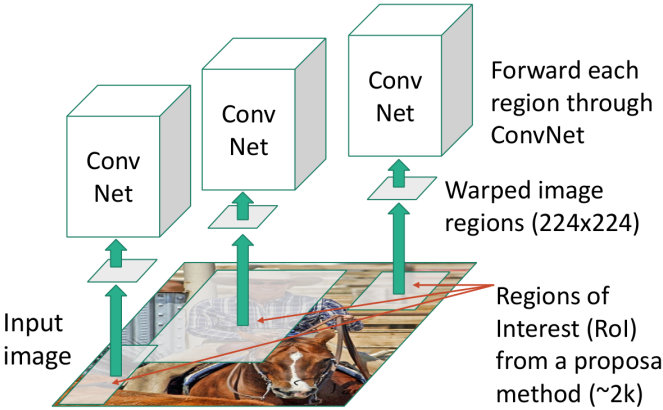
R-CNN: Region-Based CNN



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

Region Proposals

R-CNN: Region-Based CNN



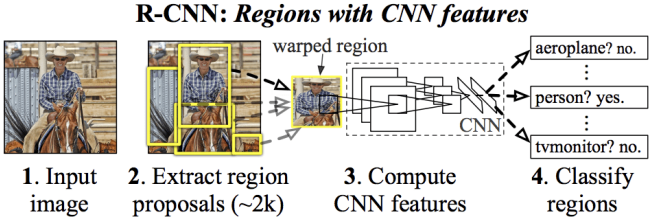
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

R-CNN

Arquitetura

A arquitetura do R-CNN é composta por três módulos:

1. **Region Proposal**. Gera e extrai as regiões candidatas a *bboxes*.
2. **Feature Extractor**. Extrai *features* de cada região usando uma rede CNN.
3. **Classificador**. Classifica cada região um modelo linear de classificação (SVM).



Paper: Rich feature hierarchies for accurate object detection and semantic segmentation

R-CNN

Desvantagens

- Alto custo de treinamento.
  - Precisa classificar 2000 regiões por imagem.
- Não pode ser implementado em tempo real.
  - em média, leva 47s para cada imagem de teste.
- O algoritmo *Selective Search* é estático.
  - Não há aprendizado na geração das *Region Proposals*.
- O treinamento não pode ser paralelizado.
  - Cada módulo deve ser treinado separadamente.

Fonte: R-CNN, Fast R-CNN, Faster R-CNN, YOLO - Object Detection Algorithms

Agenda

Fast R-CNN

- Em 2015, foi proposta a **Fast R-CNN**, com o intuito de corrigir alguns pontos fracos da *R-CNN*.
- Foram combinados três módulos (CNN, SVM e *bbox regressor*) em um, tornando-o um modelo *end-to-end*.
- Após esse modelo, a R-CNN original ficou conhecida como *Slow R-CNN*.

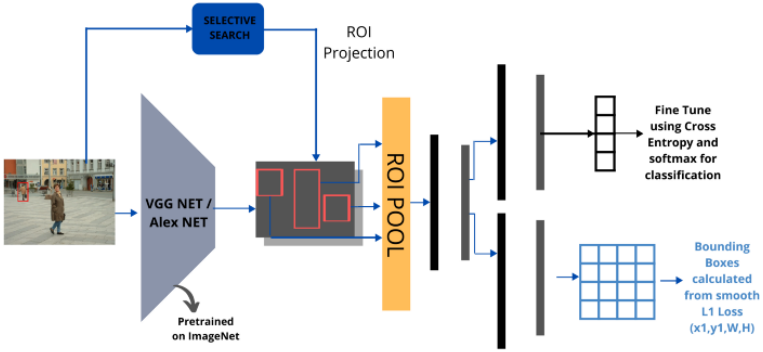
Paper: Fast R-CNN
Source code: Official Github

Fast R-CNN

- Nesse modelo, utilizou-se como extrator de *features* o modelo VGG16 pré-treinado na ImageNet.
- Ao invés de se utilizar SVM, optou-se por duas redes FC para predição dos *bboxings* e classificação.
- Isso permitiu um menor tempo de treinamento do modelo e também uma melhor acurácia.

