

# *Segmentação Semântica*

Prof. Jefersson A. dos Santos

[jefersson@dcc.ufmg.br](mailto:jefersson@dcc.ufmg.br)



DCC  
DEPARTAMENTO DE  
CIÊNCIA DA COMPUTAÇÃO

U F *m* G

# Roteiro

## Aulas anteriores

- Regressão
- Otimização
- Redes em múltiplas camadas
- Redes convolucionais

## Próximas aulas

- Aplicações de CNNs
  - Segmentação semântica
  - Detecção de objetos
- Aprendizado semi-supervisionado
  - Auto Encoders
  - Variational Auto Encoders

# Roteiro

## Aula de hoje

- Segmentação de imagens
- Segmentação semântica como classificação de pixels
- FCNs
  - Upsampling e Learnable Sampling
- U-Nets
- SegNets
- Convoluçãoes dilatadas, multi-escala

# Até agora



This image is [CC0 public domain](#)

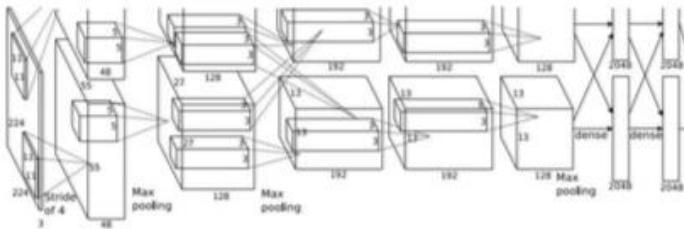


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

**Vector:**  
4096

**Fully-Connected:**  
4096 to 1000

# Até agora



This image is [CC0 public domain](#)

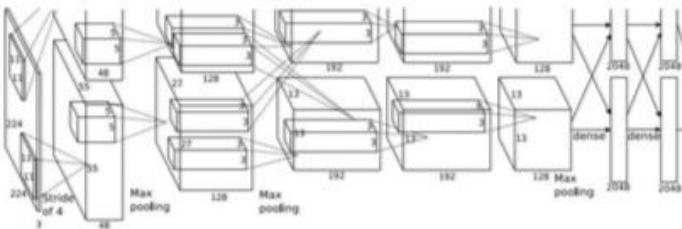


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

**Vector:**  
4096

**Fully-Connected:**  
4096 to 1000

**Class Scores**  
Cat: 0.9  
Dog: 0.05  
Car: 0.01  
...

# Aplicações de CNNs

Classification



CAT

No spatial extent

Semantic  
Segmentation



GRASS, CAT,  
TREE, SKY

No objects, just pixels

# Aplicações de CNNs

Classification



CAT

No spatial extent

Semantic  
Segmentation



GRASS, CAT,  
TREE, SKY

No objects, just pixels

Object  
Detection



DOG, DOG, CAT

# Aplicações de CNNs

Classification



CAT

No spatial extent

Semantic  
Segmentation



GRASS, CAT,  
TREE, SKY

No objects, just pixels

Object  
Detection



DOG, DOG, CAT

Multiple Object

Instance  
Segmentation

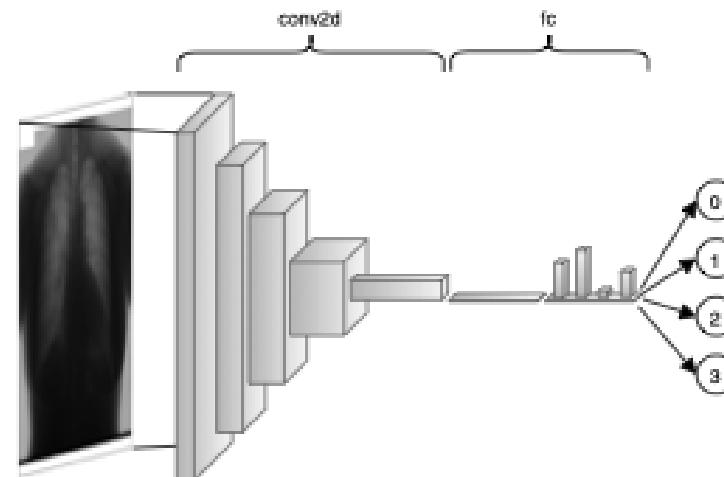


DOG, DOG, CAT

This image is CC0 public domain

# Predição esparsa

- Classificação é um problema de **predição esparsa**, pois queremos atribuir um rótulo a uma imagem
  - CNN é a arquitetura mais usada em imagens



# Predição esparsa

- Principais camadas
  - Camadas convolucionais
  - Camadas de Max-Pooling
  - Camadas Fully-Connected (FC)
- Outras camadas/funções
  - Rectified Linear Unit (ReLU)
  - Batch normalization
  - Softmax para classificação
  - Sigmóide para regressão

# Rotulação Esparsa → Rotulação Densa

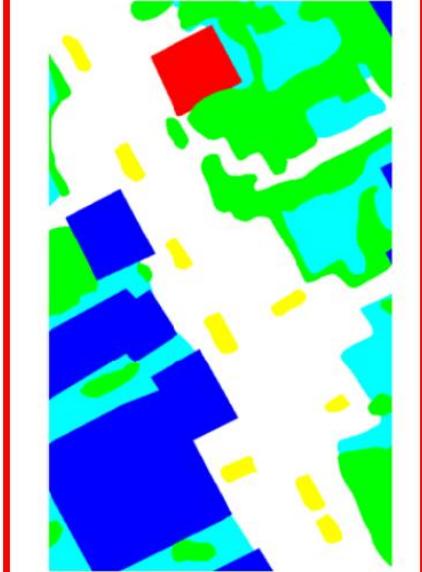
- Rotulação esparsa
  - Classificação de objetos
  - Regressão da probabilidade de malignidade em imagens radiológicas
  - Classificação entre imagens de plantação de café e de imagens urbanas
- Rotulação densa
  - Segmentação de objetos
  - Segmentação de tumores
  - Segmentação de regiões de plantação

# Rotulação Esparsa → Rotulação Densa

- Rotulações esparsas:
    - Classificação de Imagens
    - Reconhecimento de objetos
    - Classificação de classes
  - Rotulações densas:
    - Segmentação semântica
    - Segmentação de pixels
    - Segmentação de pixels
    - Segmentação de pixels
- Image Classification**



**Semantic Segmentation**

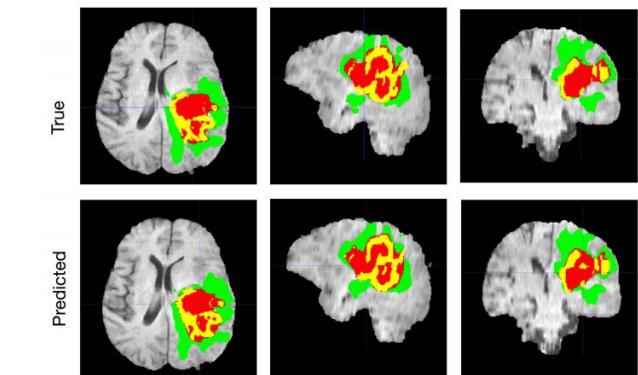
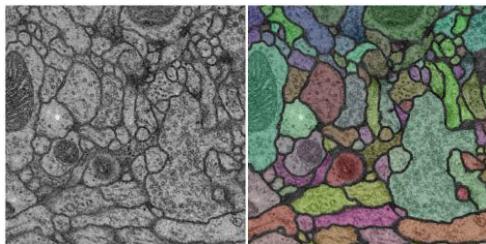
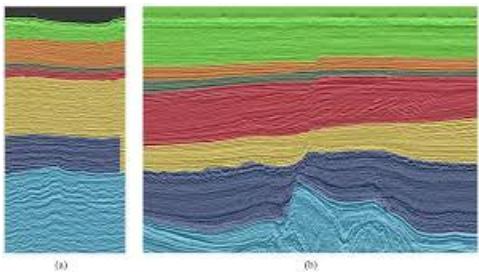


**Detection/Localization**


- lógicas  
s urbanas

# Aplicações

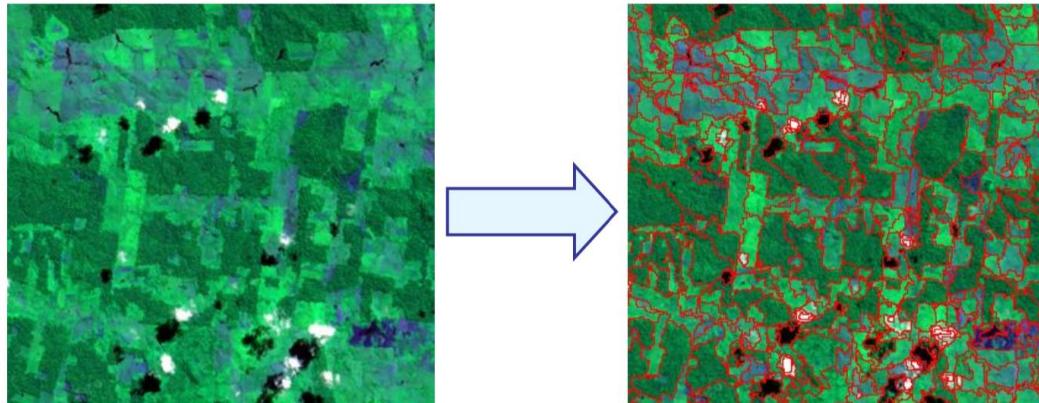
- Diagnóstico médico
- Mapeamento geográfico
- Geologia
  - Mapeamento de lavra (mineração)
  - Exploração de petróleo (sísmica)
- ...



# *Segmentação de Imagens*

# Segmentação Baseada em Regiões

- Objetivo principal:
  - identificar sub-regiões significativas/homogêneas
- Processo não-supervisionado!



$$R = R_1 \cup R_2 \cup \dots \cup R_n$$

# Segmentação Baseada em Regiões

- Objetivo
  - ideia de segmentação
- Processo

## Basic Formulation

Let  $R$  represent an image. Segmentation may be viewed as the process of partitioning  $R$  in  $n$  subregions  $R_1, R_2, \dots, R_n$ , such that:

a)  $\bigcup_{i=1}^n R_i = R$

{ covers all image

b)  $R_i$  is a connected region

{ each region is a  
connected component

c)  $R_i \cap R_j = \emptyset$  for all  $i$  and  $j$ ,  $i \neq j$

{ regions are disjoint

d)  $P(R_i) = \text{TRUE}$  for  $i=1,2,\dots,n$

{  $P$  defines the membership  
condition

e)  $P(R_i \cup R_j) = \text{FALSE}$  for  $i \neq j$

{  $P$  also differentiates the  
regions.

