Detecção e Segmentação de objetos

Aprendizado de máquina e aprendizado profundo - Lux.AI

INSTITUIÇÃO EXECUTORA

Centro de Informática UFPE

FADE UFPE

COORDENADORA

MCTI FUTURO

Softex

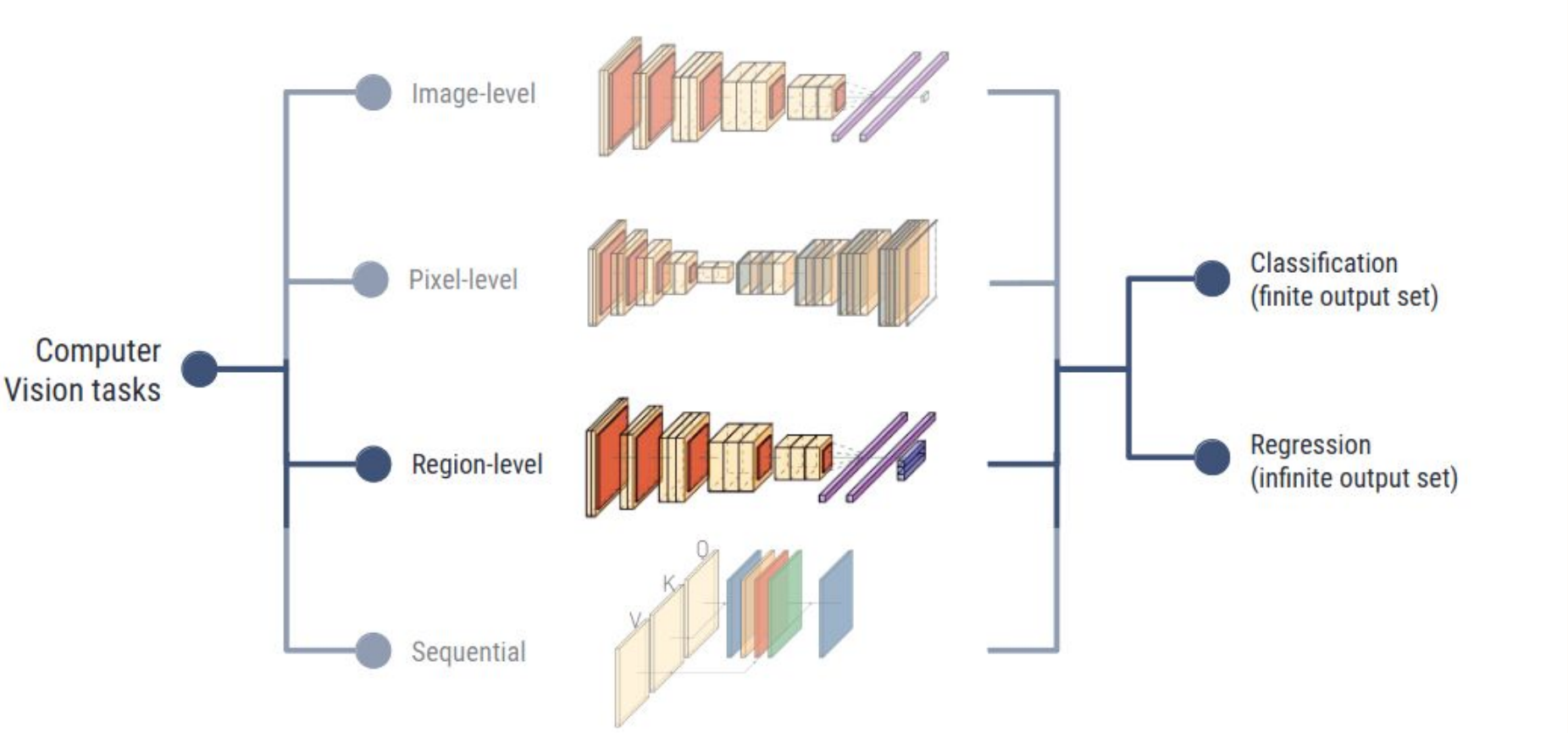
APOIO

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO

GOVERNO FEDERAL BRASIL UNIÃO E RECONSTRUÇÃO

Detecção 2D - Introdução

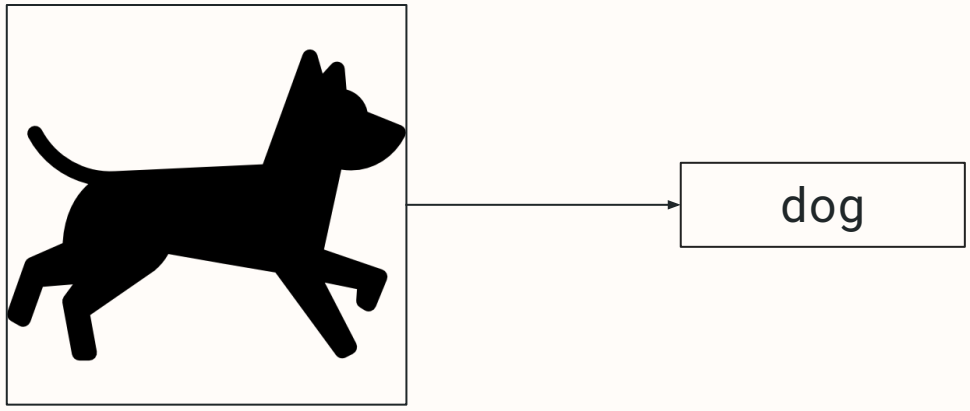
Tarefas em visão computacional



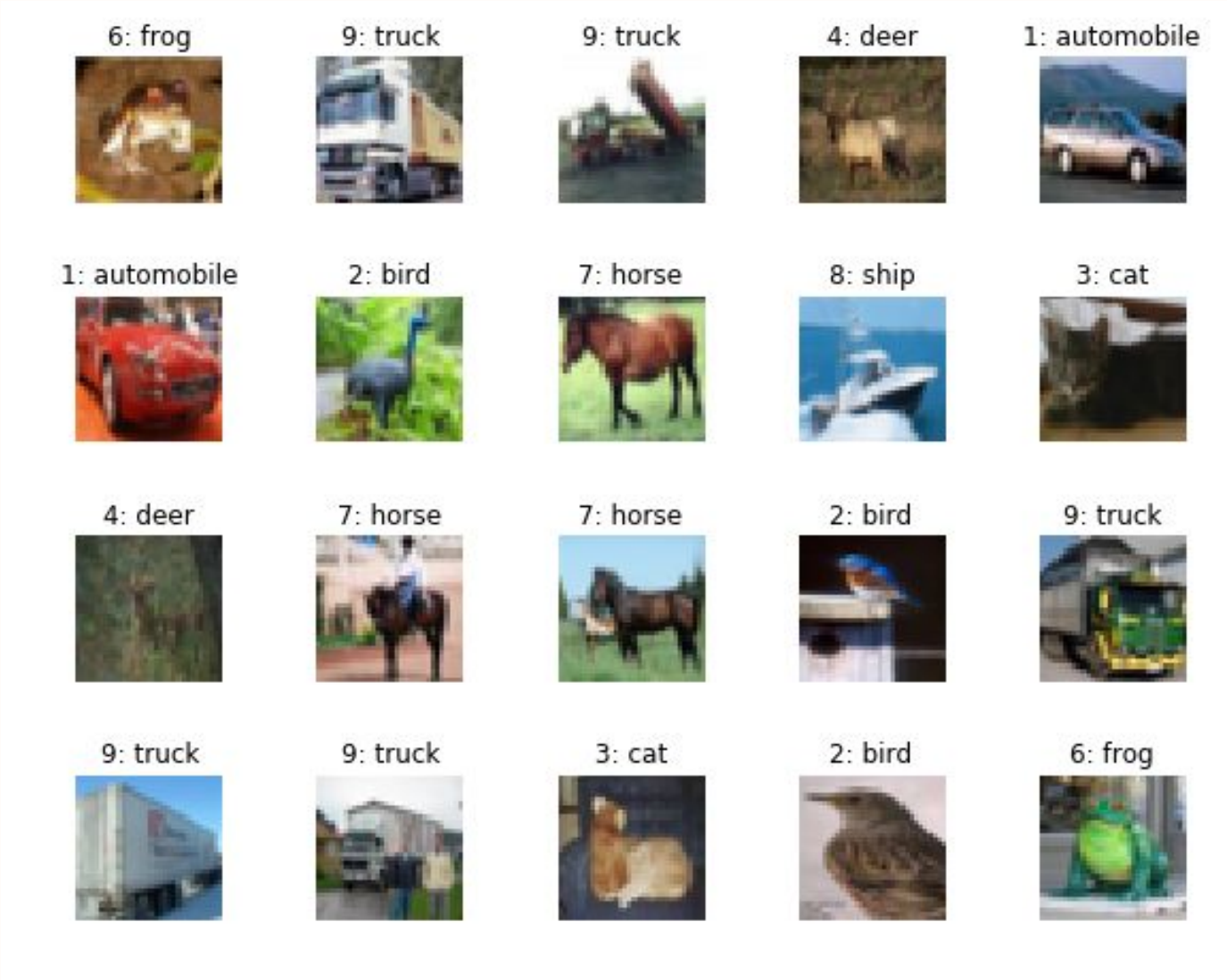
Fonte: Fidel Guerrero Peña, Tsang Ing Ren. Object Detection.

Classificação de objetos

- A **Classificação** busca categorizar se a imagem de entrada contém um objeto de interesse
  - Geralmente temos o objeto de interesse em **evidência** na imagem, e precisamos apenas estimar a qual **classe** o objeto pertence

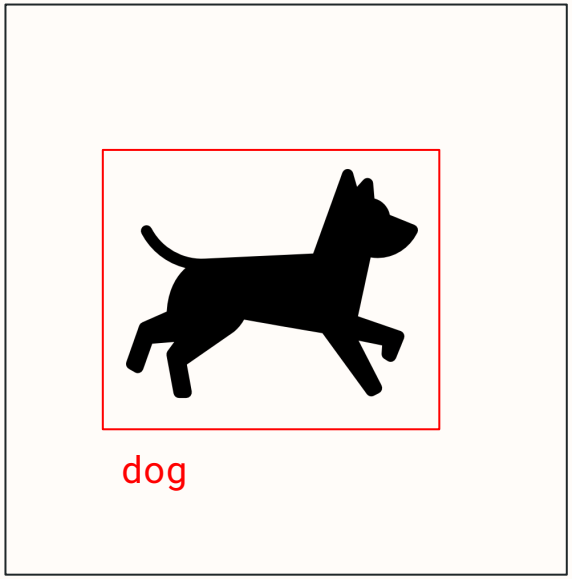


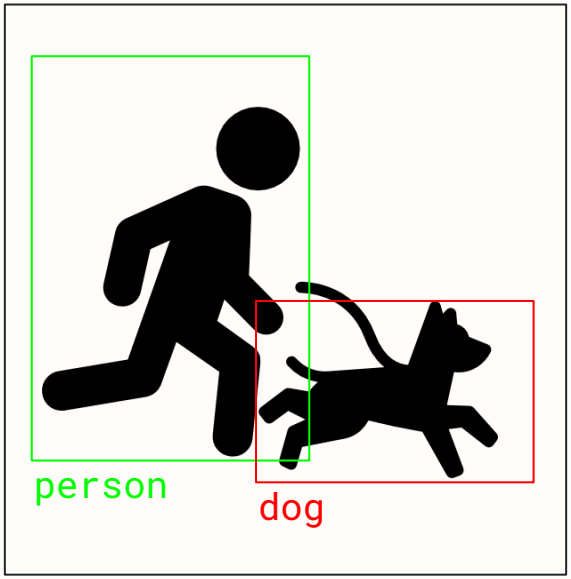
Classificação de objetos



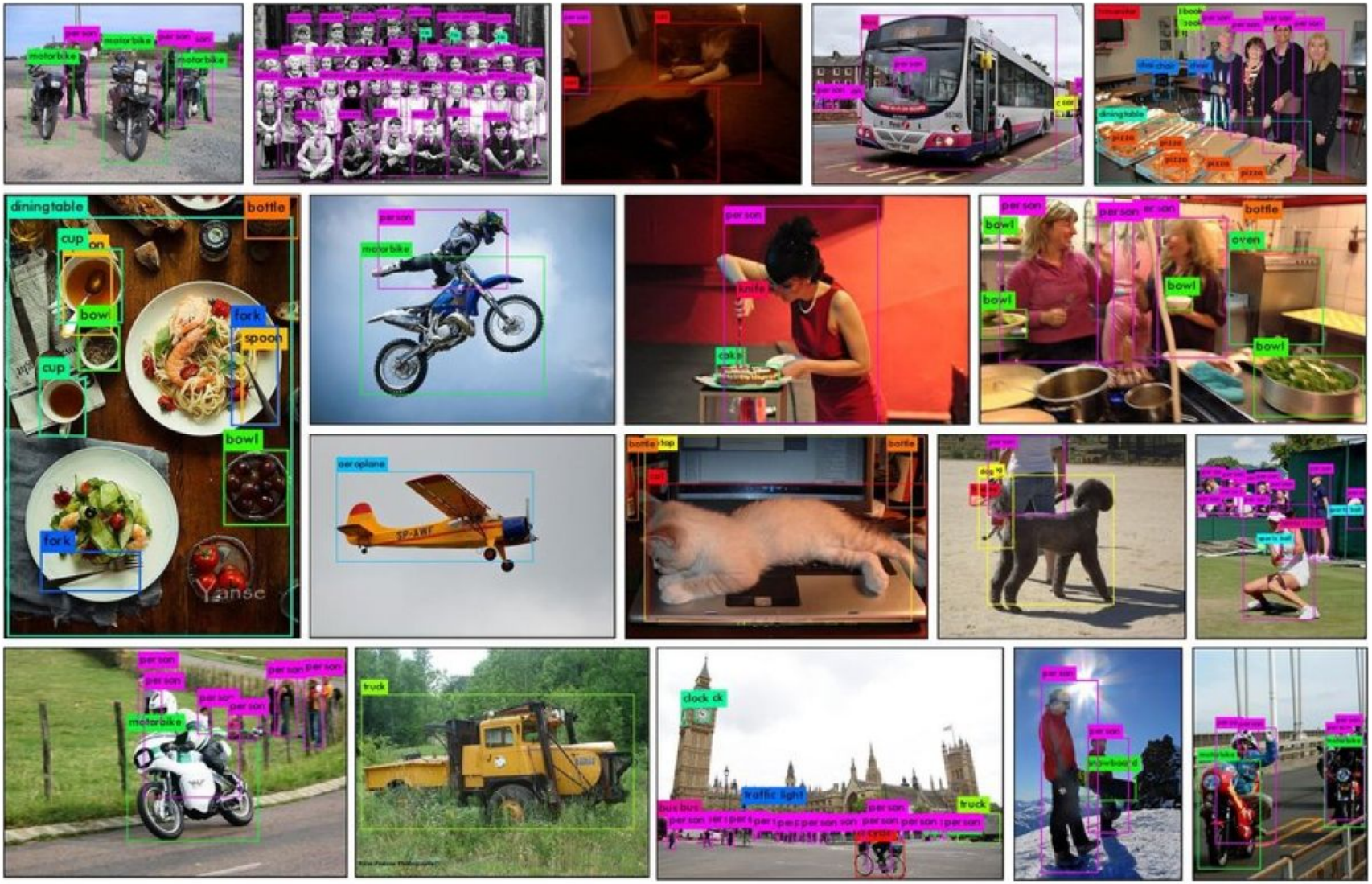
Detecção de objetos

- Algumas tarefas requerem o conhecimento da localização do objeto na imagem
  - Por exemplo, quando existem **múltiplos objetos** na imagem de entrada, inicialmente é necessário a identificação das **posições específicas** de cada objeto
  - Em seguida, podemos **categorizar** ou **classificar** cada objeto a partir da sua localização estimada
  - Chamamos de **detecção** a tarefa de **encontrar as coordenadas** ou **posição** do objeto na imagem



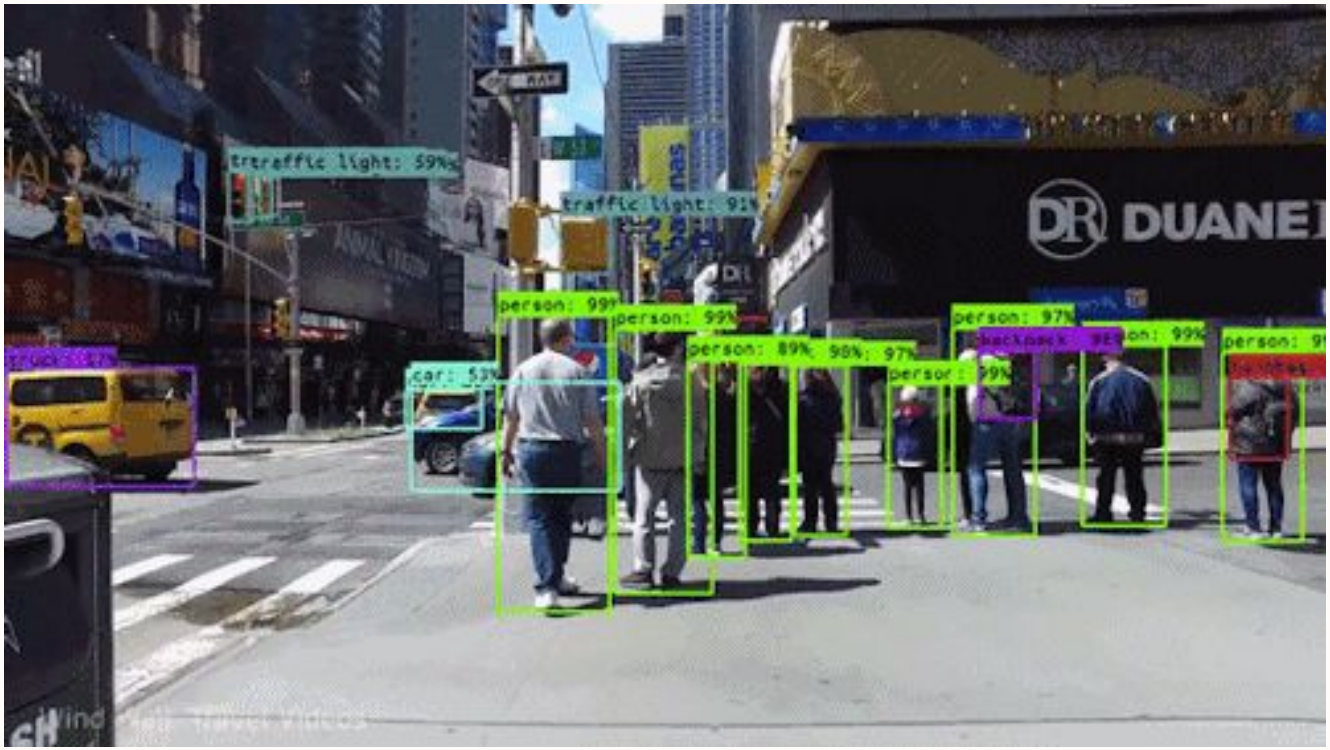


Detecção de objetos



Detecção de objetos

- ➔ Possui aplicações em diversas áreas que utilizam visão computacional
- ➔ Por exemplo: veículos autônomos, sistemas de vigilância, navegação de ambientes, manipulação robótica, entre outros.



Detecção de objetos

➔ Estratégias de detecção geralmente são divididas entre métodos que utilizam abordagens **tradicionais** e abordagens baseadas em **aprendizagem de máquina** e/ou **aprendizagem profunda**

Detecção de objetos

➔ Estratégias de detecção geralmente são divididas entre métodos que utilizam abordagens **tradicionais** e abordagens baseadas em **aprendizagem de máquina** e/ou **aprendizagem profunda**

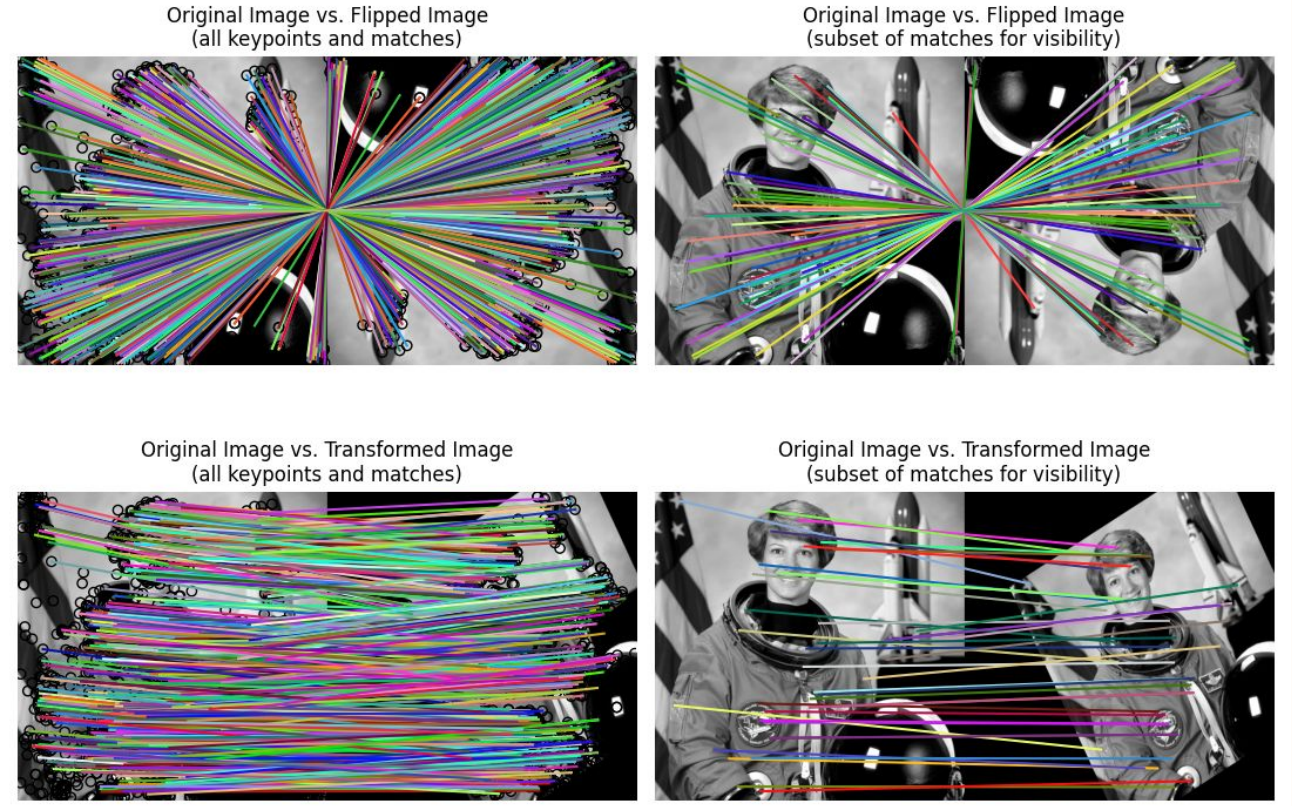
➔ Exemplos de métodos baseados em características:

- SIFT (Scale Invariant Feature Transform)
- HOG (Histogram of Oriented Gradients)
- SURF, ORB, BRISK, ...

Detecção de objetos

1 — 2 — 3 — 4

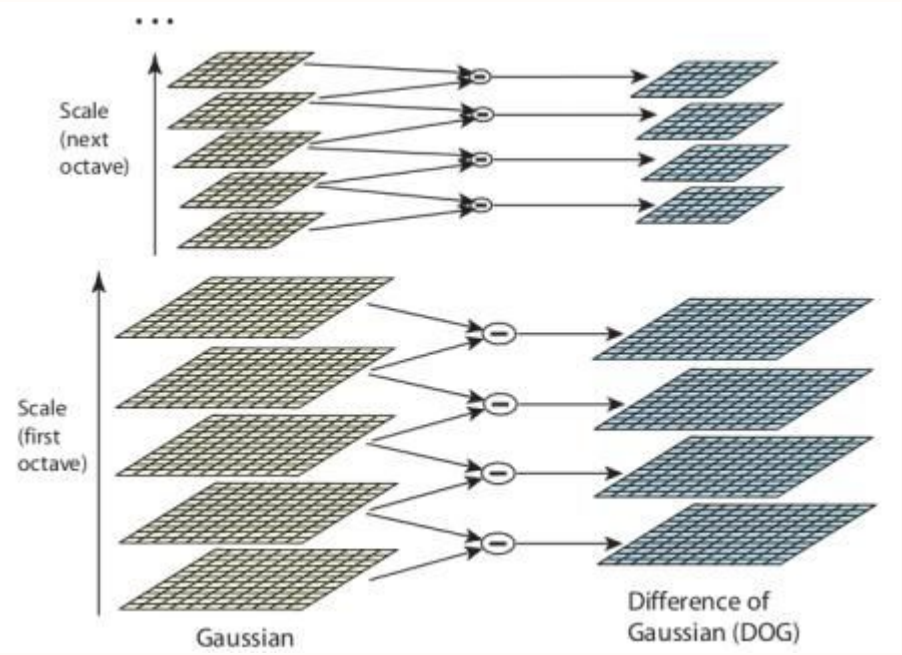
Entrada — **Extração de features** — **Matching** — **Saída**

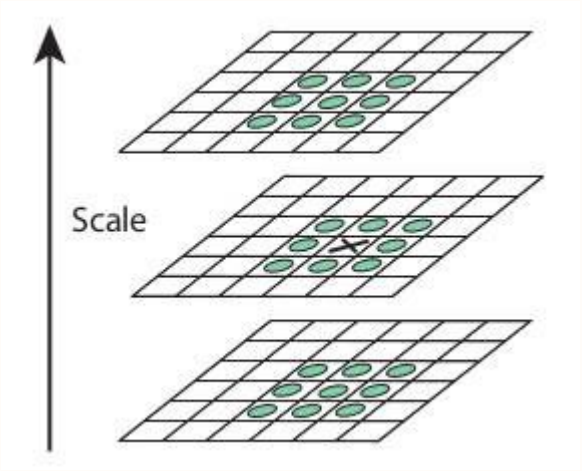


Fonte: https://scikit-image.org/docs/stable/auto_examples/features_detection/plot_sift.html

Detecção de objetos - SIFT

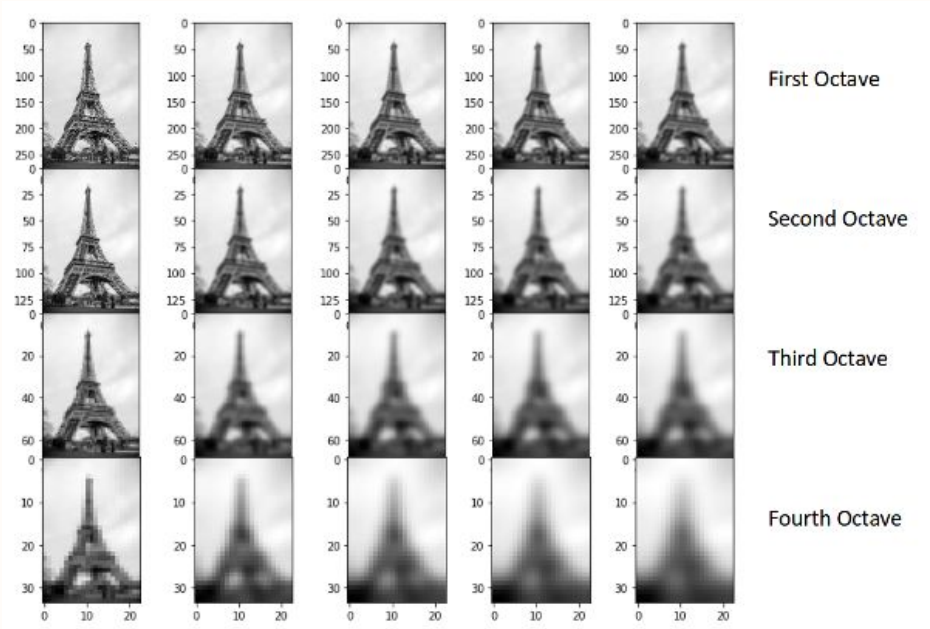
- ➔ Detecção Space-scale
- ➔ Localização dos keypoints
- ➔ Atribuição de orientação
- ➔ Criação dos descritores
- ➔ Matching

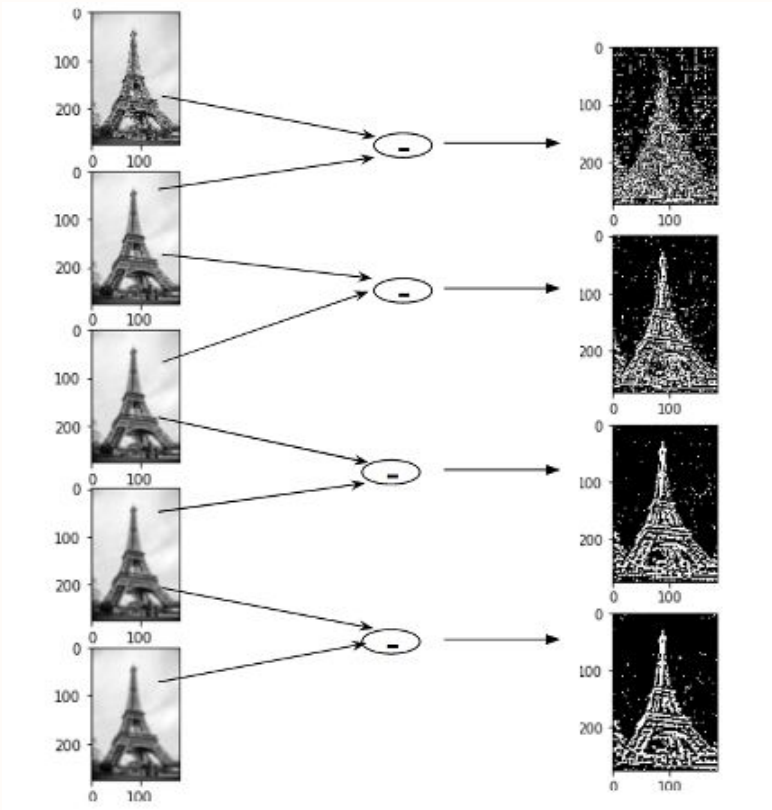




Fonte: https://docs.opencv.org/4.x/da/df5/tutorial_py_sift_intro.html

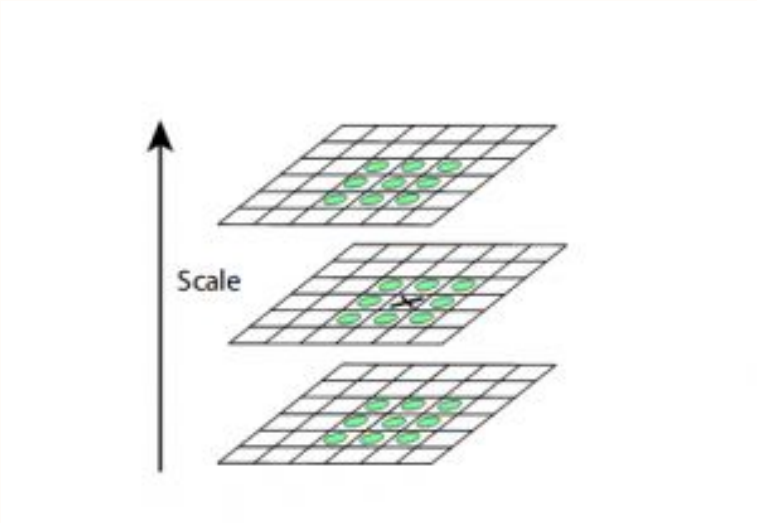
Detecção de objetos - SIFT



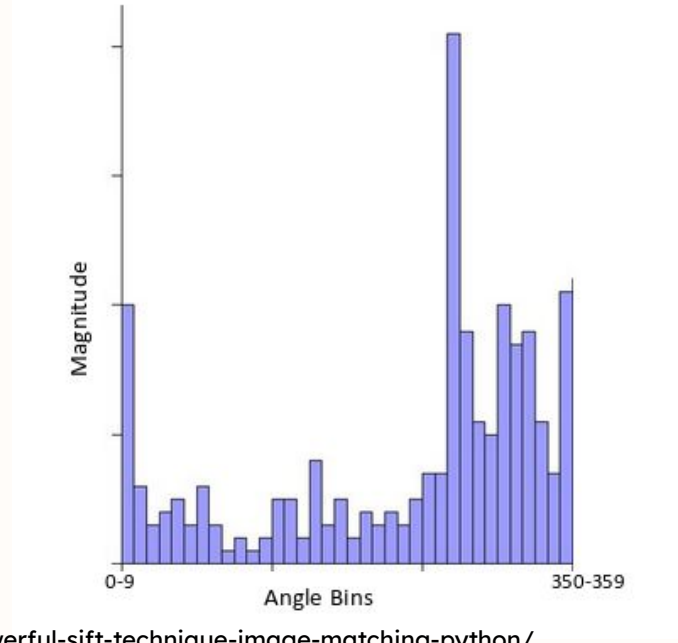


Fonte: https://www.analyticsvidhya.com/blog/2019/10/detailed-guide-powerful-sift-technique-image-matching-python/

Detecção de objetos - SIFT

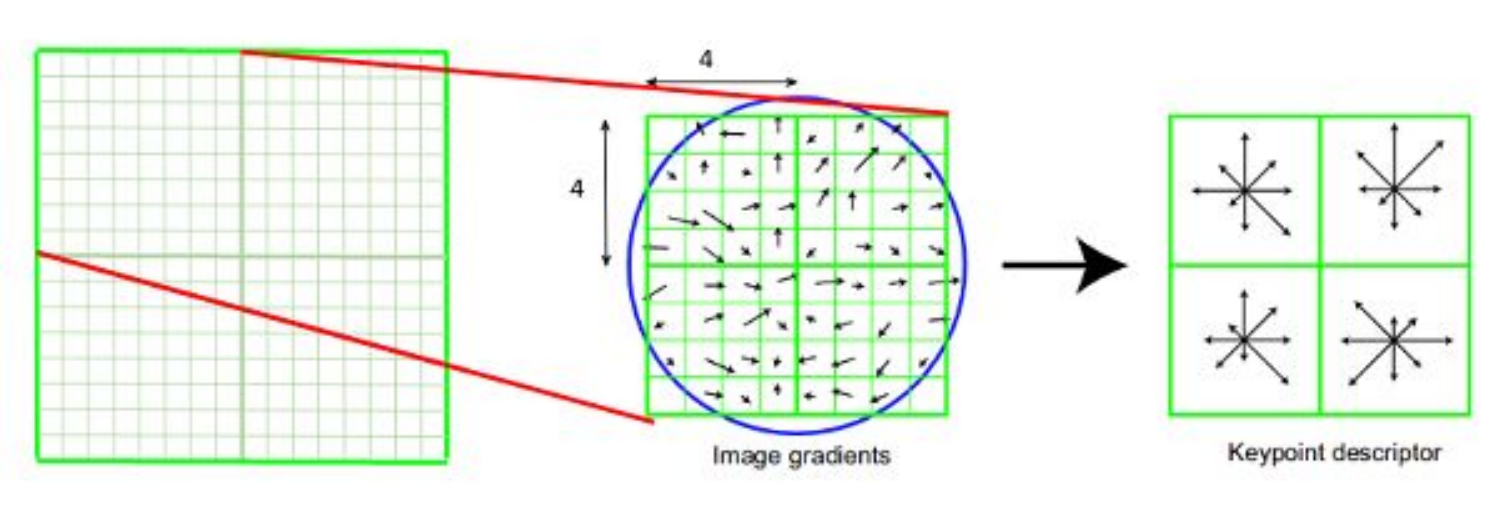


35	40	41	45	50
40	40	42	46	52
42	46	50	55	55
48	52	56	58	60
56	60	65	70	75



Fonte: https://www.analyticsvidhya.com/blog/2019/10/detailed-guide-powerful-sift-technique-image-matching-python/

Detecção de objetos - SIFT



Fonte: https://www.analyticsvidhya.com/blog/2019/10/detailed-guide-powerful-sift-technique-image-matching-python/

Detecção de objetos - SIFT

```python
import cv2


if __name__ == '__main__':

    image = cv2.imread('./data/cat1.jpg')

    gray  = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    sift = cv2.SIFT_create()
    kp   = sift.detect(gray, None)

    img_sift = image.copy()
    img_sift = cv2.drawKeypoints(gray, kp, img_sift)
```

Detecção de objetos - SIFT

Detecção de objetos - SIFT

```python
#Feature Matching

image2 = cv2.imread('./data/cat2.jpg')
image3 = cv2.imread('./data/cat3.jpg')

gray2 = cv2.cvtColor(image2, cv2.COLOR_BGR2GRAY)
gray3 = cv2.cvtColor(image3, cv2.COLOR_BGR2GRAY)

kp2, desc2 = sift.detectAndCompute(gray2, None)
kp3, desc3 = sift.detectAndCompute(gray3, None)

bf = cv2.BFMatcher(cv2.NORM_L1, crossCheck=True)

matches12 = bf.match(desc, desc2)
matches13 = bf.match(desc, desc3)

matches12 = sorted(matches12, key=lambda x: x.distance)
matches13 = sorted(matches13, key=lambda x: x.distance)

img_sift2 = image2.copy()
img_sift2 = cv2.drawMatches(gray, kp, gray2, kp2, matches12[:50], img_sift2, flags=2)

img_sift3 = image3.copy()
img_sift3 = cv2.drawMatches(gray, kp, gray3, kp3, matches13[:50], img_sift3, flags=2)
```

Detecção de objetos - SIFT

Detecção de objetos

➔ Estratégias de detecção geralmente são divididas entre métodos que utilizam abordagens **tradicionais** e abordagens baseadas em **aprendizagem de máquina** e/ou **aprendizagem profunda**

➔ Exemplos de métodos baseados em características:

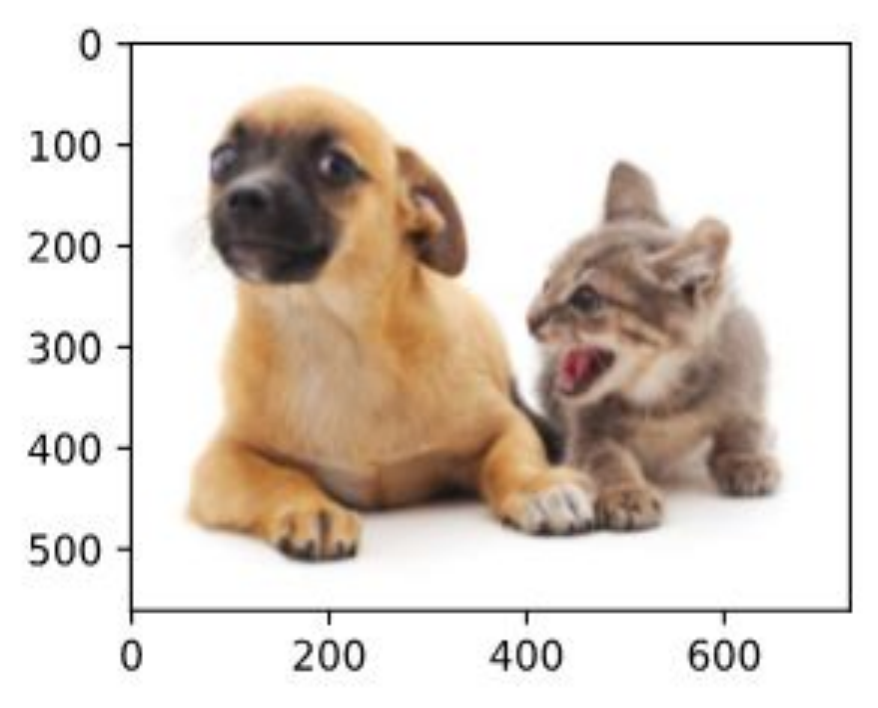
- SIFT (Scale Invariant Feature Transform)
- HOG (Histogram of Oriented Gradients)

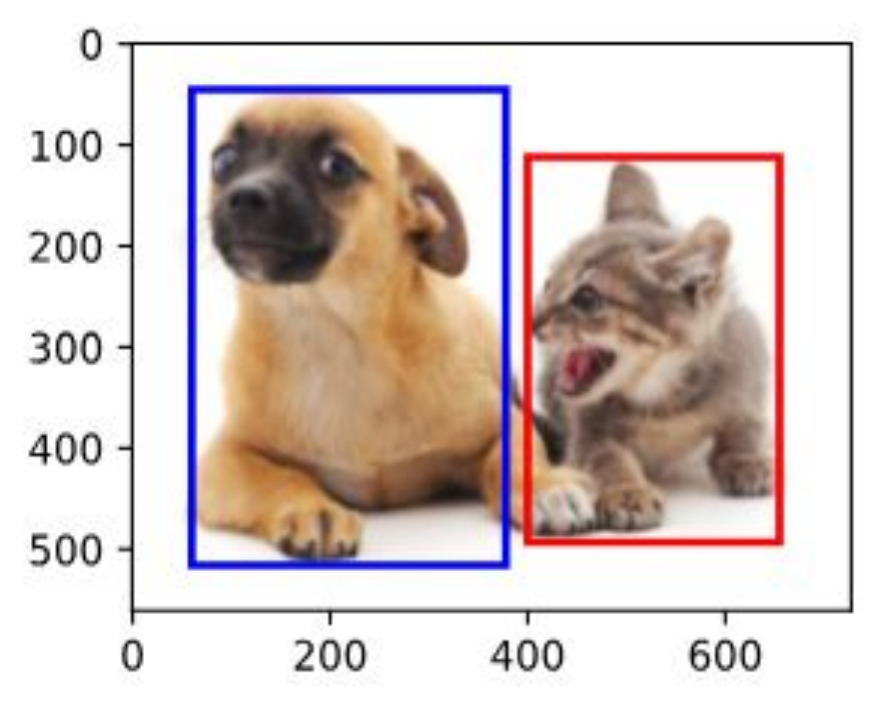
➔ Exemplos de métodos baseados em aprendizagem profunda:

- Region-based CNN (R-CNN), Fast RCNN, Faster RCNN
- YOLO, SSD, RetinaNet
- EfficientNet, MobileNet

Detecção de objetos - Bounding box 2D

- O bounding box 2D é utilizado para representar a localização espacial de um objeto na imagem