Paper: Fast R-CNN
Source code: Official Github

Fast R-CNN



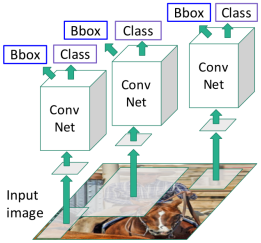
Fonte: Understanding Fast R-CNN and Faster R-CNN for Object Detection.

Fast R-CNN

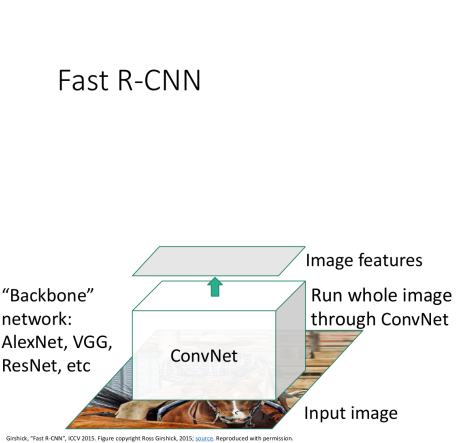
Fast R-CNN

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

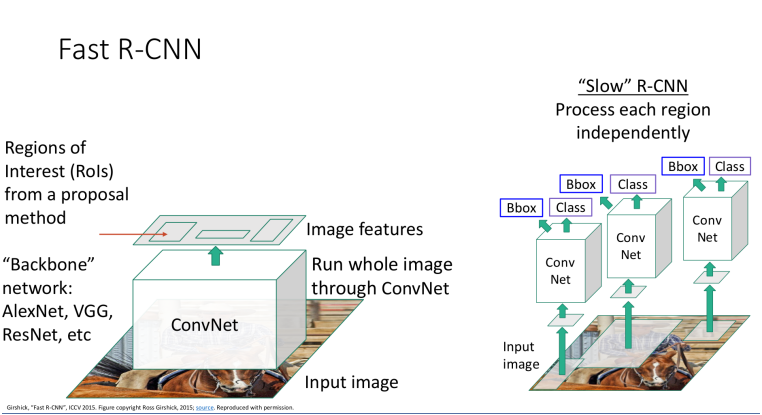
"Slow" R-CNN
Process each region independently



Fast R-CNN


Fast R-CNN
“Backbone” network: AlexNet, VGG, ResNet, etc
ConvNet
Image features
Run whole image through ConvNet
Input image
Girshick, “Fast R-CNN”, ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.
“Slow” R-CNN
Process each region independently
Bbox
Class
Conv Net
Input image

Fast R-CNN


Fast R-CNN
Regions of Interest (RoIs) from a proposal method
Image features
"Backbone" network: AlexNet, VGG, ResNet, etc
Run whole image through ConvNet
ConvNet
Input image
Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.
"Slow" R-CNN
Process each region independently
Bbox
Class
Conv Net
Input image

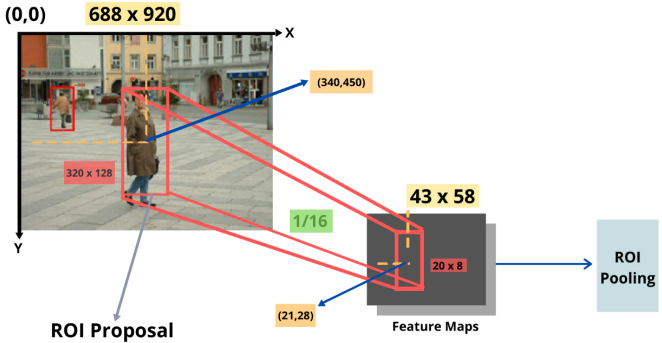
Fast R-CNN

- Na *R-CNN* são geradas 2k *Region Proposals* e cada uma é aplicada à uma CNN extratora de *features*.
  - Em um dataset com 1k imagens, seriam necessárias 2M (1000×2000) iterações.
- Na **Fast R-CNN** a imagem é enviada para a rede CNN uma única vez, criando um *features map*.
  - A *Selective Search* é realizada normalmente, definindo 2K *Region Proposals*.
- As *Region Proposals* são projetadas no *features map* gerado pela CNN, num processo denominado **Region of Interest** (ROI).

Fonte: Understanding Fast R-CNN and Faster R-CNN for Object Detection.