# Métodos Clássicos de Segmentação

- Watershed
- Mean-shift
- Graph-cut
- SLIC

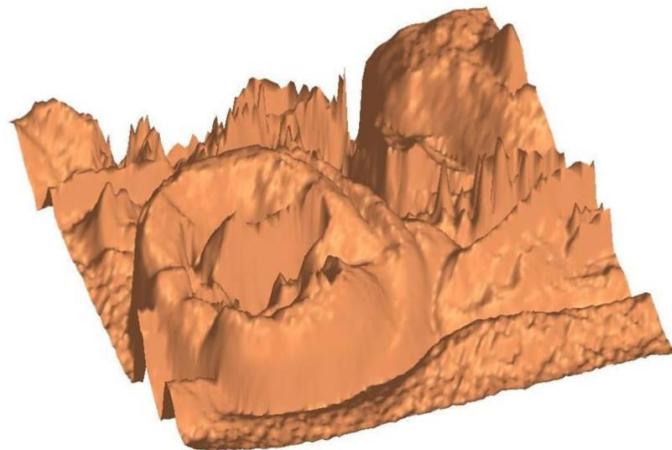
# Métodos Clássicos de Segmentação

- **Watershed**
- Mean-shift
- Graph-cut
- SLIC

A imagem é tratada em 3D (terceira dimensão é o nível de cinza).



original image



“topographic” view of the image

# Métodos Clássicos de Segmentação

- **Watershed**
- Mean-shift
- Graph-cut
- SLIC

Three types of points:

**miminum:** points belonging to a regional minimum.

**watershed of a mimimum:** points at which, a drop of water, if placed at the location of any of those points, would fall with certainty to a single minimum.

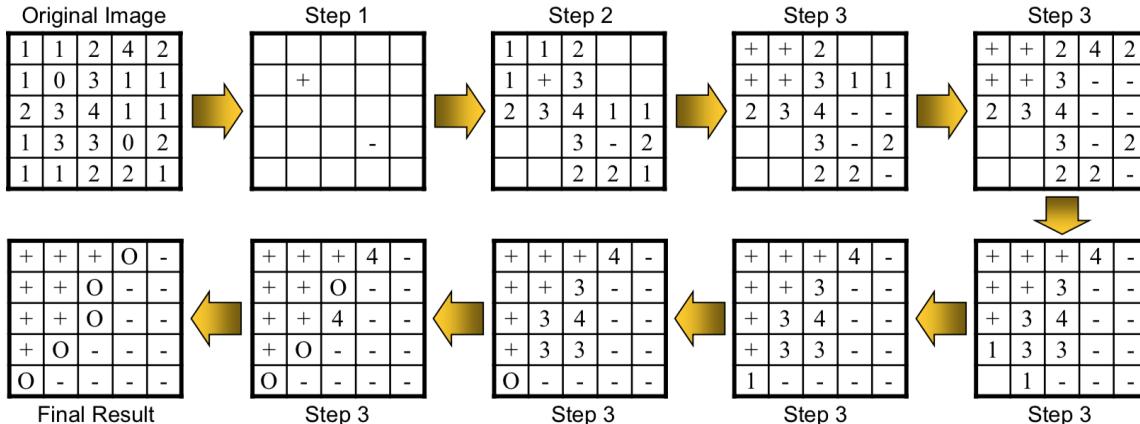
**watershed lines:** points at which water would be equally likely to fall to more than one such minimum.



“topographic” view

# Métodos Clássicos de Segmentação

- **Watershed**
- Mean-shift
- Graph-cut
- SLIC



## Meyer's flooding algorithm (Watershed):

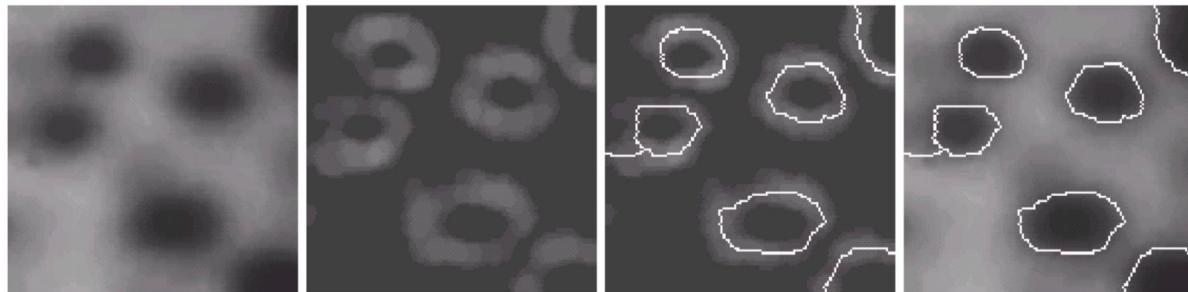
1. A set of markers, pixels where the flooding shall start, are chosen. Each is given a different label.
2. The neighbors of each marker are inserted into a priority queue with a priority level corresponding to the gray level of the pixel (low value = high priority).
3. The pixel with the highest priority level is extracted from the priority queue:
  - If all neighbors that have already been labeled have the same label, then mark the pixel with that label.
  - All non-labeled neighbors that are not yet in the priority queue are inserted into the priority queue.
4. Redo step 3 until the priority queue is empty.

The non-labeled pixels are the watershed lines.

# Métodos Clássicos de Segmentação

- **Watershed**
- Mean-shift
- Graph-cut
- SLIC

Para obter objetos homogêneos, o algoritmo *watershed* é aplicado à magnitude do gradiente.



image

gradient

borders  
superimposed to  
the gradient

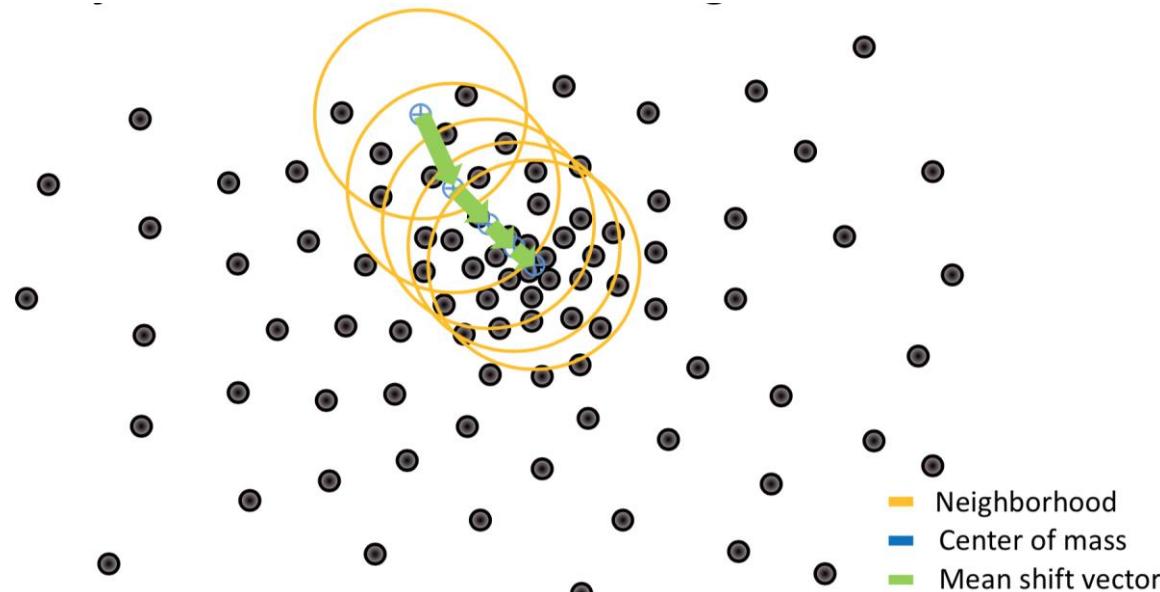
borders  
superimposed to  
the original image

# Métodos Clássicos de Segmentação

- Watershed
- **Mean-shift**
- Graph-cut
- SLIC

Uma abordagem para encontrar agrupamentos em um conjunto de amostras de dados partir de uma função de densidade de probabilidade subjacente no  $\mathbb{R}^N$

**Objetivo:** encontrar a região mais densa

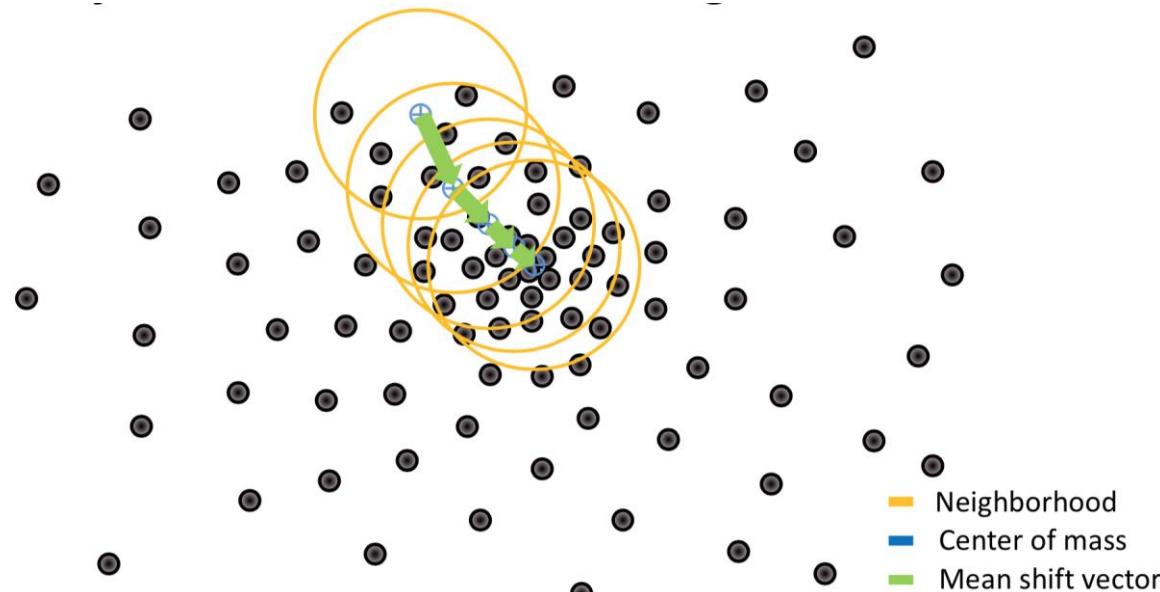


# Métodos Clássicos de Segmentação

- Watershed
- **Mean-shift**
- Graph-cut
- SLIC

Uma abordagem para encontrar agrupamentos em um conjunto de amostras de dados partir de uma função de densidade de probabilidade subjacente no  $\mathbb{R}^N$

**Objetivo:** encontrar a região mais densa



# Métodos Clássicos de Segmentação

- Watershed
- **Mean-shift**
- Graph-cut
- SLIC

Computando o vetor Mean-shift

**Arithmetic mean:**

$$\mathbf{m}(\mathbf{x}) = \left( \frac{1}{n_x} \sum_{i=1}^{n_x} \mathbf{x}_i \right) - \mathbf{x}$$

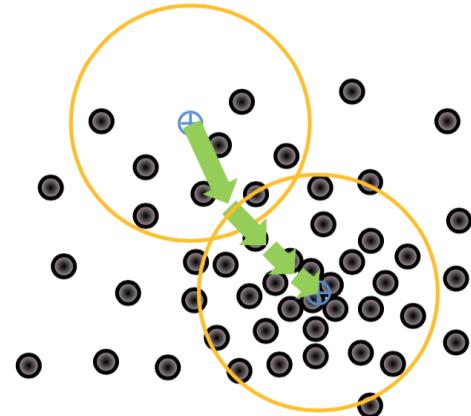
where

$\mathbf{x}$  is the current centroid

$\mathbf{m}(\mathbf{x})$  is the **mean shift** vector for  $\mathbf{x}$

$\mathbf{x}_i$  are the points inside the neighborhood centered in  $\mathbf{x}$

$n_x$  is the number of points inside the neighborhood



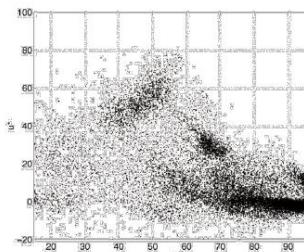
The **mean shift vector** points to the direction of the **maximum increase of density**.