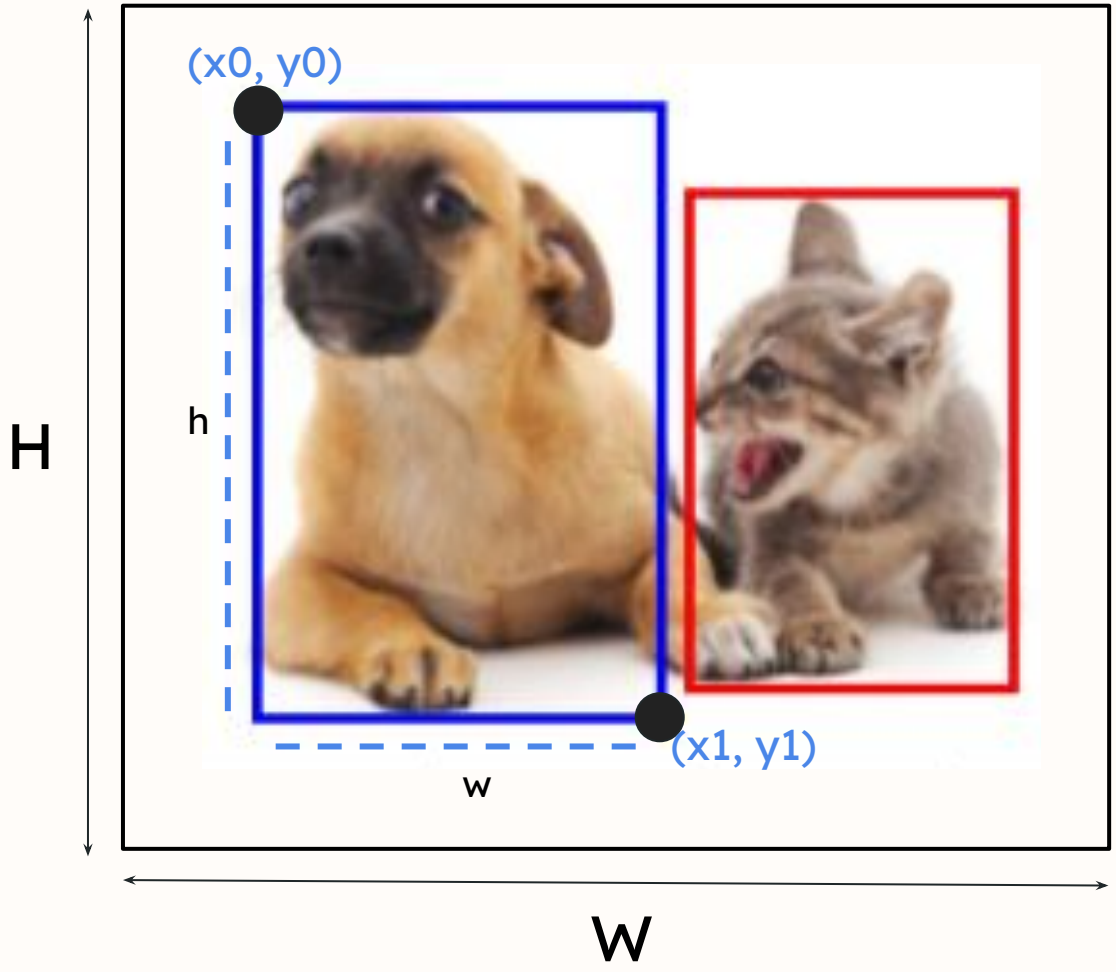




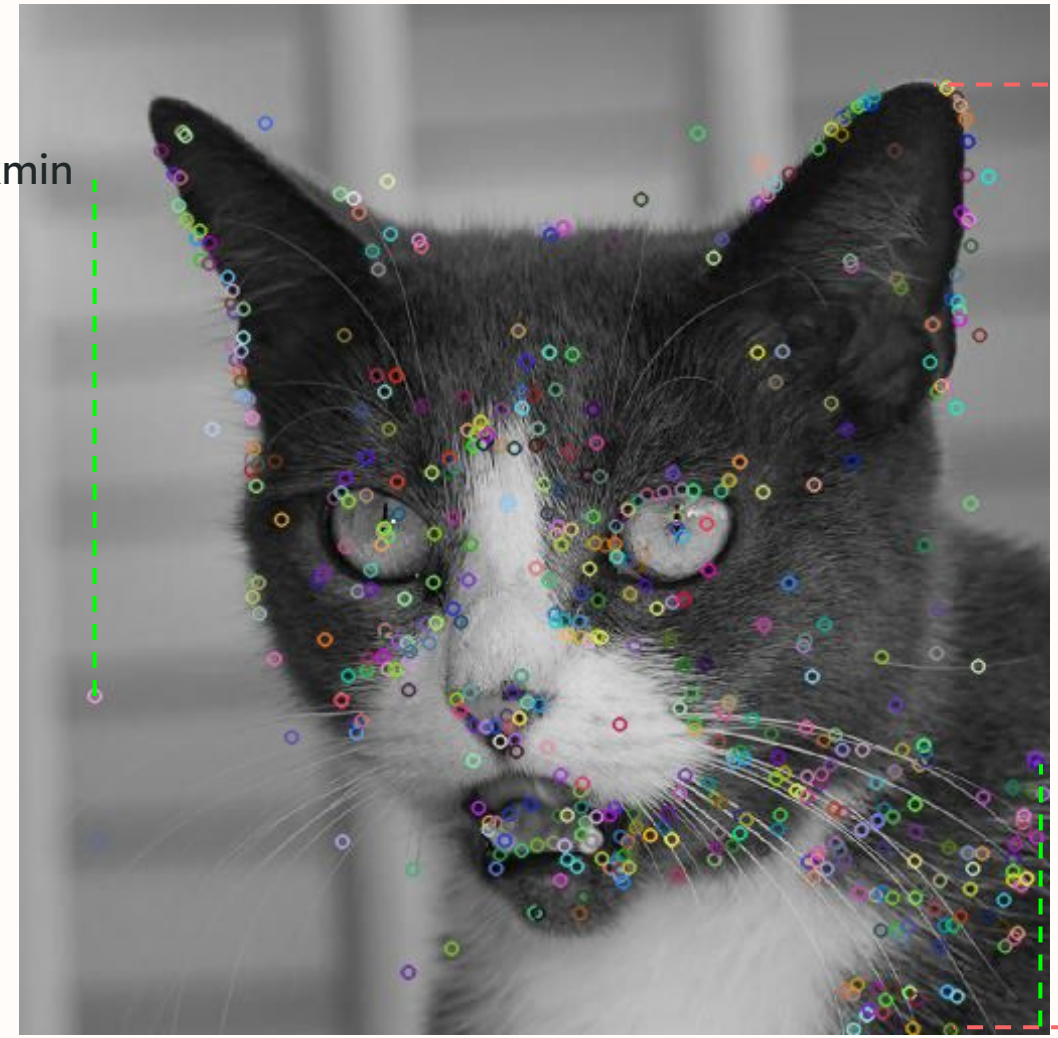
Detecção de objetos - Bounding box 2D

- ➔ Posições são determinadas pelas coordenadas do retângulo
- ➔ Considerando uma imagem de tamanho **W x H**, o bounding box 2D é comumente definido pelos parâmetros:
  - **x0:** coordenada no eixo horizontal da imagem [0, W]
  - **y0:** coordenada no eixo vertical da imagem [0, H]
  - **w:** largura do bounding box
  - **h:** altura do bounding box
- ➔ O bounding box também pode ser descrito através dos pontos superior à esquerda **(x0, y0)** e do ponto inferior à direita **(x1, y1)**, onde **x1=x0+w** e **y1=y0+h**

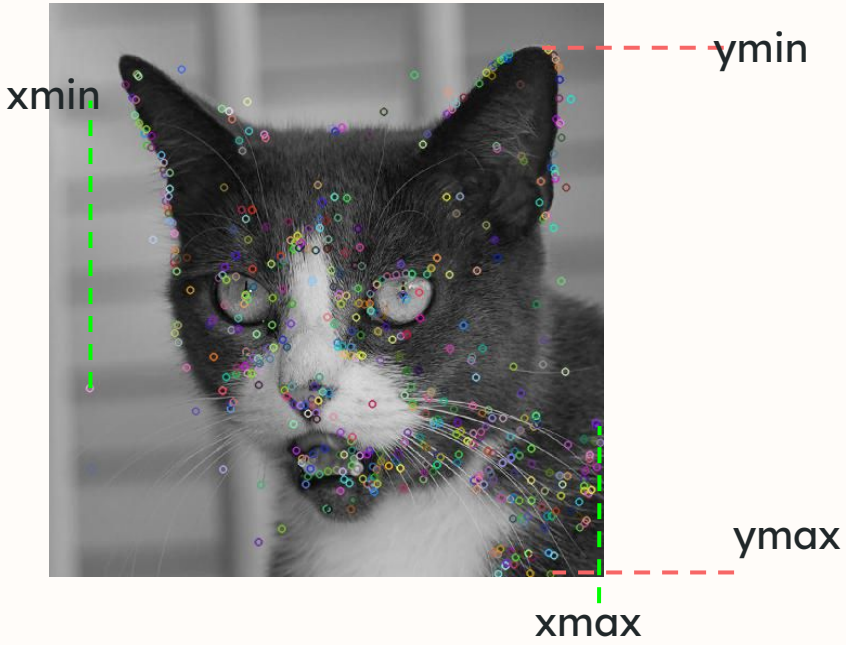
Detecção de objetos - Bounding box 2D

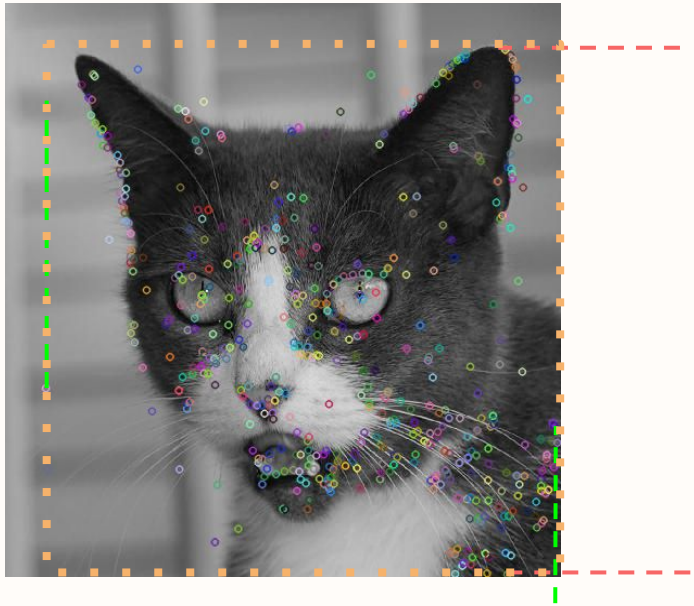


Detecção de objetos - Bounding box 2D



Detecção de objetos - Bounding box 2D





Detecção de objetos - Medindo a acurácia

➔ A **acurácia da detecção 2D** é definida através da porcentagem de sobreposição que a estimação do bounding box tem com o bounding box de ground truth (anotação) do conjunto de treino

Detecção de objetos - Medindo a acurácia

- ➔ A **acurácia da detecção 2D** é definida através da porcentagem de sobreposição que a estimação do bounding box tem com o bounding box de ground truth (anotação) do conjunto de treino
- ➔ Ou seja, calculamos a **área do bounding box estimado** e do **bounding box da ground-truth**, e avaliamos se elas ocupam a **mesma posição** na imagem

Detecção de objetos - Medindo a acurácia

- ➔ A **acurácia da detecção 2D** é definida através da porcentagem de sobreposição que a estimação do bounding box tem com o bounding box de ground truth (anotação) do conjunto de treino
- ➔ Ou seja, calculamos a **área do bounding box estimado** e do **bounding box da ground-truth**, e avaliamos se elas ocupam a **mesma posição** na imagem
- ➔ Se o valor de sobreposição for **maior que um threshold definido** a priori, consideramos a estimação correta

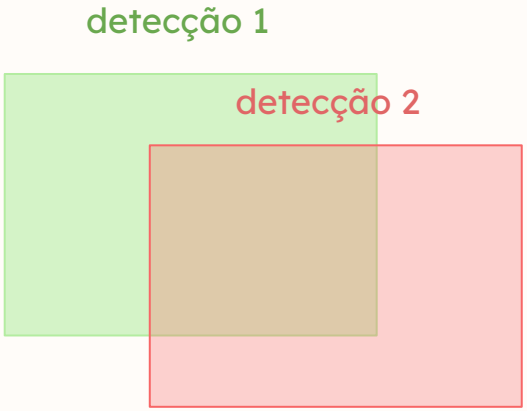
Detecção de objetos - Medindo a acurácia

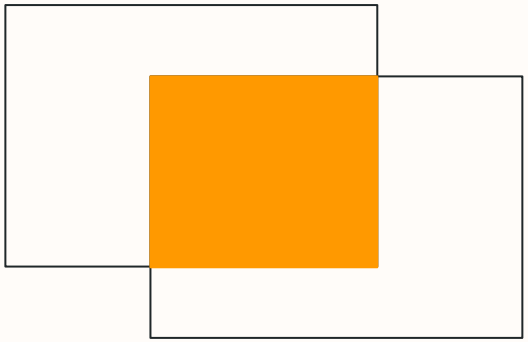
- ➔ A **acurácia da detecção 2D** é definida através da porcentagem de sobreposição que a estimação do bounding box tem com o bounding box de ground truth (anotação) do conjunto de treino
- ➔ Ou seja, calculamos a **área do bounding box estimado** e do **bounding box da ground-truth**, e avaliamos se elas ocupam a **mesma posição** na imagem
- ➔ Se o valor de sobreposição for **maior que um threshold definido** a priori, consideramos a estimação correta
- ➔ A sobreposição é medida, geralmente, através da métrica **IoU (Intersection over Union)**

IoU (Intersection over Union)

- ➔ **IoU** é uma das principais métricas para utilizada como base para avaliar a **acurácia** da detecção
  - Computa a **proximidade** das detecções comparando a relação entre a **intersecção** e **união** dos bounding boxes
  - Também pode ser utilizada como o **fator para filtrar** detecções redundantes (e.g., detecções muito próximas)

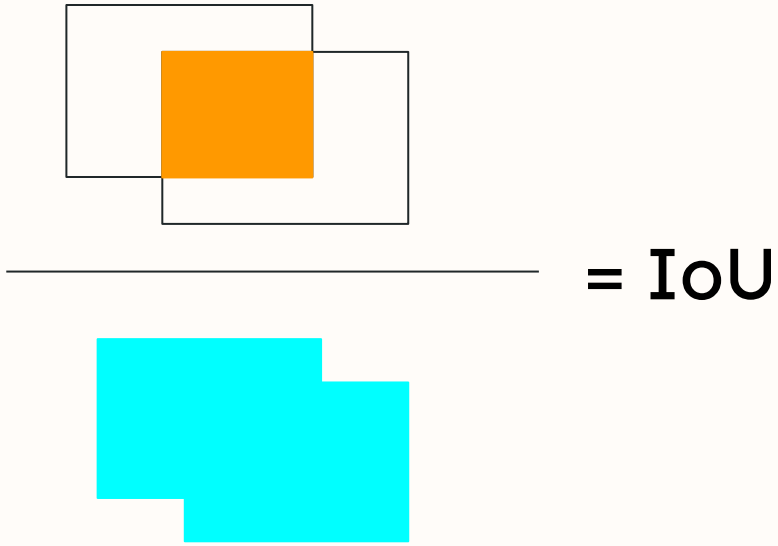
IoU (Intersection over Union)



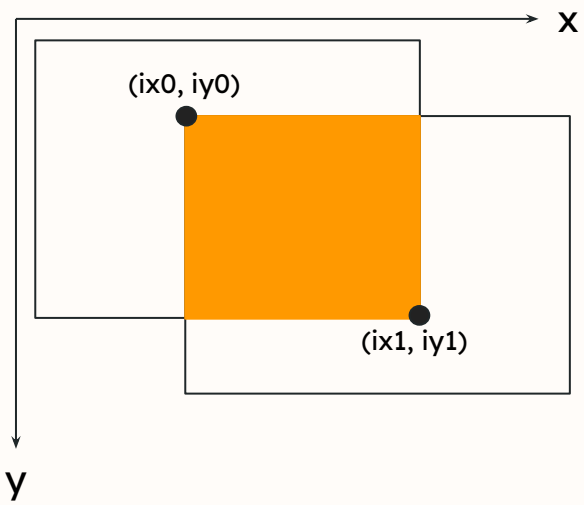


Intersecção

União

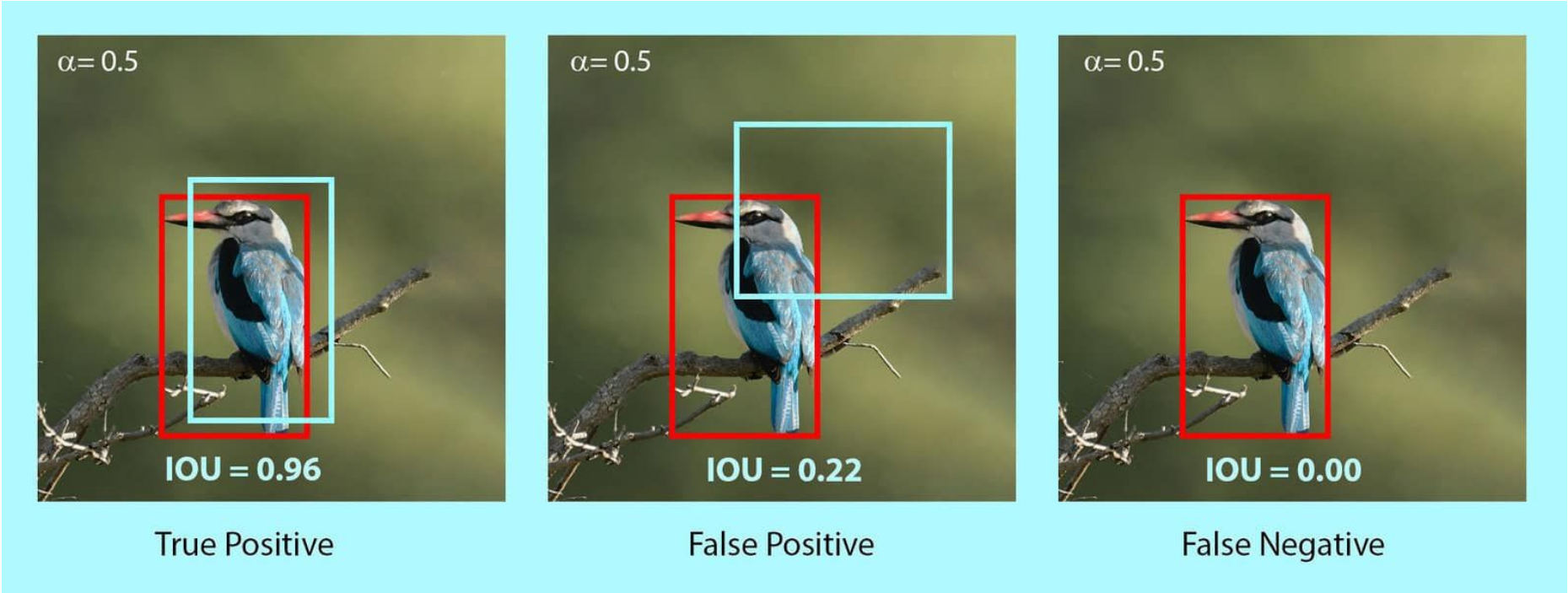


IoU (Intersection over Union)



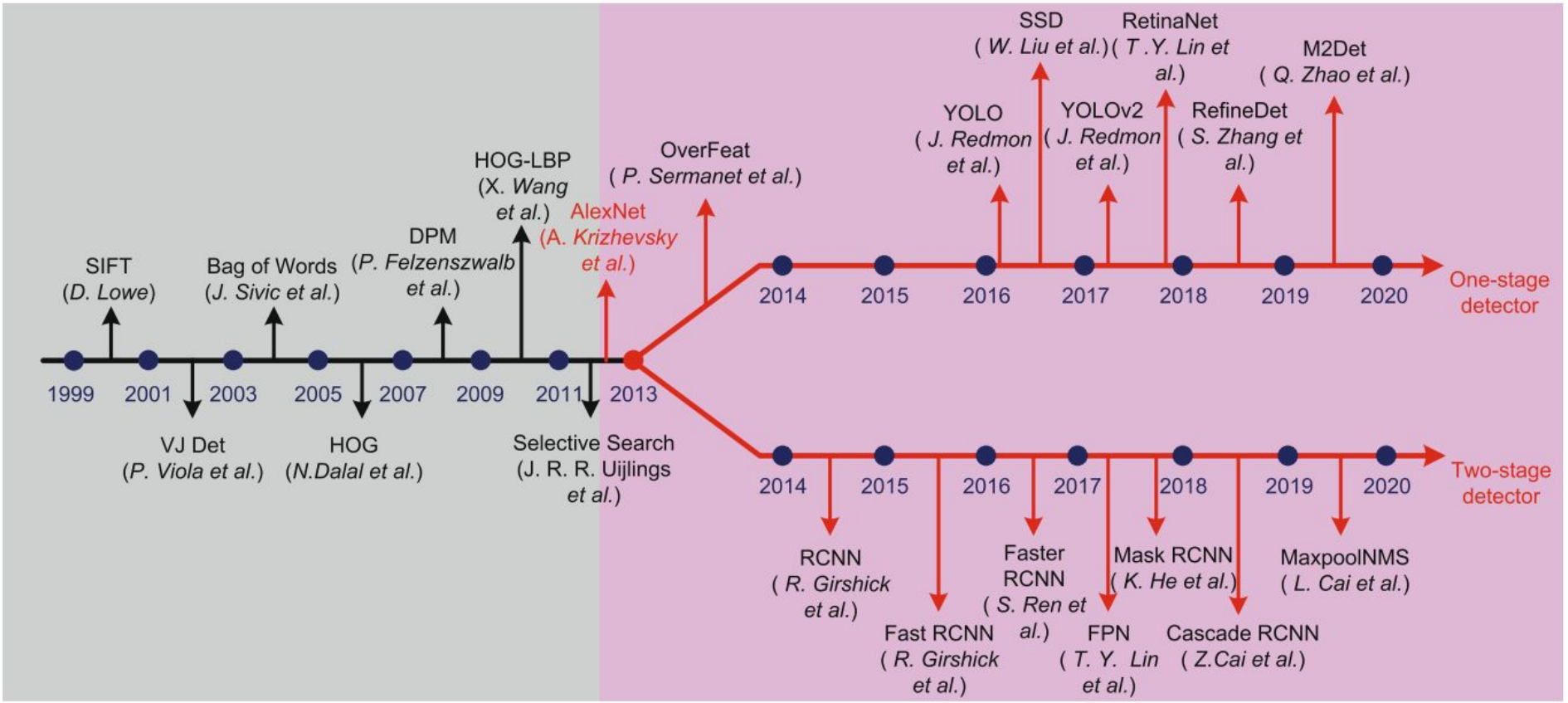

```python
def iou(box1, box2):

    area1 = (box1[2] - box1[0]) * (box1[3] - box1[1])
    area2 = (box2[2] - box2[0]) * (box2[3] - box2[1])

    ix0 = max(box2[0], box1[0])
    iy0 = max(box2[1], box1[1])
    ix1 = min(box2[2], box1[2])
    iy1 = min(box2[3], box1[3])

    intersection_area = max(0, ix1 - ix0) * max(0, iy0, iy1)

    union_area = area1 + area2 - intersection_area

    iou_value = intersection_area / union_area

    return iou_value
```

IoU (Intersection over Union)



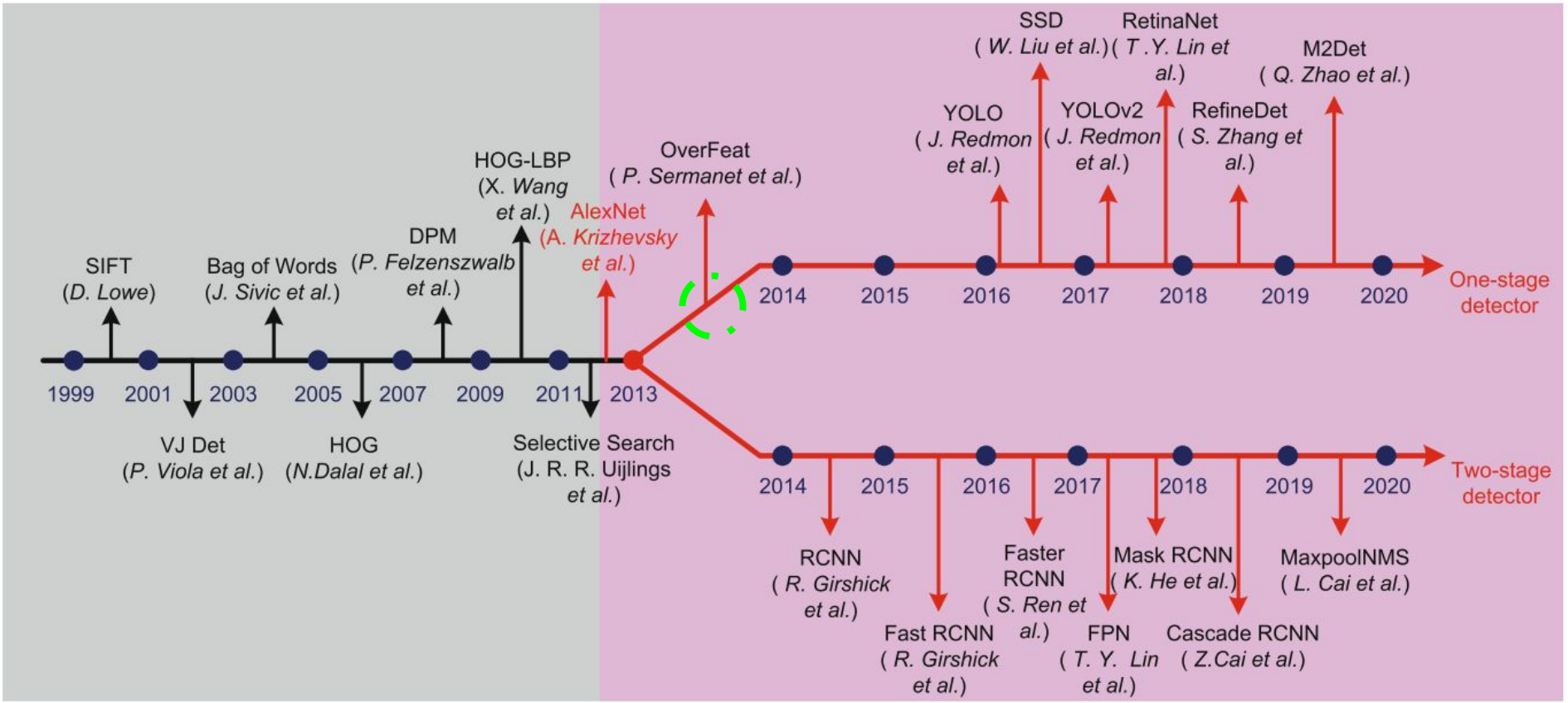
https://learnopencv.com/intersection-over-union-iou-in-object-detection-and-segmentation/

Evolução - detecção de objetos com deep learning



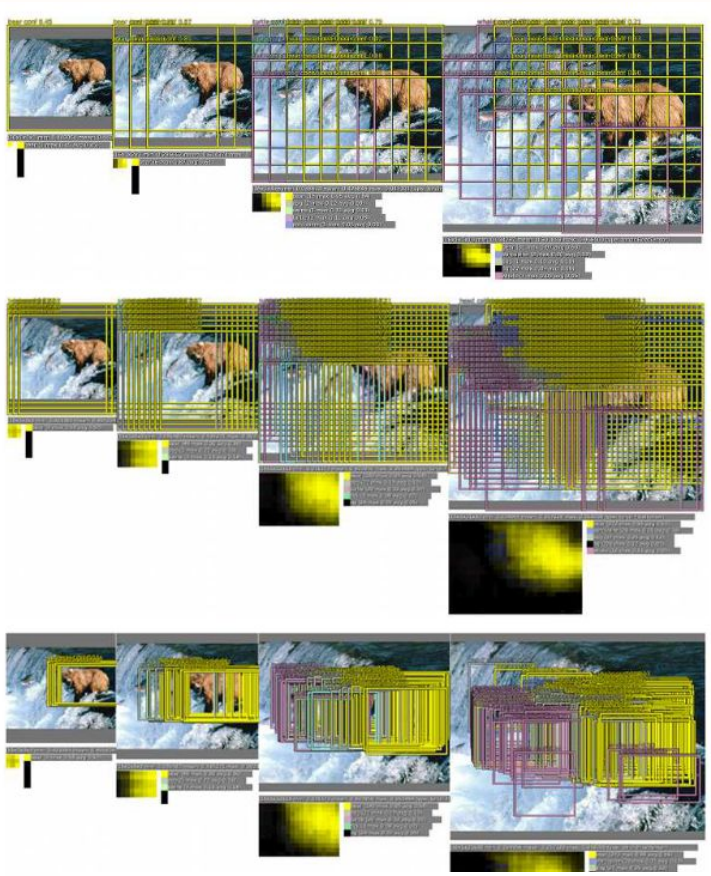
Xiao, Youzi, et al. "A review of object detection based on deep learning." *Multimedia Tools and Applications* 79 (2020): 23729-23791.

Evolução - detecção de objetos com deep learning



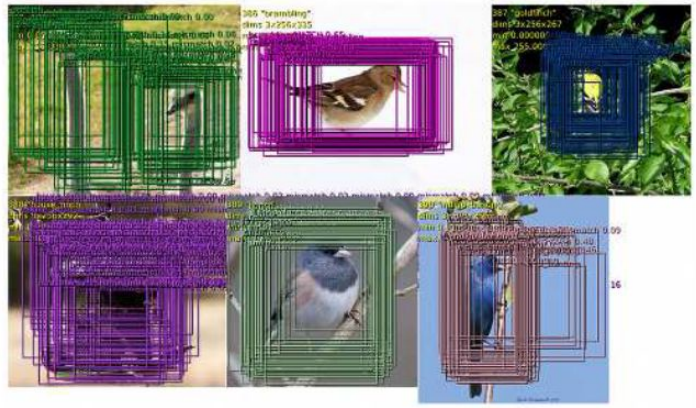
Xiao, Youzi, et al. "A review of object detection based on deep learning." *Multimedia Tools and Applications* 79 (2020): 23729-23791.

Detecção de objetos - Deep Learning (Overfeat)



Pierre Sermanet, et al. **Overfeat: Integrated Recognition, Localization and Detection using Convolutional Networks** (arxiv.org/pdf/1312.6229)

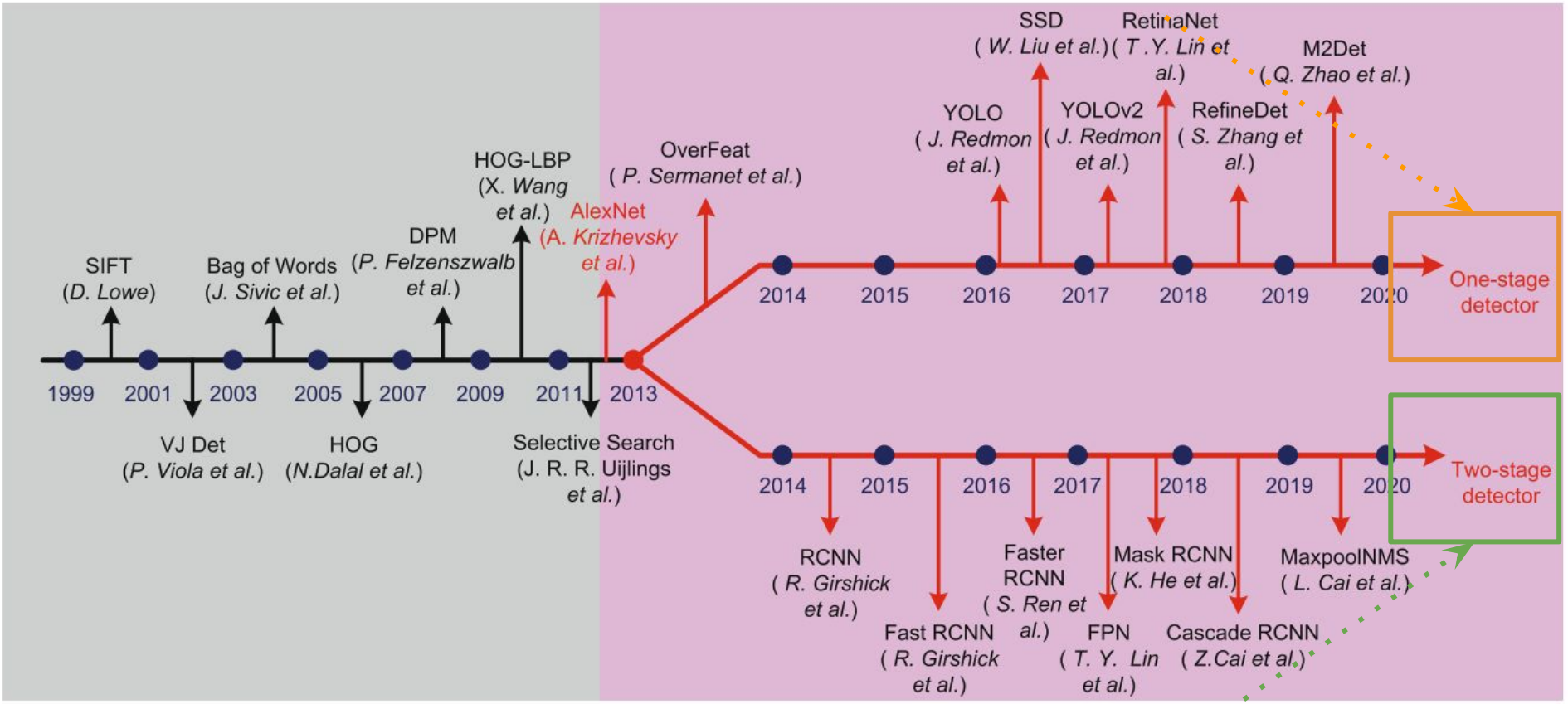
https://github.com/sermanet/OverFeat/tree/master



Layer	1	2	3	4	5	6	7	Output 8
Stage	conv + max	conv + max	conv	conv	conv + max	full	full	full
# channels	96	256	512	1024	1024	3072	4096	1000
Filter size	11x11	5x5	3x3	3x3	3x3	-	-	-
Conv. stride	4x4	1x1	1x1	1x1	1x1	-	-	-
Pooling size	2x2	2x2	-	-	2x2	-	-	-
Pooling stride	2x2	2x2	-	-	2x2	-	-	-
Zero-Padding size	-	-	1x1x1x1	1x1x1x1	1x1x1x1	-	-	-
Spatial input size	231x231	24x24	12x12	12x12	12x12	6x6	1x1	1x1

Evolução - detecção de objetos com deep learning

Realiza a detecção e classificação em um único passo



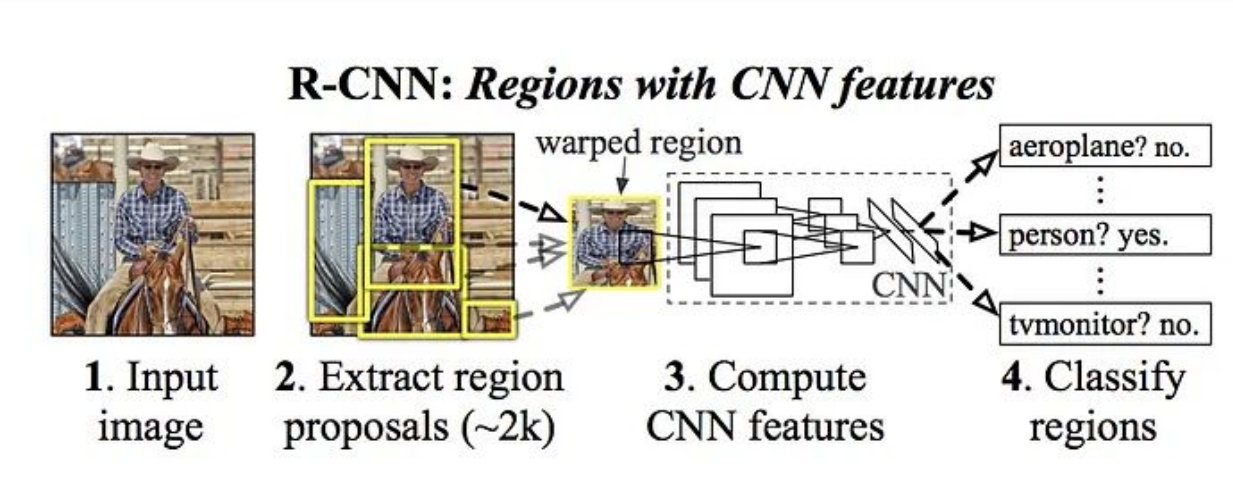
Primeiro faz a detecção do objeto na imagem e em seguida realiza a classificação

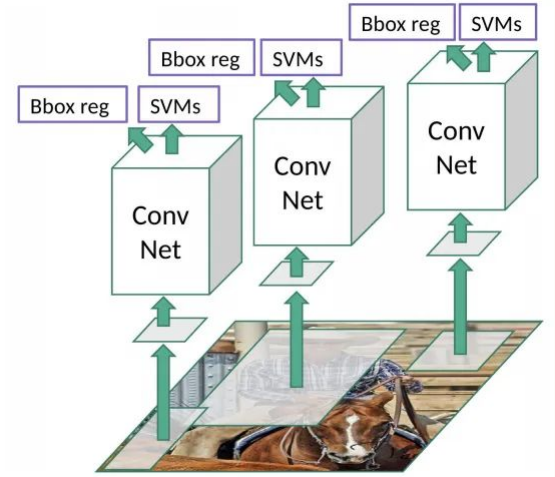
Xiao, Youzi, et al. "A review of object detection based on deep learning." *Multimedia Tools and Applications* 79 (2020): 23729-23791.

Two stage detection: RCNN

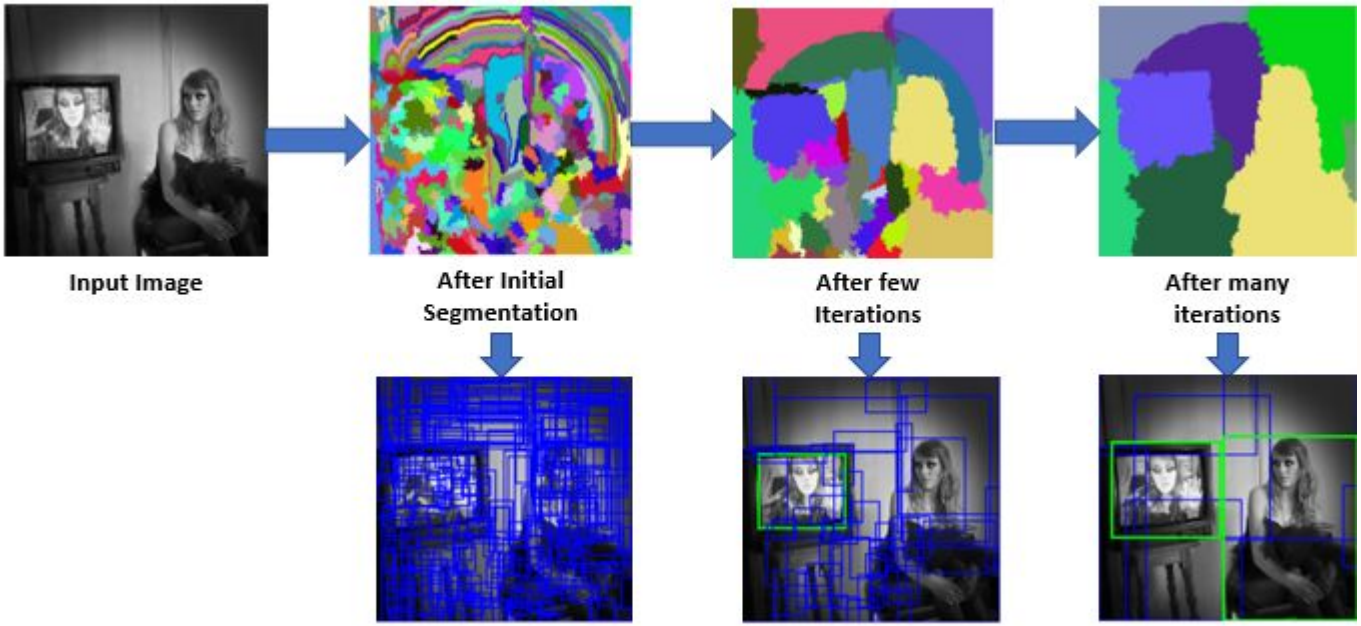
Two-stage detection - RCNN

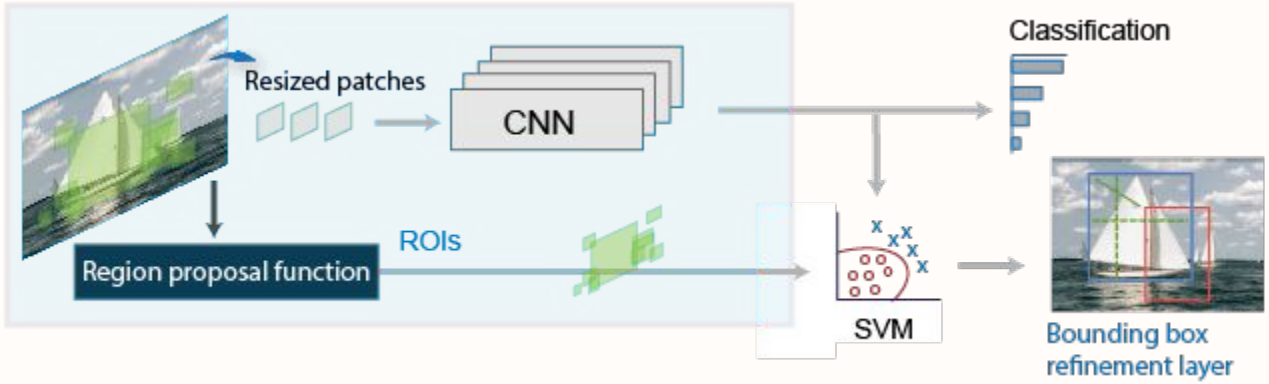
➔ Detecção de objetos em duas etapas com RCNN
- ◆ Seleciona um número pré-definido de regiões com selective search (2000)
- ◆ Avalia cada região com uma SVM, utilizando features extraídas pela CNN





Two-stage detection - RCNN



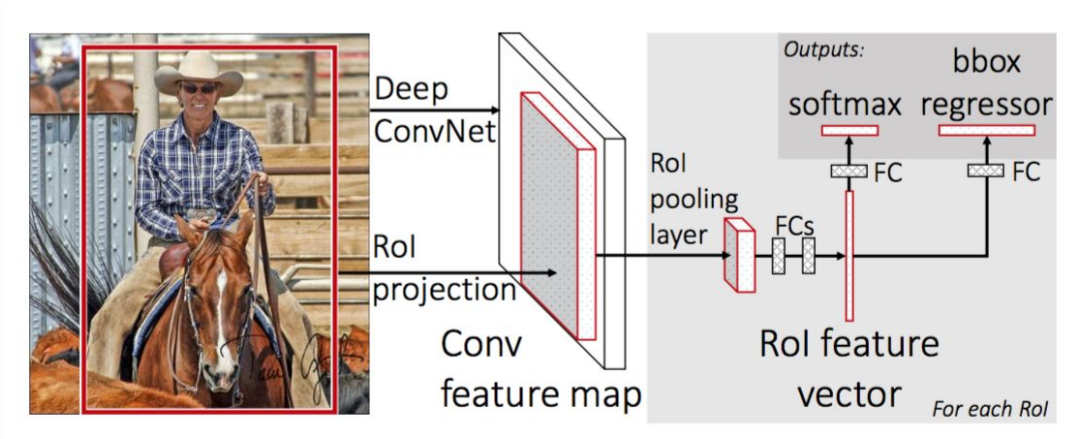


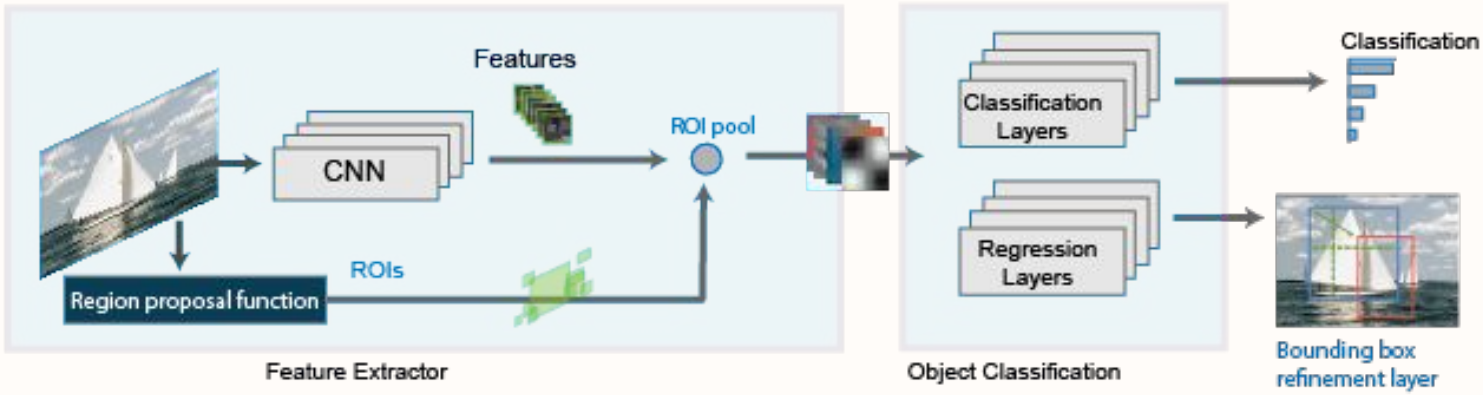
Fonte: https://www.geeksforgeeks.org/selective-search-for-object-detection-r-cnn/
https://www.mathworks.com/help/vision/ug/getting-started-with-r-cnn-fast-r-cnn-and-faster-r-cnn.html

Two-stage detection - RCNN

- ➔ Leva muito tempo para treinar (sempre classifica 2000 regiões)
- ➔ Inferência também é lenta (~47s)
- ➔ Selective search é um algoritmo que não utiliza aprendizado, consequentemente, a proposição de regiões nunca é melhorada

Two-stage detection - Fast-RCNN

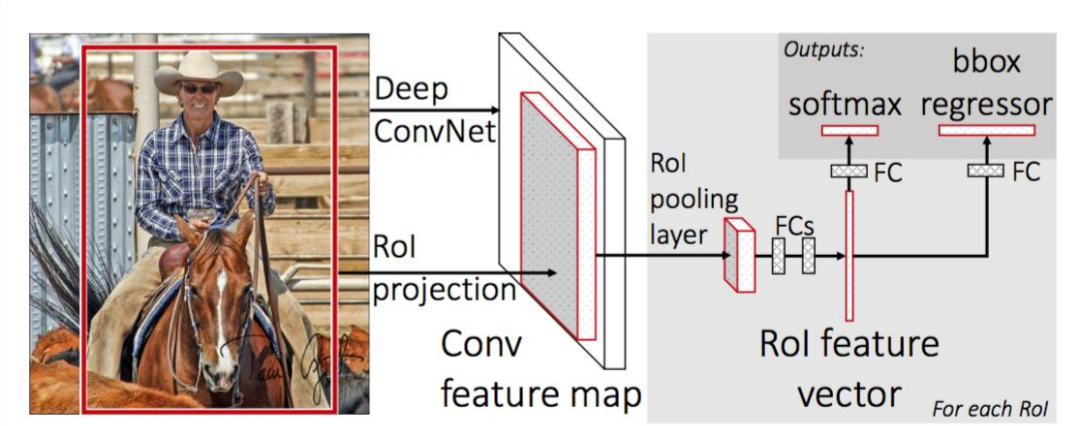


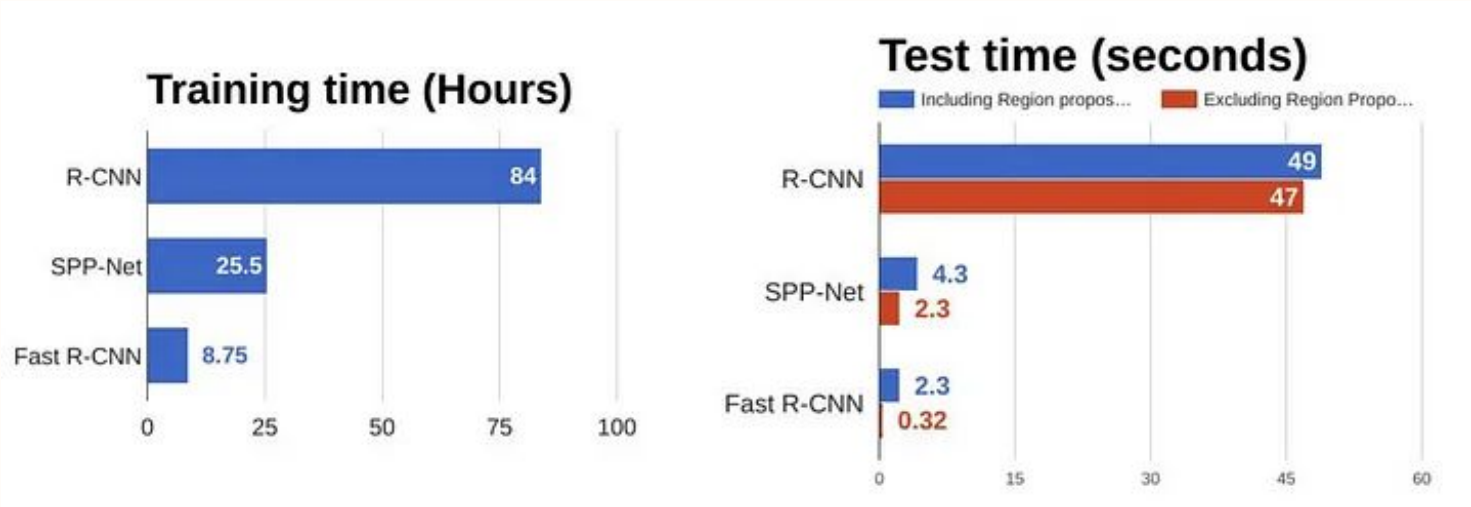


Girshick. “Fast R-CNN”, ICCV 2015. Girshick, Ross. "Fast r-cnn." *Proceedings of the IEEE international conference on computer vision*. 2015.