Fast R-CNN

As *Region Proposals* são adaptadas num processo de *subsampling ratio* e projetadas no *features map*.

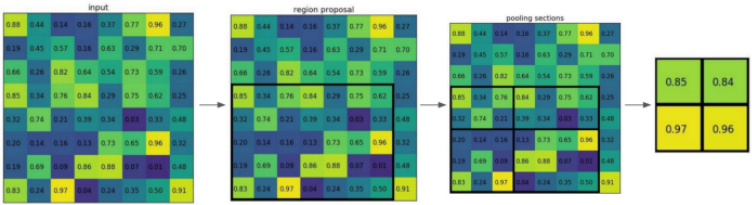


Fonte: Understanding Fast R-CNN and Faster R-CNN for Object Detection.

Fast R-CNN

Region of Interest Pooling

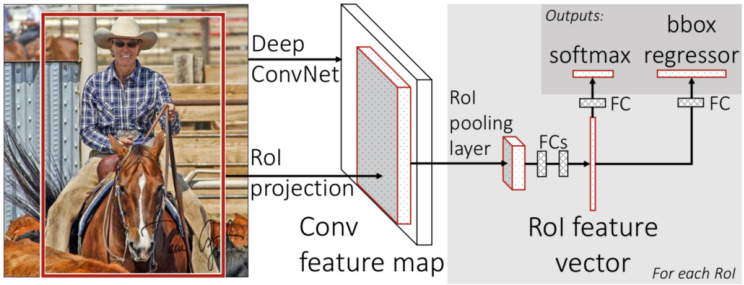
Em seguida, é aplicada uma cada de *pooling layer*, a fim de obter um tamanho fixo, independente da entrada.



A saída de tamanho fixo é importante pois será enviada à uma camada FC.

Fonte: Introduction: Fast R-CNN (Object Detection) e Region of interest pooling explained

Fast R-CNN



Fonte: Fast R-CNN

R-CNN vs Fast R-CNN

	RCNN	Fast RCNN
Training Time	84 hours	9.5 hours
(Speedup)	1x	8.8x
Test time per image	47 seconds	0.32 seconds
(Speedup)	1x	146x
Test time per image with Selective Search	50 seconds	2 seconds
(Speedup)	1x	25x

Fonte: Understanding Fast R-CNN and Faster R-CNN for Object Detection.

Fast R-CNN

- Nos modelos *R-CNN* e *Fast R-CNN* o gargalo do modelo está na definição das *Region Proposals*.
- O algoritmo *Selective Search* demora 2s por imagem e executa em CPU.
- Isso torna esses modelos inviáveis para aplicações em tempo real.

Fonte: R-CNN, Fast R-CNN, Faster R-CNN, YOLO - Object Detection Algorithms

Agenda

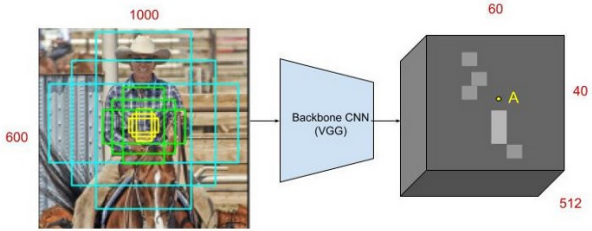
Faster R-CNN

- **Faster R-CNN** eliminou de vez o uso do algoritmo *Selective Search* substituindo-o por uma CNN.
  - Essa rede CNN é chamada *Region Proposal Network* (RPN).
- O RPN terá as funções de classificador de regressor.
  - Dessa forma, o tempo para definição das *Region Proposals* cai de 2s para 10ms por imagem.