# Métodos Clássicos de Segmentação

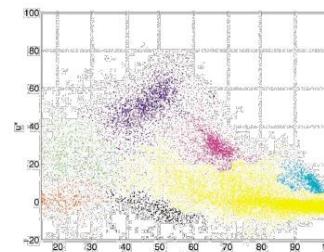
- Watershed
- **Mean-shift**
- Graph-cut
- SLIC

## Mean-shift clustering

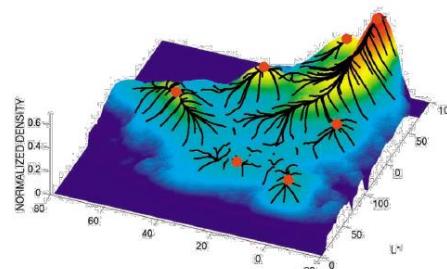
- Cluster: todos os pontos de dados na *bacia de atração*.
- *Bacia de atração*: a região para a qual todas as trajetórias levam ao mesmo modo.



(a)



(b)



Adaptive  
gradient ascent!



Comaniciu, D. and Meer, P. (2002). Mean Shift: A Robust Approach Toward Feature Space Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5), pp. 603–619.

# Métodos Clássicos de Segmentação

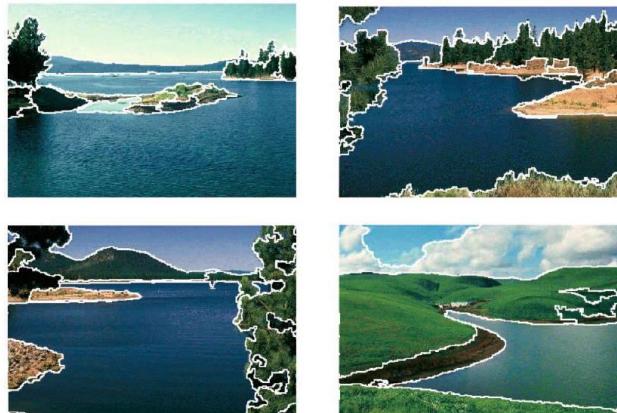
- Watershed
- **Mean-shift**
- Graph-cut
- SLIC

## Mean-shift segmentation

- As imagens são representadas em 2D (*space domain*) de  $p$  pixels dimensionais (*range domain*).
- Para usar a métrica euclidiana, cada domínio é normalizado

$$K(\mathbf{x} - \mathbf{x}_i) = \frac{c}{h_s^2 h_r^p} k\left(\left\|\frac{\mathbf{x}^s - \mathbf{x}_i^s}{h_s}\right\|^2\right) k\left(\left\|\frac{\mathbf{x}^r - \mathbf{x}_i^r}{h_r}\right\|^2\right)$$

window size in space domain      window size in range domain



### Vantagens

- Não assume formas pré-definidas
- Robusto a outliers

### Desvantagens

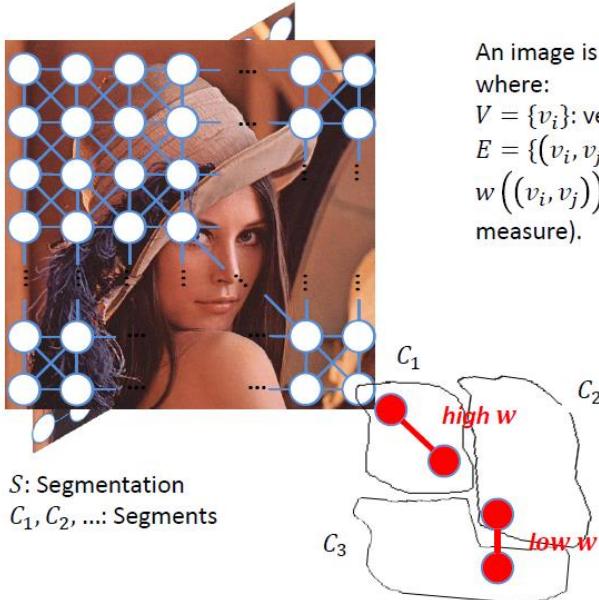
- Depende do tamanho da janela
- Caro computacionalmente

# Métodos Clássicos de Segmentação

- Watershed
- Mean-shift
- **Graph-cut**
- SLIC

## Graph-cut

- A imagem é vista como um grafo ponderado
- O particionamento ideal do grafo é aquele que minimiza o valor de corte.



An image is modeled as an undirected graph  $G = (V, E)$ ,  
where:

$V = \{v_i\}$ : vertex (○)

$E = \{(v_i, v_j)\}$ : Edge (-)

$w((v_i, v_j))$ : weight associated to an edge (affinity measure).

Shi, J. and Malik, J., (2000) Normalized Cuts and Image Segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 22 (8), pp.888-905 .

# Métodos Clássicos de Segmentação

- Watershed
- Mean-shift
- **Graph-cut**
- SLIC

## Exemplos:



## Vantagens

- Estrutura genérica, pode ser usada com muitos diferentes características e formulações de similaridade
- Fornece segmentos regulares

## Desvantagens

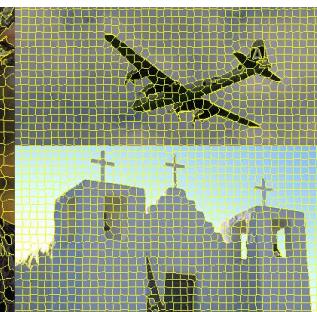
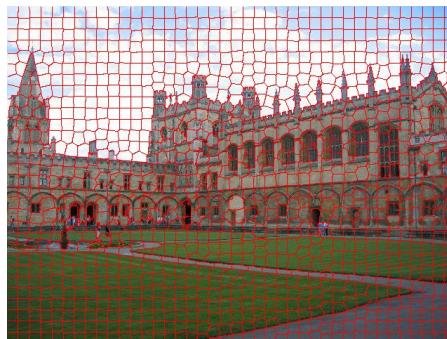
- Necessidade de escolher o número de segmentos
- Alta exigência de armazenamento e complexidade de tempo
- Tendência a particionar em tamanhos iguais segmentos

# Métodos Clássicos de Segmentação

- Watershed
- Mean-shift
- Graph-cut
- **SLIC**

## Simple Linear Iterative Clustering – SLIC

“Superpixels são regiões atômicas perceptivamente significativas. Eles fornecem uma primitiva conveniente a partir da qual é possível extrair características da imagem e reduzir bastante a complexidade de tarefas de processamento subsequentes.”



Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P. Süsstrunk, S., “SLIC Superpixels Compared to State-of-the-art Superpixel Methods,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 34(11), pp. 2274 – 2282, 2012.

# Métodos Clássicos de Segmentação

- Watershed
- Mean-shift
- Graph-cut
- **SLIC**

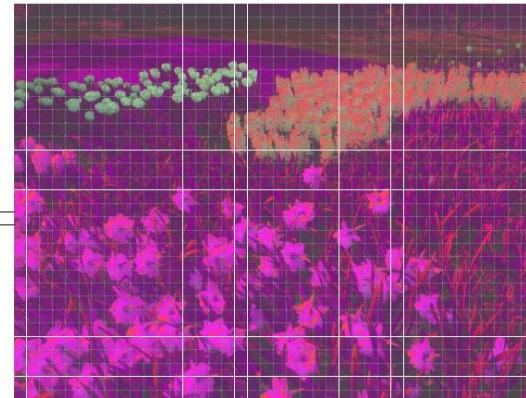
## Simple Linear Iterative Clustering – SLIC

1. Convert the RGB image to CIELAB color space.
2. Initialize cluster centers  $C_k = [l_k; a_k; b_k; x_k; y_k]^T$  by sampling pixels at regular grid steps  $S$ .

$$\sqrt{\frac{N}{K}} = S$$

number of pixels  
in the image

desired number  
of superpixels



# Métodos Clássicos de Segmentação

- Watershed
- Mean-shift
- Graph-cut
- **SLIC**

## Simple Linear Iterative Clustering – SLIC

3. Set distance  $d(i) = -\infty$  for each pixel  $i$ .

$$d = \begin{cases} d_s & \text{if } j \in \text{neighborhood}(i) \\ d_c & \text{if } j \in \text{same cluster}(i) \\ \infty & \text{otherwise} \end{cases}$$

where

$$d_s = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$$
$$d_c = \sqrt{(l_j - l_i)^2 + (a_j - a_i)^2 + (b_j - b_i)^2}$$
$$D' = \sqrt{\left(\frac{d_c}{m}\right)^2 + \left(\frac{d_s}{S}\right)^2}$$

# Métodos Clássicos de Segmentação

- Watershed
- Mean-shift
- Graph-cut
- **SLIC**

## Simple Linear Iterative Clustering – SLIC

4. Set distance  $d(i) = -\infty$  for each pixel  $i$ .

$$d = \sqrt{(d_c)^2 + \left(\frac{d_s}{S}\right)^2 m^2}$$

The diagram shows a 4x6 grid of pixels. Most pixels have a value of  $\infty$ , except for one central pixel which has a value of 1. A yellow arrow points from the term  $m^2$  in the equation to this central pixel, indicating its role in controlling the relative importance of shape and color.

controls the relative importance of shape and color

large  $m \rightarrow$  favors more compact (lower area to perimeter ratio) superpixels.  
small  $m \rightarrow$  favors more adherence to edges.