Fonte: https://www.mathworks.com/help/vision/ug/getting-started-with-r-cnn-fast-r-cnn-and-faster-r-cnn.html

Two-stage detection - Fast-RCNN

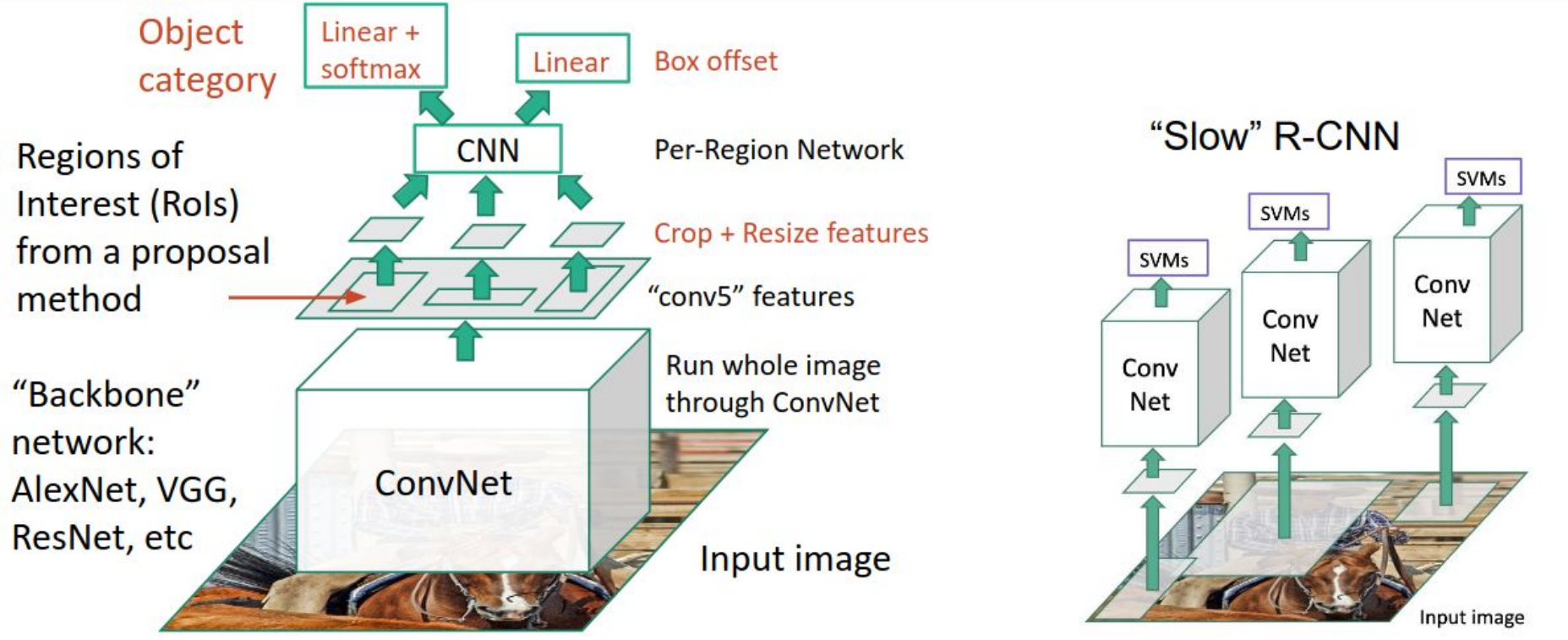




Girshick. “Fast R-CNN”, ICCV 2015. Girshick, Ross. "Fast r-cnn." *Proceedings of the IEEE international conference on computer vision*. 2015.

Fonte: https://www.mathworks.com/help/vision/ug/getting-started-with-r-cnn-fast-r-cnn-and-faster-r-cnn.html

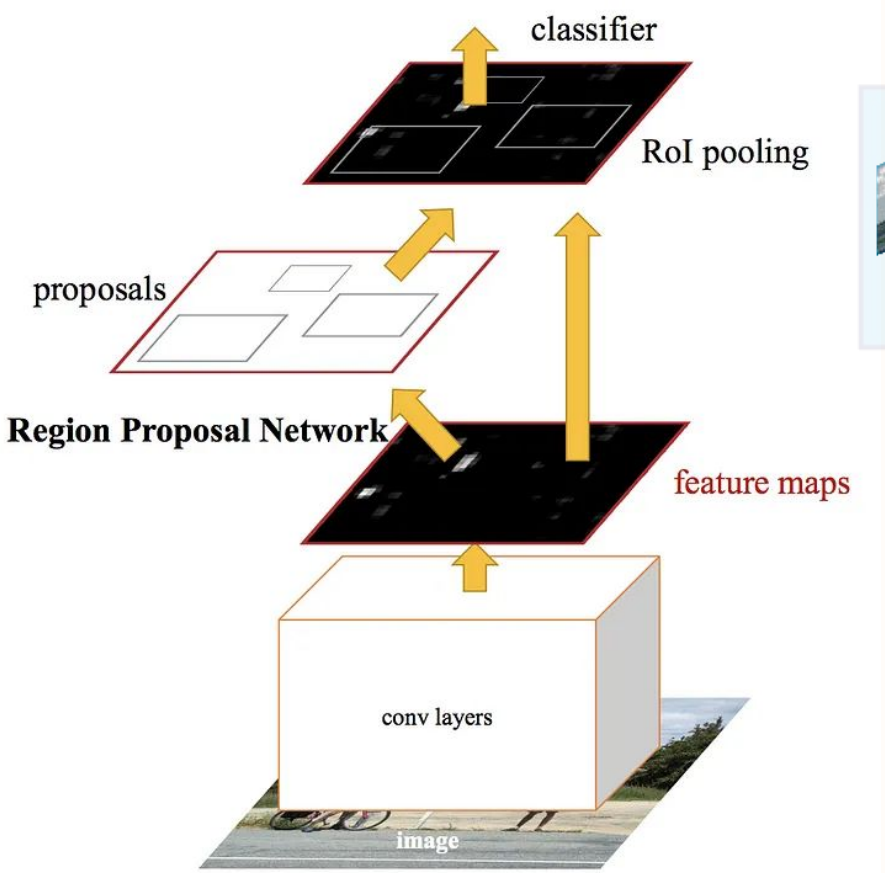
Two-stage detection - Fast-RCNN

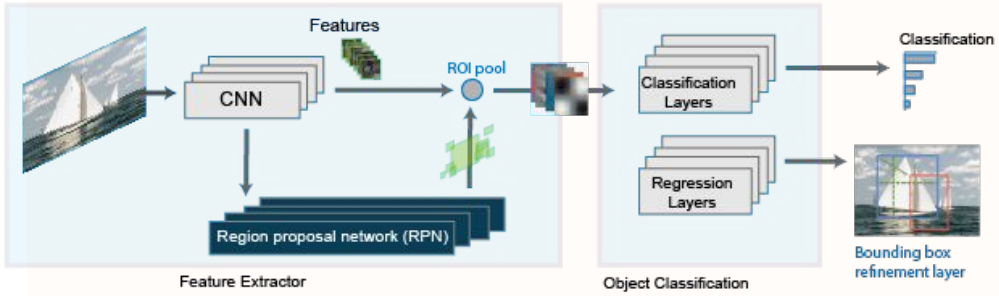


Girshick. "Fast R-CNN", ICCV 2015. Girshick, Ross. "Fast r-cnn." *Proceedings of the IEEE international conference on computer vision*. 2015.

Two-stage detection - Faster-RCNN

- ➔ Faster R-CNN
  - ◆ Region Proposal Network - Prediz regiões a partir das features
    - Treina 4 funções de custo em conjunto (classificação e regressão com RPN e bounding box e scores finais)



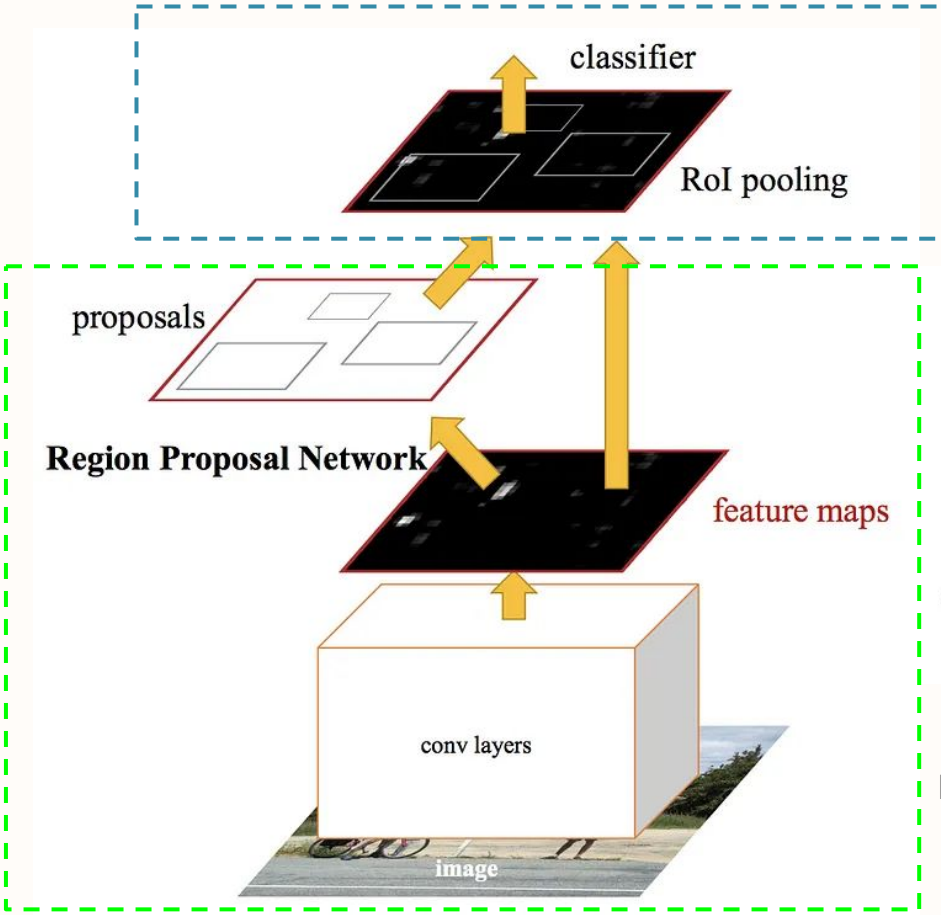


Ren, Shaoqing, et al. "**Faster r-cnn: Towards real-time object detection with region proposal networks**." *Advances in neural information processing systems* 28 (2015).

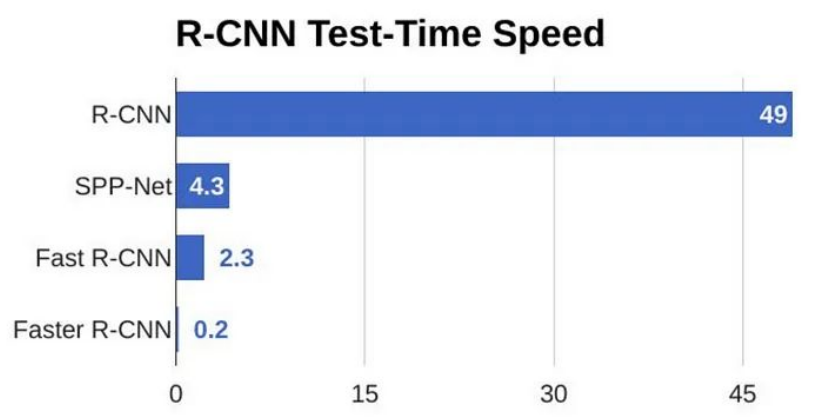
Fonte: https://www.mathworks.com/help/vision/ug/getting-started-with-r-cnn-fast-r-cnn-and-faster-r-cnn.html

Two-stage detection - Faster-RCNN

- ➔ Faster R-CNN
  - ◆ Region Proposal Network - Prediz regiões a partir das features
    - Treina 4 funções de custo em conjunto (classificação e regressão com RPN e bounding box e scores finais)



Estágio 2 - Para cada região
1. Alinha as features
2. Prediz a classe dos objetos
3. Prediz o offset do box



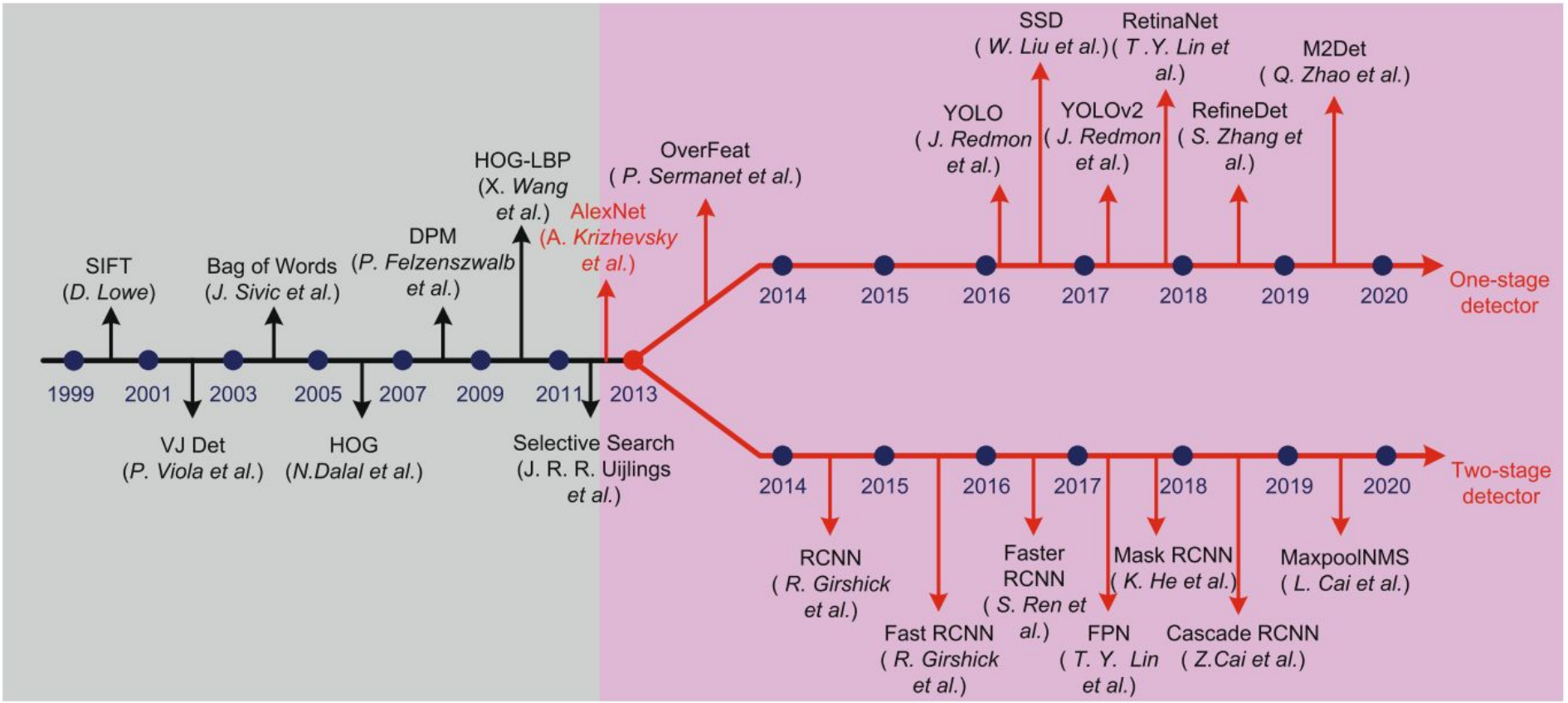
Estágio 1 - Para cada imagem
1. Extrai as features
2. Propõe regiões de interesse

Two-stage detection - Faster-RCNN

RCNN	➔ Treinamento lento ➔ RoIs definidas de acordo com a função utilizada ➔ predição utilizando svm
Fast-RCNN	➔ Utiliza toda imagem ➔ Detector utiliza features da CNN ➔ Substitui svm por softmax
Faster-RCNN	➔ Melhor desempenho de inferência e treinamento ➔ RoIs são aprendidas ➔ Remove necessidade de utilizar selective search

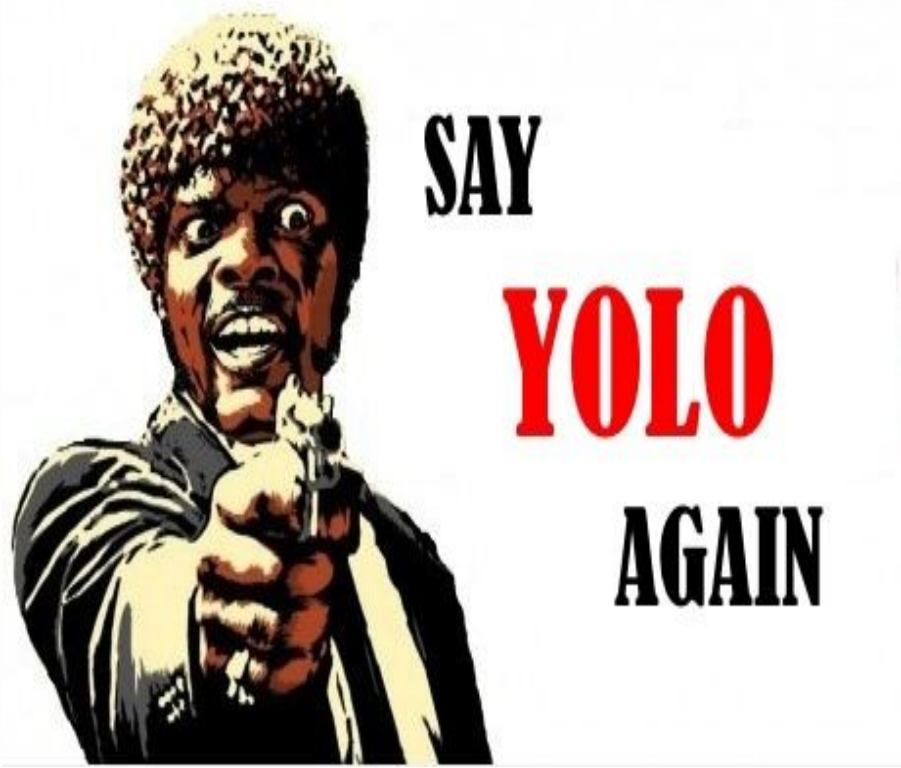
One-stage detection: YOLO

Evolução - detecção de objetos com deep learning



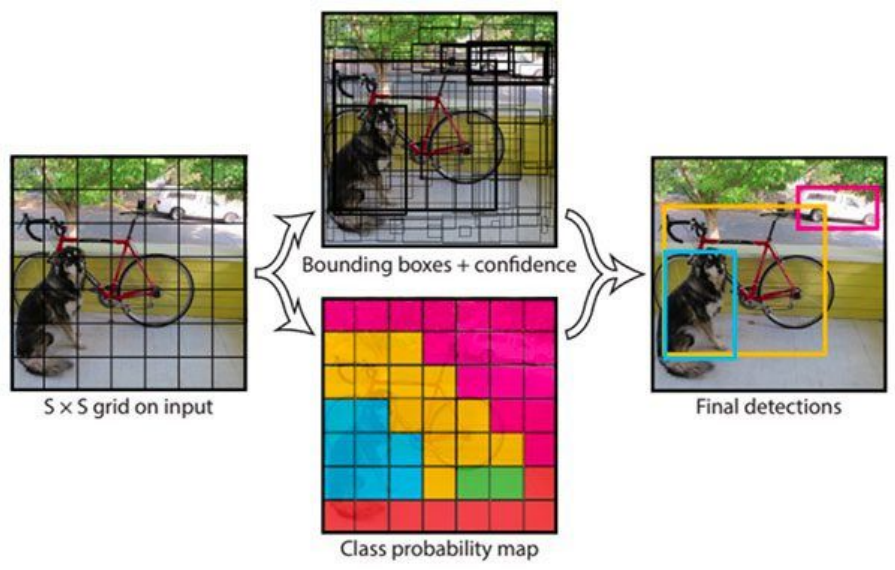
One Stage Detection - YOLO

- You Only Look Once: Unified, Real-Time Object Detection
  - YOLO é uma técnica do estado-da-arte de **detecção de objetos**
  - Correspondem à uma família de modelos *end-to-end* desenvolvidos com o intuito de realizar a detecção de objetos de forma **rápida** e com **alta acurácia**



YOLO - Características

- ➔ Utiliza uma rede neural para estimar a **localização** e **classificação** de objetos em um **único passo**
- ➔ Possui alta tolerância a **oclusões** e **sobreposições**
- ➔ Olha para a imagem inteira de uma vez, o que possibilita a utilização de **informações contextuais** ao realizar a detecção
- ➔ É altamente **eficiente** para aplicações em tempo-real

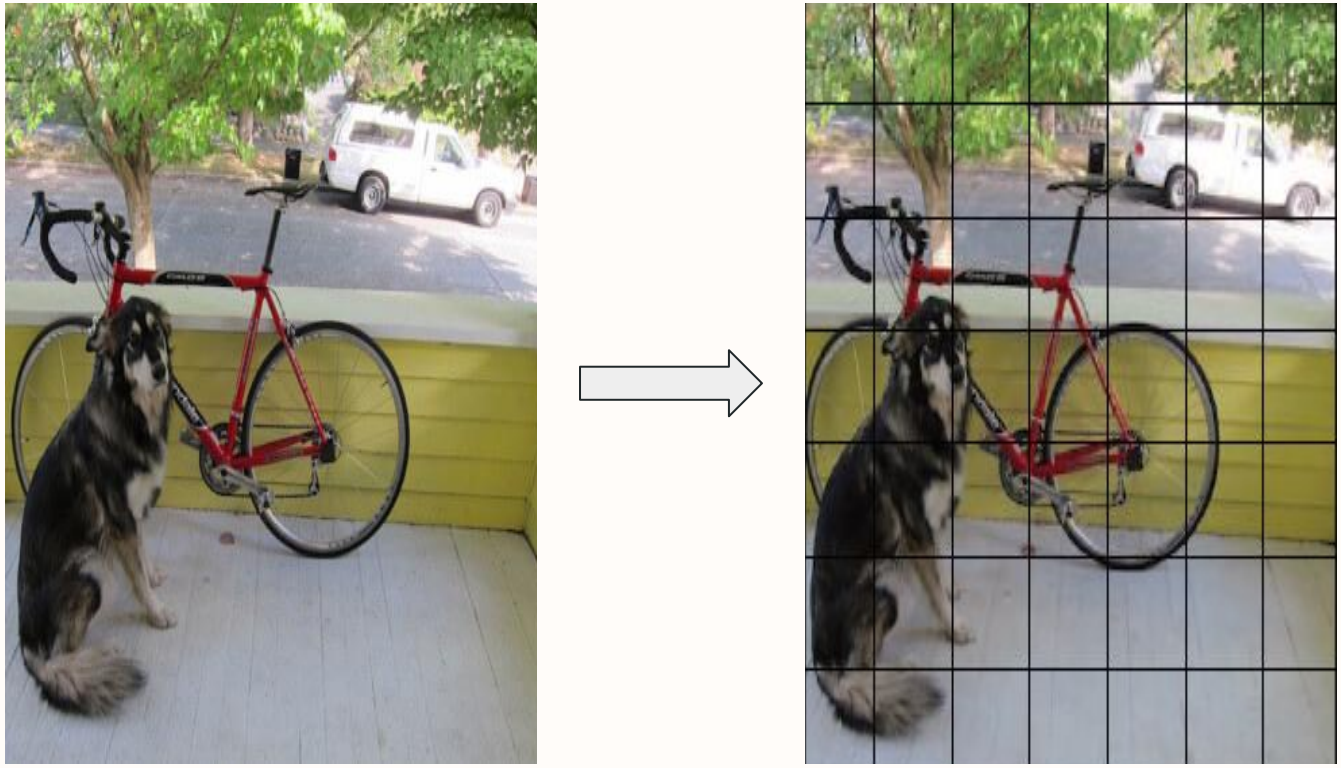


YOLO vs R-CNN

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [38]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]	2007+2012	73.2	7
Faster R-CNN ZF [28]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

YOLO - “Algoritmo”

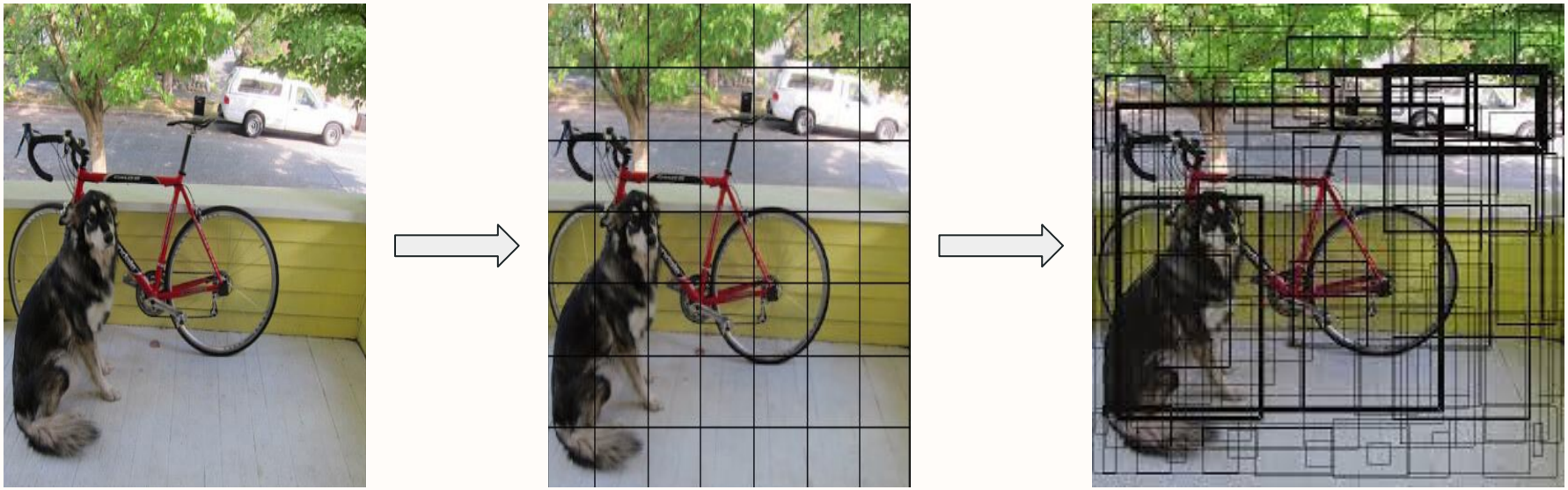
1. Inicialmente divide-se a imagem em grades de tamanho **SxS**



SxS grid

YOLO - “Algoritmo”

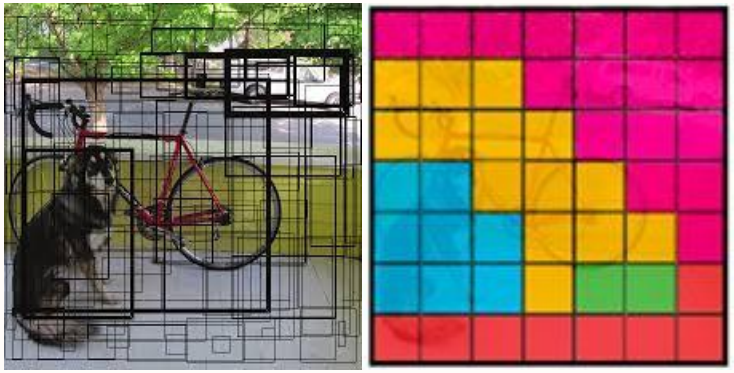
1. Inicialmente divide-se a imagem em grades de tamanho **SxS**
2. Para cada célula da grade geram-se **B** bounding boxes

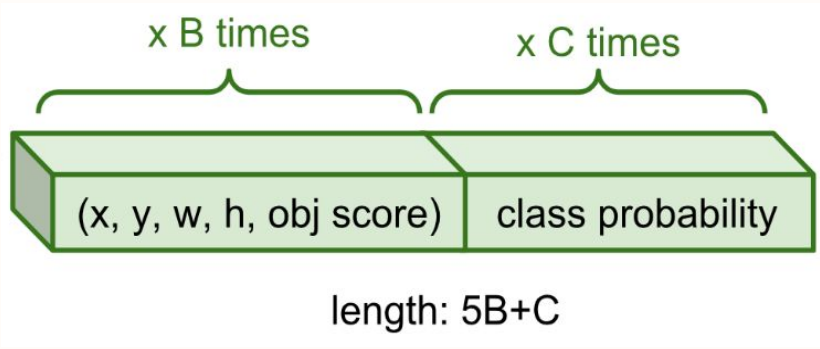


SxS grid

YOLO - “Algoritmo”

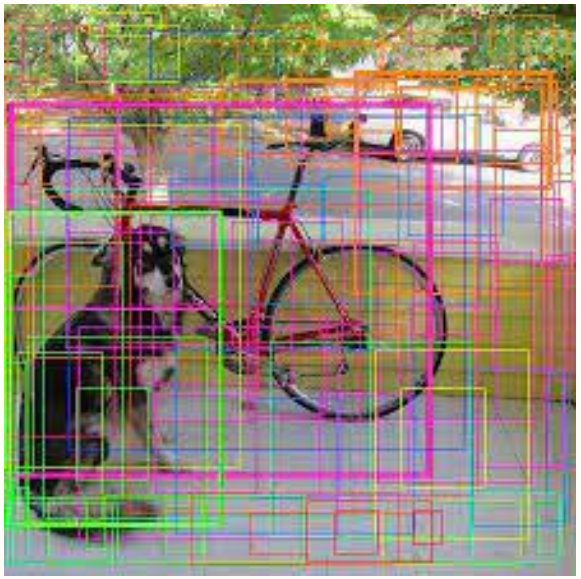
1. Inicialmente divide-se a imagem em grades de tamanho **SxS**
2. Para cada célula da grade geram-se **B** bounding boxes
3. Para cada retângulo B realizam-se as predições das coordenadas **bx**, **by**, **bw** e **bh** e de uma **object score** que indica a probabilidade da célula conter um objeto



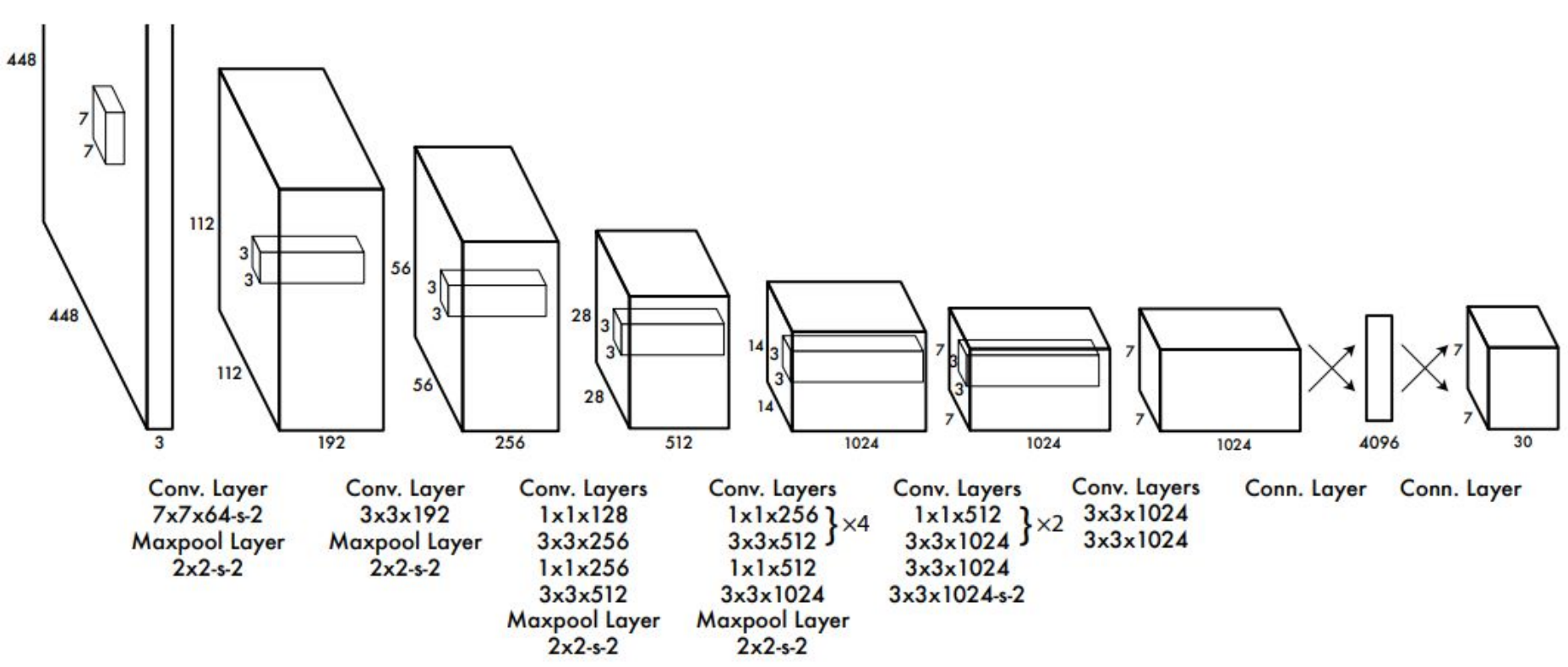


YOLO - “Algoritmo”

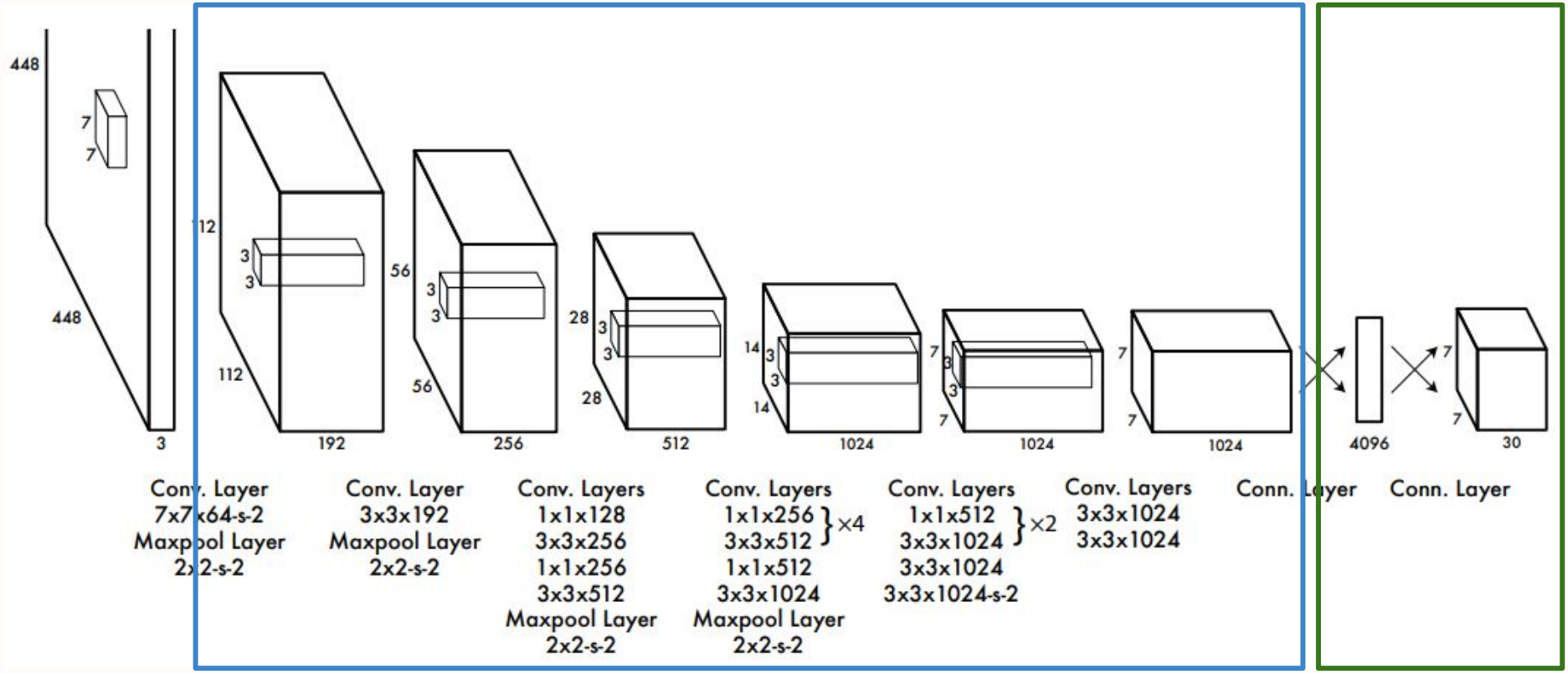
1. Inicialmente divide-se a imagem em grades de tamanho **SxS**
2. Para cada célula da grade geram-se **B** bounding boxes
3. Para cada retângulo B realizam-se as predições das coordenadas **bx**, **by**, **bw** e **bh** e de uma **object score** que indica a probabilidade da célula conter um objeto conhecido
4. Se a bounding box contiver um objeto, então teremos também uma predição da **classe** correspondente ao objeto



YOLO - Arquitetura (DarkNet)

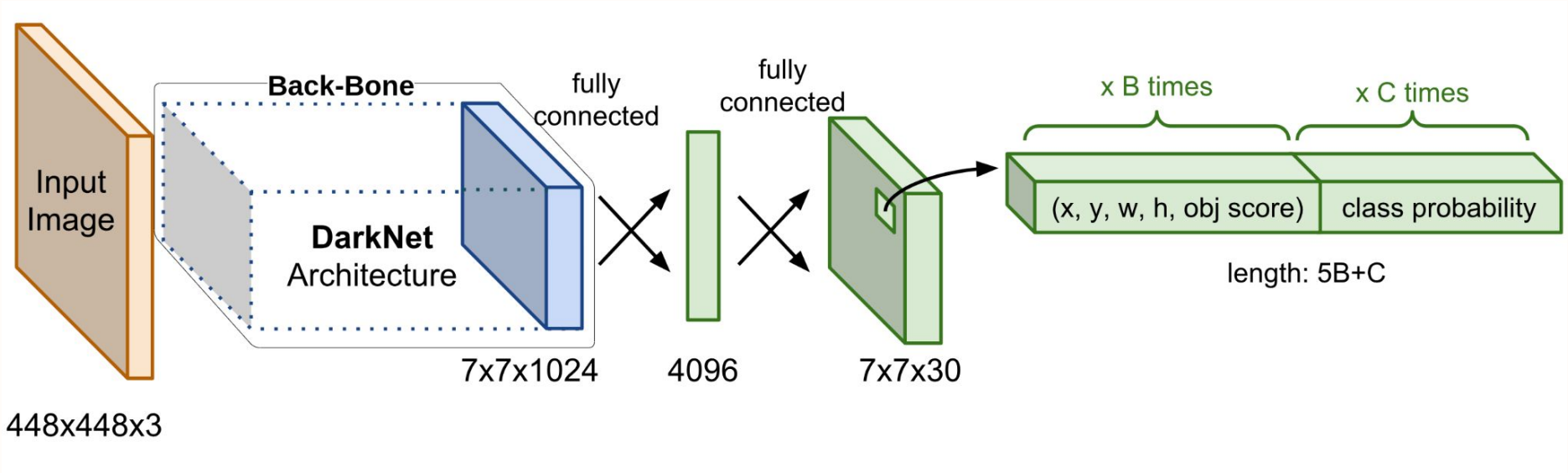


YOLO - Arquitetura (DarkNet)



DarkNet

YOLO - Arquitetura



YOLO - Loss

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

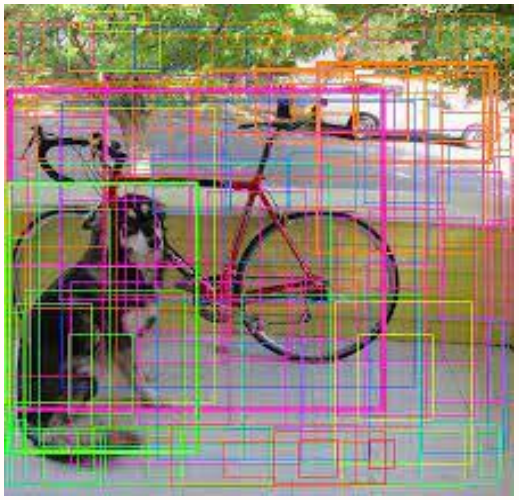
$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