Paper: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

Faster R-CNN

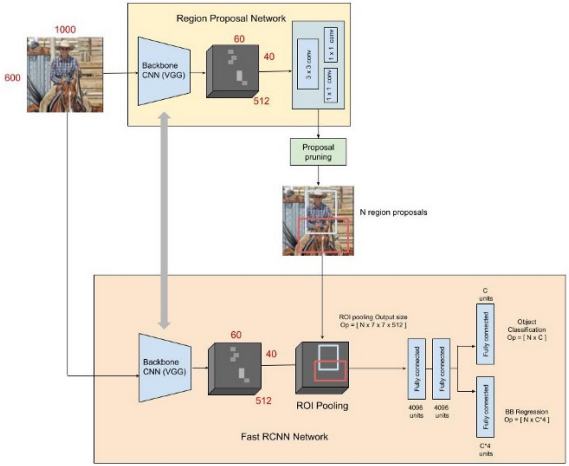
Region Proposal Network



Fonte: Faster R-CNN for object detection

Faster R-CNN

Region Proposal Network



Faster R-CNN

	RCNN	Fast RCNN	Faster RCNN
Test time per image with Proposals	50 seconds	2 seconds	0.2 seconds
(Speedup)	1x	25x	250x
mAP (PASCAL VOC 07)	66.0	66.9	66.9

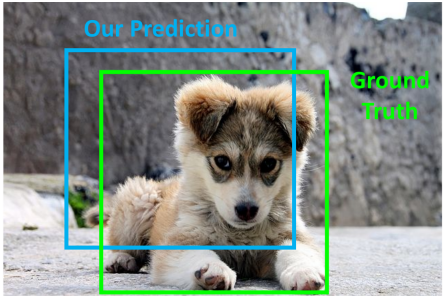
Fonte: Understanding Fast R-CNN and Faster R-CNN for Object Detection.

Agenda

Comparing Boxes

Intersection over Union (IoU)

- Como comparar o *box* da predição com o *box* do *ground-truth*?



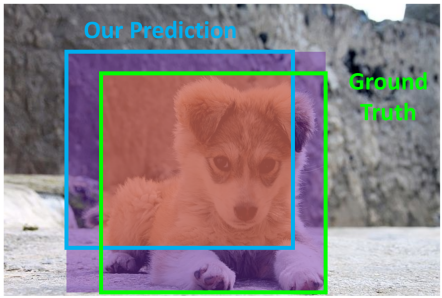
Puppy image is licensed under CC-A 2.0 Generic license. Bounding boxes and text added by Justin Johnson.

Comparing Boxes

Intersection over Union (IoU)

- Como comparar o *box* da predição com o *box* do *ground-truth*?
- **Intersection over Union** (IoU): também chamado de "*Similaridade de Jaccard*" ou "*Jaccard index*".

$$\frac{\textit{Area of Intersection}}{\textit{Area of Union}}$$



Puppy image is licensed under CC-A 2.0 Generic license. Bounding boxes and text added by Justin Johnson.

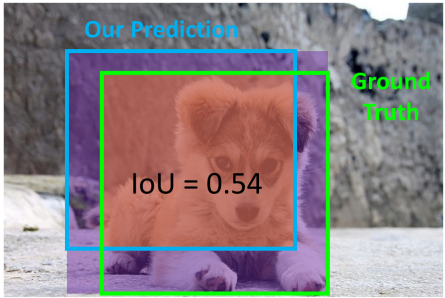
Comparing Boxes

Intersection over Union (IoU)

- Como comparar o *box* da predição com o *box* do *ground-truth*?
- **Intersection over Union** (IoU): também chamado de "*Similaridade de Jaccard*" ou "*Jaccard index*".

$$\frac{\textit{Area of Intersection}}{\textit{Area of Union}}$$

- IoU $>$ 0.5 é "*aceitável*",



Puppy image is licensed under CC-A 2.0 Generic license. Bounding boxes and text added by Justin Johnson.

Comparing Boxes

Intersection over Union (IoU)

- Como comparar o *box* da predição com o *box* do *ground-truth*?
- **Intersection over Union** (IoU): também chamado de "*Similaridade de Jaccard*" ou "*Jaccard index*".

$$\frac{\textit{Area of Intersection}}{\textit{Area of Union}}$$

- IoU $>$ 0.5 é "*aceitável*",
- IoU $>$ 0.7 é "*muito bom*",

Puppy image is licensed under CC-A 2.0 Generic license. Bounding boxes and text added by Justin Johnson.