# Segmentação Semântica “shallow”

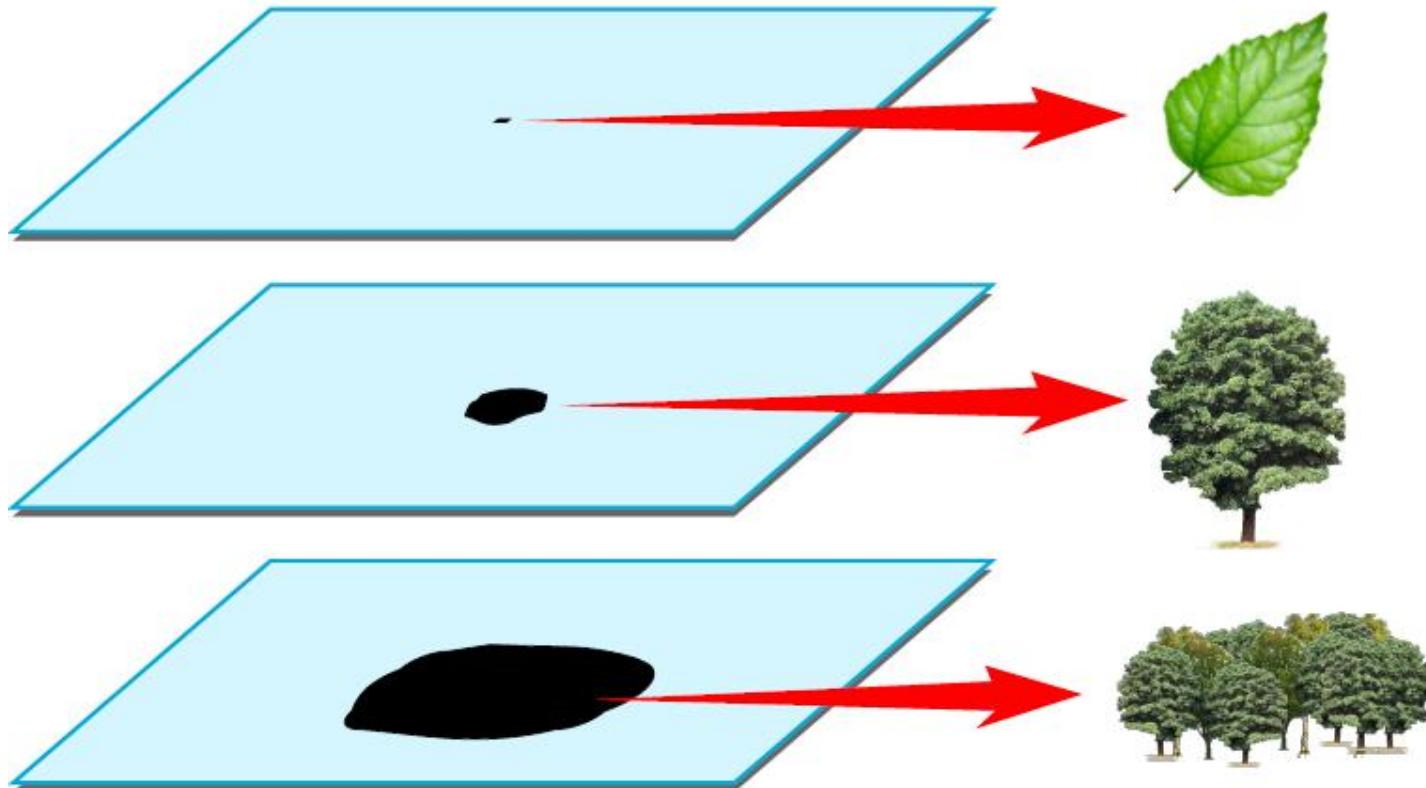
## Building



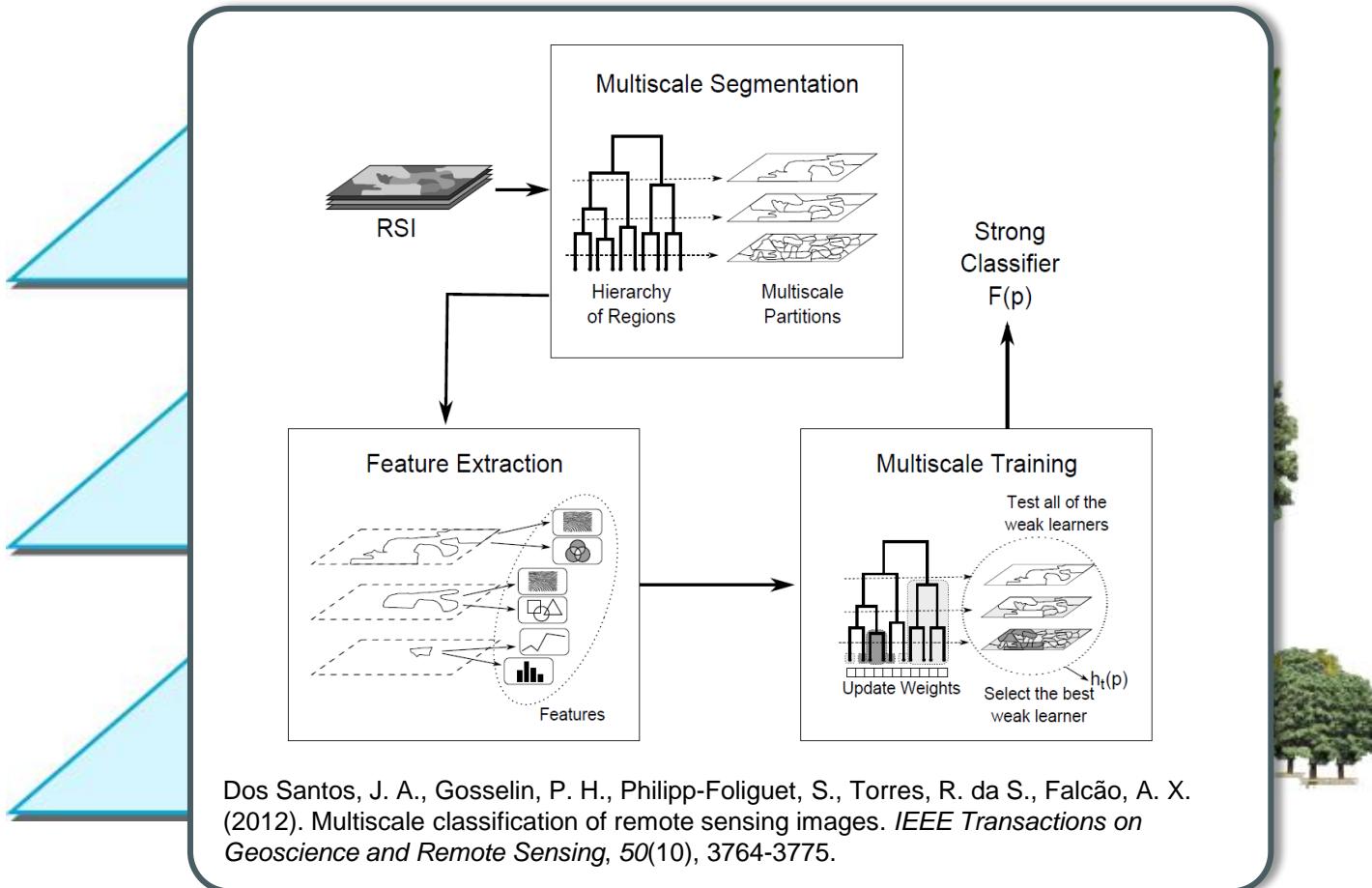
## Using



# Segmentação Semântica “shallow”



# Segmentação Semântica “shallow”

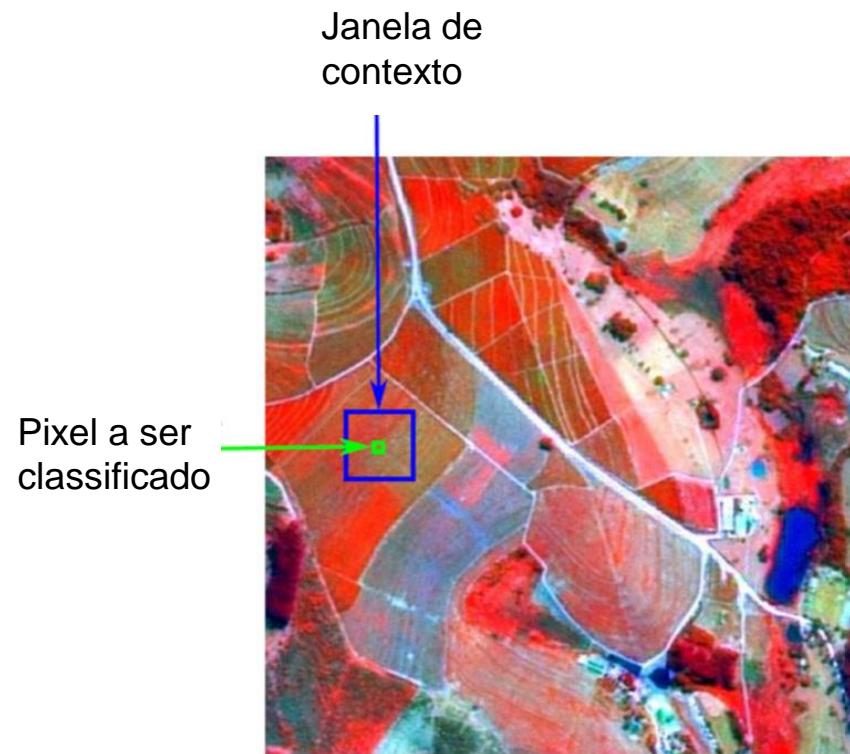


# *Abordagens profundas para segmentação semântica*

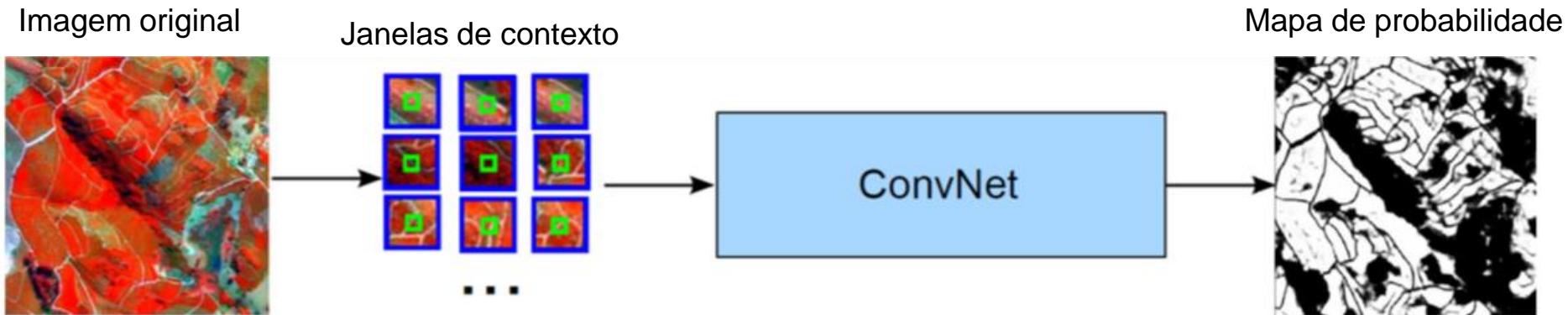
# Abordagens profundas de segmentação

- Abordagens profundas para rotulação densa
  - Classificação de pixels
  - Fully Convolutional Networks (FCNs)
  - Redes de Deconvolução
    - DeconvNets
    - U-Nets
    - SegNets