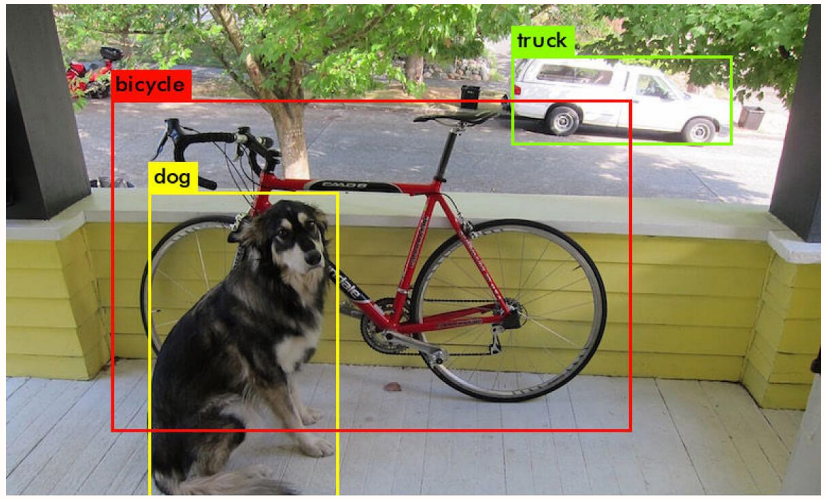
$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2$$

YOLO - NMS

- ➔ Non-Maximum Suppression (NMS) é utilizado para eliminar detecções redundantes
  - ◆ Se baseia nos valores de IoU para cada detecção realizada





YOLO - NMS (Algoritmo)

➔ Non-Maximum Suppression (NMS) é utilizado para eliminar detecções redundantes

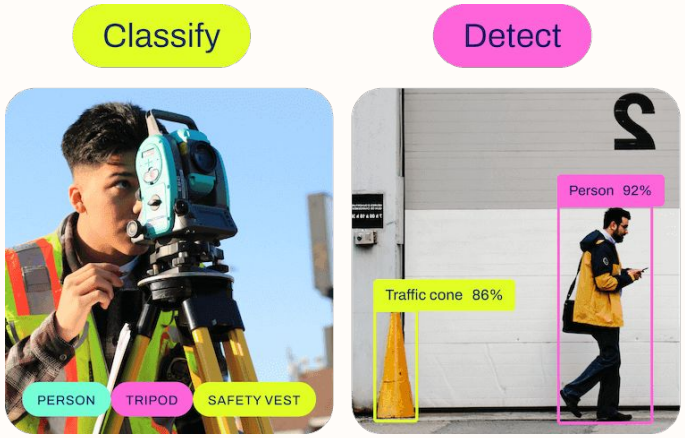
1. Ordenar os bounding boxes de acordo com seu valor de confiança
2. Selecionar os bounding boxes com maiores scores
3. Remover bounding boxes com alto valor de IoU
4. Repetir os passos 2-3 até não restar bounding boxes a serem comparados

YOLO - NMS (Algoritmo)

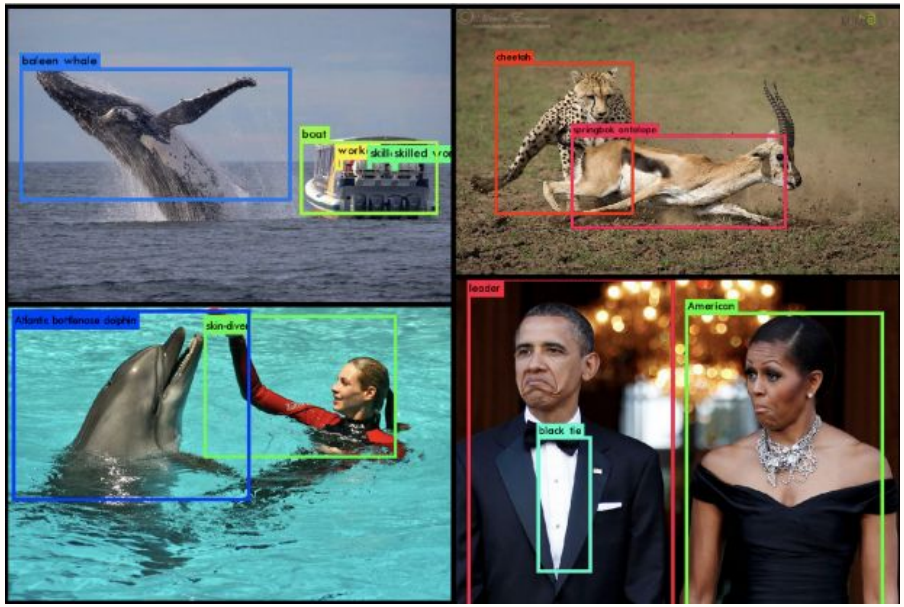
➔ Non-Maximum Suppression (NMS) é utilizada para eliminar detecções redundantes

```python
def iou(box1, box2):⋯

def nms(bounding_boxes, confidence_threshold=.8, iou_threshold=.5):
    # bounding box: x1, y1, x2, y2, confidence

    bounding_boxes = sorted(bounding_boxes, reverse=True, key = lambda x: x[5])

    bbox_filtered = [box for box in bounding_boxes if box[5] > confidence_threshold]

    has_box = True

    bbox_supressed = list()

    while has_box:
        current_box = bbox_filtered.pop(0)

        bbox_supressed.append(current_box)

        for box in bbox_filtered:
            iou_value = iou(current_box[:4], box[:4])
            if iou_value > iou_threshold: bbox_filtered.remove(box)

        if len(bbox_filtered) == 0: has_box = False

    return bbox_supressed
```

YOLO Aplicações



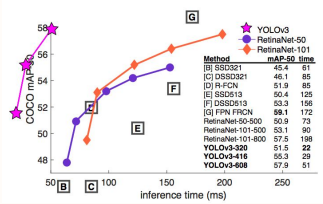
Backbone	Top-1	Top-5	Bn Ops	BFLOP/s	FPS
Darknet-19 [15]	74.1	91.8	7.29	1246	**171**
ResNet-101[5]	77.1	93.7	19.7	1039	53
ResNet-152 [5]	**77.6**	**93.8**	29.4	1090	37
Darknet-53	77.2	**93.8**	18.7	**1457**	78



- ➔ Utilizado em diversas aplicações de visão (detecção veículos, pessoas, objetos)
- ➔ Eficiente (rápido e leve)

Evolução YOLO

YOLOv1: Concepção da arquitetura YOLO, popularizado por suas características (e.g., singleshot, rápido e preciso); Disponibilização do darknet e cfgs



YOLOv3: Melhoria da performance do modelo, utilizando múltiplas anchors e spatial pyramid pooling; Último modelo implementado na darknet;

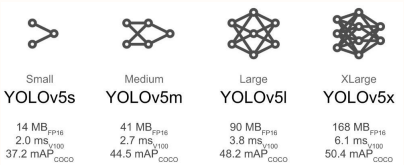
YOLOS: Integração de transformers e detecção sequencial

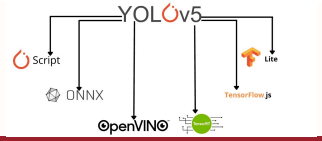
YOLOv8
Atualização do modelo v5 da ultralytics; Framework unificado para treinamento e avaliação do modelo; integração com principais ferramentas de exp. tracking;
YOLO-NAS: Utilização do NAS para desing da arquitetura; Novas estrategias de treinamento e quantização do modelo;

2015 — 2016 — 2018 — 2020 — 2021 — 2022 — 2023

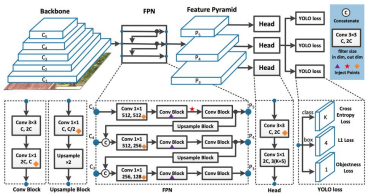
YOLOv2, YOLO9000: Melhoria do modelo original, incorporando os conceitos de batch normalization, anchor boxes e clusters de dimensão

YOLOv4: Novas estratégias utilizando mosaic augmentation, nova função de loss e a eliminação de anchors;
YOLOv5: Esforço de integração com otimização de hiperparâmetros, experiment tracking e conversão para diferentes formatos dos frameworks mais populares

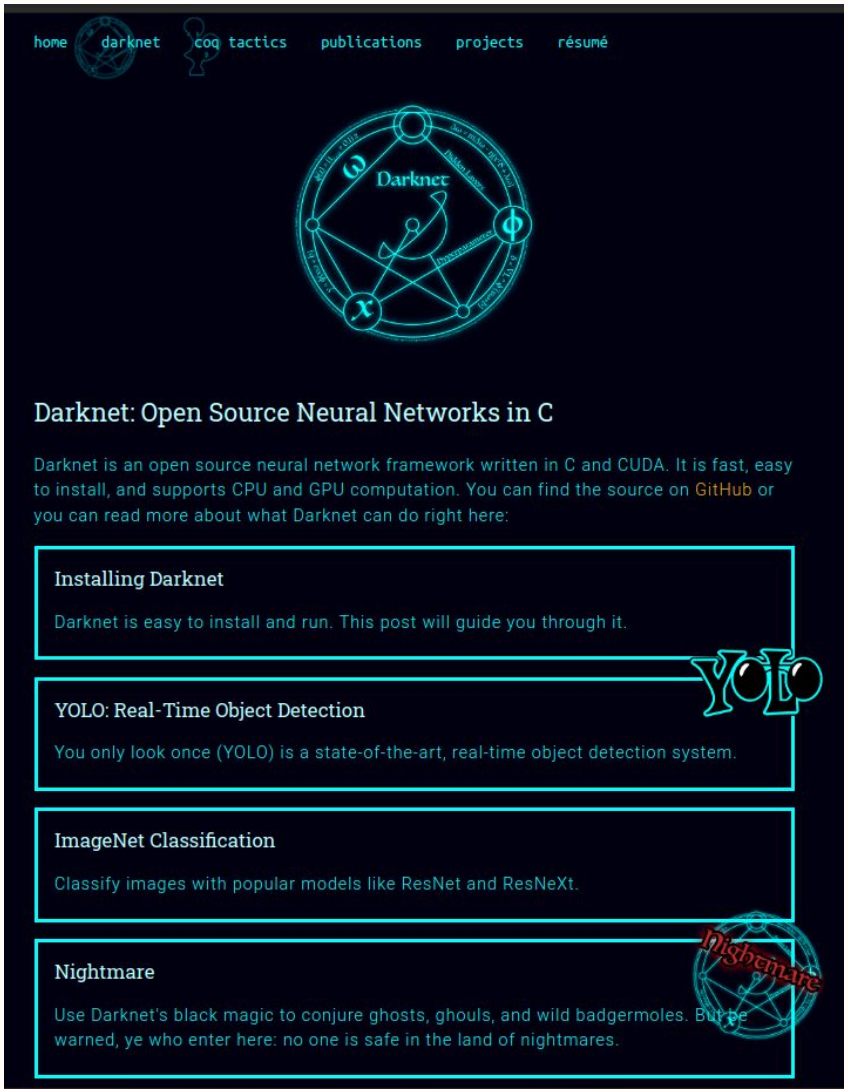


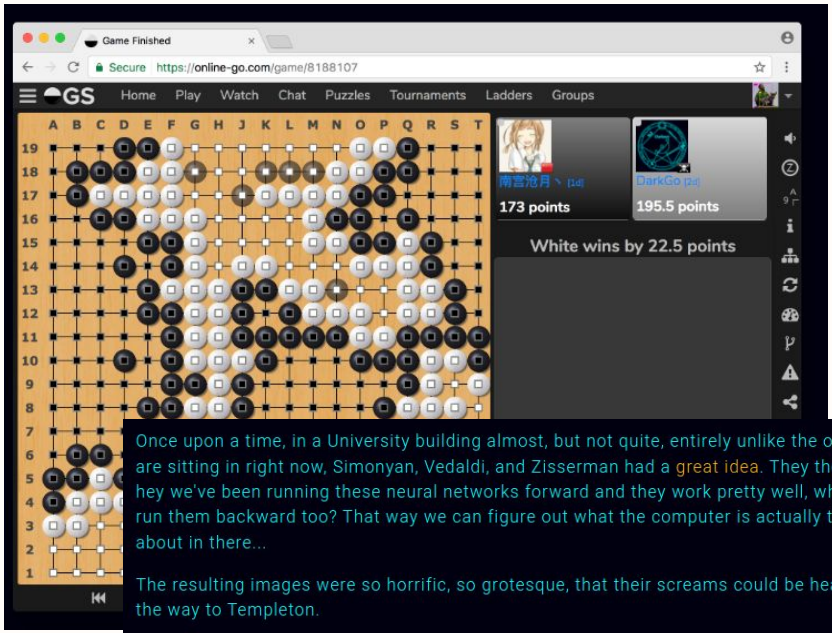


YOLOv6, YOLOv7: Extensão do modelos para novas tasks como aplicação em robótica e estimação de pose.



DarkNet (2015 - 2018)





Once upon a time, in a University building almost, but not quite, entirely unlike the one you are sitting in right now, Simonyan, Vedaldi, and Zisserman had a great idea. They thought, hey we've been running these neural networks forward and they work pretty well, why not run them backward too? That way we can figure out what the computer is actually thinking about in there...

The resulting images were so horrific, so grotesque, that their screams could be heard all the way to Templeton.

https://pjreddie.com/darknet/
https://online-go.com/user/view/434218

DarkNet - Arquivos CFG

➔ Definição dos parâmetros de treino, camadas do modelo e valores de threshold

```
[net]
# how many images are in each bat
batch=1
# into how many sub-batches shall
subdivisions=1
# input size of the network
height=448
width=448
channels=3
# learning parameters
momentum=0.9
decay=0.0005
b_debug=0

# base learning rate
learning_rate=0.001
# change learning rate after the
policy=steps
# need to have as many steps as s
steps=200,400,600,20000,30000
# re-scale the current learning r
scales=2.5,2,2,.1,.1
# max number of "iterations"
max_batches = 40000
# snapshow the learned weights af
i_snapshot_iteration=1000
#
#
c_ending_gt_files=.txt
```

```
# thereby, we can use the lea
#
[convolutional]
filters=64
size=7
stride=2
pad=1
activation=leaky

[maxpool]
size=2
stride=2

[convolutional]
filters=192
size=3
stride=1
pad=1
activation=leaky

[maxpool]
size=2
stride=2

[convolutional]
filters=128
size=1
stride=1
pad=1
activation=leaky
```

```
[connected]
#needs to be side*side*((1 + l.coord
# l.coords -- 4 coordinates to speci
# l.n -- how many bounding boxes are
# e.g., 20 classes, side 7 -> 1470
# e.g., 20 class, side 9 -> 2430
output= 1470
activation=linear

[detection]
# 20 for pascal voc (l.classes)
classes=20
# bounding boxes -> 4 parameters (l.
coords=4
rescore=1
b_debug=1
# number of cell in x and y directio
side=7
# number of predicted boxes per cell
num=2
softmax=0
# adapt loss function on bounding bo
sqrt=1
jitter=.2

# relative weights for the combined
object_scale=1
noobject_scale=.5
class_scale=1
coord_scale=5
```

YOLO9000 (YOLOv2) - Arquitetura (DarkNet19)

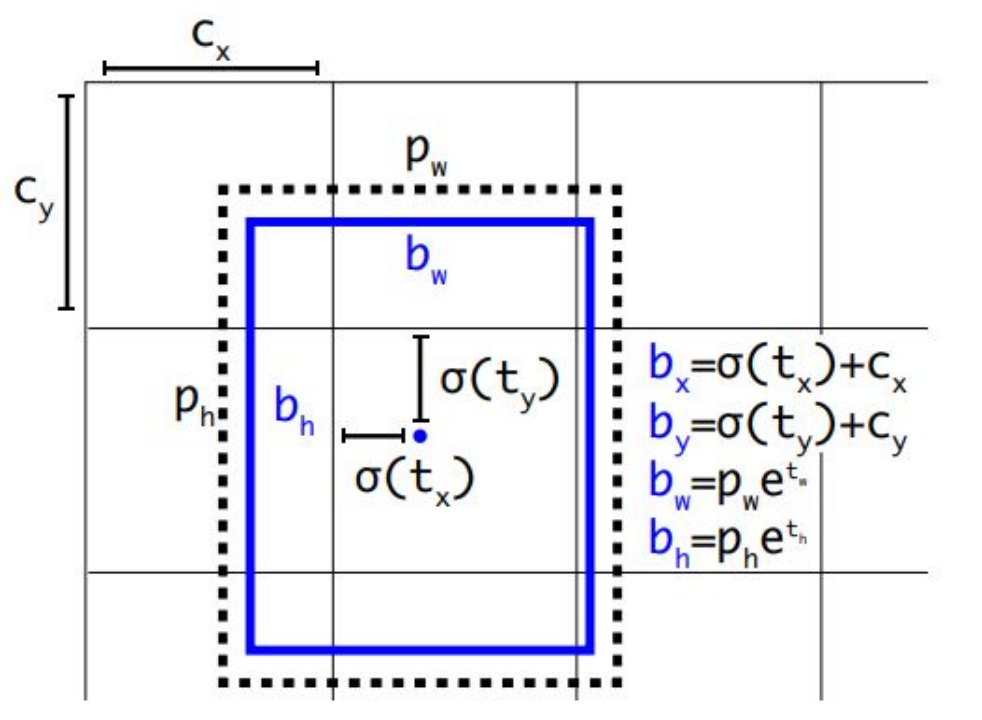
Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

- ➔ YOLO9000: Better, Faster, Stronger
  - Batch Normalization
  - Variação de resolução no treino (Multi-scale training)
  - Anchors boxes
  - Dimension clusters

https://arxiv.org/pdf/1612.08242.pdf

YOLO9000 (YOLOv2)

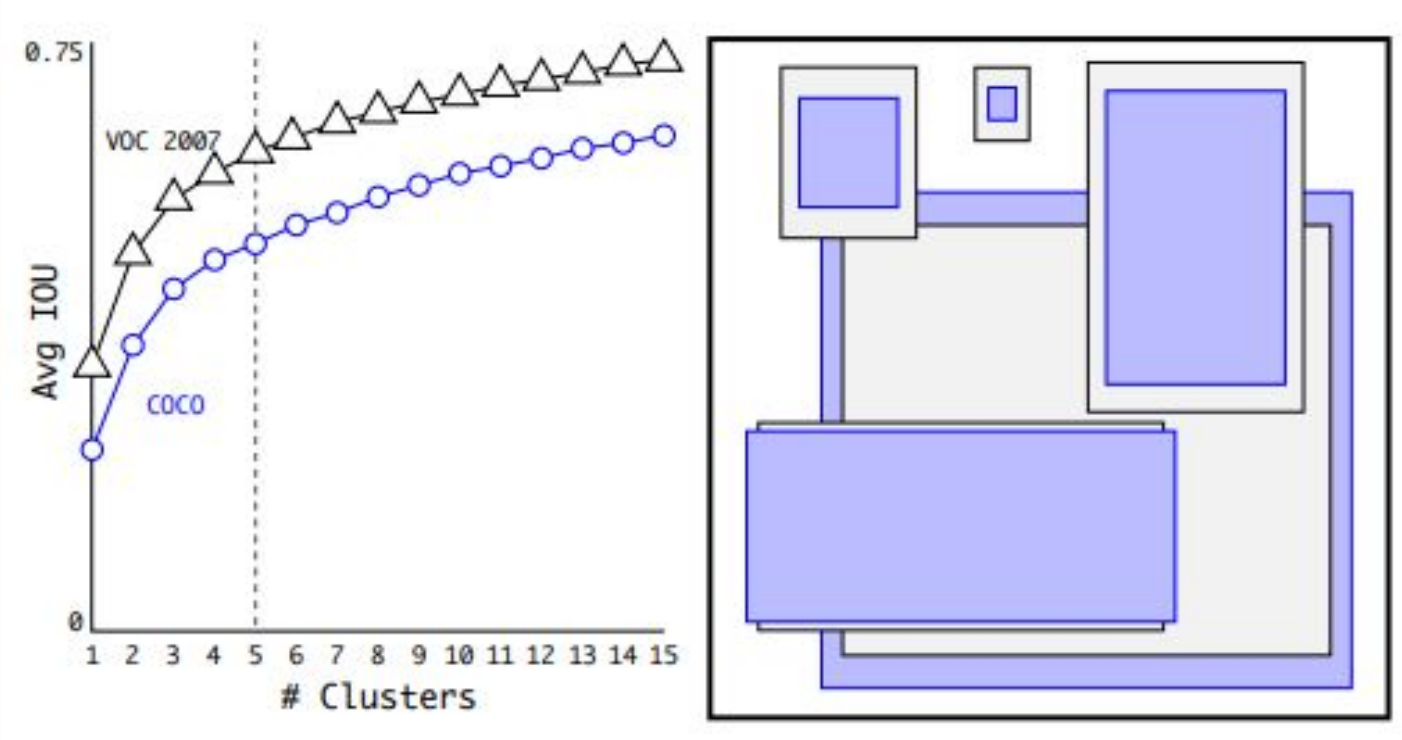
- ➔ A predição é realizada através da estimação do ponto central e dos valores que definem a dimensão do objeto na posição avaliada
- ➔ A rede retorna N*B predições, onde N é o valor de âncoras previamente definido e B o número de predições máxima realizada em cada célula/posição da imagem



$$x = (t_x * w_a) - x_a$$
$$y = (t_y * h_a) - y_a$$

YOLO9000 (YOLOv2) - Clustering Dimensions

- ➔ K-means é utilizado no conjunto de treino para gerar grupos de bounding boxes com dimensões similares
  - ◆ Número de grupos define dimensões válidas de referência para as âncoras



YOLOv3: An incremental Improvement

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
	Convolutional	32	1 × 1	
1×	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
	Convolutional	64	1 × 1	
2×	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
	Convolutional	128	1 × 1	
8×	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
	Convolutional	256	1 × 1	
8×	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
	Convolutional	512	1 × 1	
4×	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Table 1. **Darknet-53.**

Backbone	Top-1	Top-5	Bn Ops	BFLOP/s	FPS
Darknet-19 [15]	74.1	91.8	7.29	1246	**171**
ResNet-101[5]	77.1	93.7	19.7	1039	53
ResNet-152 [5]	**77.6**	**93.8**	29.4	1090	37
Darknet-53	77.2	**93.8**	18.7	**1457**	78

Abstract

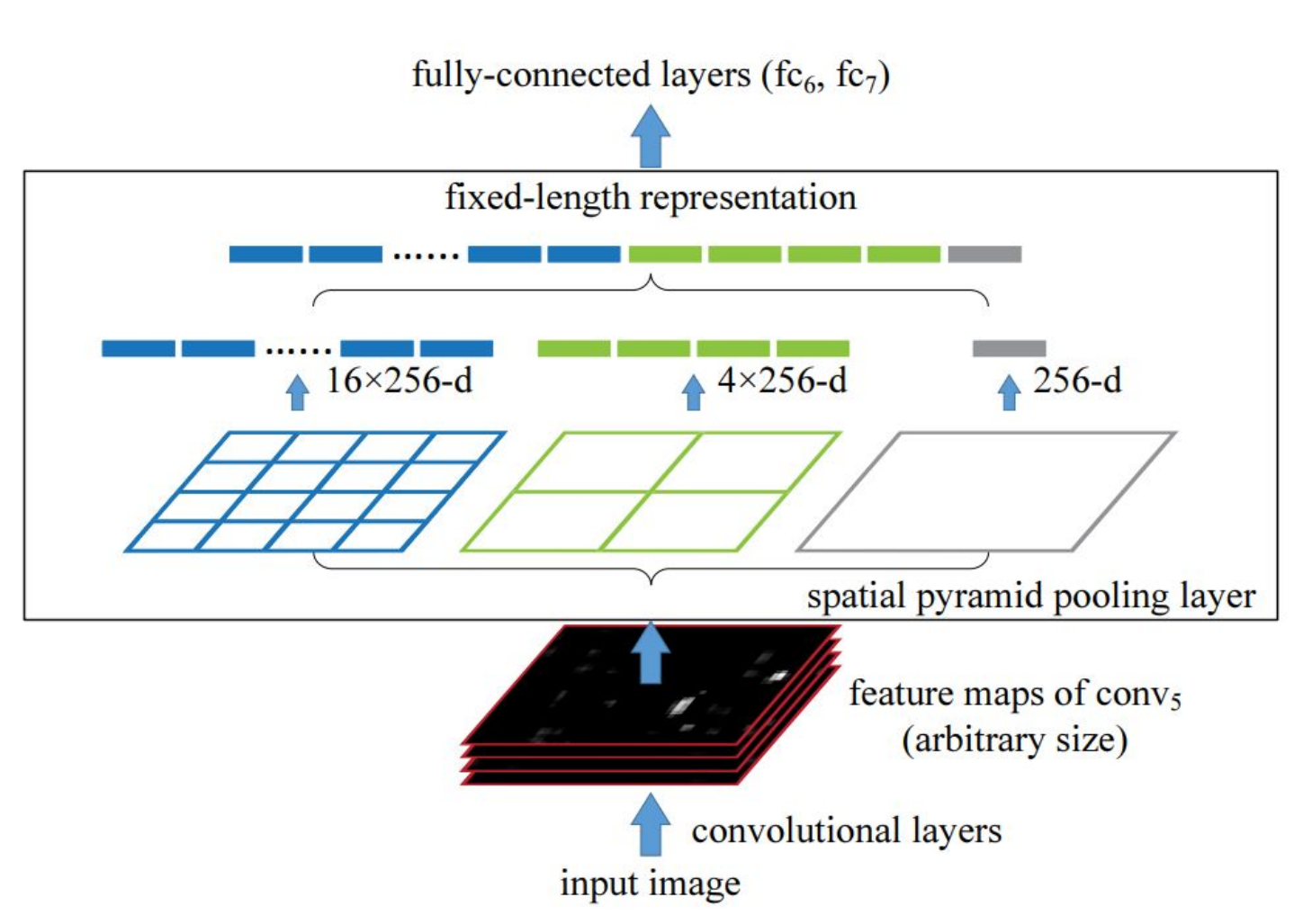
We present some updates to YOLO! We made a bunch of little design changes to make it better. We also trained this new network that's pretty swell. It's a little bigger than last time but more accurate. It's still fast though, don't worry. At 320 × 320 YOLOv3 runs in 22 ms at 28.2 mAP, as accurate as SSD but three times faster. When we look at the old .5 IOU mAP detection metric YOLOv3 is quite good. It achieves 57.9 AP_{50} in 51 ms on a Titan X, compared to 57.5 AP_{50} in 198 ms by RetinaNet, similar performance but 3.8× faster. As always, all the code is online at https://pjreddie.com/yolo/.

2. The Deal

So here's the deal with YOLOv3: We mostly took good ideas from other people. We also trained a new classifier network that's better than the other ones. We'll just take you through the whole system from scratch so you can understand it all.

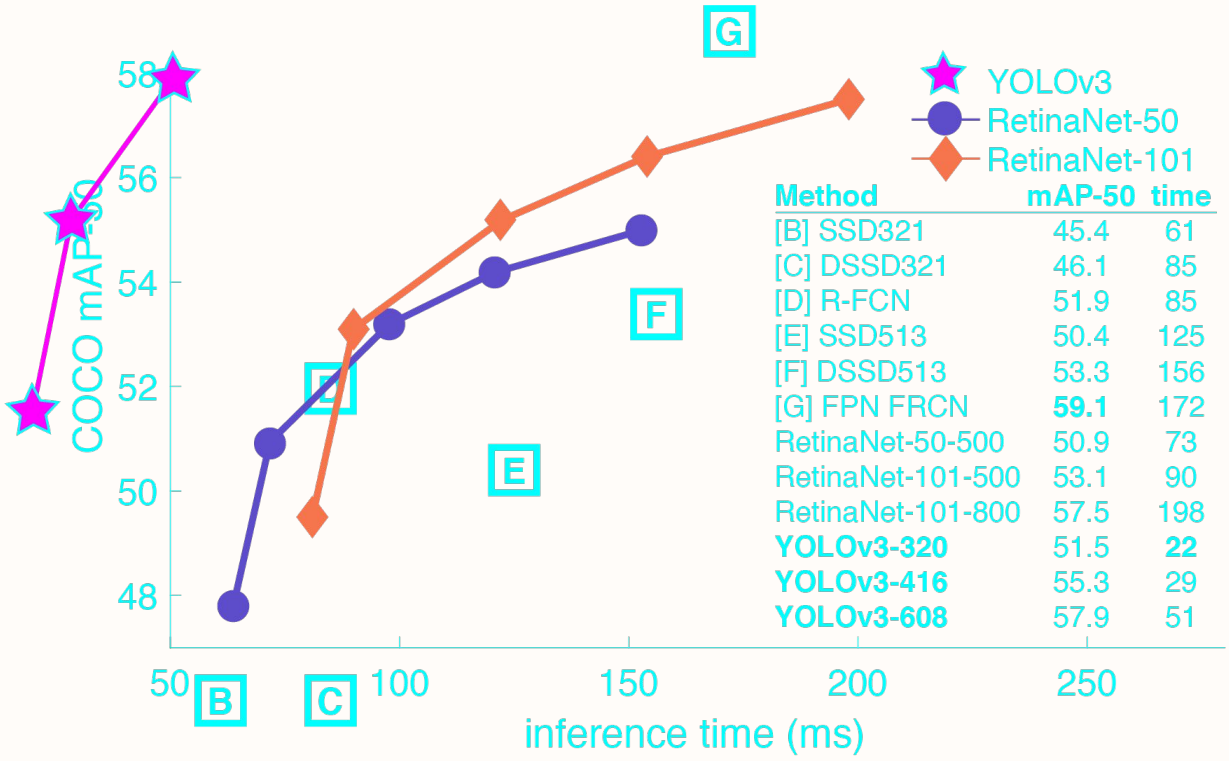
https://pjreddie.com/media/files/papers/YOLOv3.pdf

SPFF



Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition (https://arxiv.org/pdf/1406.4729v4.pdf)

YOLOv3



Method	mAP-50	time
[B] SSD321	45.4	61
[C] DSSD321	46.1	85
[D] R-FCN	51.9	85
[E] SSD513	50.4	125
[F] DSSD513	53.3	156
[G] FPN FRCN	**59.1**	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	**22**
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51

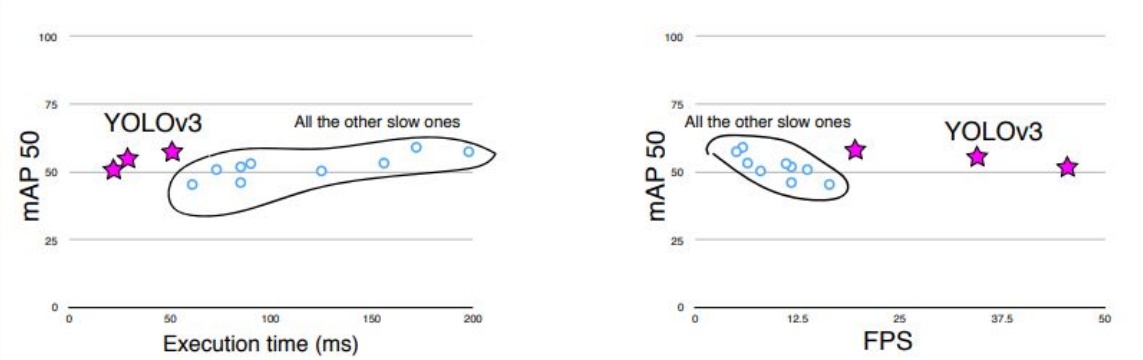
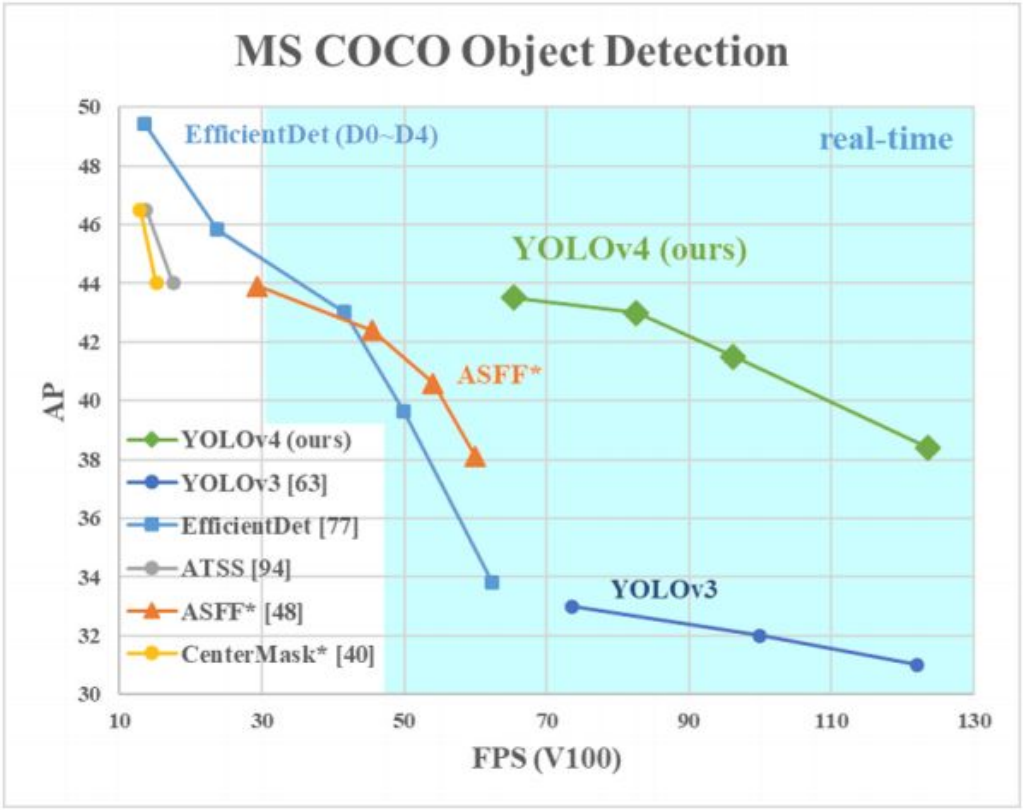


Figure 4. Zero-axis charts are probably more intellectually honest... and we can still screw with the variables to make ourselves look good!

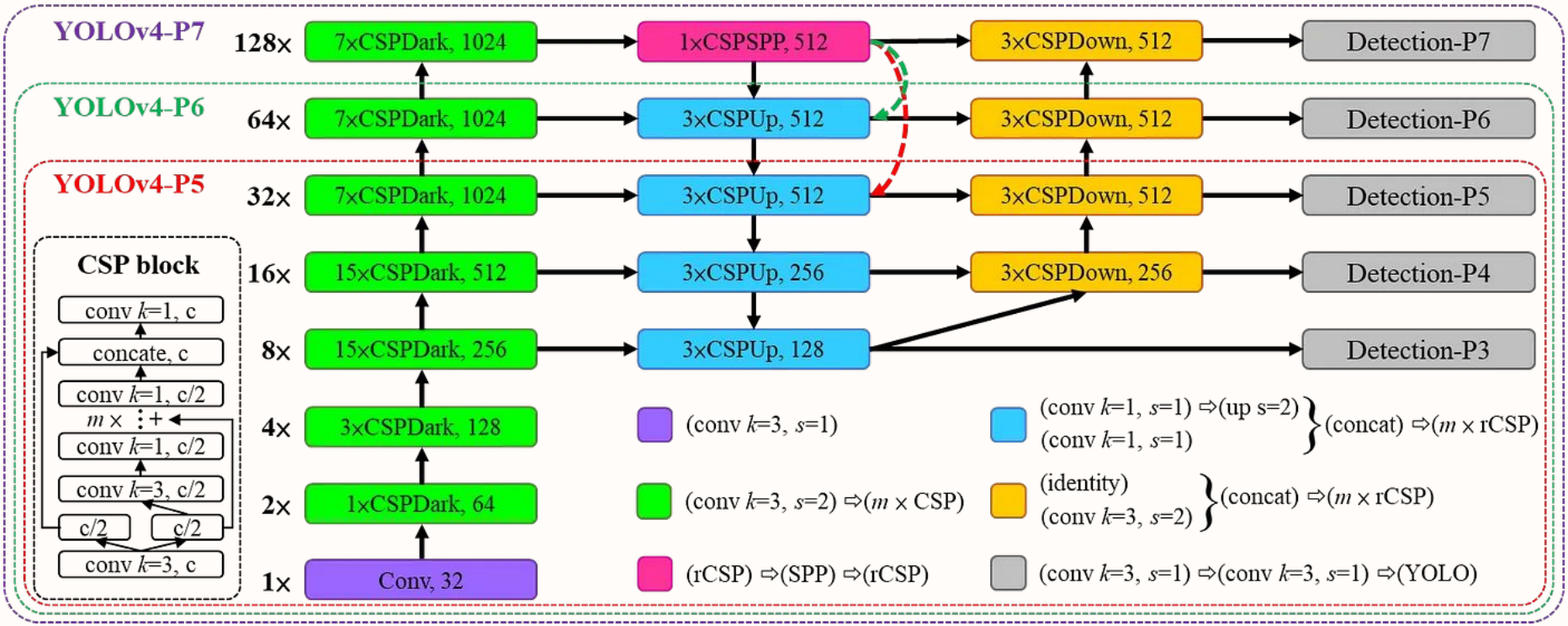
YOLOv4

- ➔ Novo backbone CSPDarknet53, melhorando acurácia e performance
- ➔ YOLOv4 otimizado para melhorar o uso de recursos (bom para edges device)
- ➔ Otimização de hiperparâmetros com algoritmos genéticos, novas estratégias de augmentation



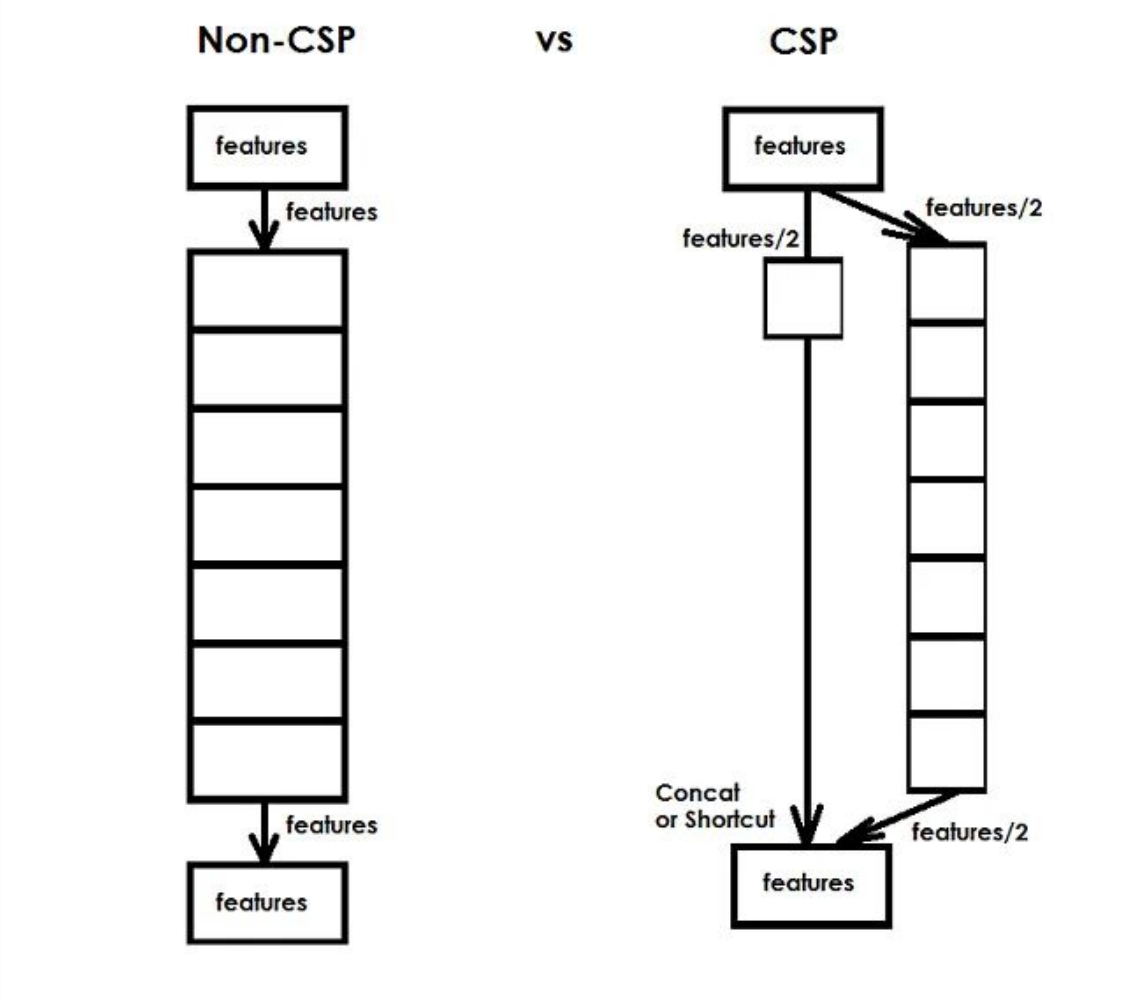
YOLOv4: Optimal Speed and Accuracy for object detection
https://arxiv.org/pdf/2004.10934.pdf

YoloV4



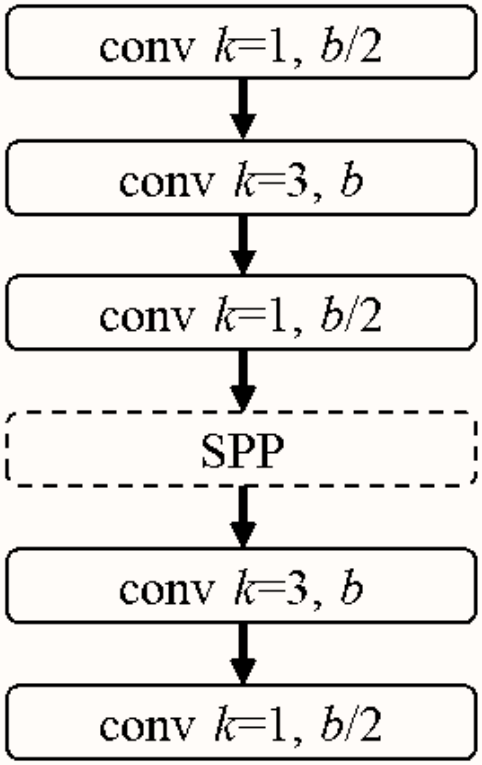
https://alexeyab84.medium.com/scaled-yolo-v4-is-the-best-neural-network-for-object-detection-on-ms-coco-dataset-39dfa22fa982

Yolov4

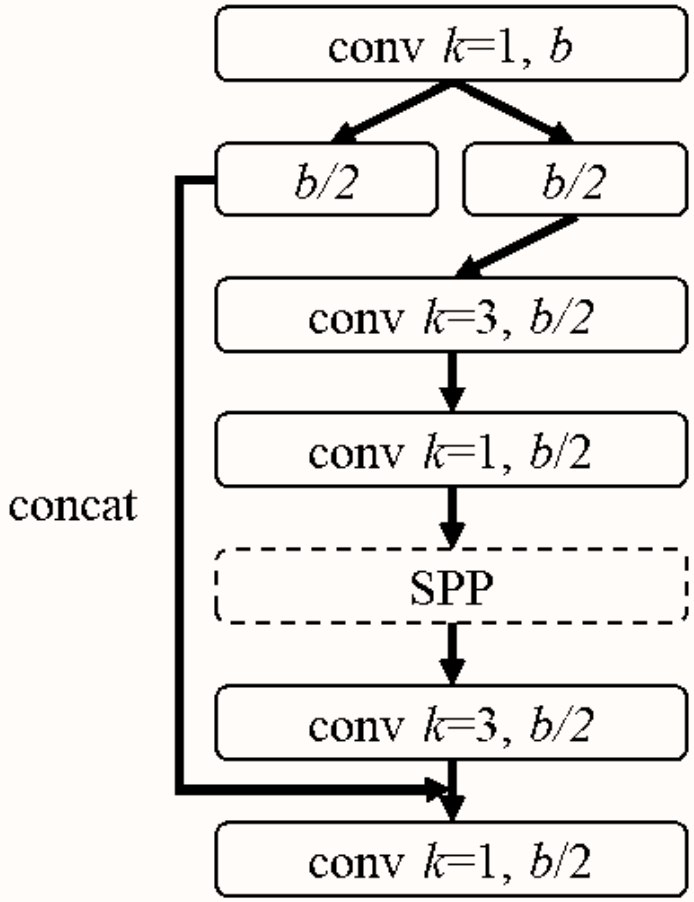


https://alexeyab84.medium.com/scaled-yolo-v4-is-the-best-neural-network-for-object-detection-on-ms-coco-dataset-39dfa22fa982

Yolov4



(a) reversed dark layers (SPP)



(b) reversed CSP dark layers (SPP)

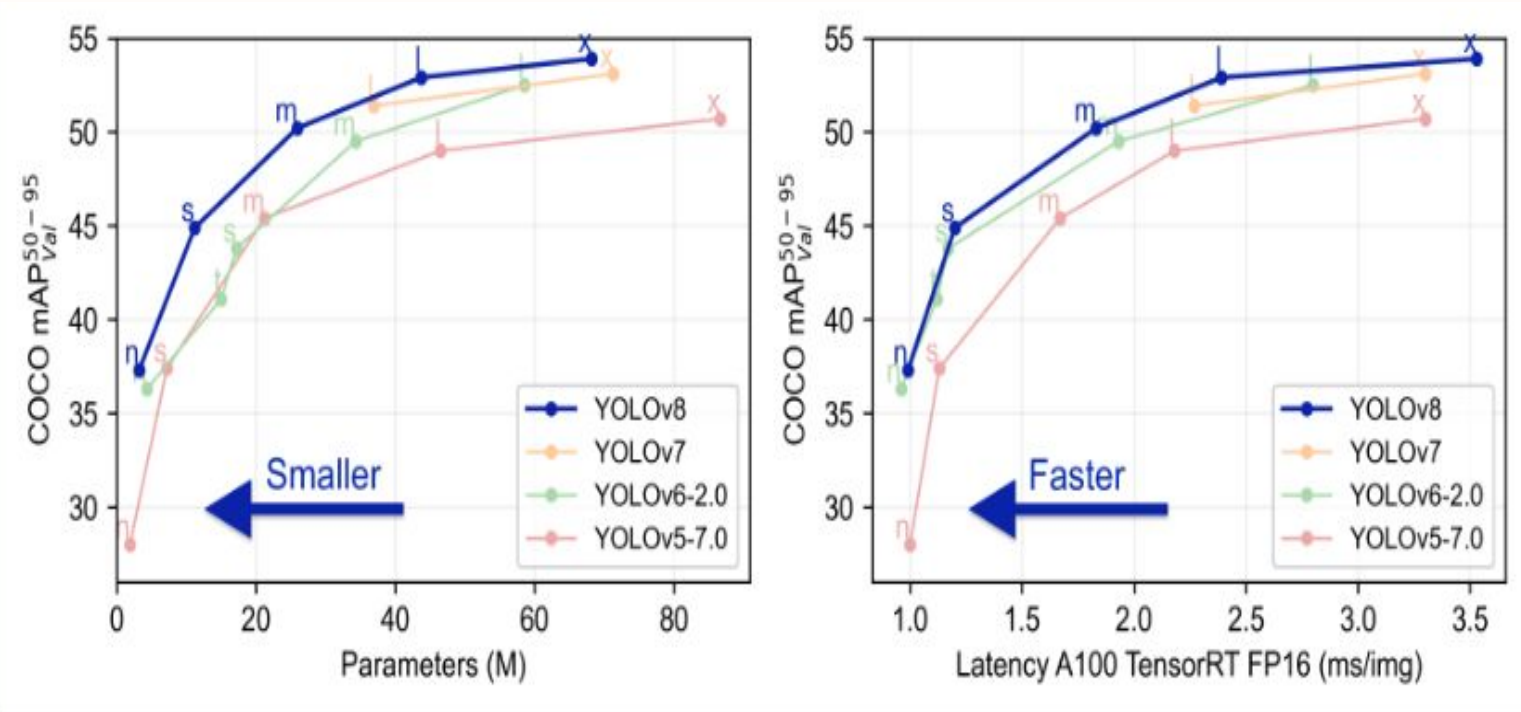
https://alexeyab84.medium.com/scaled-yolo-v4-is-the-best-neural-network-for-object-detection-on-ms-coco-dataset-39dfa22fa982

YOLOv5, YOLOv8

Ultralytics YOLOv8 is a cutting-edge, state-of-the-art (SOTA) model that builds upon the success of previous YOLO versions and introduces new features and improvements to further boost performance and flexibility. YOLOv8 is designed to be fast, accurate, and easy to use, making it an excellent choice for a wide range of object detection and tracking, instance segmentation, image classification and pose estimation tasks.