Comparing Boxes

Intersection over Union (IoU)

- Como comparar o *box* da predição com o *box* do *ground-truth*?
- **Intersection over Union** (IoU): também chamado de "*Similaridade de Jaccard*" ou "*Jaccard index*".

$$\frac{\textit{Area of Intersection}}{\textit{Area of Union}}$$

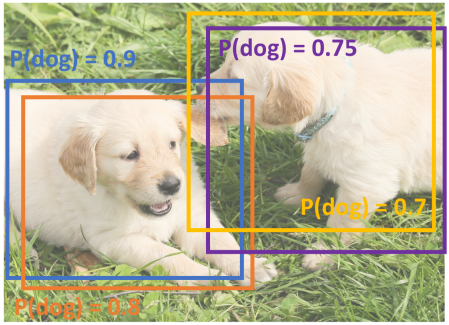
- IoU $>$ 0.5 é "*aceitável*",
- IoU $>$ 0.7 é "*muito bom*",
- IoU $>$ 0.9 é "*quase perfeito*".

Puppy image is licensed under CC-A 2.0 Generic license. Bounding boxes and text added by Justin Johnson.

Comparing Boxes

Overlapping Boxes

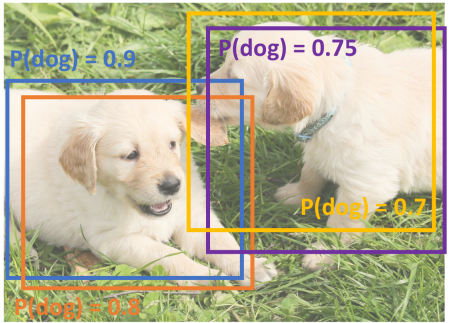
- **Problema:** *Object Detectors* frequentemente detectam *boxes* sobrepostos:



Comparing Boxes

Overlapping Boxes: Non-Max Suppression (NMS)

- **Problema:** *Object Detectors* frequentemente detectam *boxes* sobrepostos:
- **Solução:** pós-processar detecções utilizando **Non-Max Suppression (NMS)**
  1. Selecione o *box* com maior *score*.
  2. Elimine *boxes* com baixo *score*.
     - ⋆ IoU $>$ threshold (ex. 0.7).
  3. Se restarem *boxes*, GOTO 1.



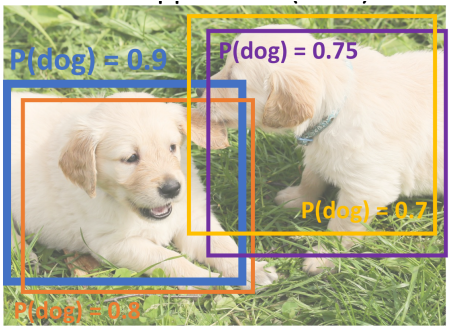
Puppy image is CC0 Public Domain

Comparing Boxes

Overlapping Boxes: Non-Max Suppression (NMS)

- **Problema:** *Object Detectors* frequentemente detectam *boxes* sobrepostos:
- **Solução:** pós-processar detecções utilizando **Non-Max Suppression (NMS)**
  1. Selecione o *box* com maior *score*.
  2. Elimine *boxes* com baixo *score*.
     - ★ IoU $>$ threshold (ex. 0.7).
  3. Se restarem *boxes*, GOTO 1.

IoU(■, ■) = **0.78**
IoU(■, ■) = 0.05
IoU(■, ■) = 0.07

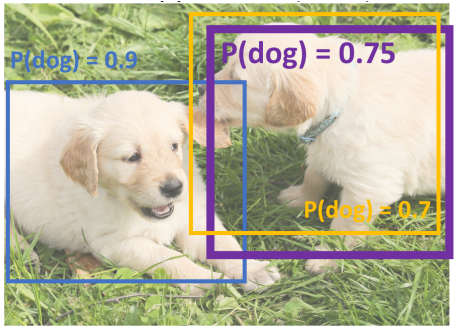


Comparing Boxes

Overlapping Boxes: Non-Max Suppression (NMS)

- **Problema:** *Object Detectors* frequentemente detectam *boxes* sobrepostos:
- **Solução:** pós-processar detecções utilizando **Non-Max Suppression (NMS)**
  1. Selecione o *box* com maior *score*.
  2. Elimine *boxes* com baixo *score*.
     - ⋆ IoU $>$ threshold (ex. 0.7).
  3. Se restarem *boxes*, GOTO 1.