# Classificação de pixels

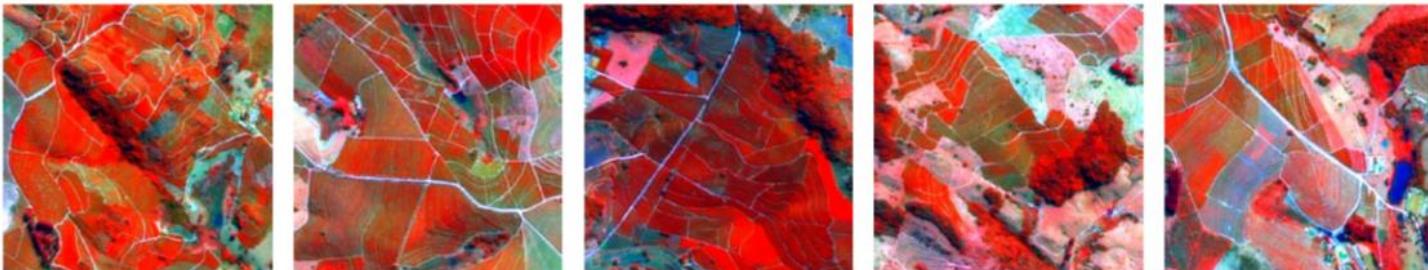


# Classificação de pixels



# Resultados

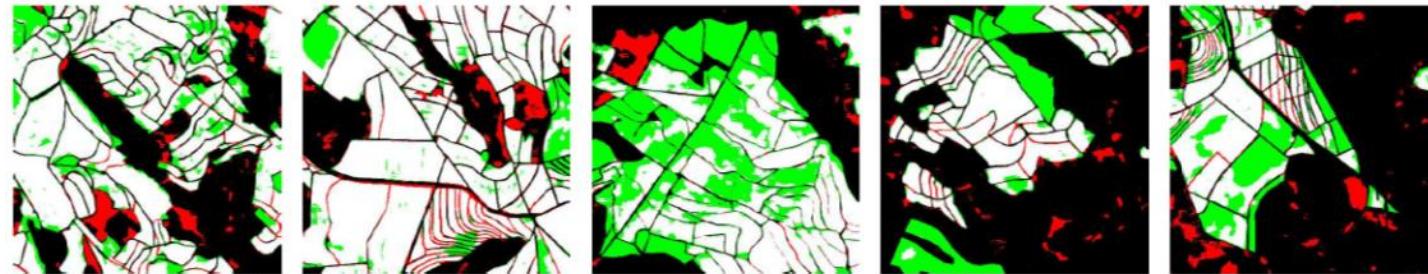
Imagen



ground-truth

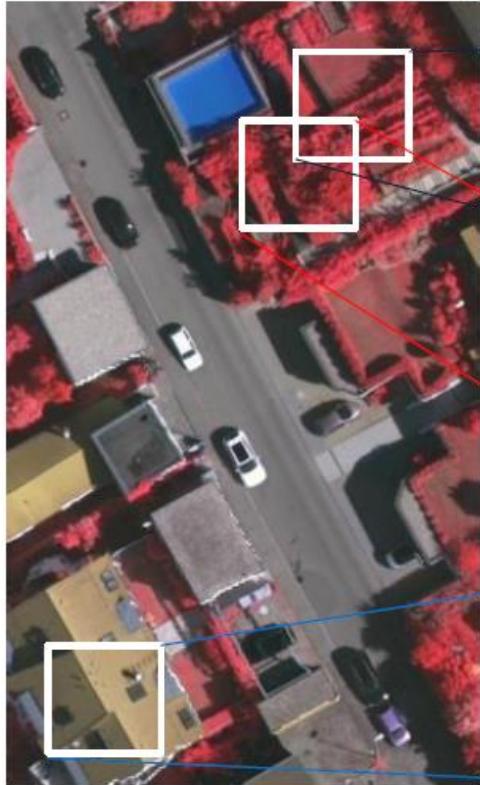


Mapa de  
relevância



□ True Positive ■ True Negative ■ False Positive ■ False Negative

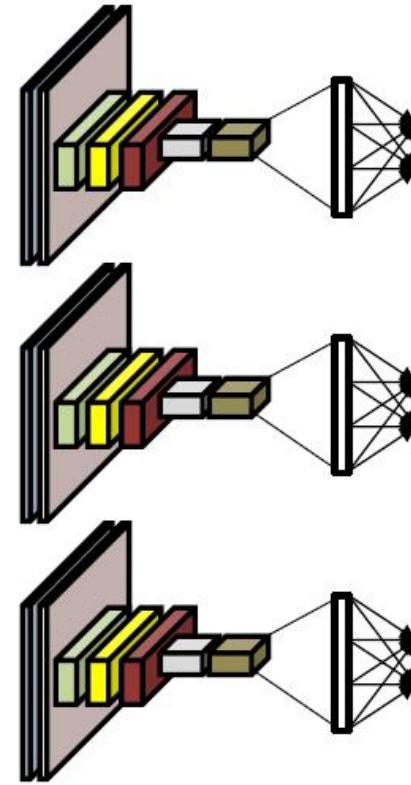
input image



crop  
patch



classify center  
pixel with a CNN



garden

garden

roof

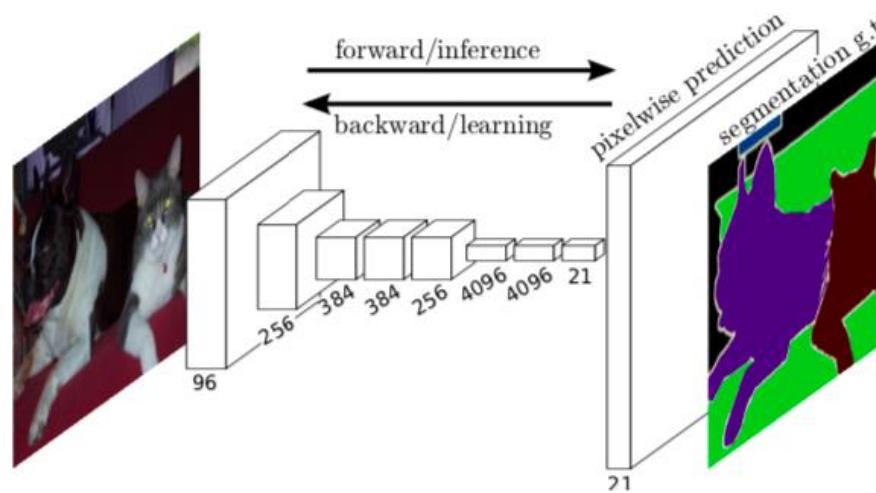
Operações redundantes devido aos overlaps. Muito ineficiente!

# *Fully Convolutional Networks (FCN)*

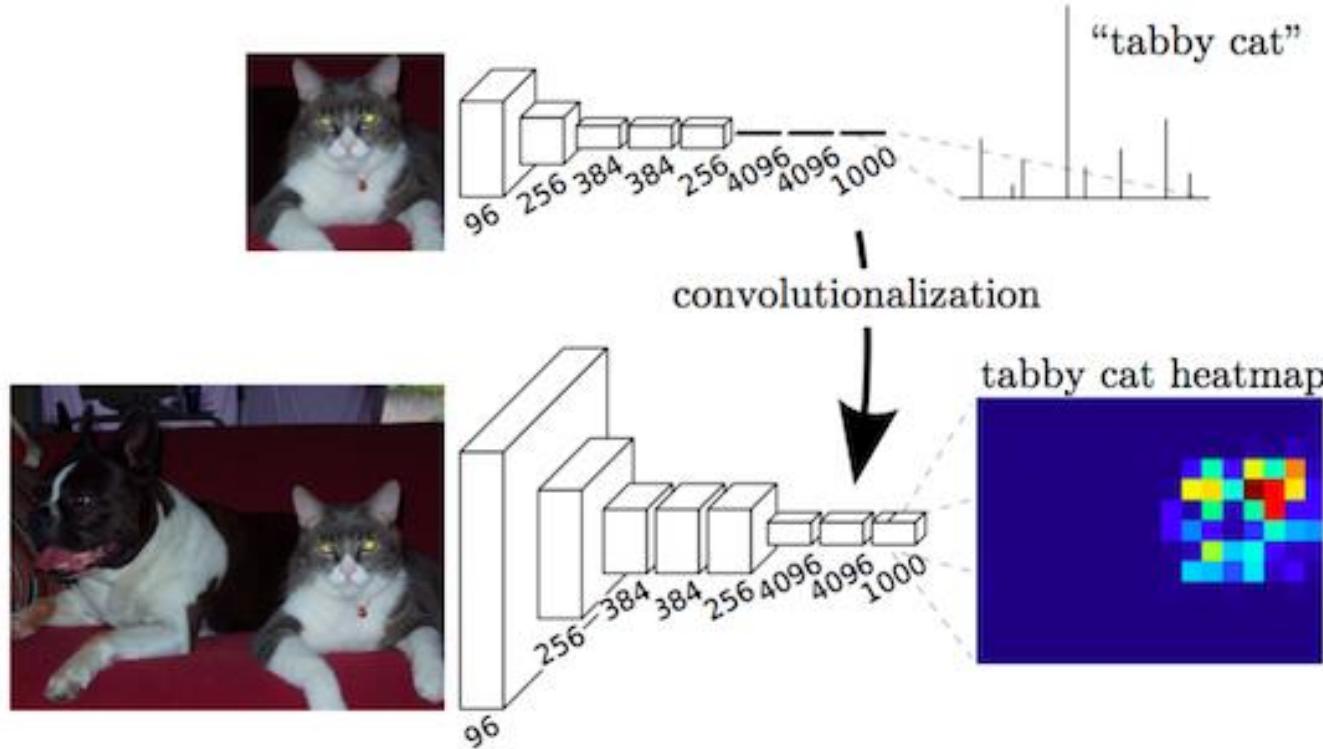
# Fully Convolutional Networks

- Fully Convolutional Networks (FCNs)

"Fully" tem a ver com o fato usar imagem inteira ao invés de patches



# Fully Convolutional Networks



# Última camada da FCN

- A última camada deve ter a mesma resolução da imagem
- Todas as camadas que vimos até agora fazem downsampling da entrada, i.e. diminuem a resolução
- Como fazer o upscaling na última camada para retornar ao tamanho original? Como recuperar a localização que foi perdida durante as operações de pooling?  
Vamos retornar a esta pergunta em breve.
- Note que a penúltima camada contém um feature map com uma resolução muito menor que a da imagem original, e que deve ter um número de canais/kernels igual ao número de classes do dataset

# Como fazer *upsampling in-network*?

Dada a seguinte entrada  $2 \times 2$ , como obter uma saída  $4 \times 4$ ?