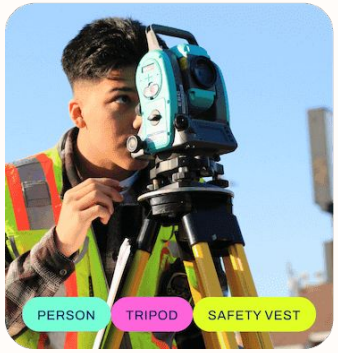
We hope that the resources here will help you get the most out of YOLOv8. Please browse the YOLOv8 Docs for details, raise an issue on GitHub for support, and join our Discord community for questions and discussions!

To request an Enterprise License please complete the form at Ultralytics Licensing.

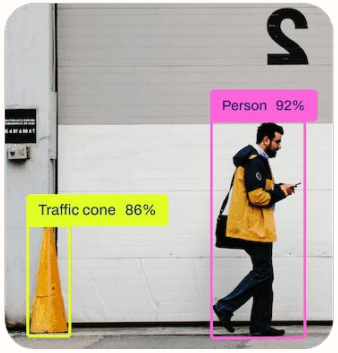


YOLOv5, YOLOv8

Classify

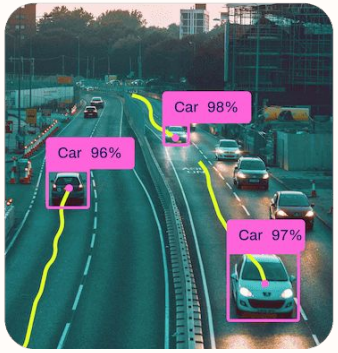


Detect

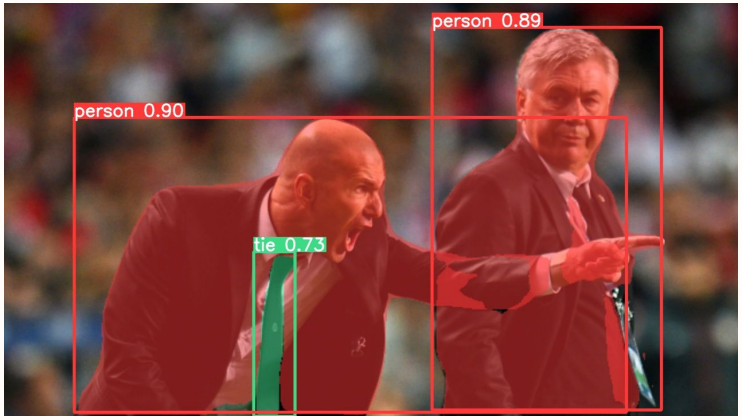


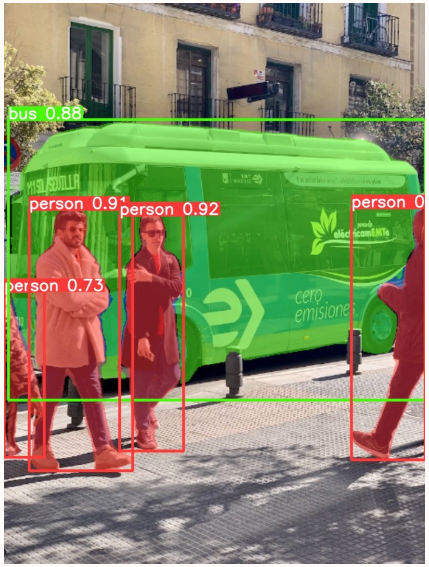
Segment

Track

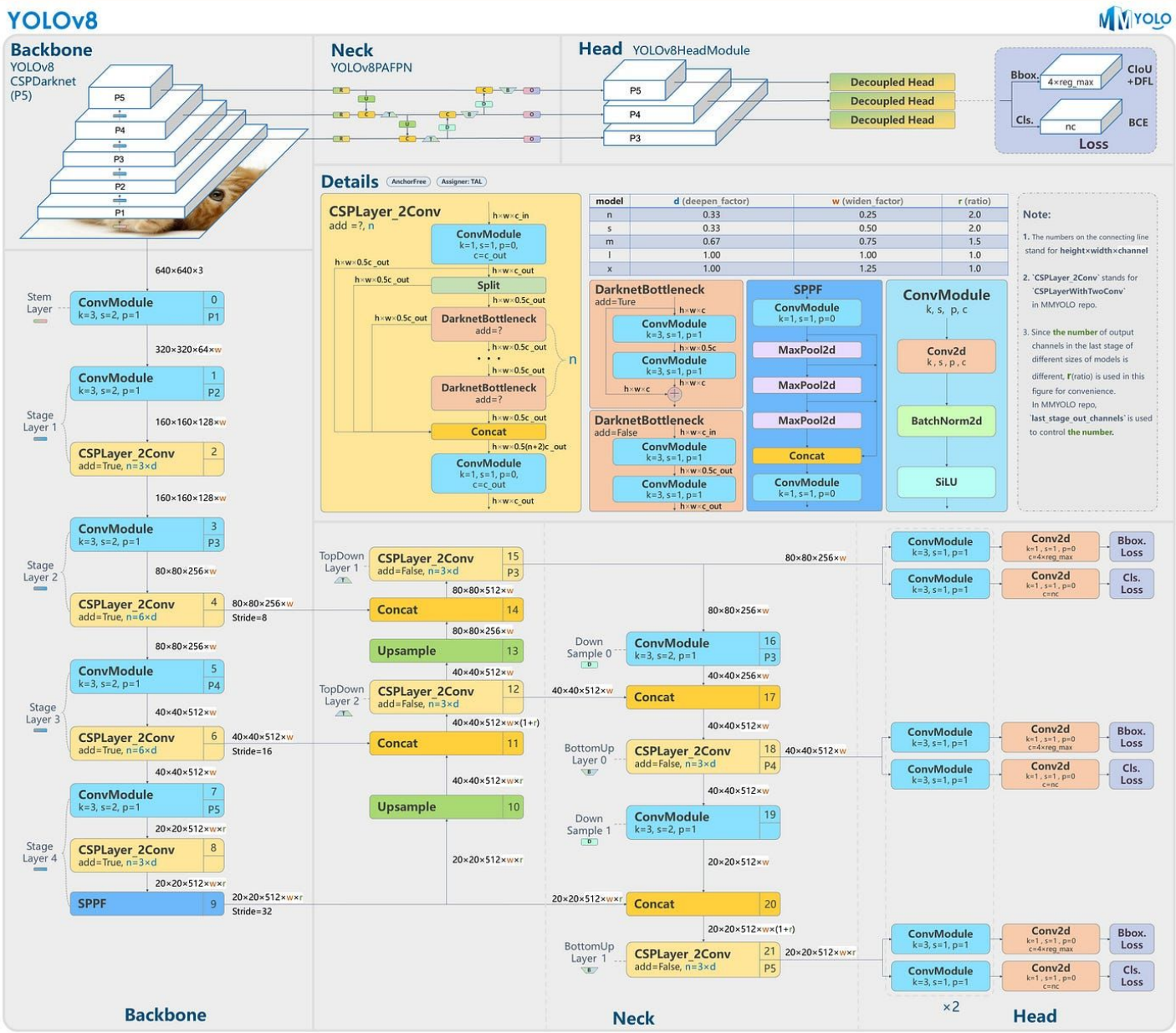


Pose

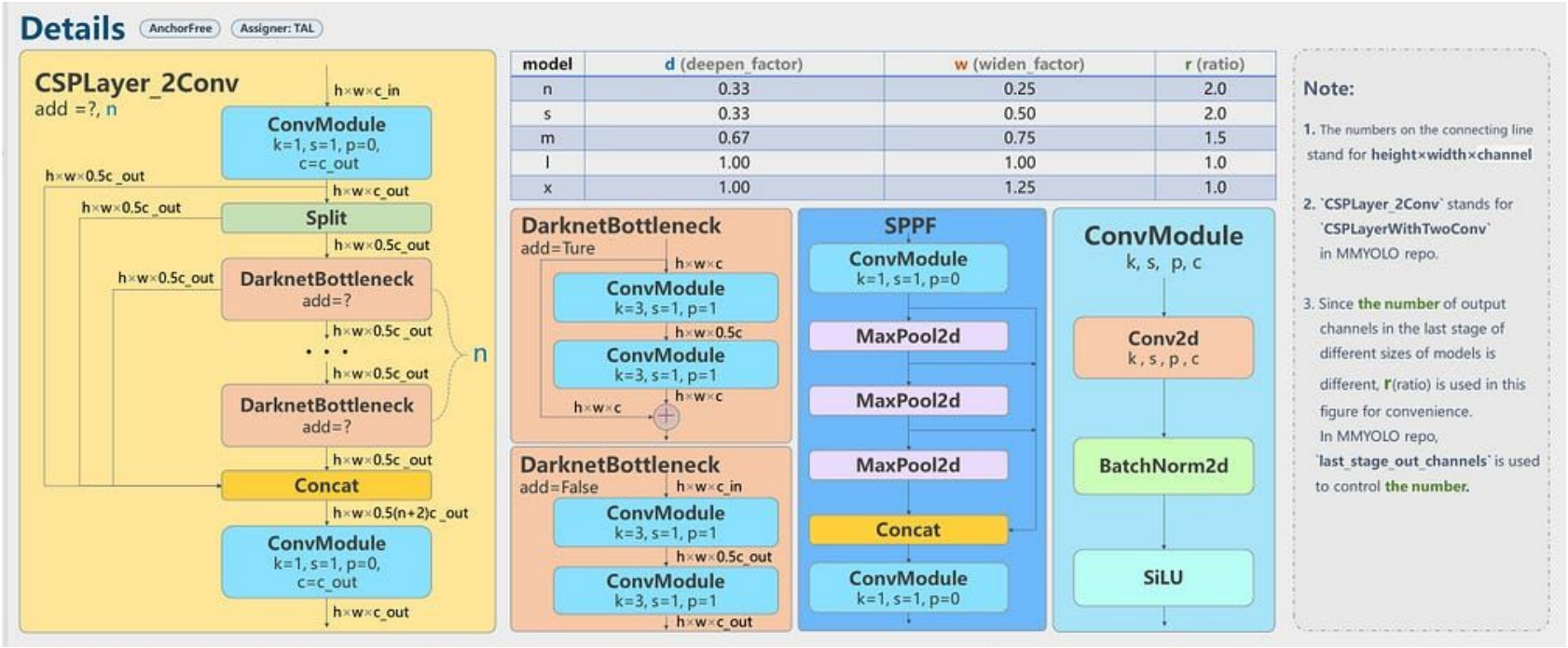




YOLOv8

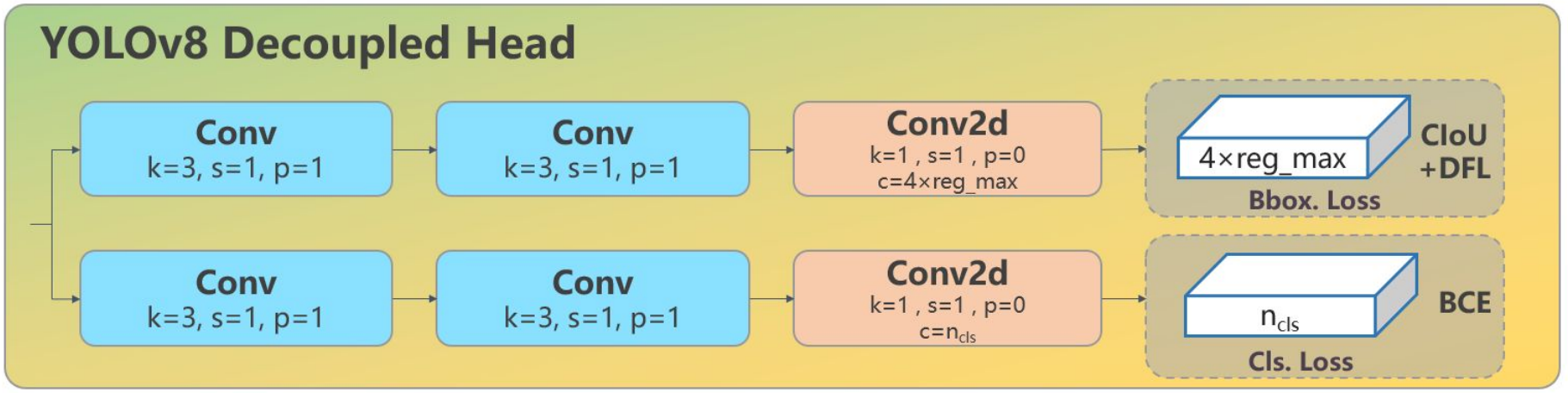

YOLOv8
Backbone
YOLOv8
CSPDarknet
(P5)
Neck
YOLOv8PAFPN
Head
YOLOv8HeadModule
Decoupled Head
Decoupled Head
Decoupled Head
Bbox.
4×reg_max
CIoU
+DFL
Cls.
nc
BCE
Loss
Details
AnchorFree
Assigner: TAL
CSPLayer_2Conv
add =?, n
ConvModule
k=1, s=1, p=0,
c=c_out
Split
DarknetBottleneck
add=?
Concat
n
model
d (deepen_factor)
w (widen_factor)
r (ratio)
n
0.33
0.25
2.0
s
0.33
0.50
2.0
m
0.67
0.75
1.5
l
1.00
1.00
1.0
x
1.00
1.25
1.0
Note:
1. The numbers on the connecting line stand for height×width×channel
2. `CSPLayer_2Conv` stands for `CSPLayerWithTwoConv` in MMYOLO repo.
3. Since the number of output channels in the last stage of different sizes of models is different, r(ratio) is used in this figure for convenience. In MMYOLO repo, `last_stage_out_channels` is used to control the number.
DarknetBottleneck
add=Ture
DarknetBottleneck
add=False
SPPF
MaxPool2d
Concat
ConvModule
k, s, p, c
Conv2d
k , s , p , c
BatchNorm2d
SiLU
640×640×3
Stem Layer
320×320×64×w
Stage Layer 1
160×160×128×w
Stage Layer 2
80×80×256×w
Stage Layer 3
40×40×512×w
Stage Layer 4
20×20×512×w×r
SPPF
Stride=8
Stride=16
Stride=32
TopDown Layer 1
TopDown Layer 2
Upsample
Down Sample 0
Down Sample 1
BottomUp Layer 0
BottomUp Layer 1
Bbox. Loss
Cls. Loss
×2
Backbone
Neck
Head

CSPLayer



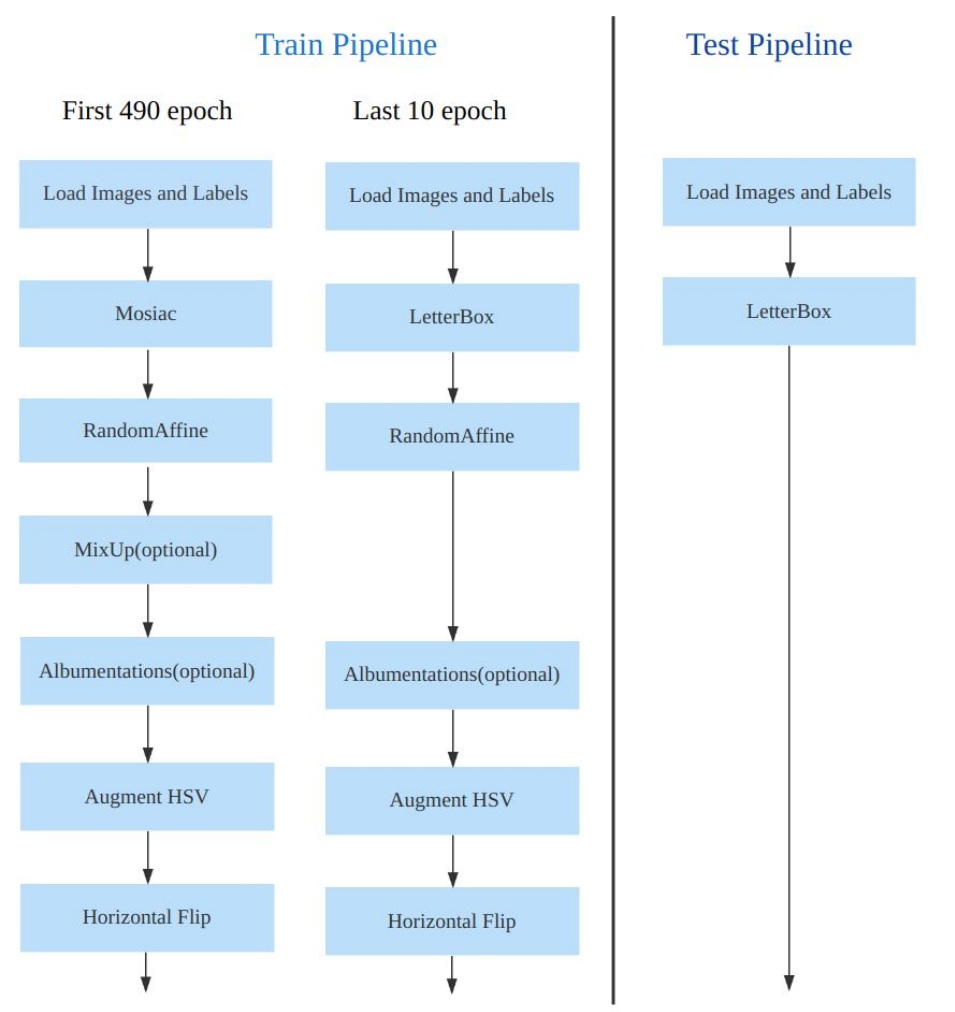
https://mmyolo.readthedocs.io/en/latest/recommended_topics/algorithm_descriptions/yolov8_description.html#yolov8-overview

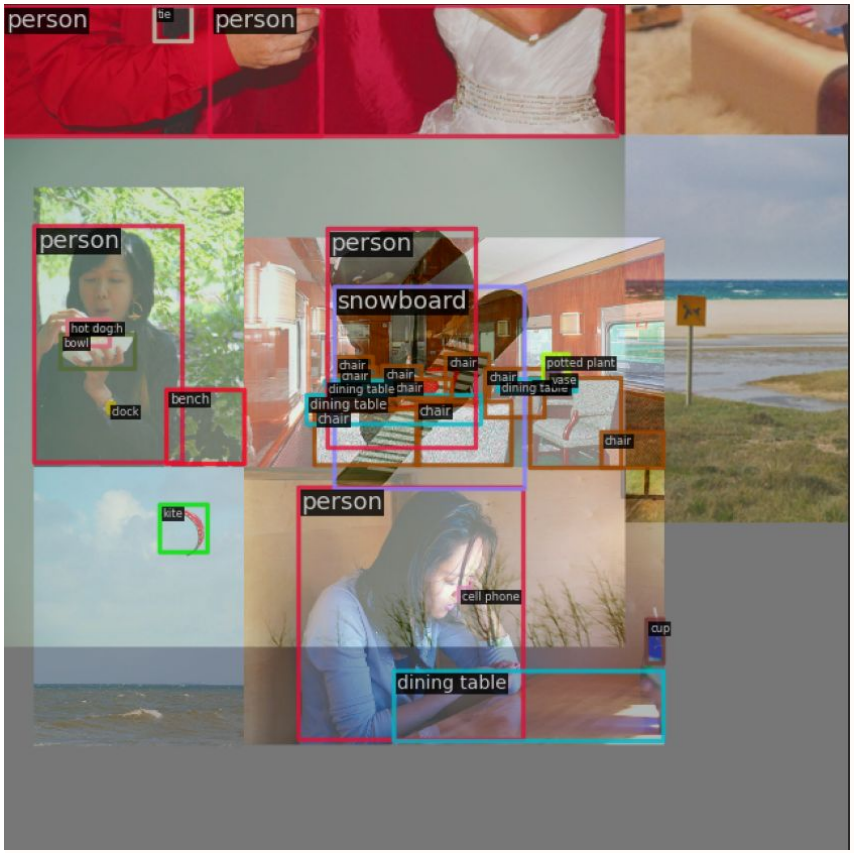
Decoupled Head



https://mmyolo.readthedocs.io/en/latest/recommended_topics/algorithm_descriptions/yolov8_description.html#yolov8-overview

Augmentations



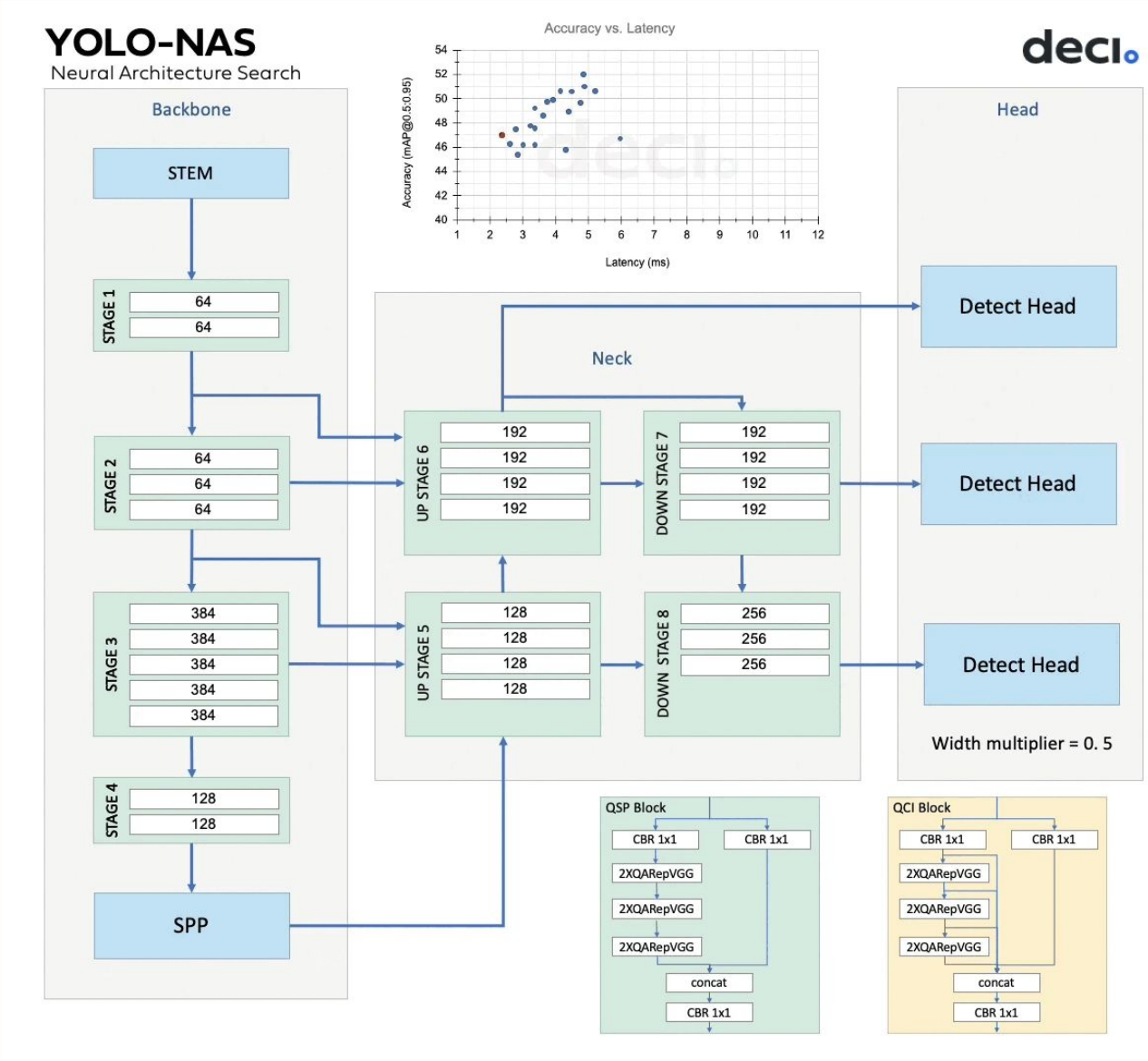


https://mmyolo.readthedocs.io/en/latest/recommended_topics/algorithm_descriptions/yolov8_description.html#yolov8-overview

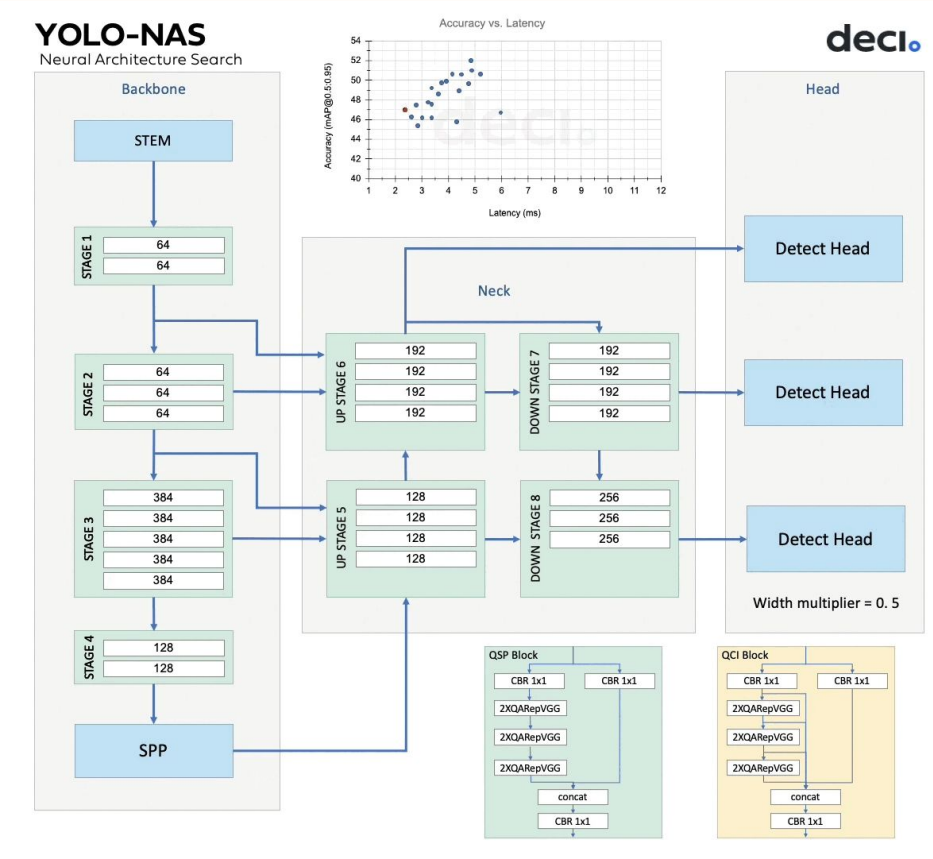
YOLOv8

Model	size (pixels)	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

YOLO-NAS (SOTA Object Detection)



YOLO-NAS (SOTA Object Detection)

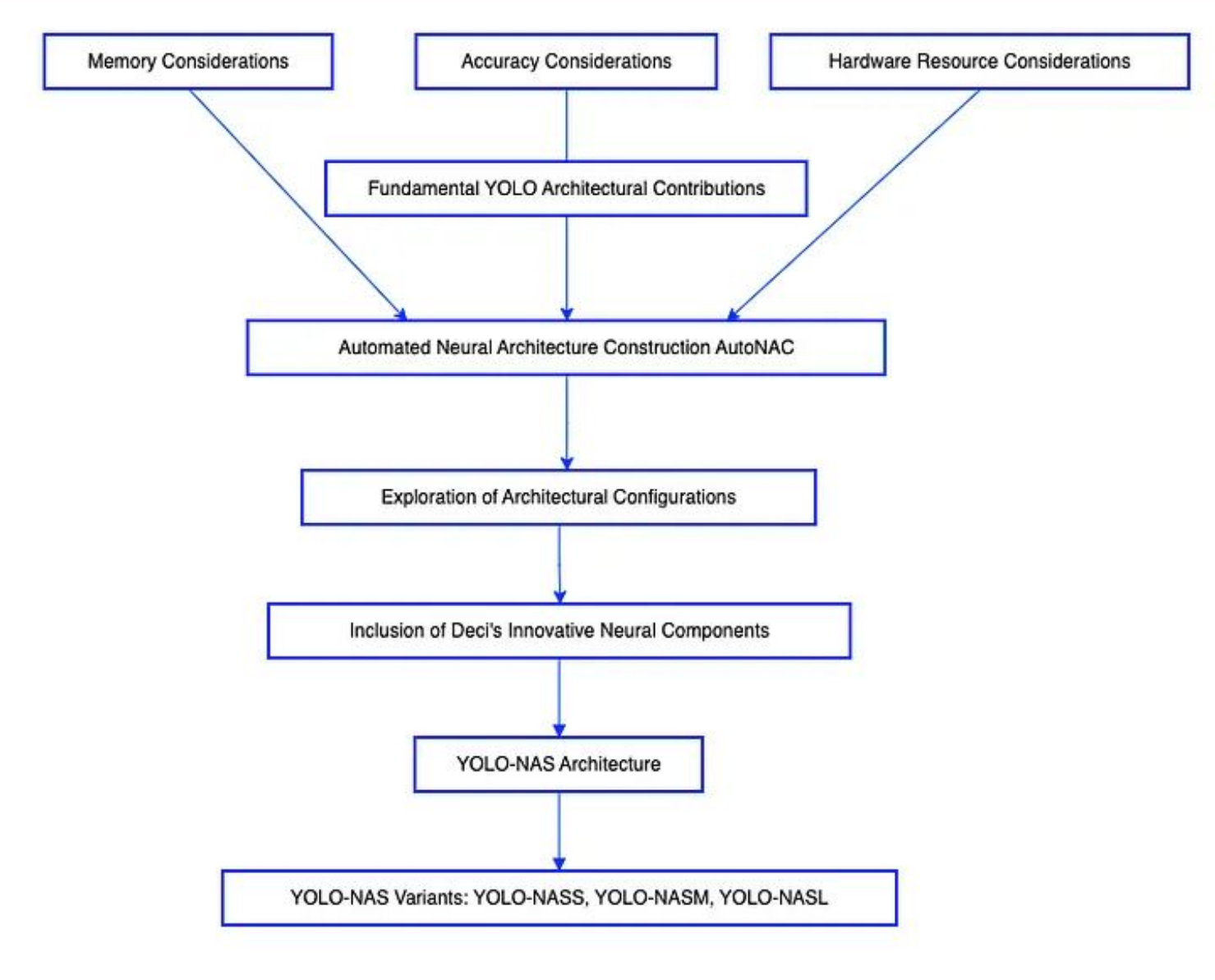


➔ Modelo proposto pela Deci, utilizando o autoNAC (tecnologia NAS proprietária que facilita a busca e design de modelos)

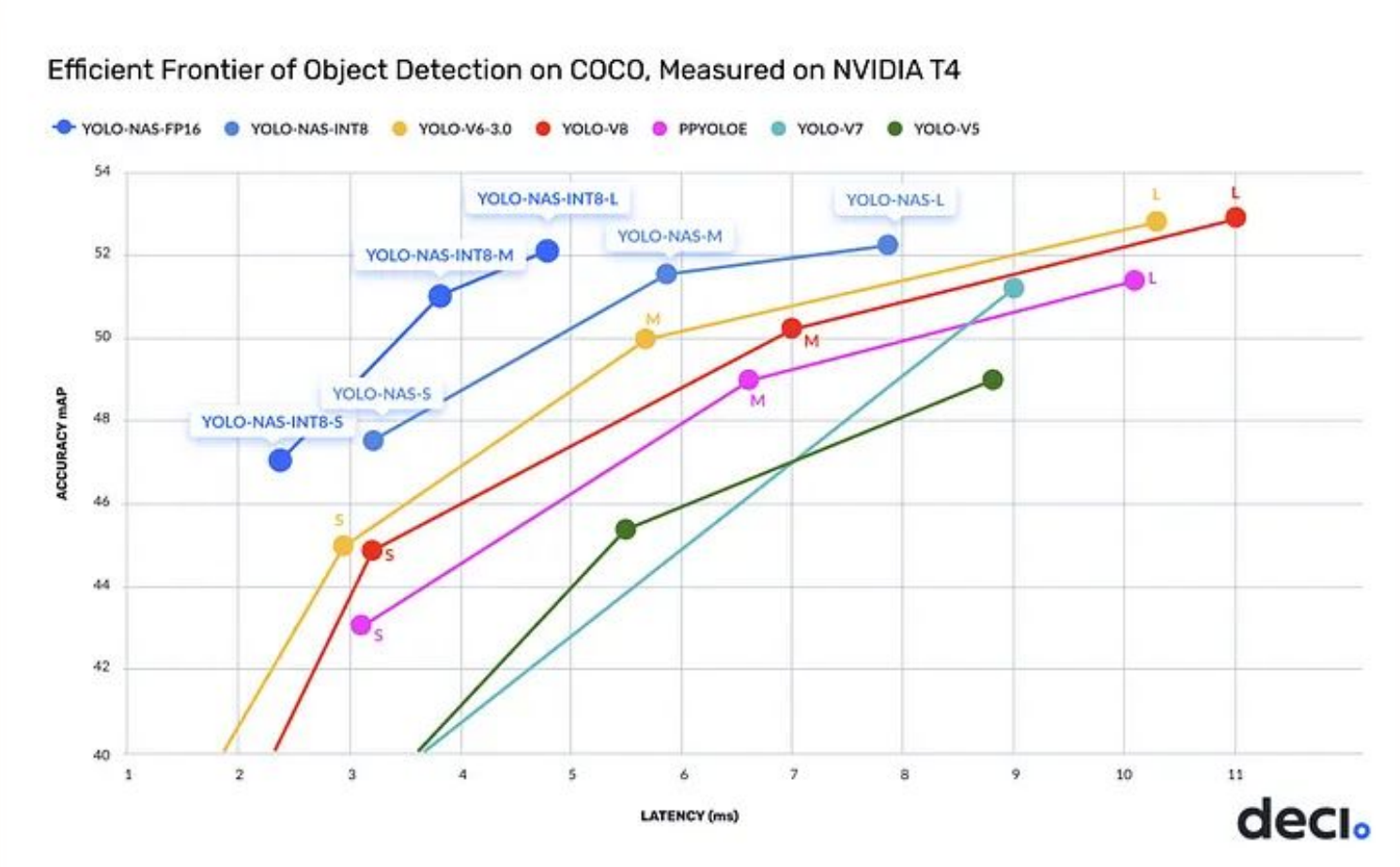
YOLO-NAS - Network Architecture Search

- ➔ NAS envolve o desenvolvimento e design de modelos de machine learning de forma automática, encontrando melhores configurações que superem o desempenho obtido pelo modelo original
  - ◆ Fatores de melhoria incluem tempo de inferência, recursos computacionais, complexidade da arquitetura e acurácia

YOLO-NAS - Network Architecture Search

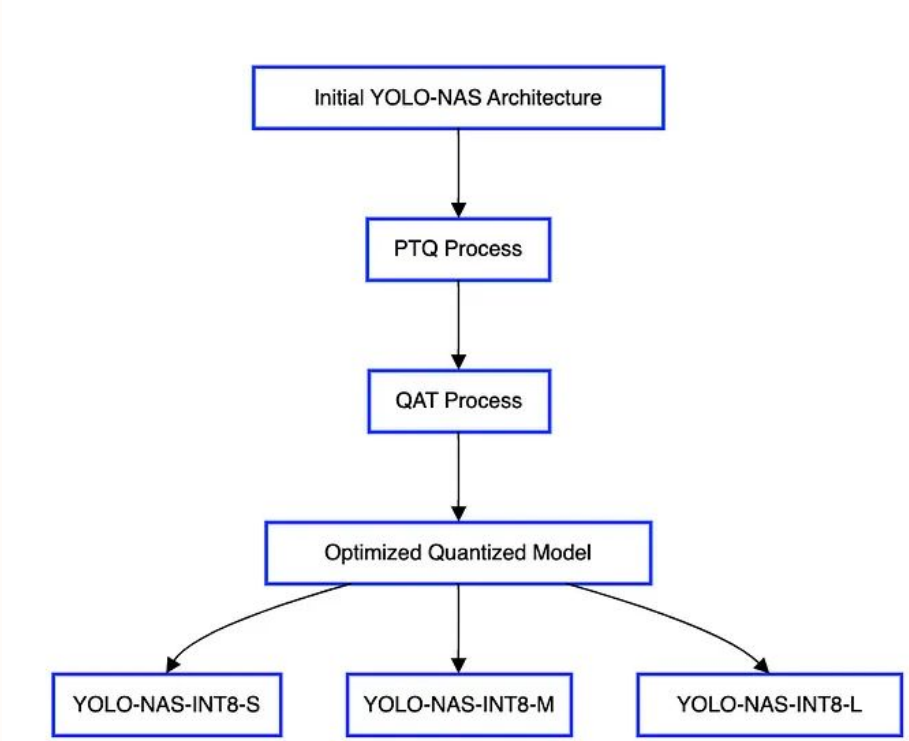


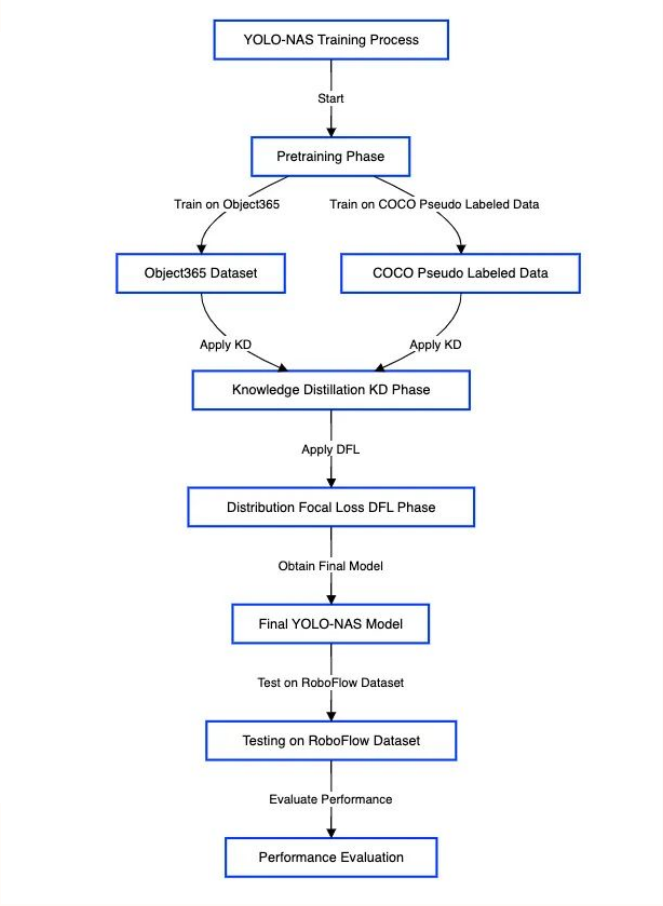
YOLO-NAS - Resultados e variações



YOLO-NAS: Treinamento e otimização

➔ A arquitetura da Deci também propõe contribuições para melhoria do treinamento com estratégias de labeling e quantization





YOLO architectures review

Juan R. Terven, Diana M. Cordova-Esparza. **A COMPREHENSIVE REVIEW OF YOLO: FROM YOLOV1 AND BEYOND**. Oct. 2023.

https://arxiv.org/pdf/2304.00501.pdf

Datasets Detecção 2D

Detecção de objetos - Principais datasets

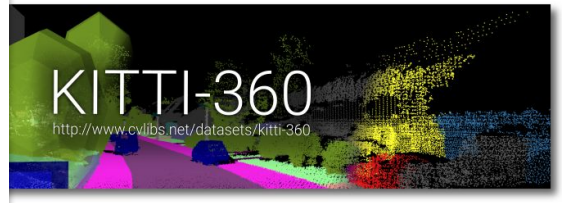
IMAGENET

14,197,122 images, 21841 synsets indexed

Home Download Challenges About

Not logged in. Login | Signup

ImageNet is an image database organized according to the WordNet hierarchy (currently only the nouns), in which each node of the hierarchy is depicted by hundreds and thousands of images. The project has been instrumental in advancing computer vision and deep learning research. The data is available for free to researchers for non-commercial use.



KITTI Vision Benchmark Suite!

driving platform Annieway to develop novel challenging real-world computer vision benchmarks. Our tasks of interest are: stereo, optical flow, visual odometry, 3D object detection and 3D tracking. For this purpose, we equipped a standard station wagon with two high-resolution color and grayscale video cameras. Accurate ground truth is provided by a Velodyne laser scanner and a GPS localization system. Our datasets are captured by driving around the mid-size city of Karlsruhe, in rural areas and on highways. Up to 15 cars and 30 pedestrians are visible per image. Besides providing all data in raw format, we extract benchmarks for each task. For each of our benchmarks, we also provide an evaluation metric and this evaluation website. Preliminary experiments show that methods ranking high on established benchmarks such as Middlebury perform below average when being moved outside the laboratory to the real world. Our goal is to reduce this bias and complement existing benchmarks by providing real-world benchmarks with novel difficulties to the community.