IoU(,) = **0.74**

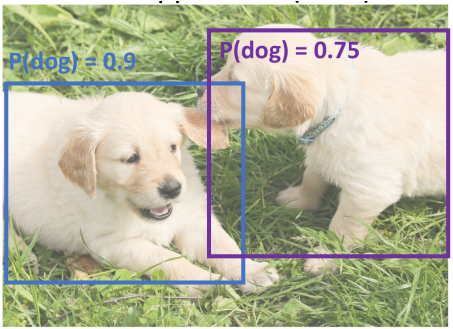


Puppy image is CC0 Public Domain

Comparing Boxes

Overlapping Boxes: Non-Max Suppression (NMS)

- **Problema:** *Object Detectors* frequentemente detectam *boxes* sobrepostos:
- **Solução:** pós-processar detecções utilizando **Non-Max Suppression (NMS)**
  1. Selecione o *box* com maior *score*.
  2. Elimine *boxes* com baixo *score*.
     - ★ IoU $>$ threshold (ex. 0.7).
  3. Se restarem *boxes*, GOTO 1.



Puppy image is CC0 Public Domain

Comparing Boxes

Overlapping Boxes: Non-Max Suppression (NMS)

- **Problema:** *Object Detectors* frequentemente detectam *boxes* sobrepostos:
- **Solução:** pós-processar detecções utilizando **Non-Max Suppression (NMS)**
  1. Selecione o *box* com maior *score*.
  2. Elimine *boxes* com baixo *score*.
     - ⋆ IoU $>$ threshold (ex. 0.7).
  3. Se restarem *boxes*, GOTO 1.
- Problema: NMS pode eliminar *boxes* "bons" quando os objetos estão sob muita sobreposição... nesse caso não existe solução boa.

Agenda

Tensorflow Object Detection API

Tutorial

- Para utilizar os modelos descritos é possível utilizar a API de **Object Detection** do Tensorflow.
- É um *framework Open Source* baseado em Tensorflow.
- Fornece suporte para criação de modelos capazes de localizar e identificar objetos em uma imagem.

Tensorflow Object Detection API

Tutorial

Vamos aprender:

- Preparação do ambiente utilizando Google Colab.
- Ferramentas para criação de datasets.
- Organização dos dados de treinamento.
- Configurar o treinamento.
- Treinar e salvar o modelo.

Existem outras ferramentas e é possível encontrar algumas outras através do link:

- https://en.wikipedia.org/wiki/List_of_manual_image_annotation_tools

Encontre a que mais goste...

Agenda

Detecção de Objetos

Arquiteturas

- **R-CNN** (Regions with Convolutional Network)
- **Fast-RCNN**
- **Faster-RCNN**
- **SSD** (Single Shot Detection)
- **YOLO** (You Only Look Once)
- **R-FCNN** (Region-Based Convolutional Network)
- **SqueezeDet**
- Dentre outras...

Fonte: https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/

Detecção de Objetos

Datasets

Os principais datasets utilizados para Detecção de objetos são:

- **Open Images v4**: 9M imagens com 600 classes
  - Download: https://storage.googleapis.com/openimages/web/index.html
- **ImageNet**: 450k imagens com 200 classes
  - https://image-net.org/challenges/LSVRC/
- **COCO**: 120k imagens com 80 classes
  - Download: https://cocodataset.org/
- **Pascal VOC**: 12k imagens com 20 classes
  - http://host.robots.ox.ac.uk/pascal/VOC/.
- **Oxford-IIIT Pet**: 7k imagens com 37 classes
  - https://www.robots.ox.ac.uk/ vgg/data/pets/
- **KITTI Vision**: 7k imagens com 3 classes
  - http://www.cvlibs.net/datasets/kitti/

Referências

- Deep Learning for Computer Vision - University of Michigan
  - Lecture 15: Object Detection
- Object Detection for Dummies
  - Object Detection for Dummies Part 1: Gradient Vector, HOG, and SS
  - Object Detection for Dummies Part 2: CNN, DPM and Overfeat
  - Object Detection for Dummies Part 3: R-CNN Family
  - Object Detection for Dummies Part 4: Fast Detection Models

Curso Inteligência Artificial: do Zero ao Infinito

Detecção de Objetos: Modelos R-CNN

Universidade Federal de Mato Grosso