Entrada:  
 $2 \times 2$

1	2
3	4

# Como fazer *upsampling in-network*?

Dada a seguinte entrada  $2 \times 2$ , como obter uma saída  $4 \times 4$ ?

Entrada:  
 $2 \times 2$

1	2
3	4

Saída:  $4 \times 4$

**Nearest Neighbor**

1	1	2	2
1	1	2	2
3	3	4	4
3	3	4	4

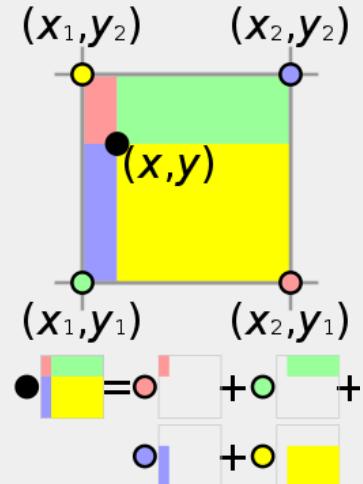
**Bed of Nails  
"Cama de pregos"**

1	0	2	0
0	0	0	0
3	0	4	0
0	0	0	0

**Interpolação bilinear**

1		2	
3		4	

Interpolação bilinear.

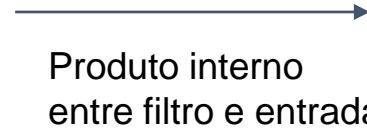


calcula cada saída  $y_{ij}$  a partir das 4 entradas mais próximas usando um mapa linear que depende apenas das posições relativas da saída e das entradas

# Upsampling com aprendizado: transpose convolution

Relembrando: convolução 3 x 3 típica, stride=1, pad=1


Entrada: 4x4

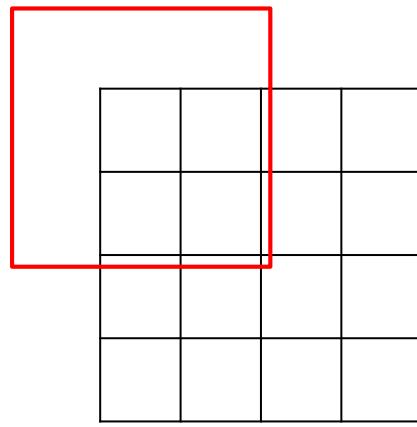


→  
Produto interno  
entre filtro e entrada


Saída: 4x4

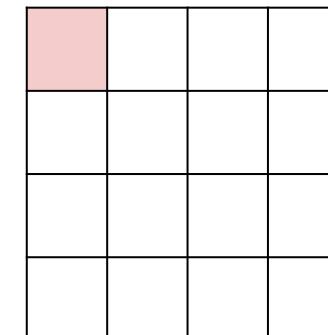
# Upsampling com aprendizado: transpose convolution

Relembrando: convolução 3 x 3 típica, stride=1, pad=1



Entrada: 4x4

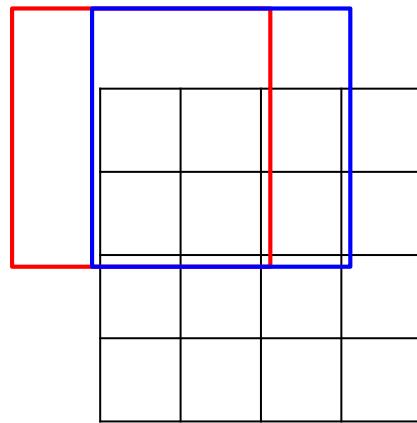
→  
Produto interno  
entre filtro e entrada



Saída: 4x4

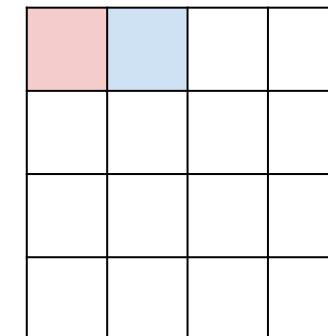
# Upsampling com aprendizado: transpose convolution

Relembrando: convolução 3 x 3 típica, stride=1, pad=1



Entrada: 4x4

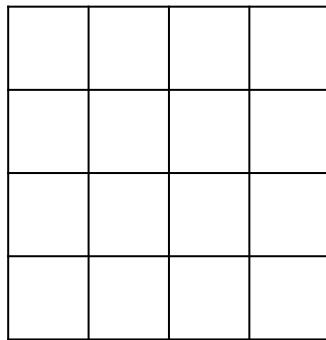
→  
Produto interno  
entre filtro e entrada



Saída: 4x4

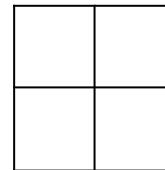
# Upsampling com aprendizado: transpose convolution

Relembrando: convolução 3 x 3 típica, stride=2, pad=1



Entrada: 4x4

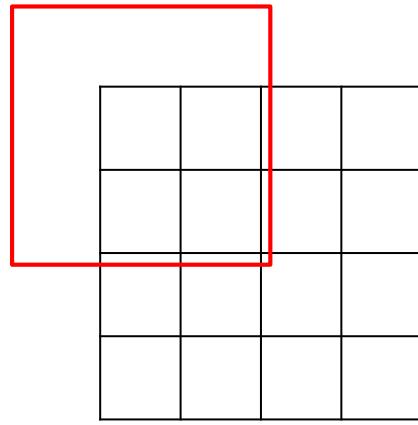
→  
Produto interno  
entre filtro e entrada



Saída: 2x2

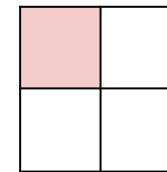
# Upsampling com aprendizado: transpose convolution

Relembrando: convolução 3 x 3 típica, stride=2, pad=1



Entrada: 4x4

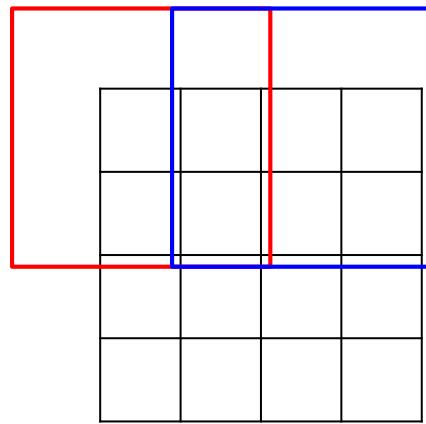
→  
Produto interno  
entre filtro e entrada



Saída: 2x2

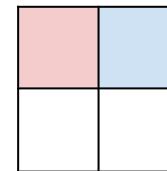
# Upsampling com aprendizado: transpose convolution

Relembrando: convolução 3 x 3 típica, stride=2, pad=1



Entrada: 4x4

→  
Produto interno  
entre filtro e entrada



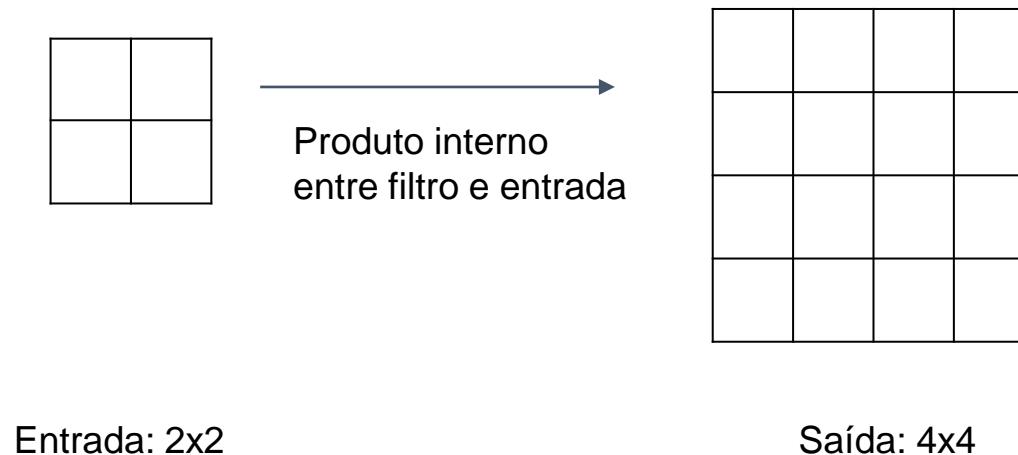
Saída: 2x2

Filtro move 2 pixels na  
entrada para cada pixel  
na saída

Stride dá a razão entre o  
deslocamento na  
entrada e na saída

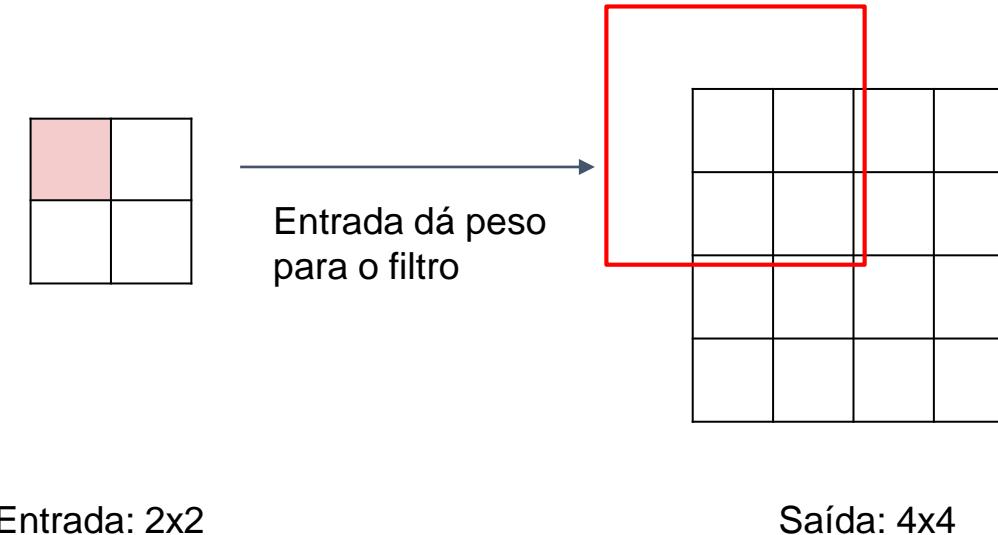
# Upsampling com aprendizado: transpose convolution