COCO
Common Objects in Context

info@cocodataset.org

Home People Dataset Tasks Evaluate

News

- We are pleased to announce the LVIS 2021 Challenge and Worksho
- Please note that there will not be a COCO 2021 Challenge, instead, participate in the LVIS 2021 Challenge.
- We have partnered with the team behind the open-source tool Fifty download, visualize, and evaluate COCO
- FiftyOne is an open-source tool facilitating visualization and access and serves as an evaluation tool for model analysis on COCO.

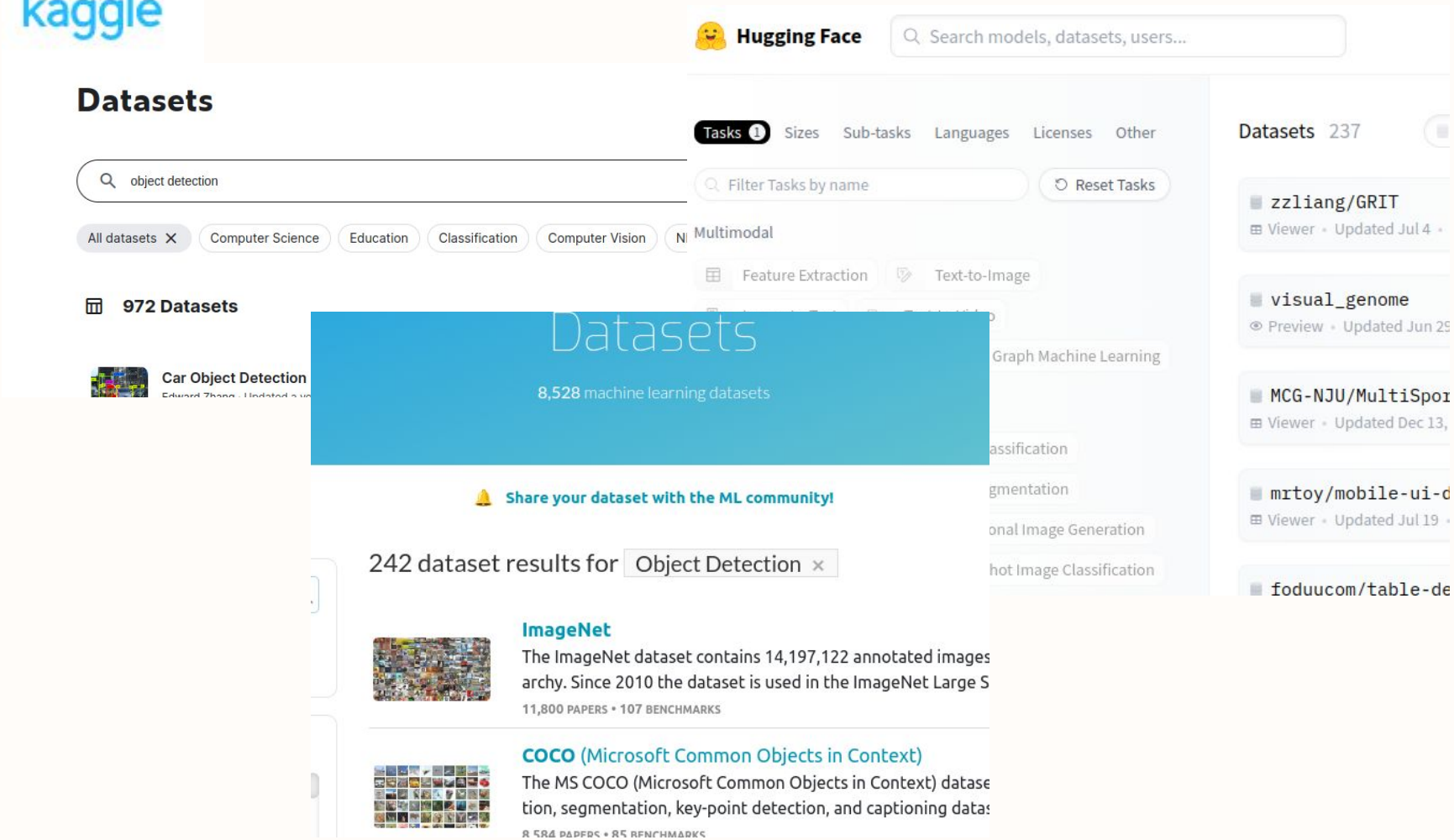
Visual Object Classes Challenge 2012 (VOC2012)

PASCAL2
Pattern Analysis, Statistical Modelling and Computational Learning

[click on an image to see the annotation]

Detecção de objetos - Datasets

kaggle



Segmentação

Segmentação de objetos - Introdução

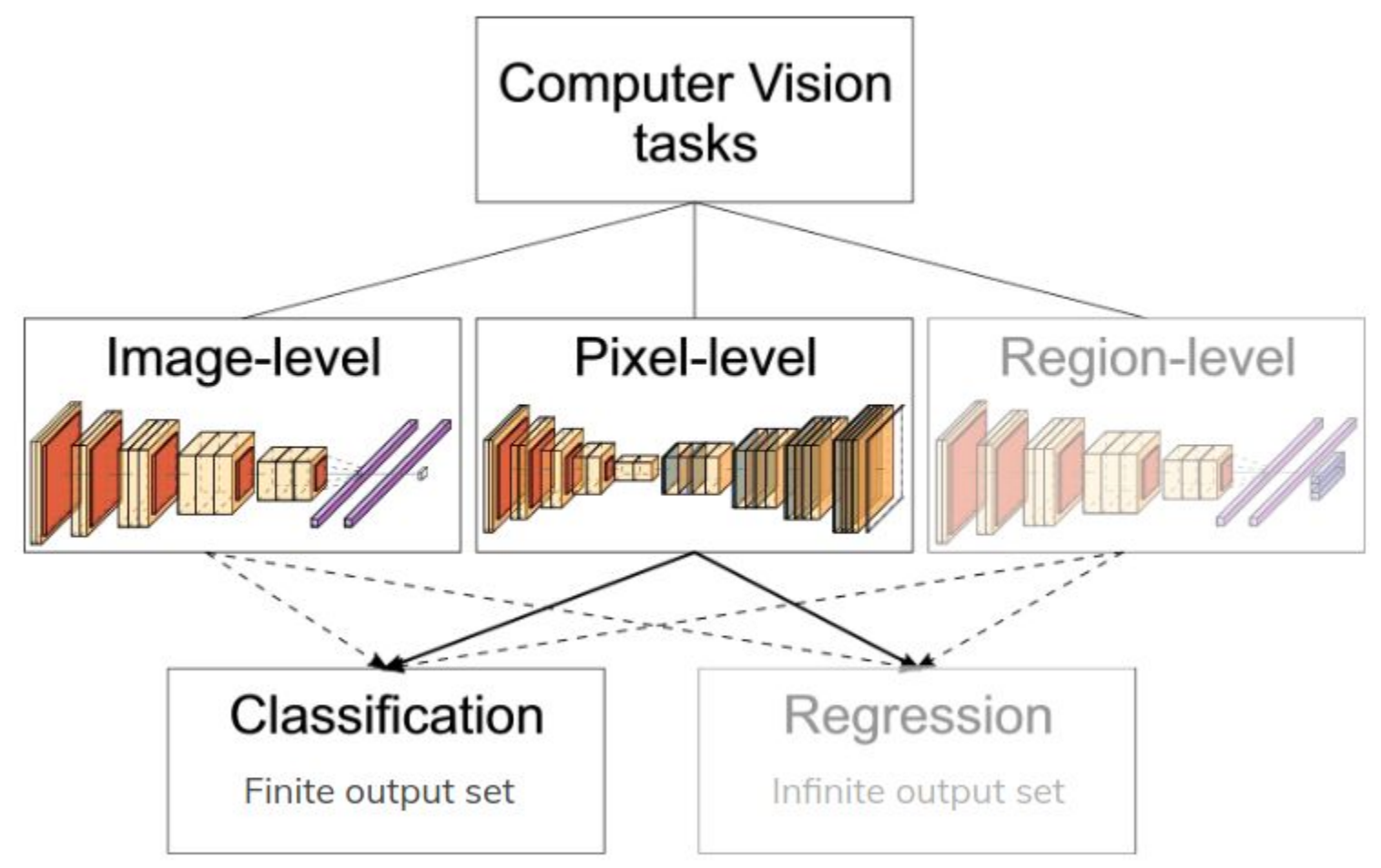
Segmentação de objetos

- ➔ Classificação: Identifica um ou mais objetos na imagem, retornando uma label específica para cada região na imagem

- ➔ Detecção: Identifica a posição do objeto na imagem, retornando uma região de interesse definida pela caixa que delimita o objeto na imagem

- ➔ Segmentação: Identifica a área que um objeto ocupa na imagem, fracionando a região do objeto em segmentos com base em suas características visíveis.

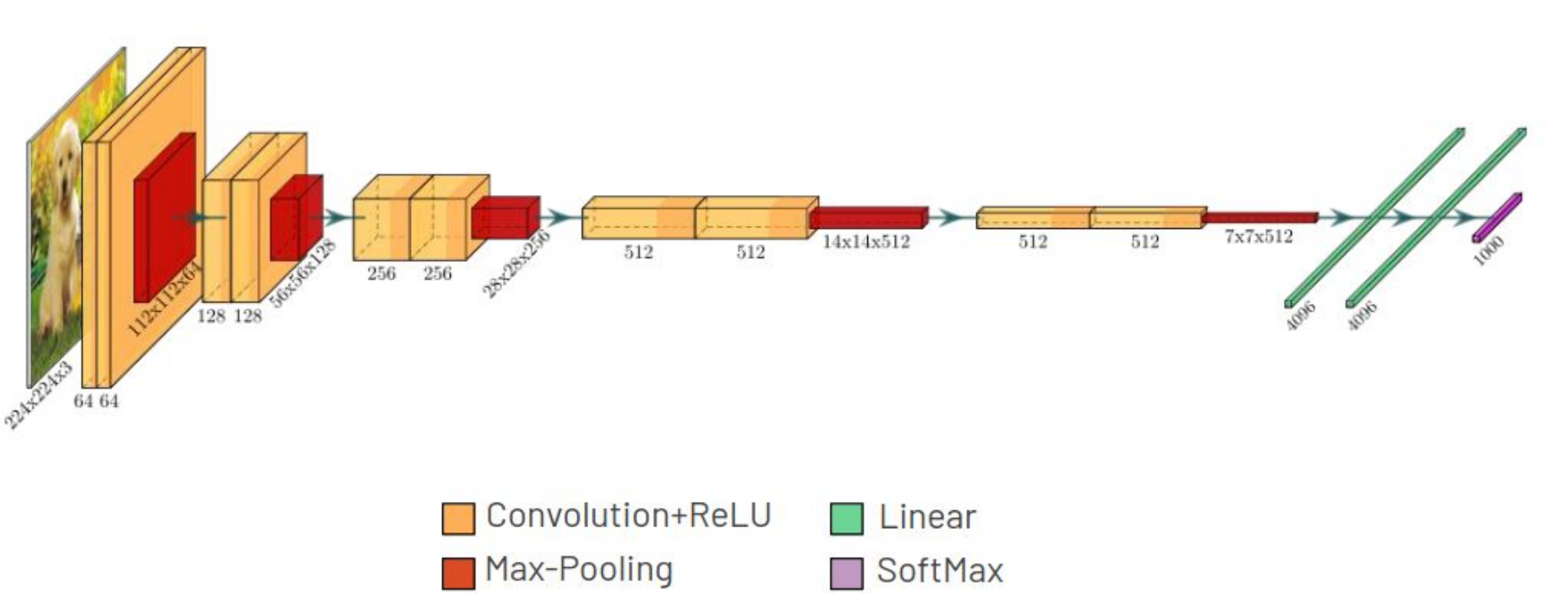
Segmentação de objetos



Fonte: Fidel Guerrero Peña, Tsang Ing Ren. Classification for image and pixel-level tasks

Image-level classification

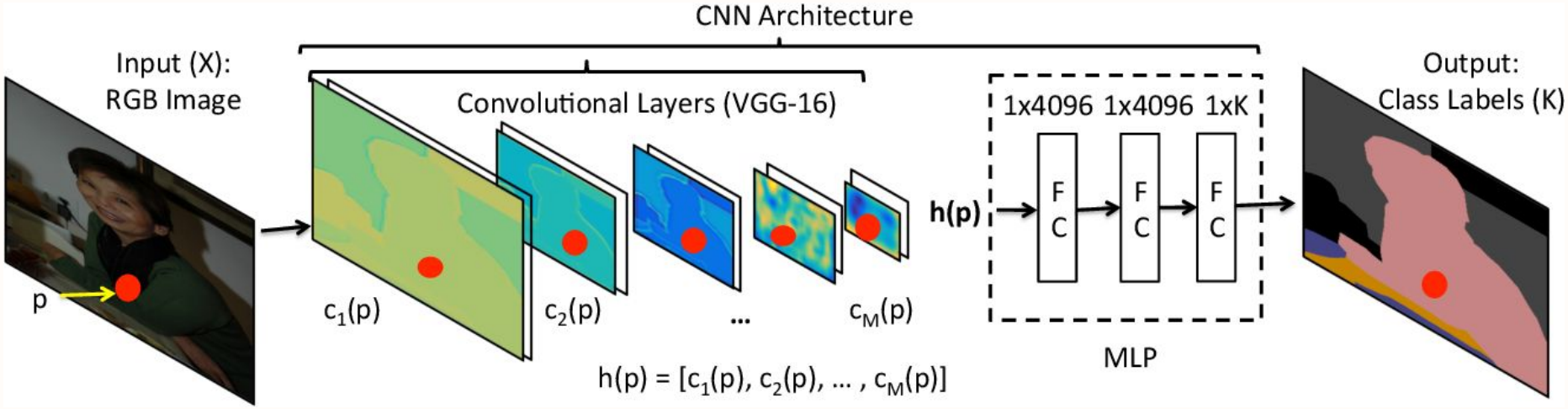
➔ AlexNet, VGG, ResNet



Fonte: Fidel Guerrero Peña, Tsang Ing Ren. Classification for image and pixel-level tasks

Pixel-level classification

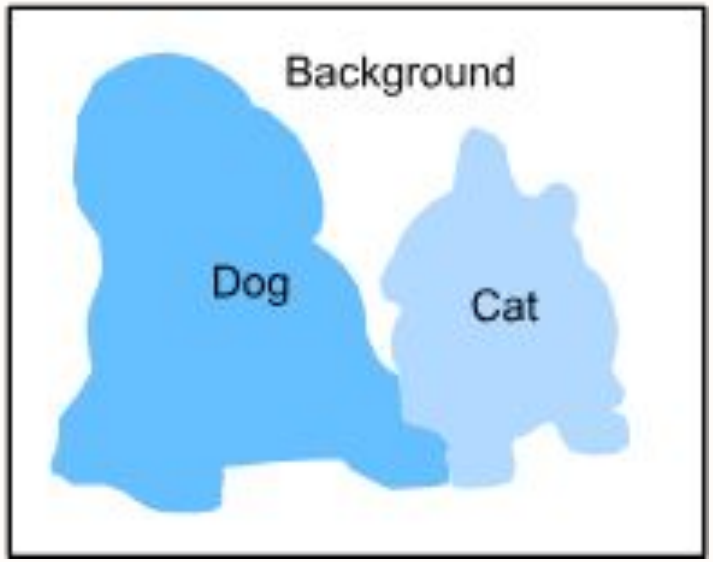
➔ Entrada e saída tem mesma resolução espacial (largura e altura)



Fonte da imagem: Bansal, Aayush, et al. "Pixelnet: Towards a general pixel-level architecture." *arXiv preprint arXiv:1609.06694* (2016).

Pixel-level classification - Segmentação de objetos

- ➔ Consiste em dividir a imagem em regiões pertencentes a diferentes classes semânticas
- ➔ Diferentemente da detecção, a segmentação tem por objetivo identificar e classificar o que está presente na imagem a nível de pixel



Segmentação de objetos

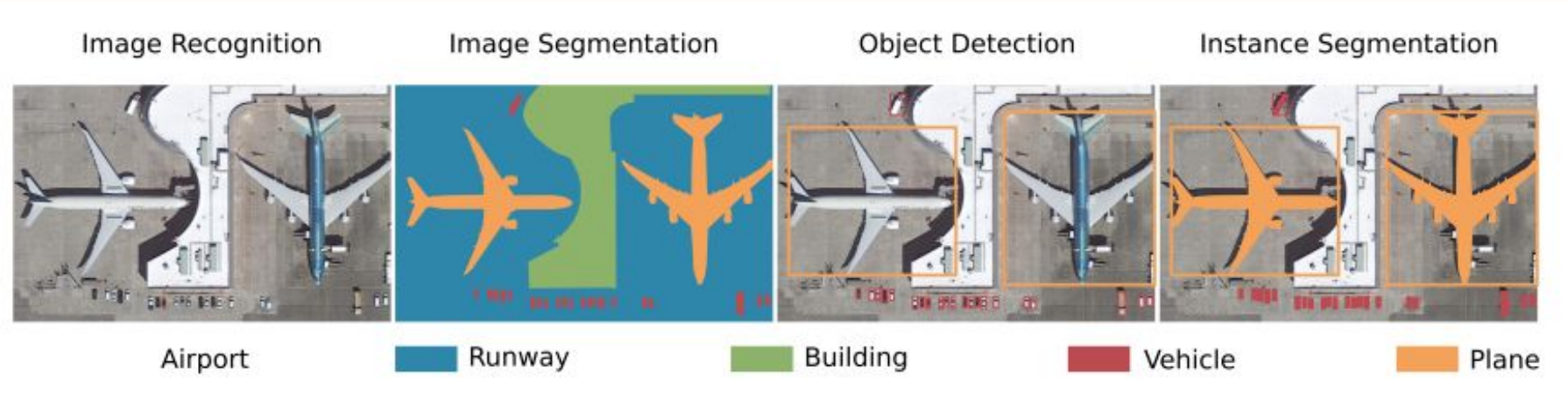


Imagem obtida de : Hoeser, Thorsten, and Claudia Kuenzer. "Object detection and image segmentation with deep learning on earth observation data: A review-part i: Evolution and recent trends." *Remote Sensing* 12.10 (2020): 1667.

Segmentação

➔ A segmentação pode se basear em informações de descontinuidade, gradiente ou regiões similares

Segmentação

➔ Em visão computacional existem duas tarefas correspondentes à segmentação: **segmentação semântica** e **segmentação de instância**

Segmentação

- ➔ Em visão computacional existem duas tarefas correspondentes à segmentação: **segmentação semântica** e **segmentação de instância**
  - **Segmentação semântica**: refere-se ao processo de encontrar e rotular determinados objetos na imagem, de tal forma a destacar cada pixel correspondente ao objeto

Segmentação

➔ Em visão computacional existem duas tarefas correspondentes à segmentação: **segmentação semântica** e **segmentação de instância**

- **Segmentação semântica**: refere-se ao processo de encontrar e rotular determinados objetos na imagem, de tal forma a destacar cada pixel correspondente ao objeto
- **Segmentação de instância:** os objetos detectados de uma mesma classe não são agrupados, mas sim são destacados como instâncias separadas

Segmentação - Semântica vs Instâncias

Grupos são identificados e segmentados (e.g., Pessoas, Carros, Pista)

Objetos são identificados de forma única, sem agrupamento de classes (e.g., pessoa 1, pessoa 2, pessoa 3, etc)

Segmentação Panóptica

➔ Essencialmente corresponde à **junção da segmentação semântica e de instância**, já que segmenta e rotula simultaneamente a instância de um objeto na imagem, de tal forma a distinguir instâncias de um mesmo objeto, ao passo que atribui o mesmo rótulo da classe semântica correspondente

Evolução da Segmentação de objetos com Deep Learning

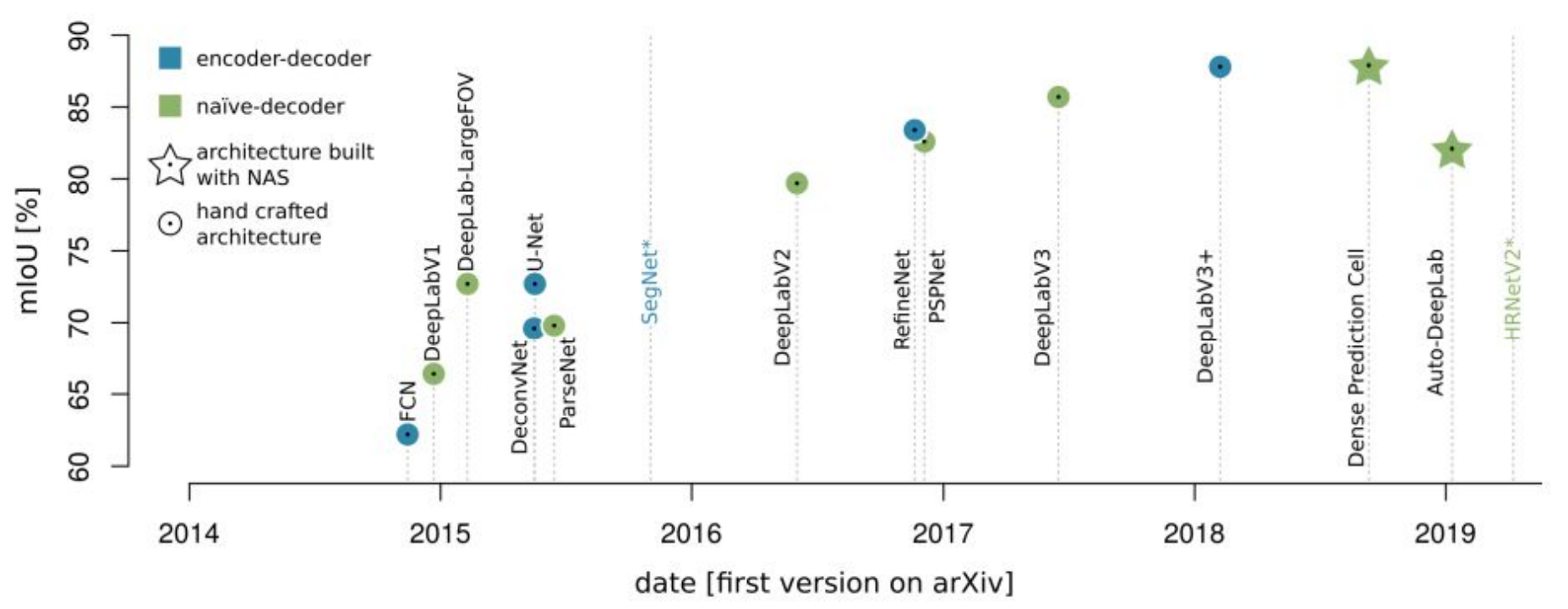
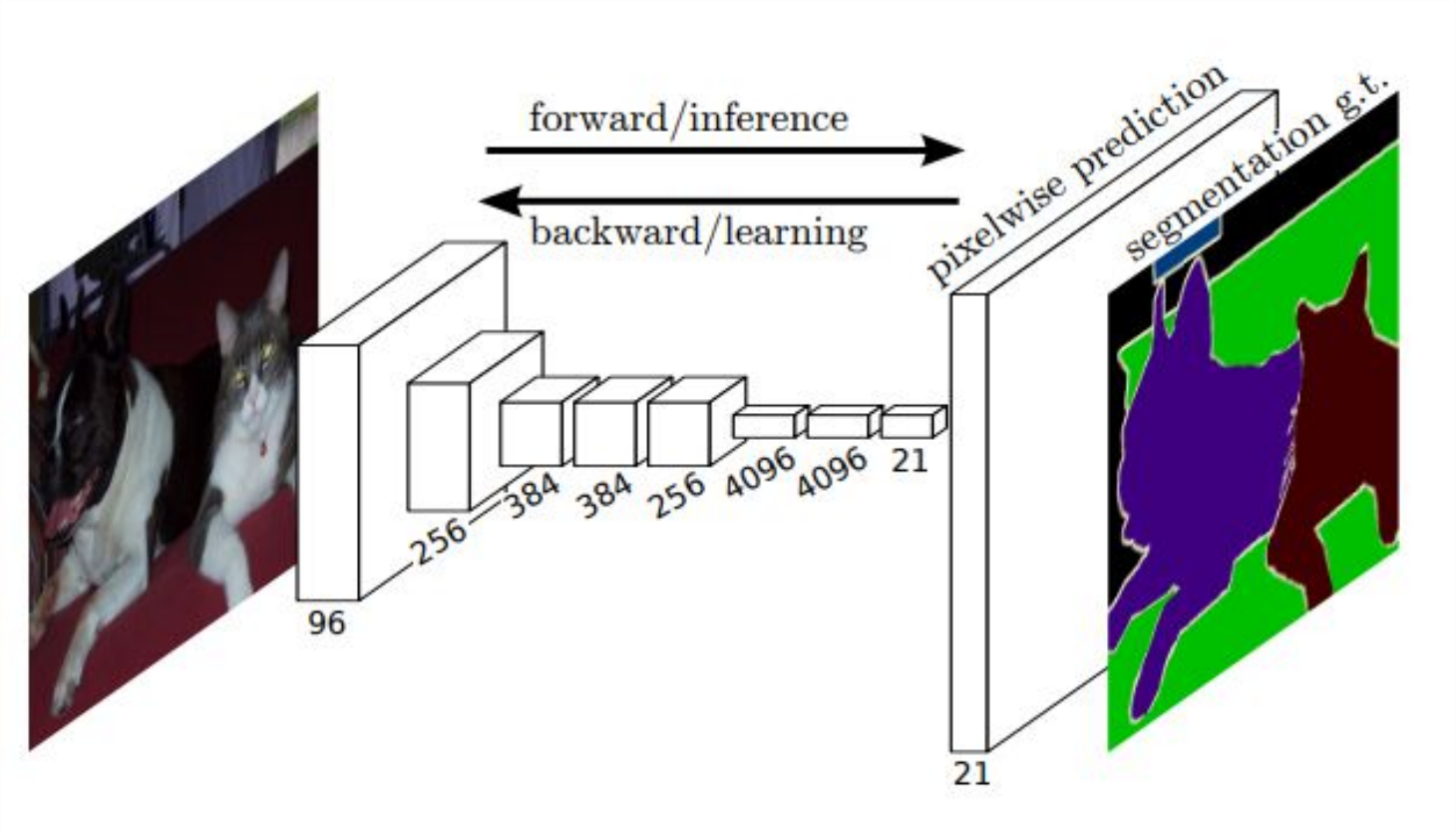


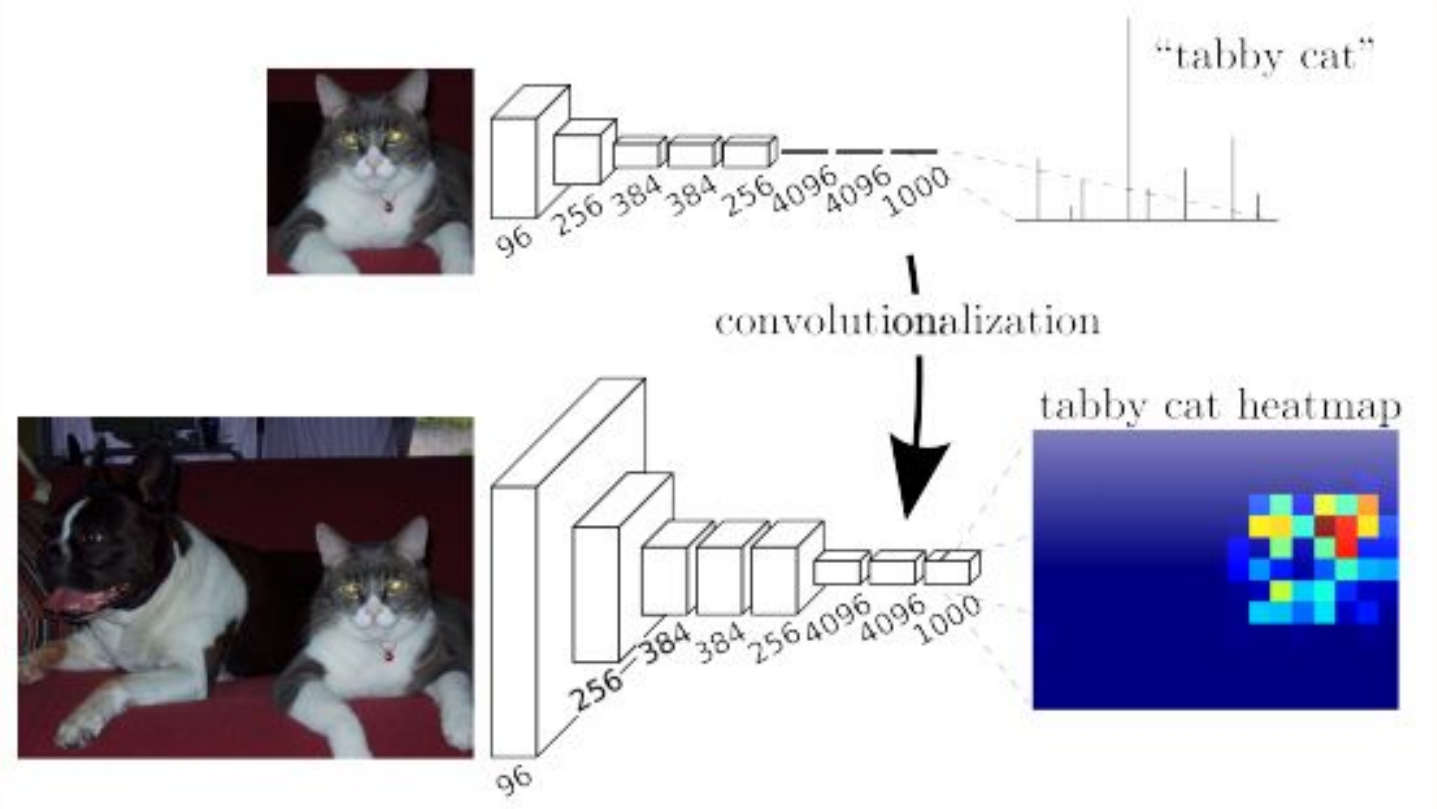
Imagem obtida de : Hoeser, Thorsten, and Claudia Kuenzer. "Object detection and image segmentation with deep learning on earth observation data: A review-part i: Evolution and recent trends." *Remote Sensing* 12.10 (2020): 1667.

Segmentação e Deep Learning - Fully Convolutional Networks



Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

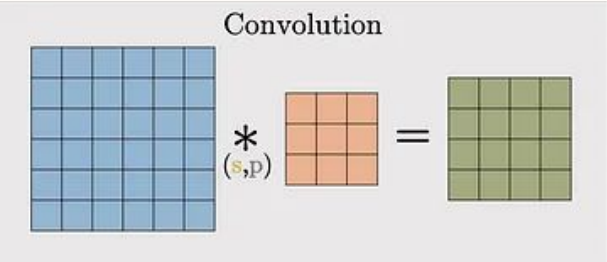
Fully Convolutional Networks - FC, upsampling e correspondências espaciais

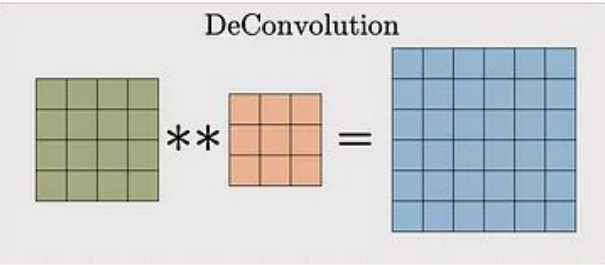


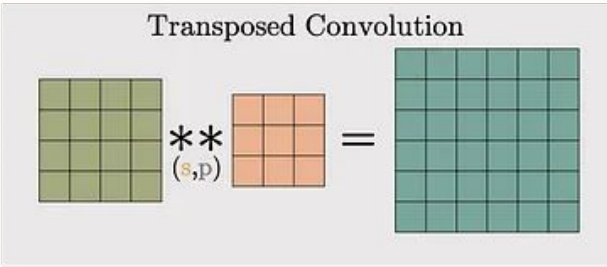
Transformando camadas fully connected em camadas convolucionais possibilita a classificação dos mapas de ativação espacial.

Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

Fully Convolutional Networks - FC, upsampling e correspondências espaciais

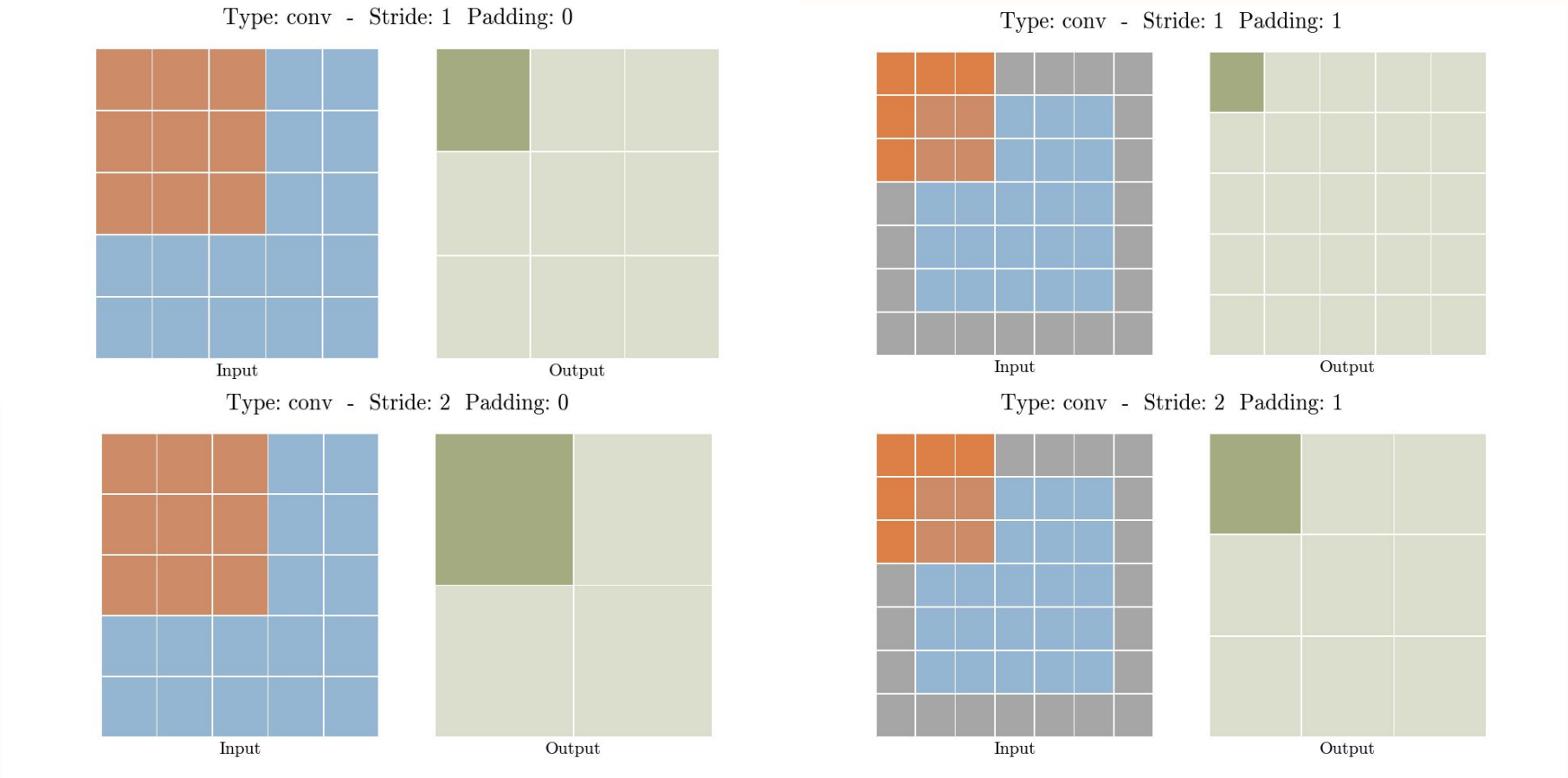






Fonte: https://towardsdatascience.com/what-is-transposed-convolutional-layer-40e5e6e31c11

Fully Convolutional Networks - FC, upsampling e correspondências espaciais

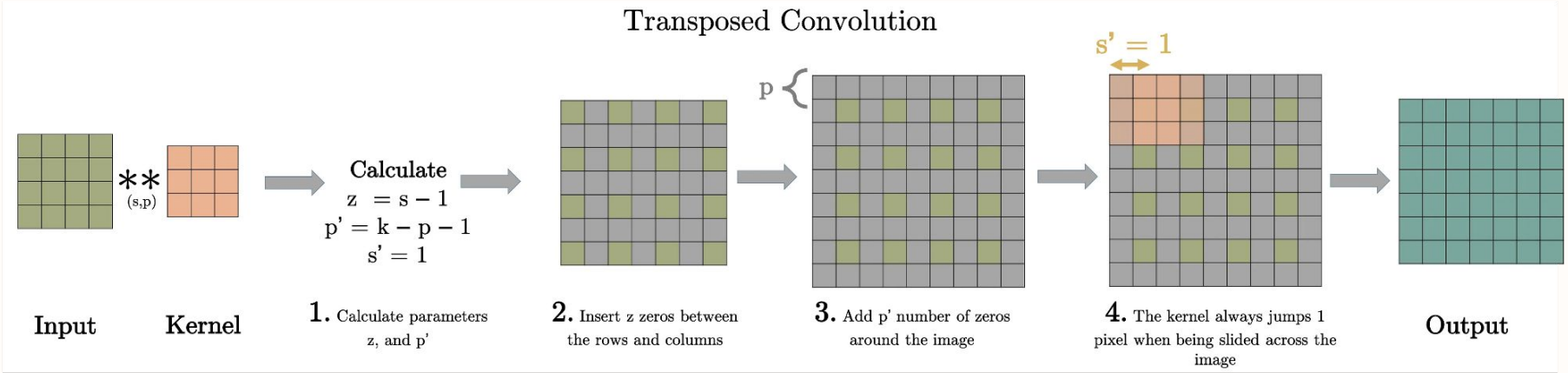


$$o = \frac{i + 2p - k}{s} + 1$$

Fonte: https://towardsdatascience.com/what-is-transposed-convolutional-layer-40e5e6e31c11

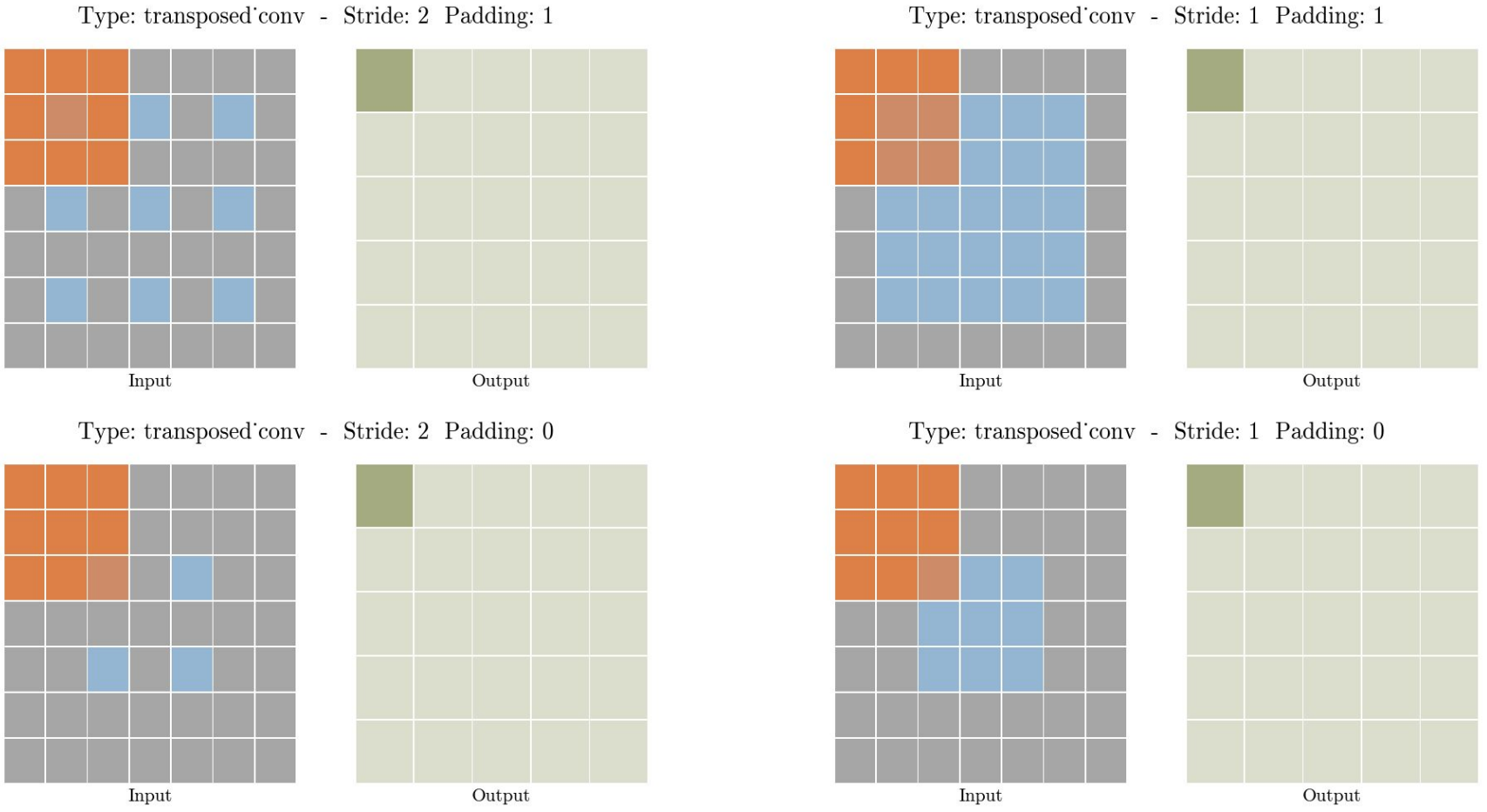
Fully Convolutional Networks - FC, upsampling e correspondências espaciais

1. Calcular parâmetros z' e p'
2. Adicionar entre cada linha e coluna z números 0s.
3. Realizar o pad com p' números 0s.
4. Executar a convolução com o resultado e stride 1



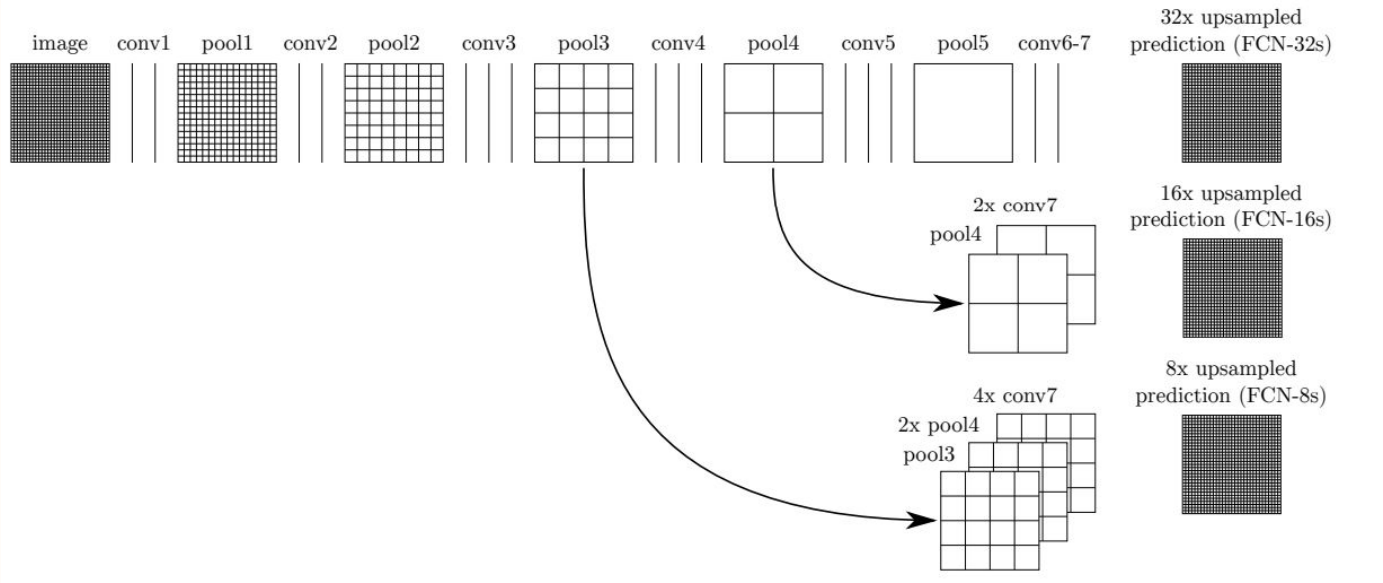
Fonte: https://towardsdatascience.com/what-is-transposed-convolutional-layer-40e5e6e31c11

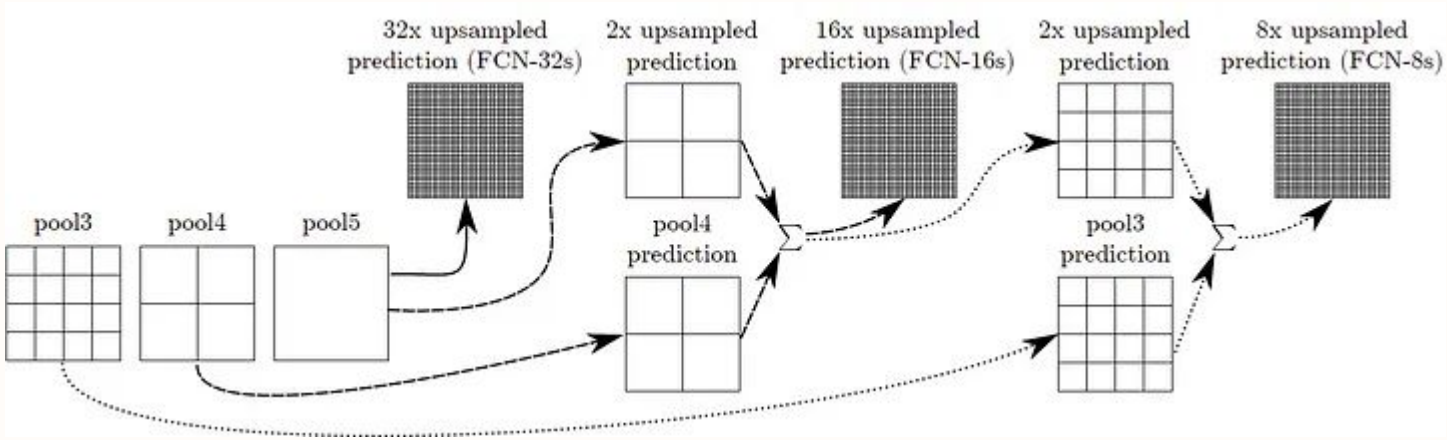
Fully Convolutional Networks - FC, upsampling e correspondências espaciais



Fonte: https://towardsdatascience.com/what-is-transposed-convolutional-layer-40e5e6e31c11

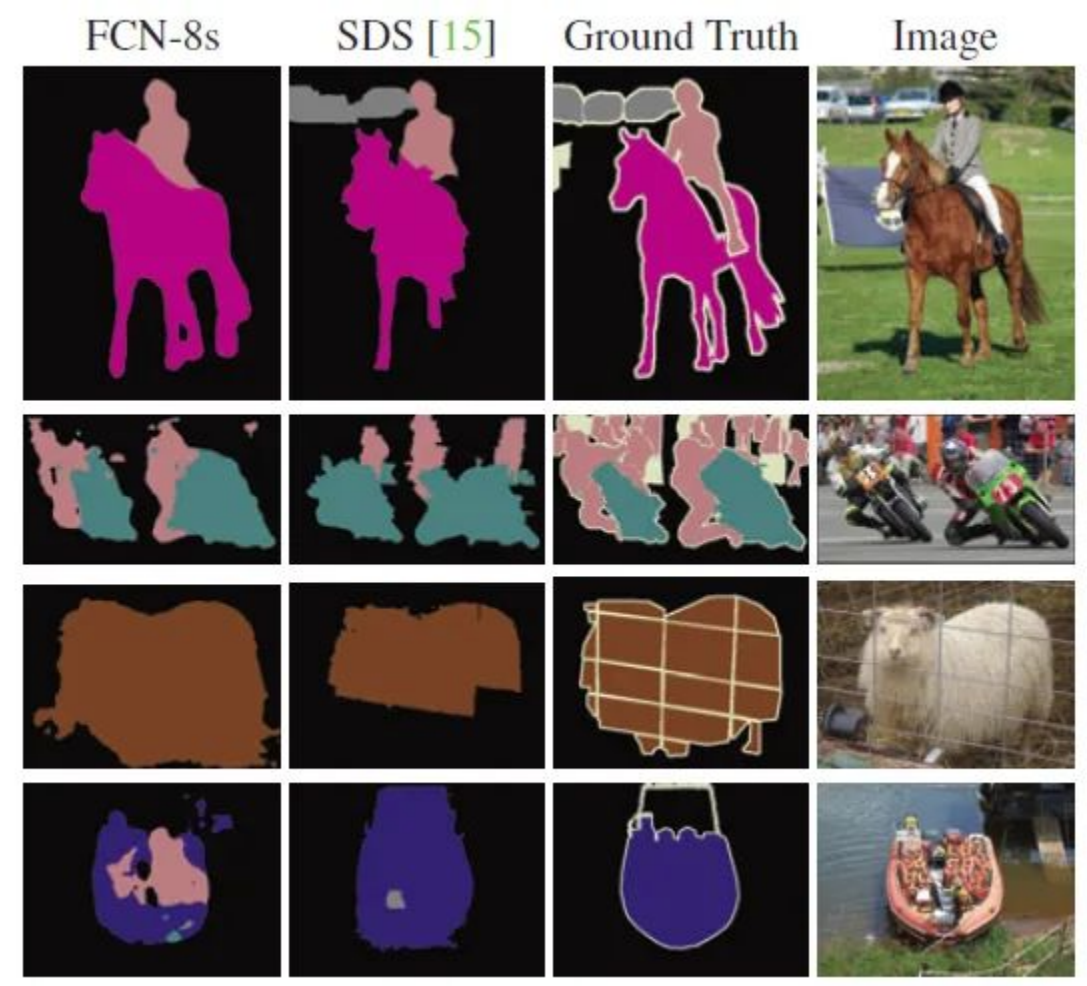
Fully Convolutional Networks - Fusão de camadas





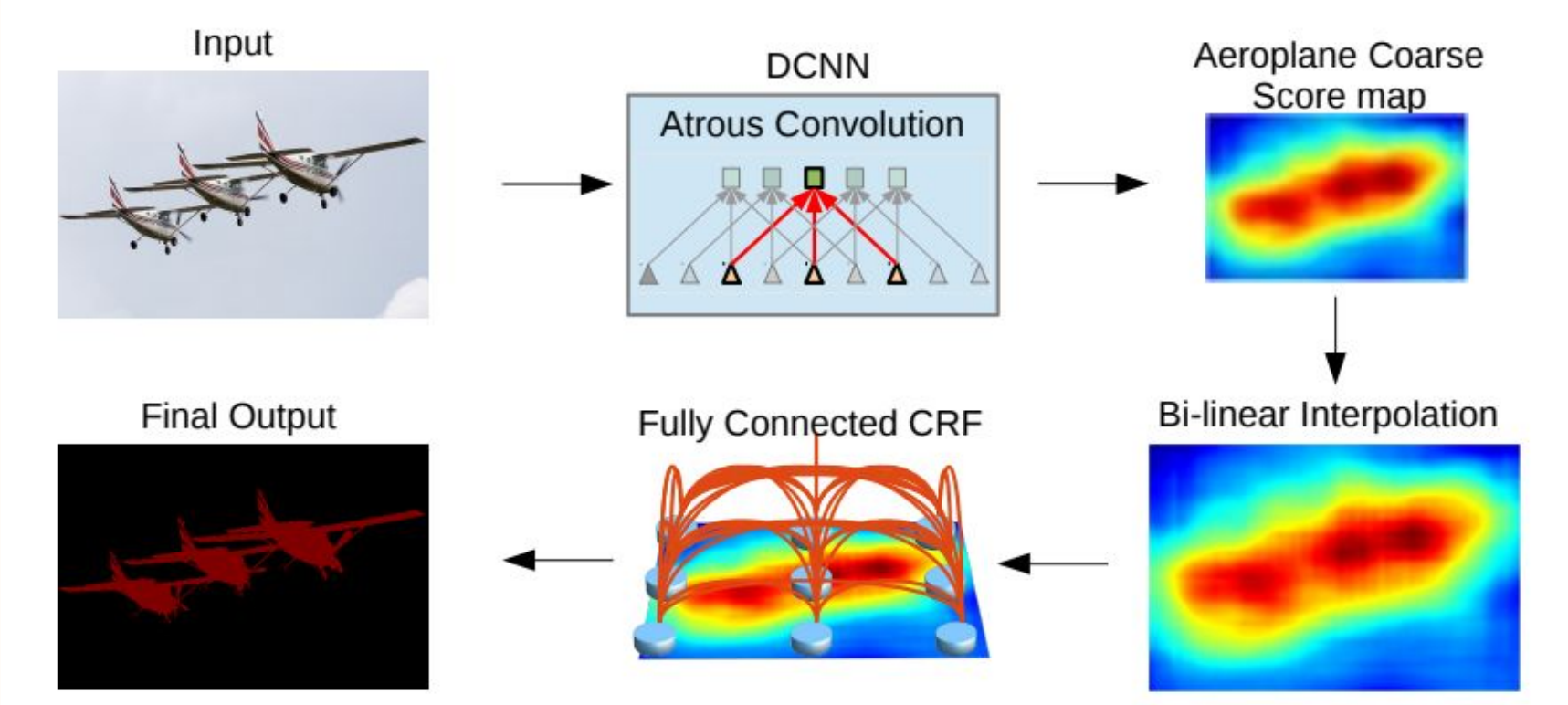
Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

Fully Convolutional Networks



Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

Segmentação e Deep Learning - DeepLab



Chen, Liang-Chieh, et al. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs." *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2017): 834-848.

DeepLab - Atrous Convolution

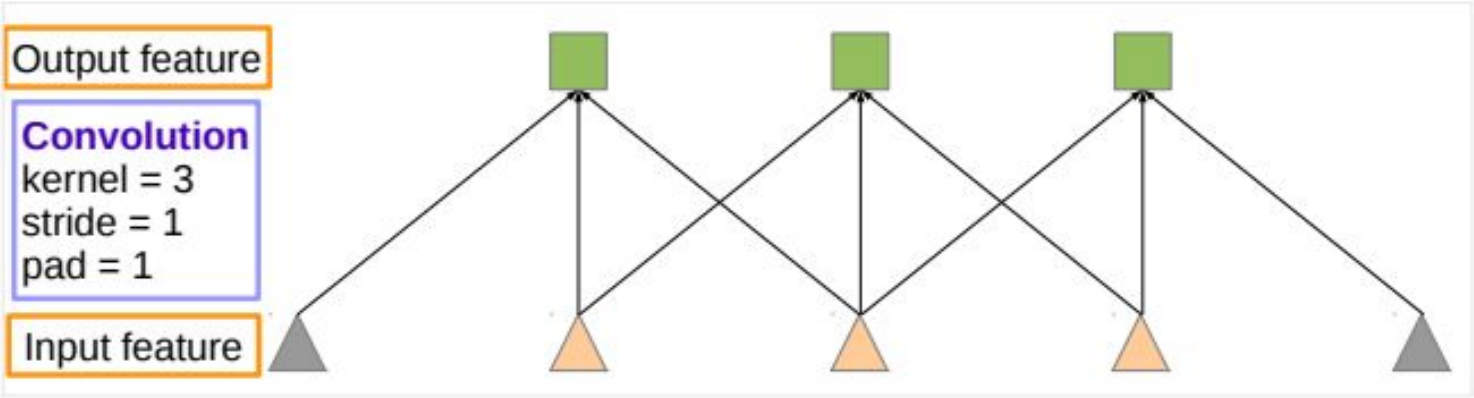
- ➔ Convolução Atrous: Inspirado no termo francês “à trous”, que significa buraco, alguns trabalhos se referem ao algoritmo como “dilated convolution”.
  - ◆ É aplicado normalmente com a transformada de wavelet.

$$y_i = \sum_{k=1}^{K} x[i + r \cdot k] w[k]$$

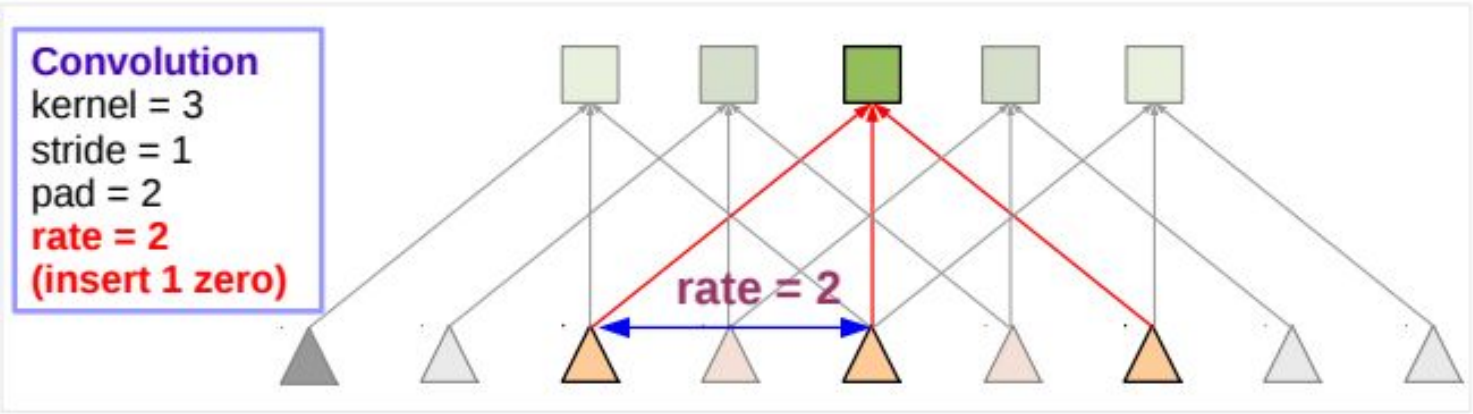
- ➔ Se r=1, temos a convolução padrão.
- ➔ Quando r>1, obtemos a convolução atrous, aumentando o stride dos exemplos de entradas

Chen, Liang-Chieh, et al. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs." *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2017): 834-848.

DeepLab – Atrous Convolution



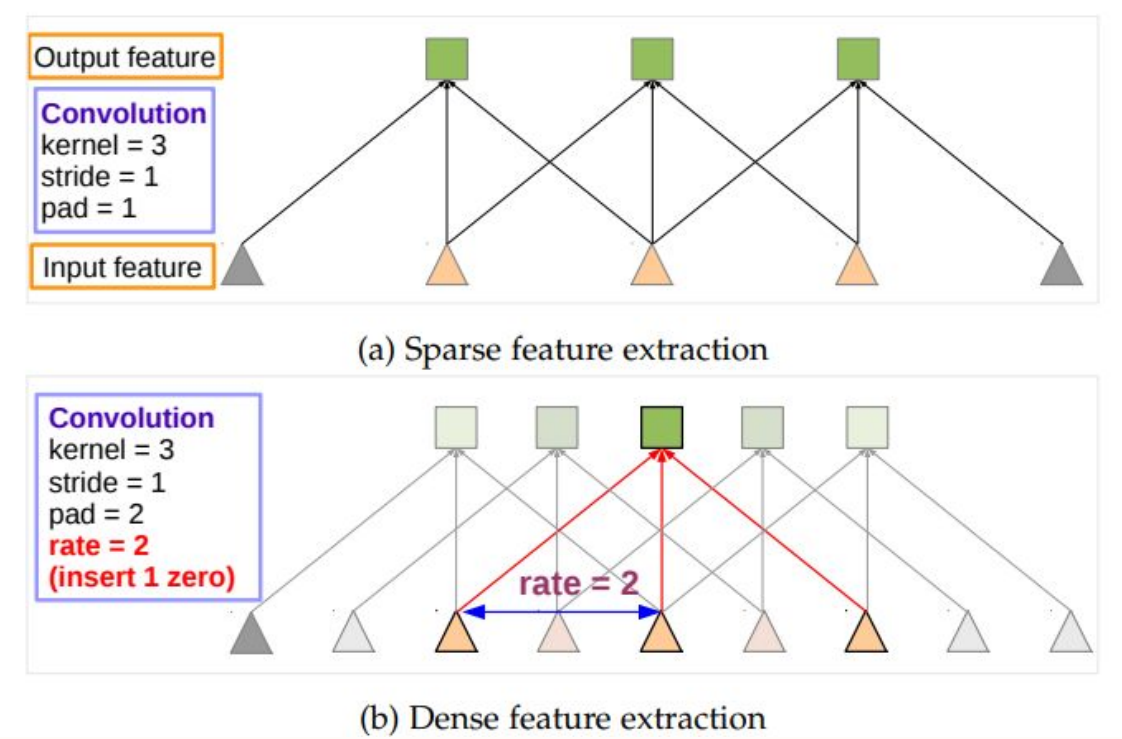
(a) Sparse feature extraction



(b) Dense feature extraction

Chen, Liang-Chieh, et al. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs." *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2017): 834-848.

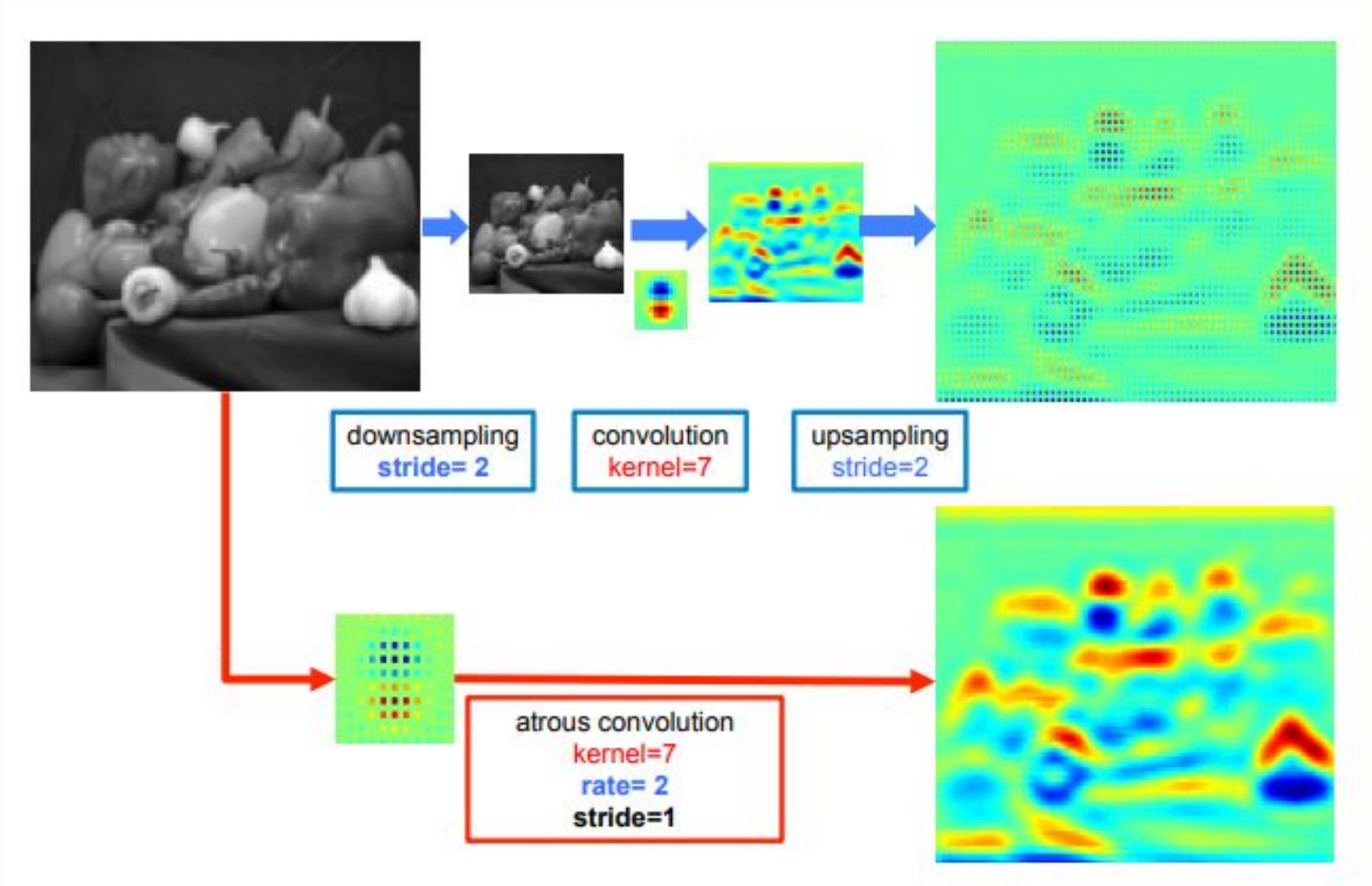
DeepLab - Atrous Convolution



➔ Acima temos a convolução padrão
➔ Na figura abaixo, temos um exemplo da convolução atrous com r=2

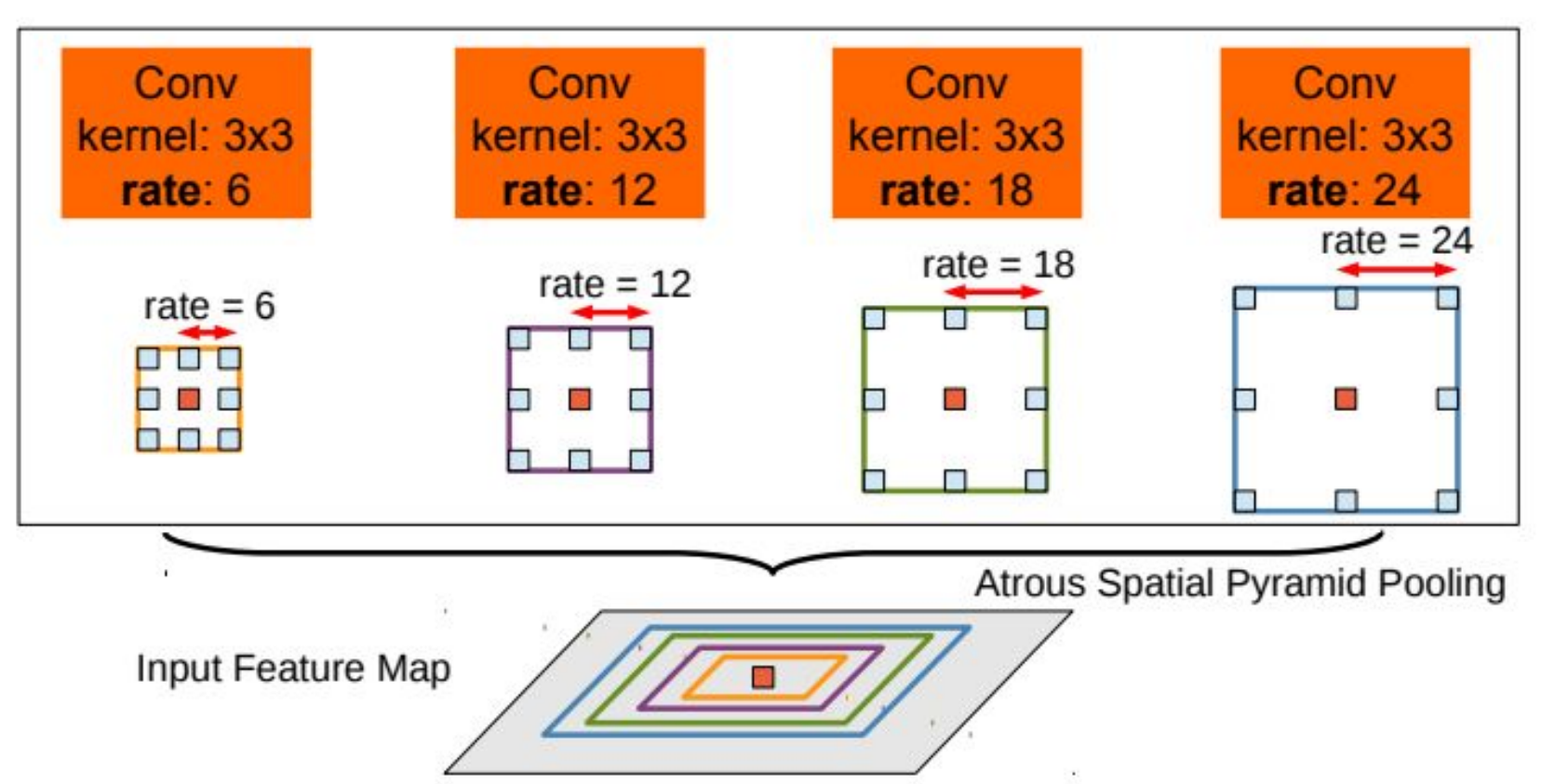
Chen, Liang-Chieh, et al. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs." *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2017): 834-848.

DeepLab - Atrous Convolution



Chen, Liang-Chieh, et al. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs." *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2017): 834-848.

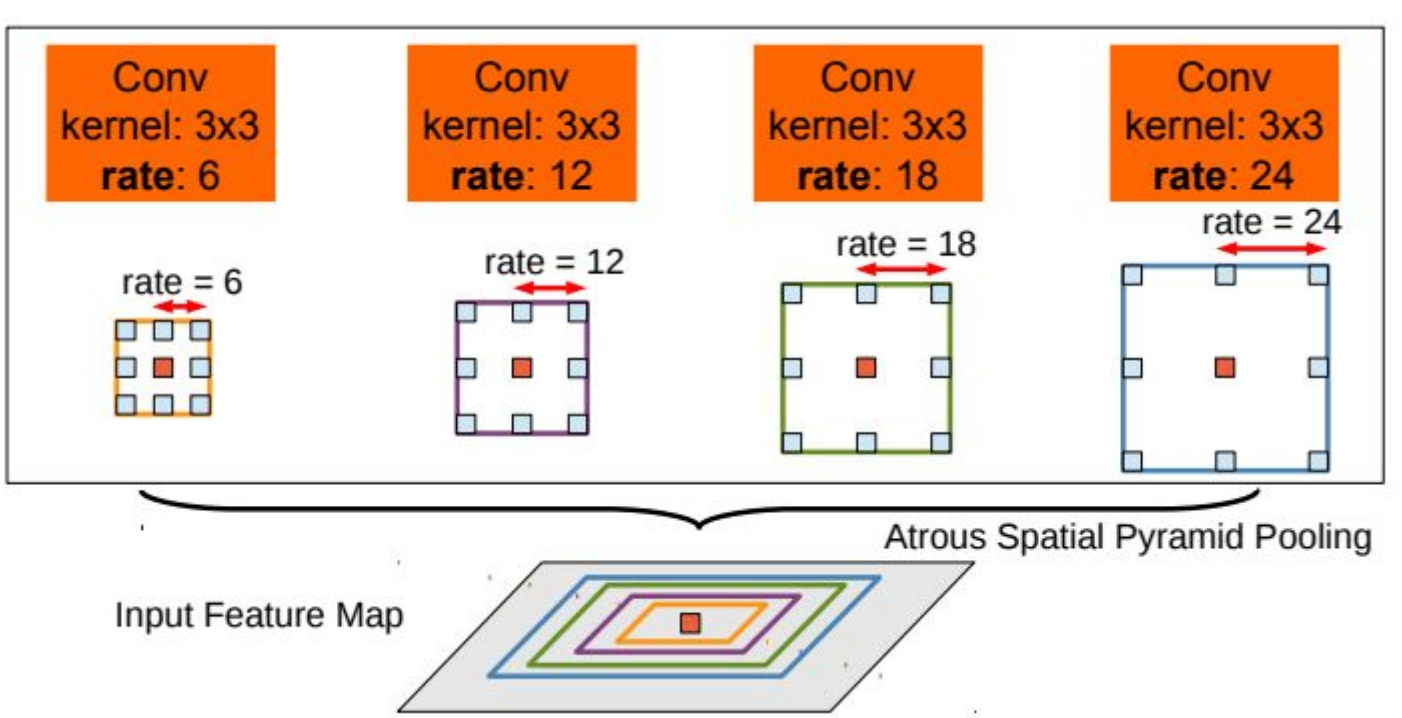
DeepLab - Atrous Spatial Pyramid Pooling (ASPP)



Chen, Liang-Chieh, et al. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs." *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2017): 834-848.

DeepLab - Atrous Spatial Pyramid Pooling (ASPP)

- ➔ Aplicação do algoritmo atrous com SPP (Spatial Pyramid Pooling)
  - ◆ Melhora a acurácia para segmentação de objetos com diferentes escalas



Chen, Liang-Chieh, et al. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs." *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2017): 834-848.

DeepLab - Fully Connected CRF (Conditional Random Field)

- ➔ FC-CRF é aplicado após a interpolação bilinear
- ➔ x é o label correspondente ao pixel e P(x) é a probabilidade associada ao pixel.
- ➔ θij funciona como um filtro
  - ◆ μ = 1 se xj é diferente de xi e μ = 0 quando xi = xj
  - ◆ Entre colchetes o primeiro termo é similar a um filtro bilateral (preserva características das arestas)
  - ◆ O segundo termo depende apenas da posição dos pixels, e funciona como um filtro gaussiano

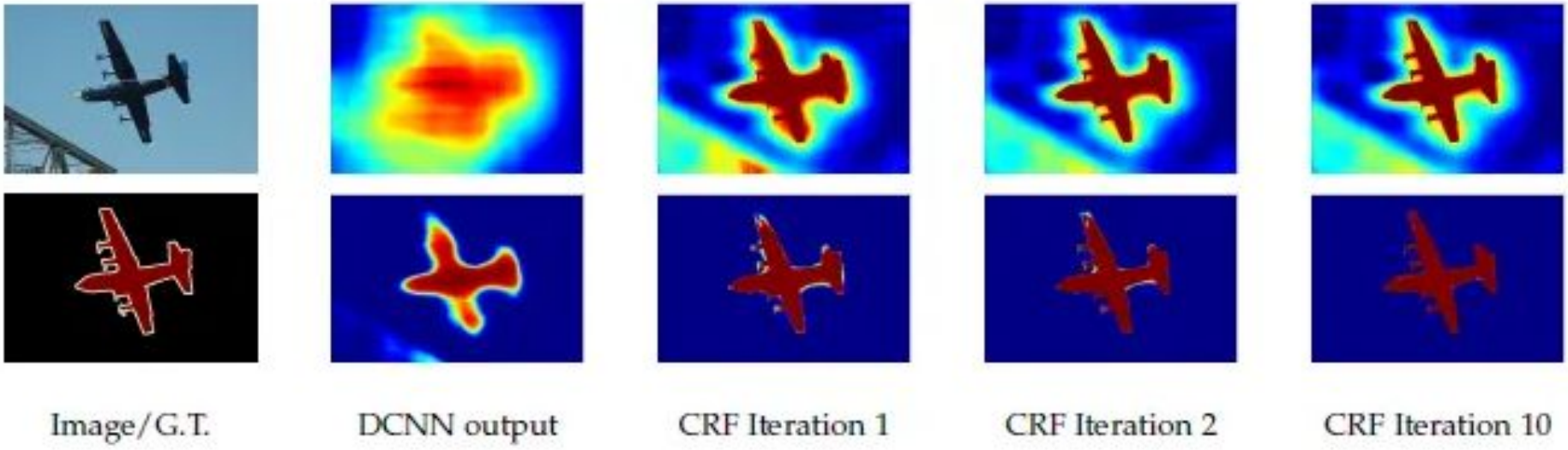
$$E(x) = \sum_i \theta_i(x_i) + \sum_{ij} \theta_{ij}(x_i, x_j)$$

$$\theta_i = -\log P(x_i)$$

$$\theta_{ij}(x_i, x_j) = \mu(x_i, x_j)\left[w_1 \exp\left(-\frac{||p_i - p_j||^2}{2\delta_\alpha^2} - \frac{||I_i - I_j||^2}{2\delta_\beta^2}\right) + w_2 \exp\left(-\frac{||p_i - p_j||^2}{2\delta_\gamma^2}\right)\right]$$

Chen, Liang-Chieh, et al. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs." *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2017): 834-848.

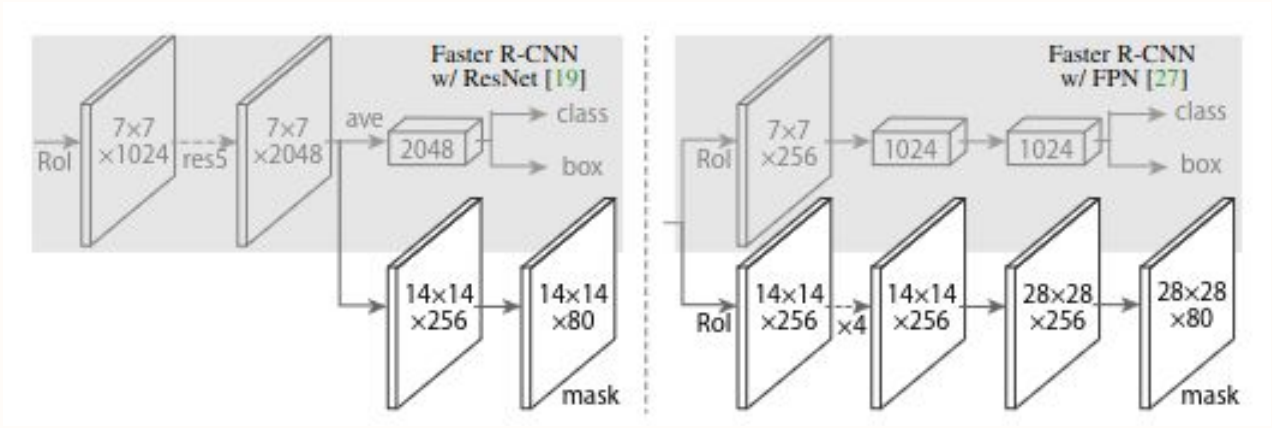
DeepLab - Fully Connected CRF (Conditional Random Field)

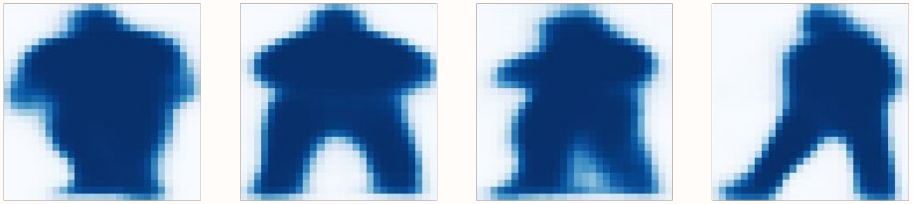


Top: Score map (input before softmax function), Bottom: belief map (output of softmax function)

Chen, Liang-Chieh, et al. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs." *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2017): 834-848.

Mask-RCNN





He, Kaiming, et al. "Mask r-cnn." *Proceedings of the IEEE international conference on computer vision*. 2017.

https://github.com/matterport/Mask_RCNN

Segmentação e Deep Learning - UNet

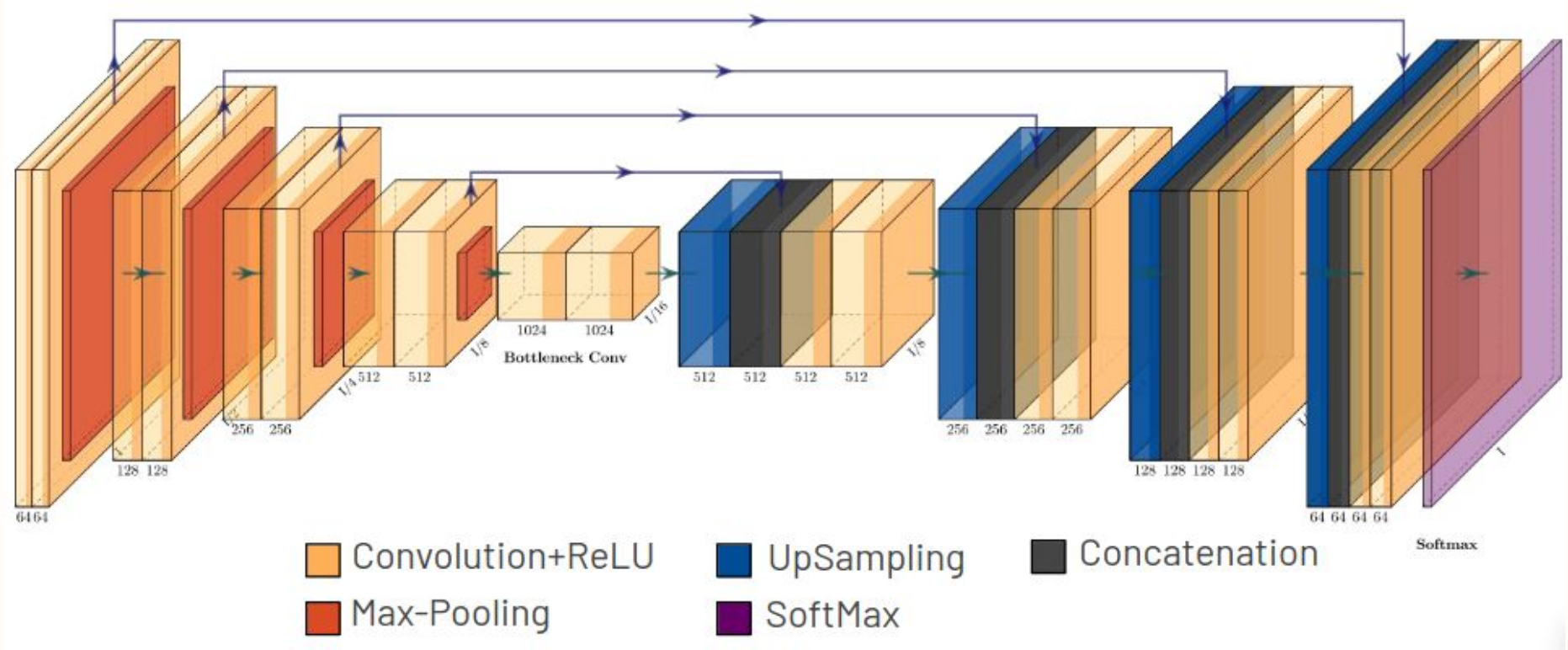
- ➔ U-Net apresenta uma arquitetura de rede **mais estruturada**
- ➔ Arquitetura baseada na ideia de Encoder-Decoder
  - ◆ Não utiliza camadas FC
  - ◆ Adiciona skip-connections por **concatenação**
- ➔ Treinamento se beneficia com operações de augmentation

U-Net - Arquitetura Encoder-Decoder

➔ Arquitetura utilizada em problemas onde devemos alinhar uma entrada de tamanho variável em um saída específica

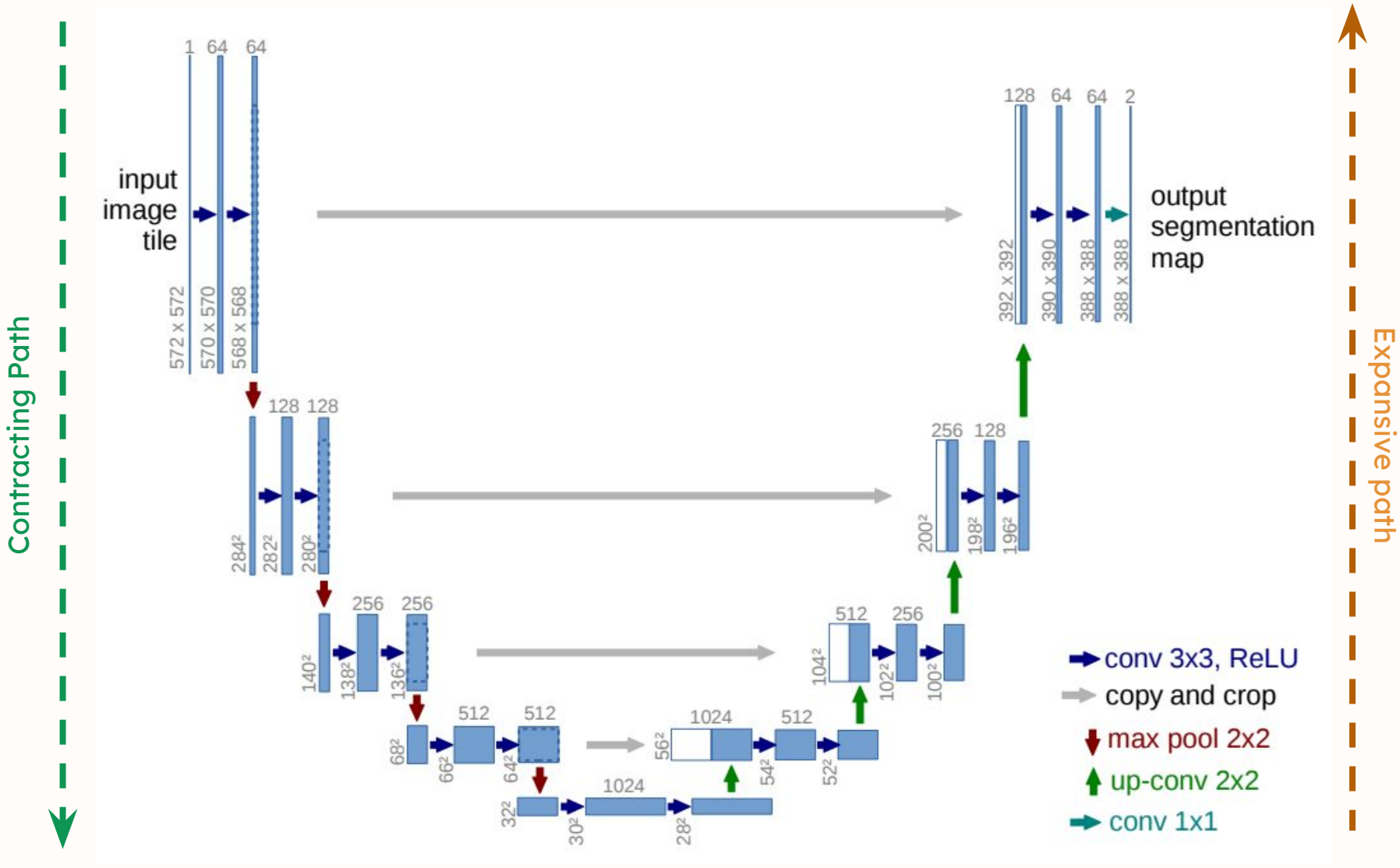
- Encoder: Recebe uma entrada e aplica uma série de transformações, mapeando o sinal recebido para um novo espaço de características através uma representação intermediária
- Decoder: Recebe a representação gerada e analisa as características, mapeando-as para o espaço desejado

U-Net - Arquitetura Encoder-Decoder



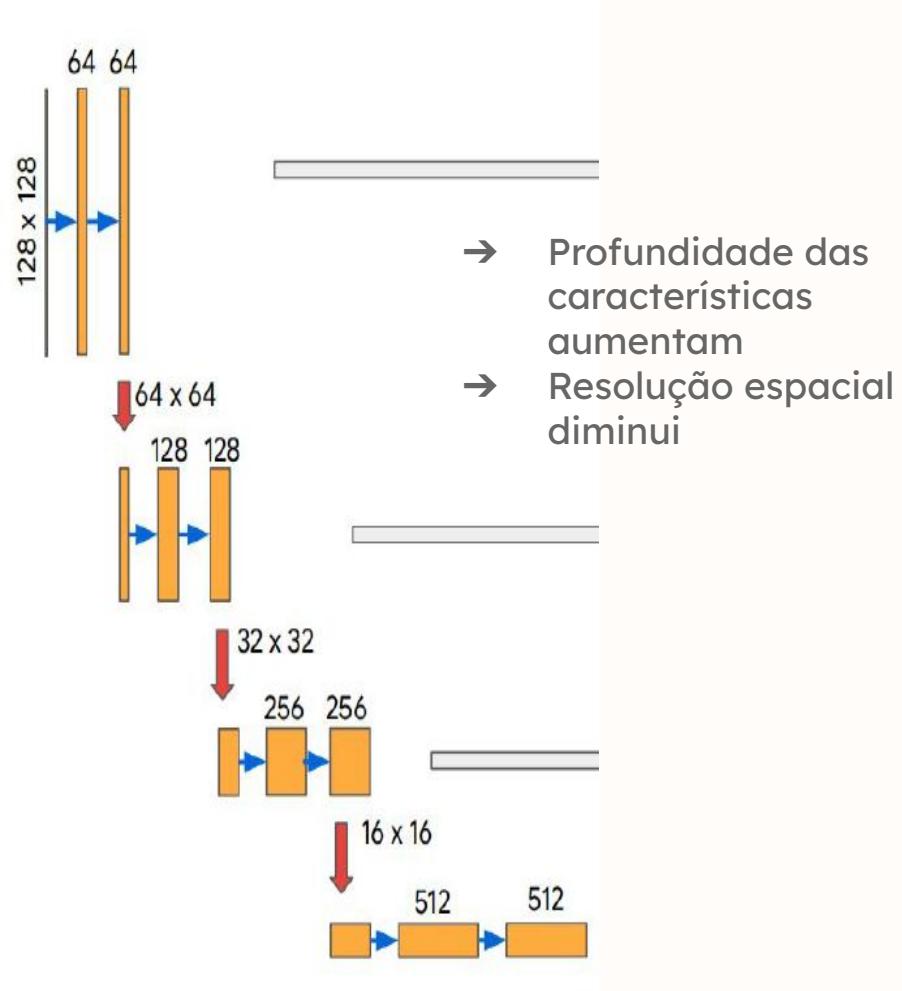
Fonte: Fidel Guerrero Peña, Tsang Ing Ren. Classification for image and pixel-level tasks