convolução **transposta**  $3 \times 3$ , stride=2, pad=1



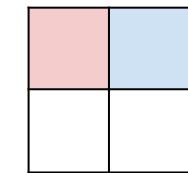
# Upsampling com aprendizado: transpose convolution

convolução **transposta**  $3 \times 3$ , stride=2, pad=1

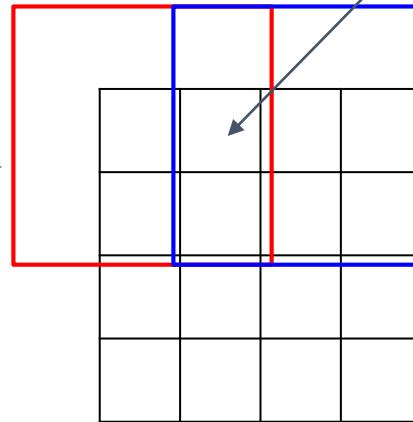


# Upsampling com aprendizado: transpose convolution

convolução **transposta**  $3 \times 3$ , stride=2, pad=1



Entrada dá peso para o filtro



Somamos onde saída se sobrepõe

Filtro move 2 pixels na saída para cada pixel na entrada

Stride dá a razão entre o movimento na saída e na entrada

Entrada:  $2 \times 2$

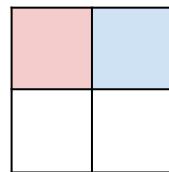
Saída:  $4 \times 4$

# Upsampling com aprendizado: transpose convolution

convolução **transposta**  $3 \times 3$ , stride=2, pad=1

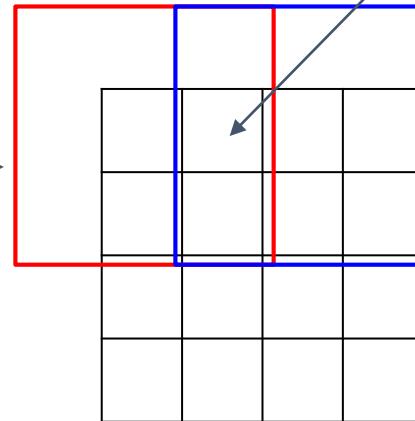
**Outros nomes:**

- Deconvolution
- Upconvolution
- Fractionally strided convolution
- Backward strided convolution



Entrada: 2x2

Entrada dá peso para o filtro



Saída: 4x4

Somamos onde saída se sobrepõe

Filtro move 2 pixels na saída para cada pixel na entrada

Stride dá a razão entre o movimento na saída e na entrada

# Convolução transposta: exemplo 1D

Exemplo com stride=2

**Input**

a
b

**Filter**

x
y
z

**Output**

ax
ay
az + bx
by
bz

Saída contém cópias do filtro ponderadas pela entrada, somando onde ocorre sobreposição da saída

O chão de o/i é 2,  
o: tamanho da saída  
i: tamanho da entrada

# Convolução como multiplicação de matrizes (Exemplo 1D)

Seja  $\mathbf{x} = (x, y, z)$  o vetor de pesos e  $\mathbf{a} = (a, b, c, d)$  o vetor de entrada

Podemos expressar a convolução em termos  
de multiplicação de matriz

$$\mathbf{x} * \mathbf{a} = \mathbf{Xa}$$

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & x & y & z & 0 & 0 \\ 0 & 0 & x & y & z & 0 \\ 0 & 0 & 0 & x & y & z \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ ax + by + cz \\ bx + cy + dz \\ cx + dy \end{bmatrix}$$

Exemplo: 1D Conv, tamanho 3,  
stride=1, padding=1

# Convolução como multiplicação de matrizes (Exemplo 1D)

Seja  $\mathbf{x} = (x, y, z)$  o vetor de pesos e  $\mathbf{a} = (a, b, c, d)$  o vetor de entrada

Podemos expressar a convolução em termos de multiplicação de matriz

$$\mathbf{x} * \mathbf{a} = \mathbf{X}\mathbf{a}$$

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & x & y & z & 0 & 0 \\ 0 & 0 & x & y & z & 0 \\ 0 & 0 & 0 & x & y & z \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ ax + by + cz \\ bx + cy + dz \\ cx + dy \end{bmatrix}$$

Convolução transposta multiplica pela transposta da mesma matriz

$$\mathbf{x} *^\top \mathbf{a} = \mathbf{X}^\top \mathbf{a}$$

$$\begin{bmatrix} x & 0 & 0 & 0 \\ y & x & 0 & 0 \\ z & y & x & 0 \\ 0 & z & y & x \\ 0 & 0 & z & y \\ 0 & 0 & 0 & z \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} ax \\ ay + bx \\ az + by + cx \\ bz + cy + dx \\ cz + dy \\ dz \end{bmatrix}$$

Exemplo: 1D Conv, tamanho 3, stride=1, padding=1

Quando stride=1, convolução transposta é uma convolução regular (com diferentes regras de padding)

# Convolução como multiplicação de matrizes (Exemplo 1D)

Seja  $\mathbf{x} = (x, y, z)$  o vetor de pesos e  $\mathbf{a} = (a, b, c, d)$  o vetor de entrada

Podemos expressar a convolução em termos  
de multiplicação de matriz

$$\mathbf{x} * \mathbf{a} = \mathbf{Xa}$$

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & 0 & x & y & z & 0 \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ bx + cy + dz \end{bmatrix}$$

Exemplo: 1D Conv, tamanho 3,  
stride=2, padding=1

# Convolução como multiplicação de matrizes (Exemplo 1D)

Seja  $\mathbf{x} = (x, y, z)$  o vetor de pesos e  $\mathbf{a} = (a, b, c, d)$  o vetor de entrada

Podemos expressar a convolução em termos de multiplicação de matriz

$$\mathbf{x} * \mathbf{a} = \mathbf{X}\mathbf{a}$$

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & 0 & x & y & z & 0 \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ bx + cy + dz \end{bmatrix}$$

Convolução transposta multiplica pela transposta da mesma matriz

$$\mathbf{x} *^\top \mathbf{a} = \mathbf{X}^\top \mathbf{a}$$

$$\begin{bmatrix} x & 0 \\ y & 0 \\ z & x \\ 0 & y \\ 0 & z \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} ax \\ ay \\ az + bx \\ by \\ bz \\ 0 \end{bmatrix}$$

Exemplo: 1D Conv, tamanho 3,  
stride=2, padding=1

Quando stride >1, convolução transposta não é mais uma convolução normal

# Segmentação Semântica: Fully Convolutional Network

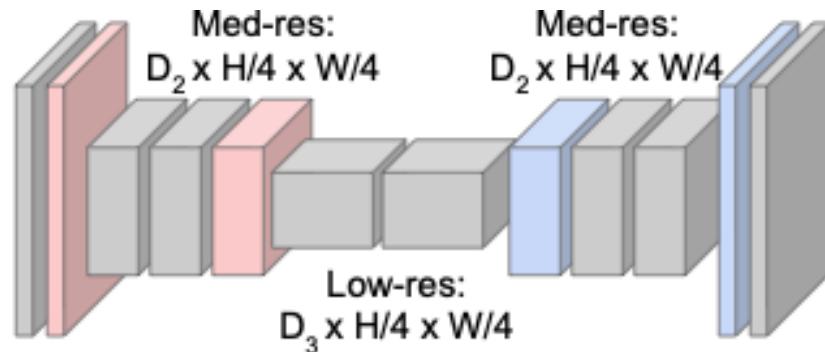
**Downsampling:** pooling,  
strided convolution



Input:  
 $3 \times H \times W$

High-res:  
 $D_1 \times H/2 \times W/2$

Projete rede como várias camadas  
convolucionais, com **downsampling** e  
**upsampling** dentro da rede