UNet - Arquitetura

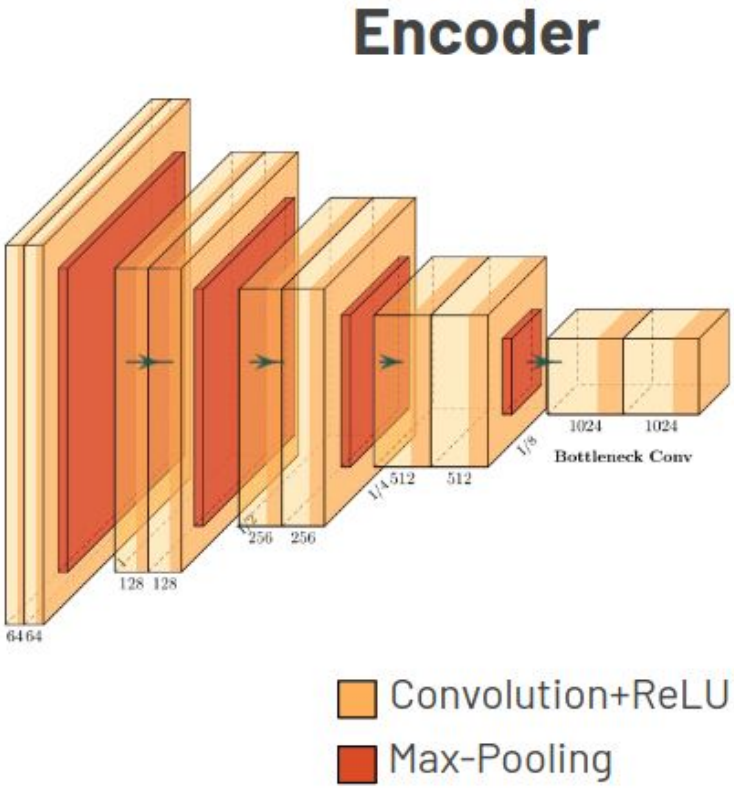


UNet Arquitetura - Contração (Encoder)

- ➔ Caminho de contração: Arquitetura convolucional padrão, são aplicadas repetidas camadas convolucionais 3x3 (sem pad) seguidas de ativação ReLU e max pooling 2x2
  - ◆ Cada downsample do max pooling dobra o número de features (profundidade das ativações)



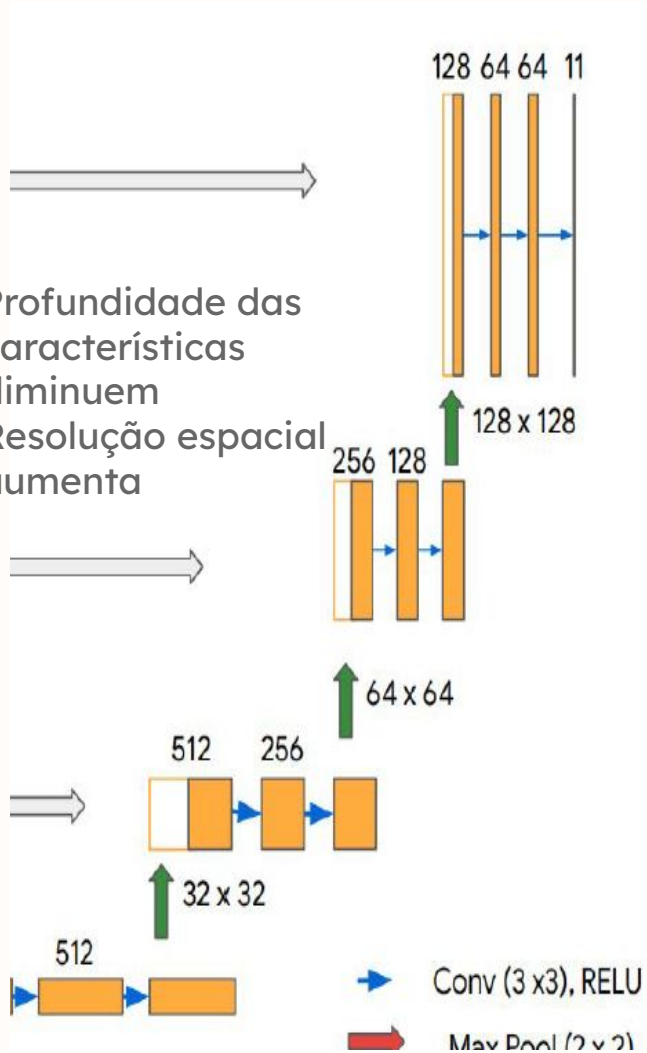
UNet Arquitetura - Contração (Encoder)



Layer	Input	Type	Output
Block1	572 x 572 x 3	Encoder	284 x 284 x 64
Block2	284 x 284 x 64	Encoder	140 x 140 x 128
Block3	140 x 140 x 128	Encoder	68 x 68 x 256
Block4	68 x 68 x 256	Encoder	32 x 32 x 512
Block5	32 x 32 x 512	Bottleneck	28 x 28 x 1024

Fonte: Fidel Guerrero Peña, Tsang Ing Ren. Classification for image and pixel-level tasks

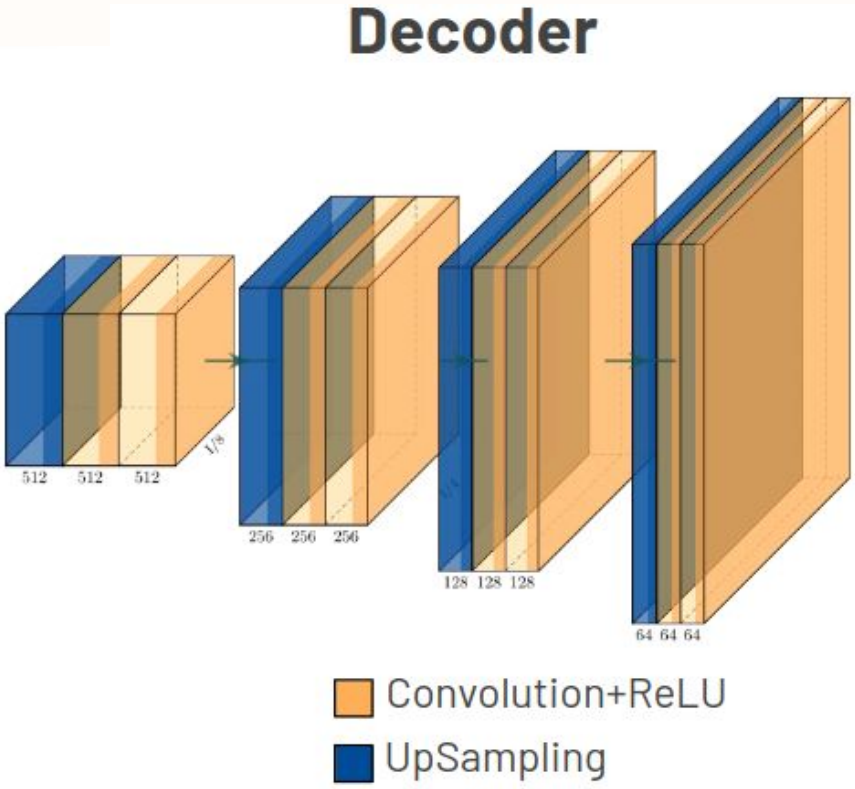
UNet Arquitetura - Expansão (Decoder)



- ➔ Profundidade das características diminuem
- ➔ Resolução espacial aumenta

- ➔ Caminho de expansão: Upsample dos mapas de ativação seguidas por up-convolution 2x2 que divide por 2 a profundidade do mapa (reduz o número de canais das features), uma concatenação com o mapa de ativação correspondente a contração do mesmo nível e convoluções 3x3 com ativação ReLU
- ➔ Na última camada uma convolução 1x1 é utilizada para mapear o vetor de features (dimensão 64) para o número de classes desejada

UNet Arquitetura - Expansão (Decoder)

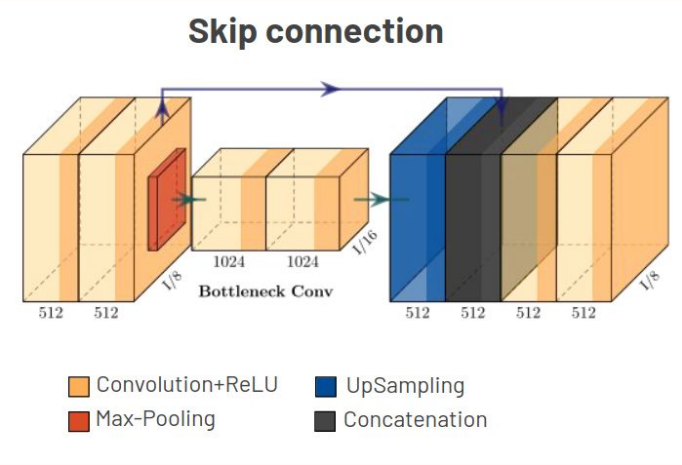


Layer	Input	Type	Output
Block6	28 x 28 x 1024	Decoder	52 x 52 x 512
Block7	52 x 52 x 512	Decoder	100 x 100 x 256
Block8	100 x 100 x 256	Decoder	196 x 196 x 128
Block9	196 x 196 x 128	Decoder	388 x 388 x 64

Fonte: Fidel Guerrero Peña, Tsang Ing Ren. Classification for image and pixel-level tasks

UNet Arquitetura - Skip connections

- ➔ DNNs podem esquecer certas características na medida que avança na arquitetura por sucessivas camadas
- ➔ Skip connections reforça o aprendizado, introduzindo as informações de camadas anteriores em pontos mais avançados da arquitetura
- ➔ U-Net utiliza skip connections em cada passo da expansão (decoder) a partir das ativações correspondentes de cada passo da contração (encoder)



UNet Arquitetura - Skip connections

- ➔ U-Net segmenta com base em detalhes de baixo nível e características genéricas
  - ◆ Arquiteturas Encoder-Decoder extraem características mais genéricas na medida em que avançam na arquitetura
  - ◆ Skip connections reintroduz features de baixo nível (detalhes)

UNet - Otimização e Loss

➔ Categorical cross-entropy

$$L = \sum_{c=1}^{k} x_c \odot \log\left(soft\max(\hat{x}_c)\right)$$

$$soft\max(\hat{x}_c) = \frac{\exp\left(\hat{x}_c\right)}{\sum_{j=1}^{k} \exp\left(\hat{x}_j\right)}$$

UNet - Otimização e Loss

➔ Dice

$$L = 1 - \frac{2\sum_{i=1}^{N} soft\max(\hat{x_c}) \odot x_c + \epsilon}{\sum_{i=1}^{N} x_c + \sum_{i=1}^{N} soft\max(\hat{x}_c) + \epsilon}$$

Segmentação - Comparação de utilização em segmentação das arquiteturas de deep learning

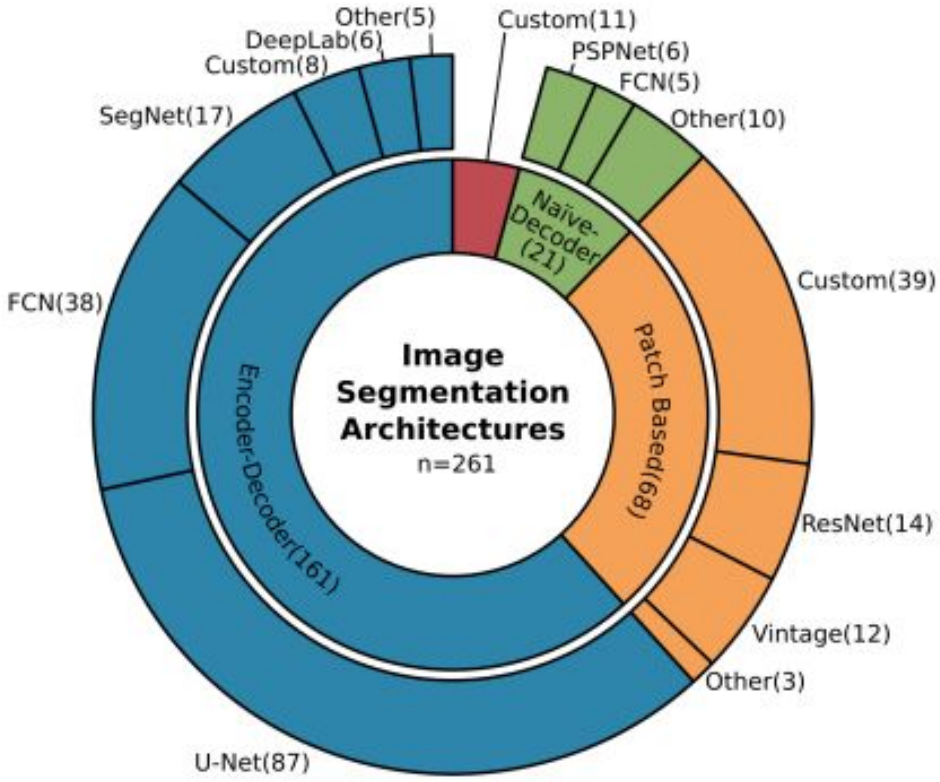
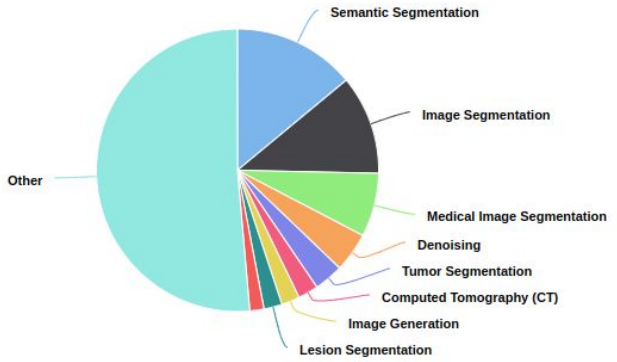


Figure 9. Overview of the commonly employed architectures for image segmentation. With 62% encoder-decoder models are the most frequently used especially the U-Net design (33%). Followed by patch-based approaches (26%) and naïve-decoder models (8%).

Imagem obtida de: Hoeser, Thorsten, Felix Bachofer, and Claudia Kuenzer. "Object detection and image segmentation with deep learning on Earth observation data: A review—Part II: Applications." *Remote Sensing* 12.18 (2020): 3053.

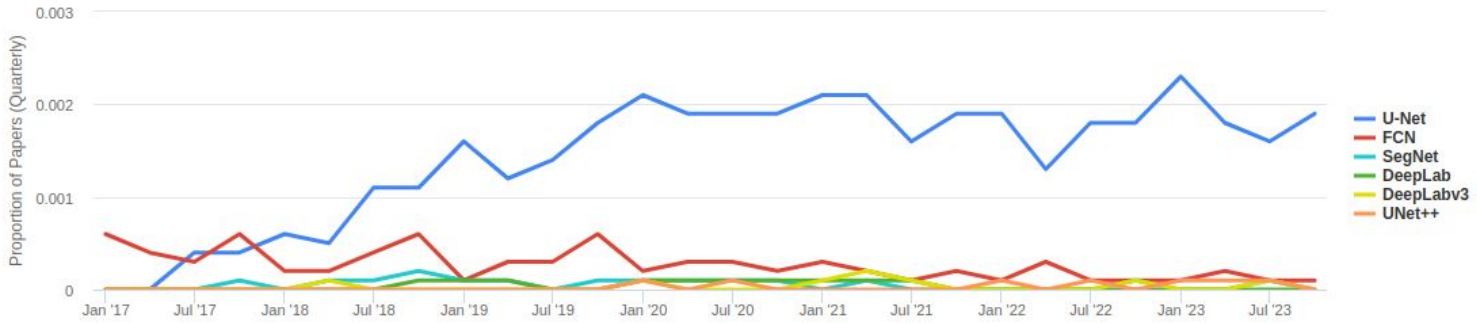
Segmentação - UNet - Aplicações e utilização

Tasks



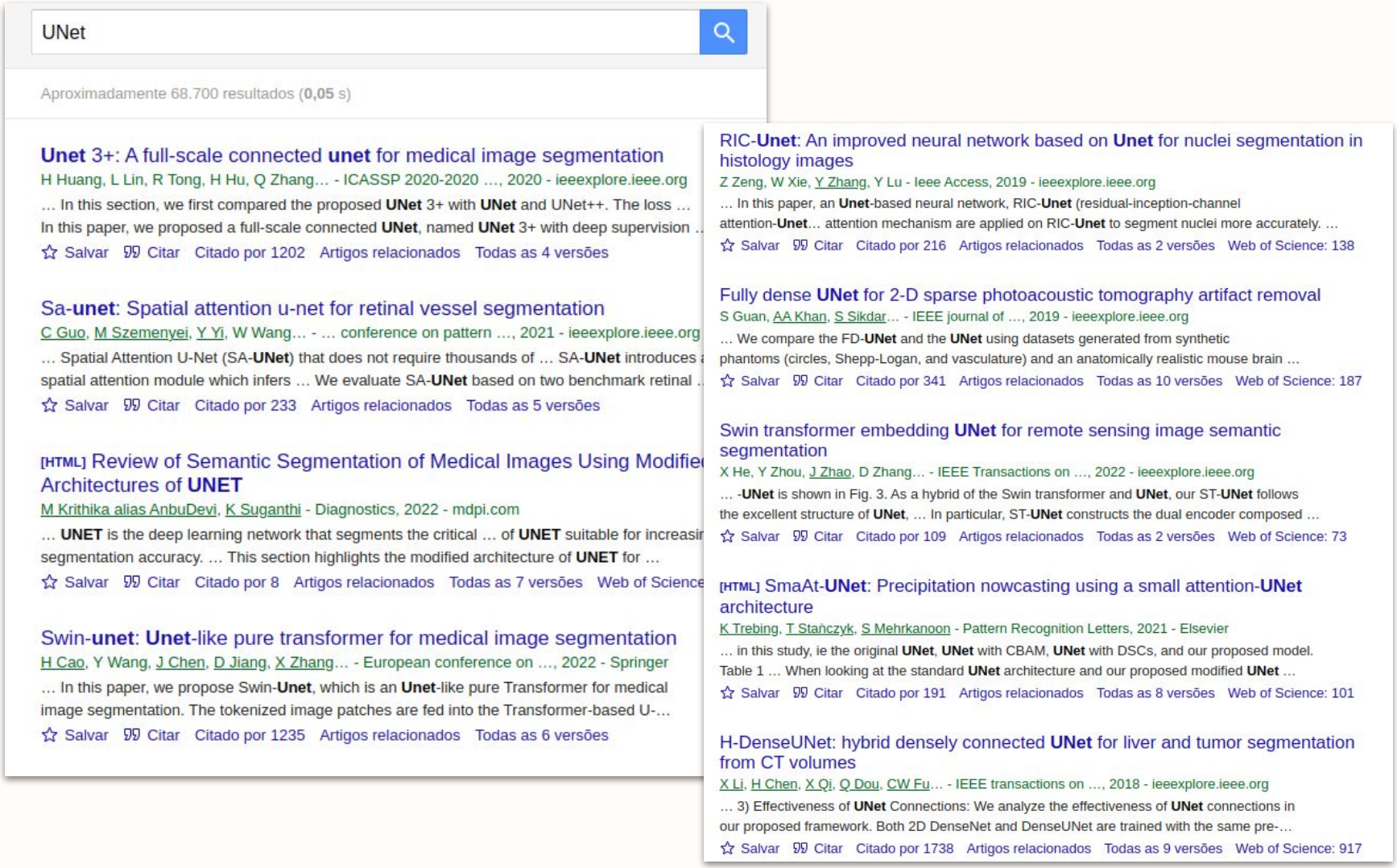
Task	Papers	Share
Semantic Segmentation	121	14.00%
Image Segmentation	98	11.34%
Medical Image Segmentation	63	7.29%
Denoising	39	4.51%
Tumor Segmentation	29	3.36%
Computed Tomography (CT)	20	2.31%
Image Generation	18	2.08%
Lesion Segmentation	18	2.08%
Super-Resolution	14	1.62%

Usage Over Time

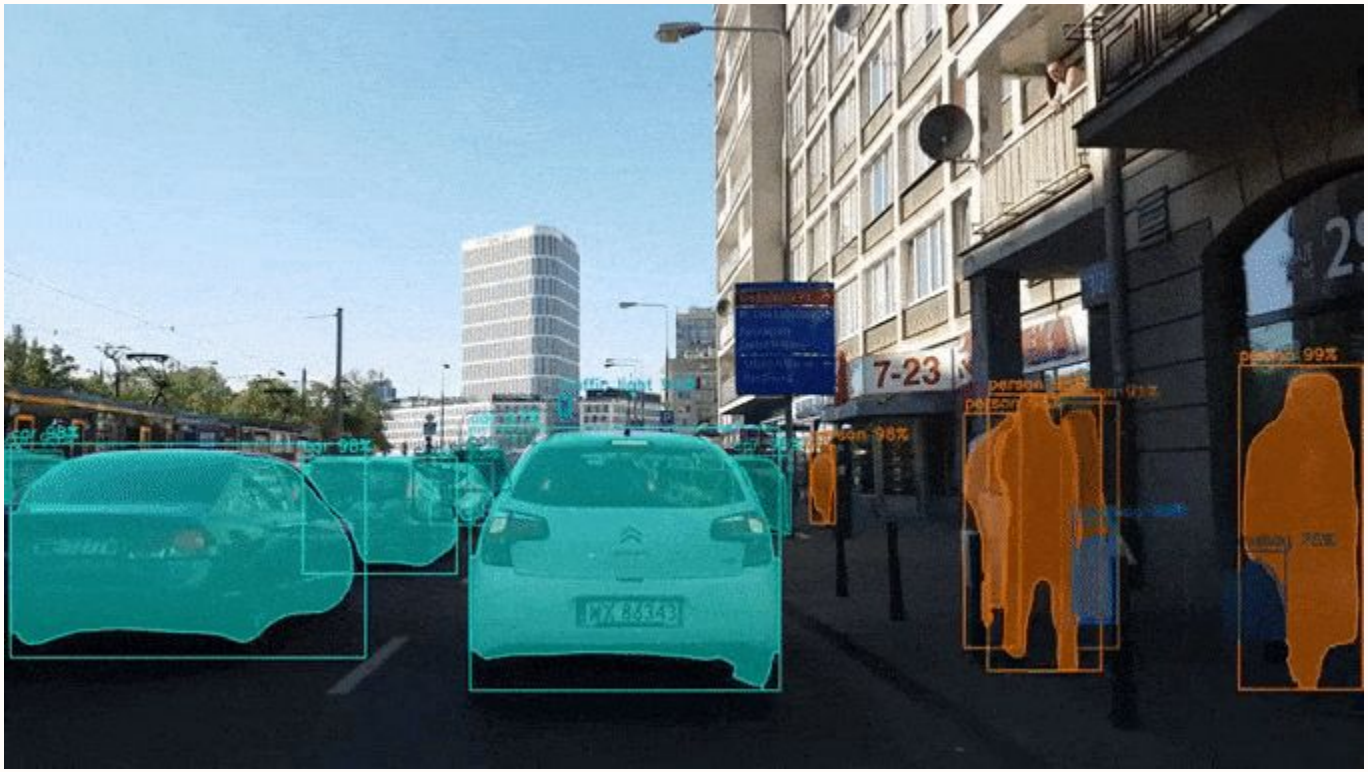


Dados e imagens obtidos de: https://paperswithcode.com/method/u-net. Acesso em out-2023

Variações UNet



Aplicações



https://github.com/matterport/Mask_RCNN

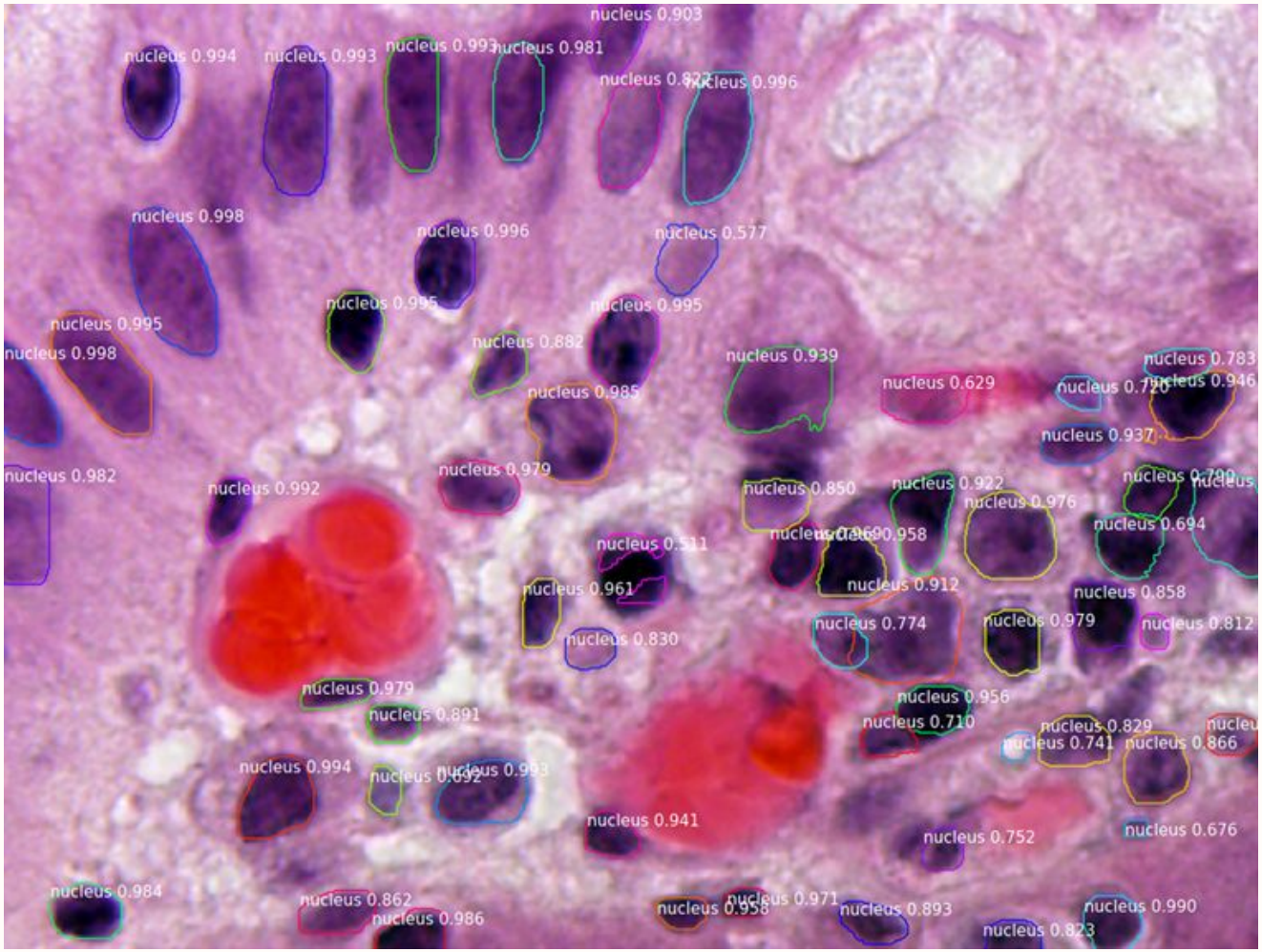
https://www.youtube.com/watch?v=OOT3UIXZztE

Aplicações

https://github.com/matterport/Mask_RCNN

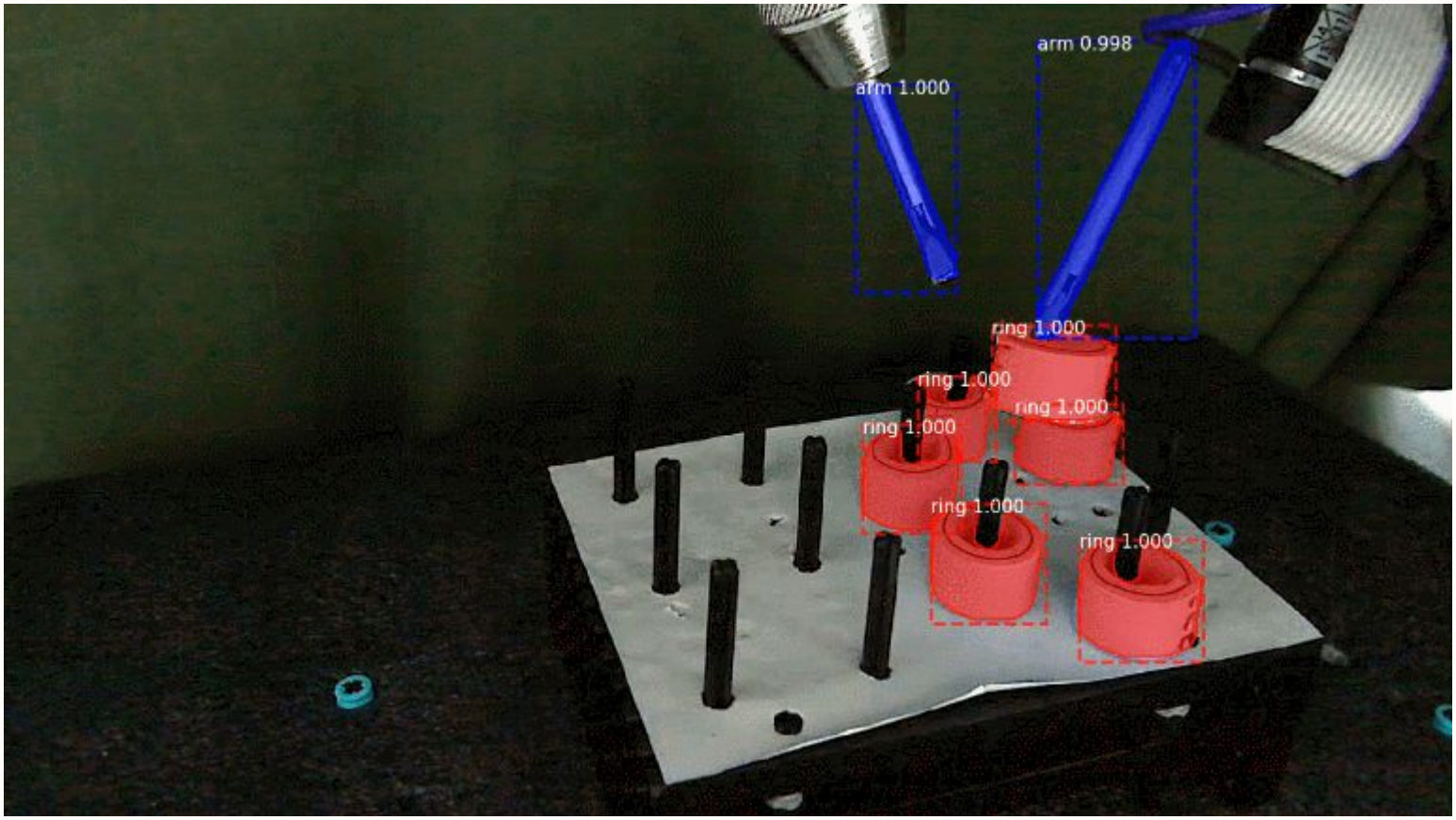
https://github.com/jremillard/images-to-osm

Aplicações



https://github.com/matterport/Mask_RCNN/tree/master/samples/nucleus

Aplicações



https://github.com/SUYEgit/Surgery-Robot-Detection-Segmentation

Aplicações - Restauração de imagens e deep learning

Visão geral

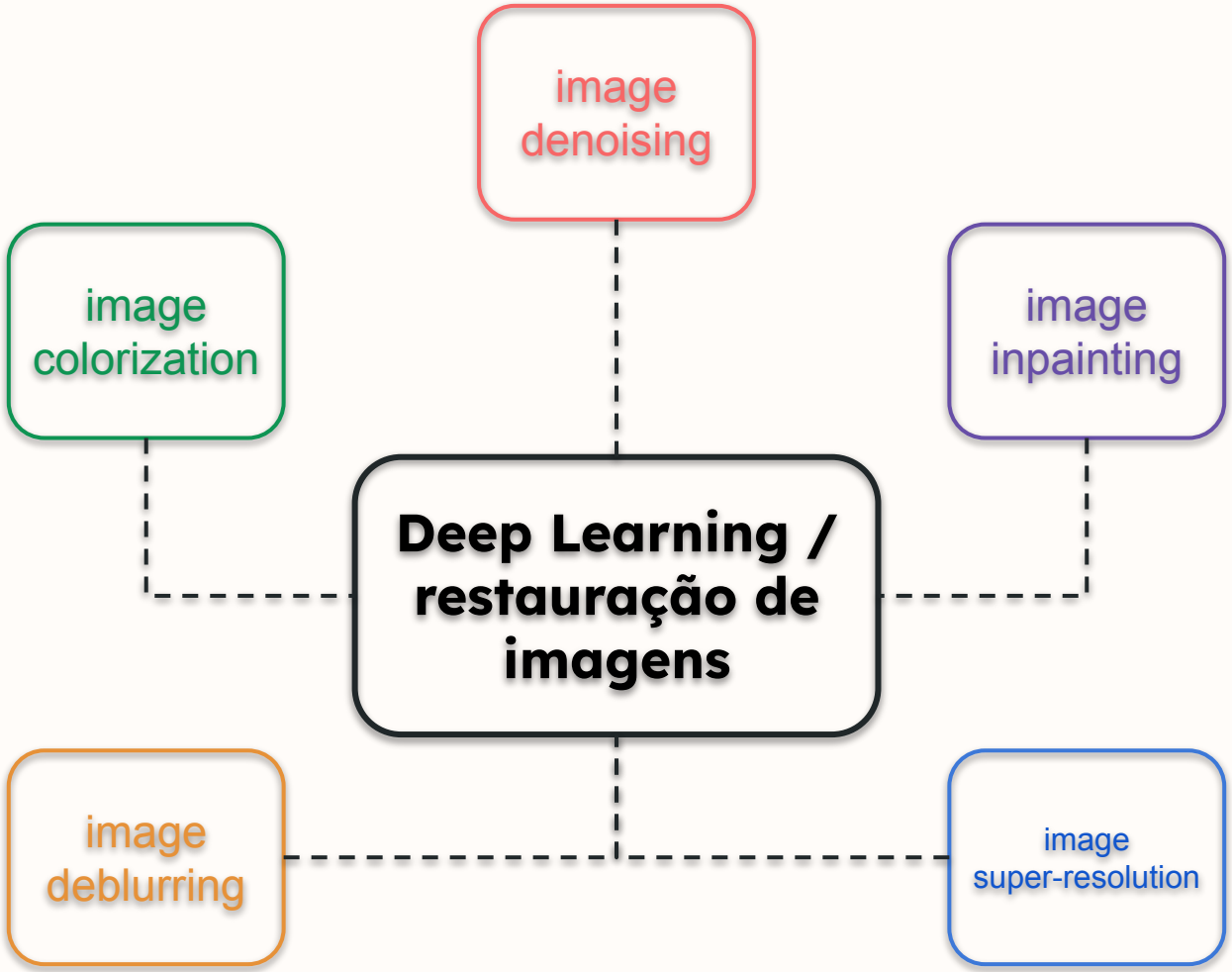


image denoising
image colorization
image inpainting
Deep Learning / restauração de imagens
image deblurring
image super-resolution

Restauração

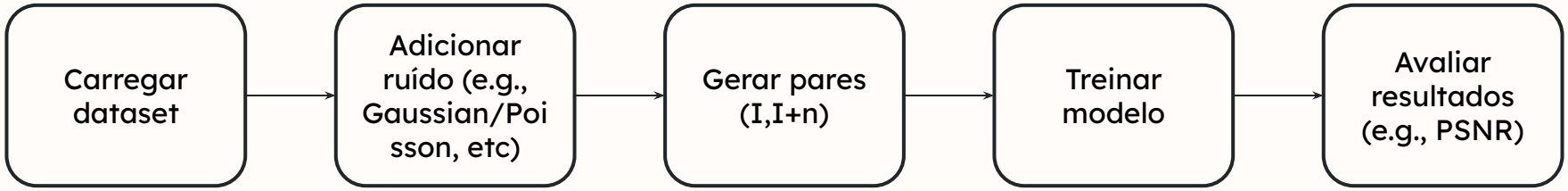


Image denoising e deblurring

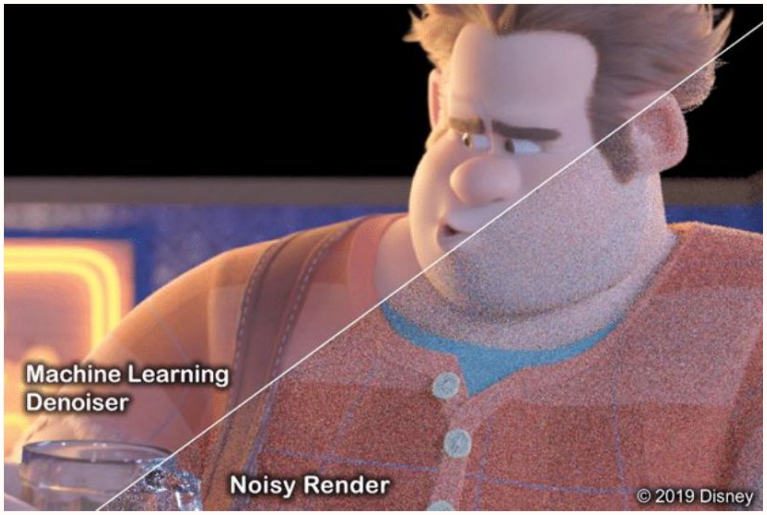


Image colorization

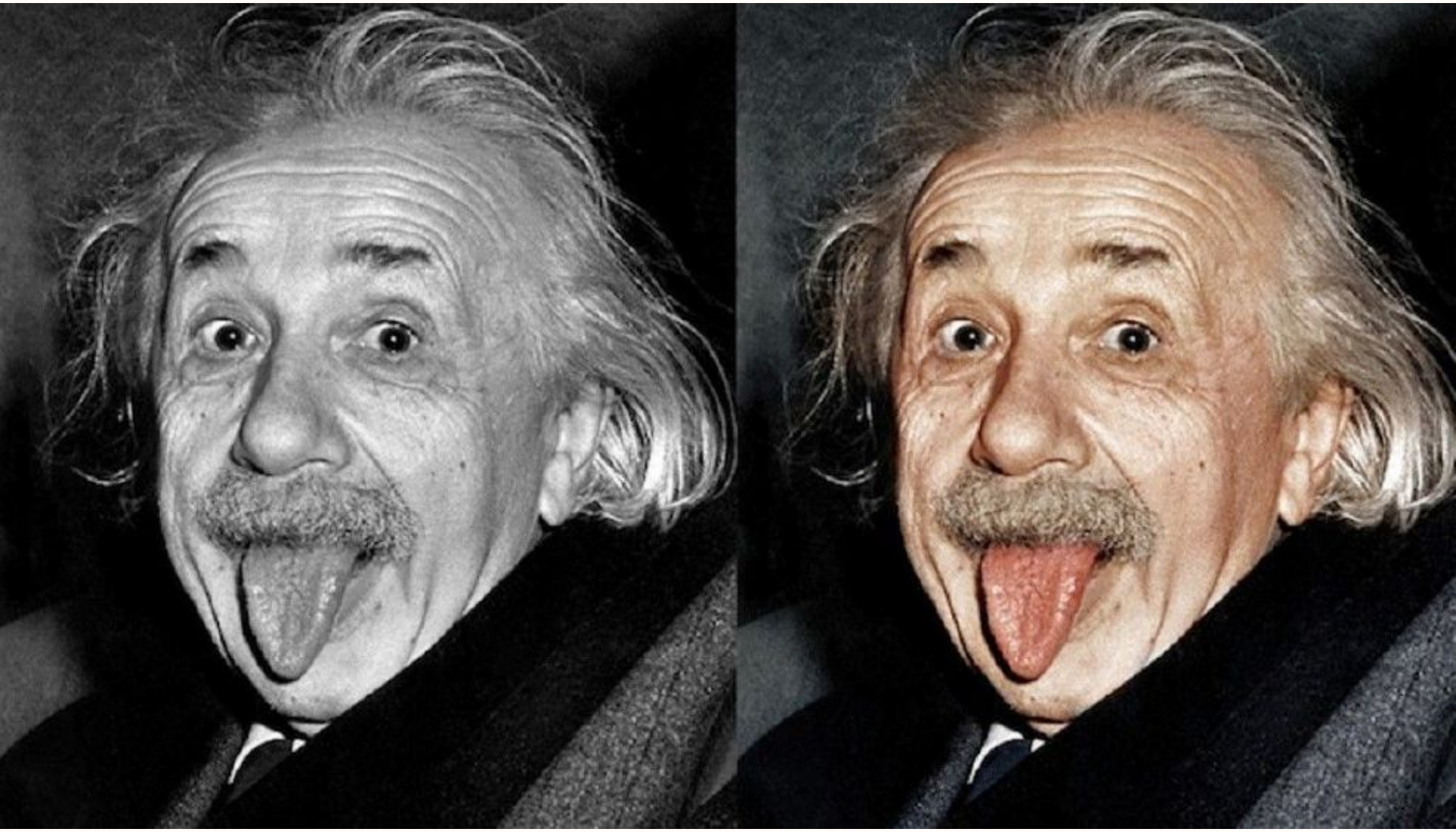
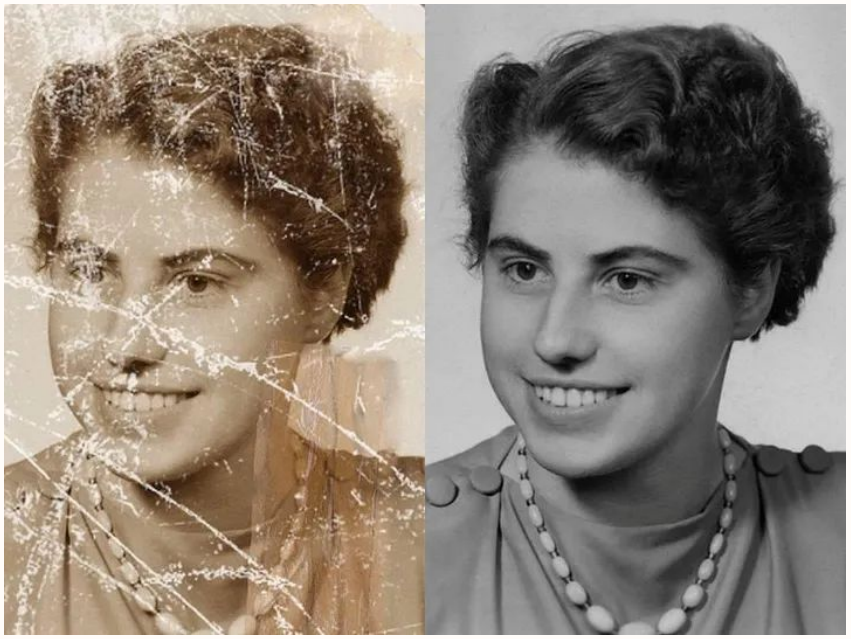


Image super-resolution

Image inpainting



Fonte das imagens: https://medium.com/jamieai/image-inpainting-with-deep-learning-dd8555e56a32. Acesso em Out-2023

Detecção e Segmentação de objetos

Aprendizado de máquina e aprendizado profundo - Lux.AI

luxai.cin.ufpe.br

INSTITUIÇÃO EXECUTORA

Centro de Informática UFPE

FADE UFPE

COORDENADORA

MCTI FUTURO

Softex

APOIO

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO

GOVERNO FEDERAL BRASIL UNIÃO E RECONSTRUÇÃO