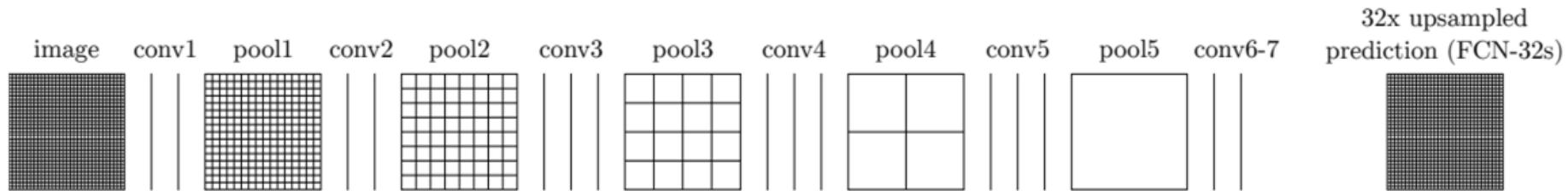


**Downsampling:** unpooling  
ou strided transpose  
convolution

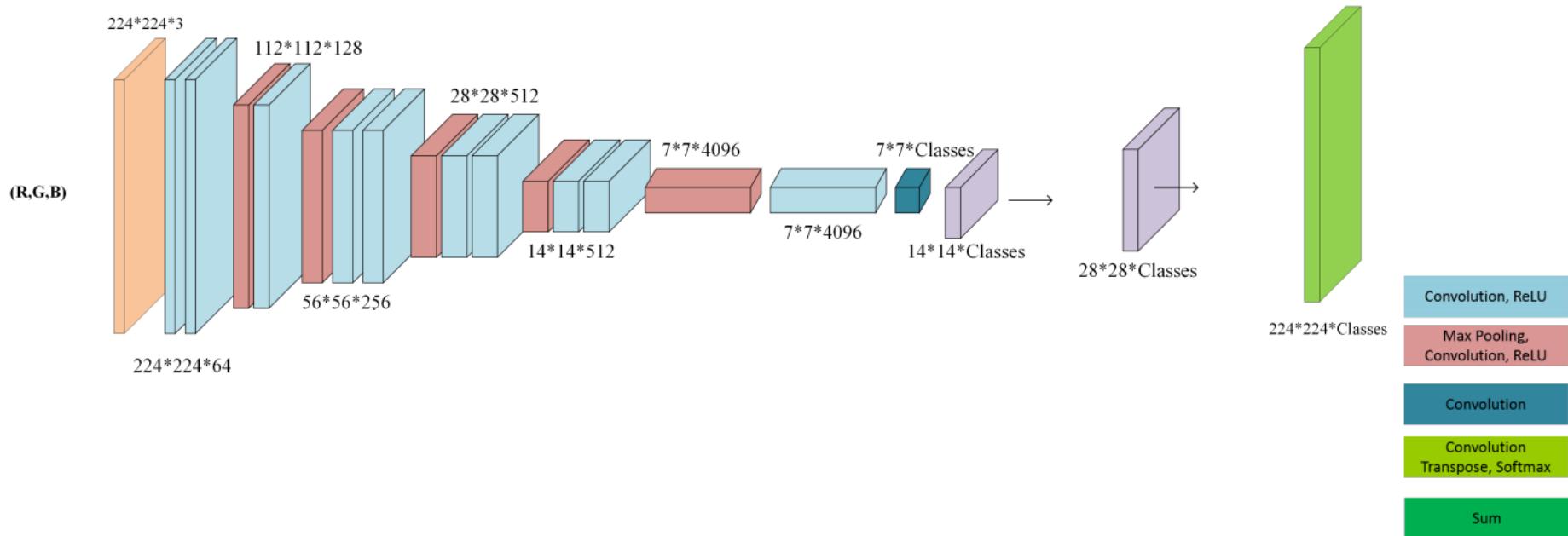


Predictions:  
 $H \times W$

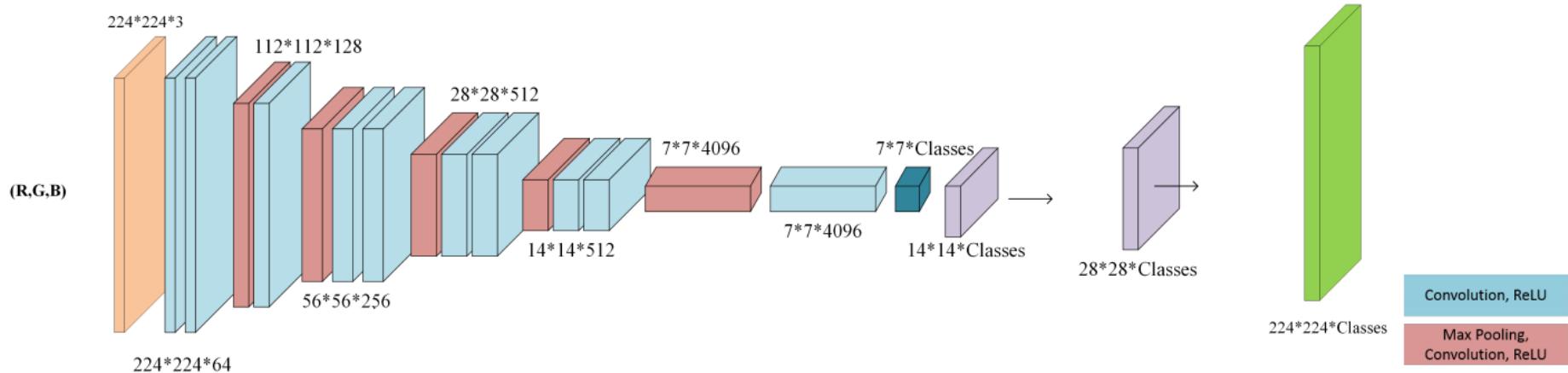
# Arquitetura da FCN-32s



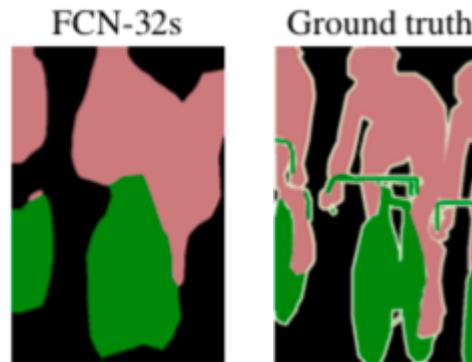
# Arquitetura da FCN-32s



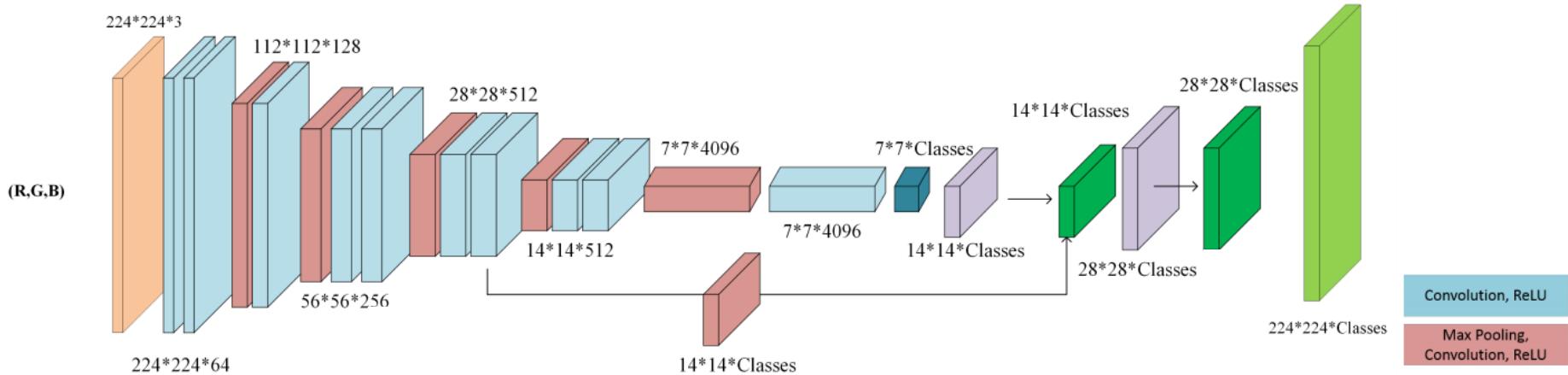
# Arquitetura da FCN-32s



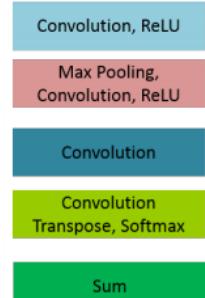
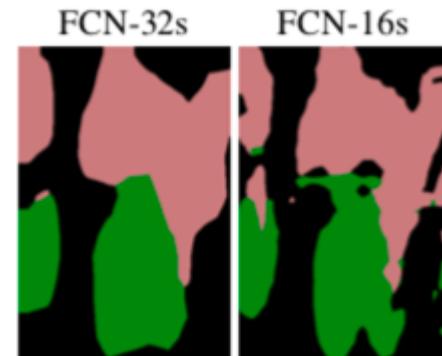
- FCN-32s: 32x conv7 (camada conv7 é upscaled 32x)
- Mesmo após o fine-tuning das redes pré-treinadas, os resultados são insatisfatoriamente grossos



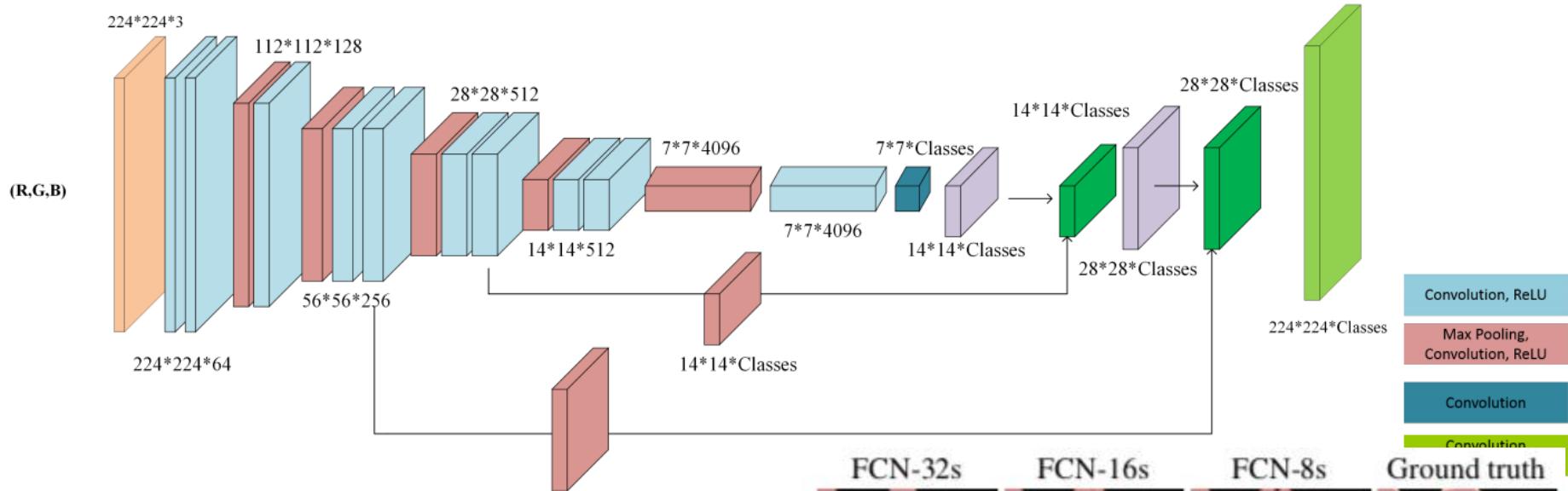
# Arquitetura da FCN-16s



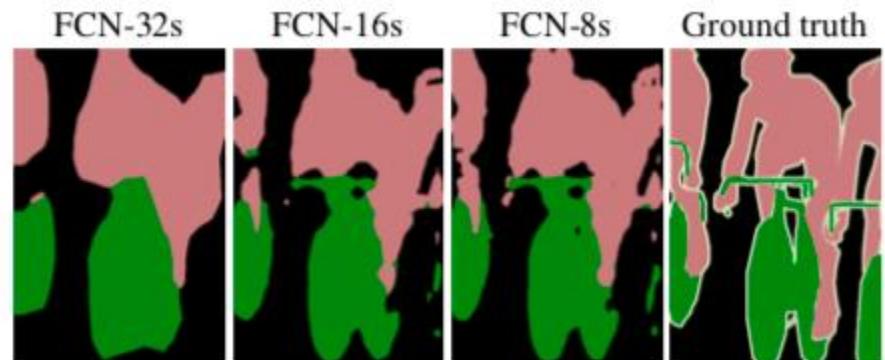
- Skip Connections p/ ligar as 1as camadas às últimas
- FCN-16s: 2x conv7 + pool4; resultado é upsampled 16x
- Diversas vantagens
  - Fusão de informação de alto nível semântico com informação de baixo nível semântico
  - Mitigação do problema do Vanishing Gradient em redes maiores



# Arquitetura da FCN-8s



- FCN-8s: 4x conv7 + 2x pool4 + pool3; resultado é upsampled 8x
- Para recuperar informação espacial mais fina nas camadas finais do decoder, preciso ligar com camadas mais rasas do encoder (receptive field menor)



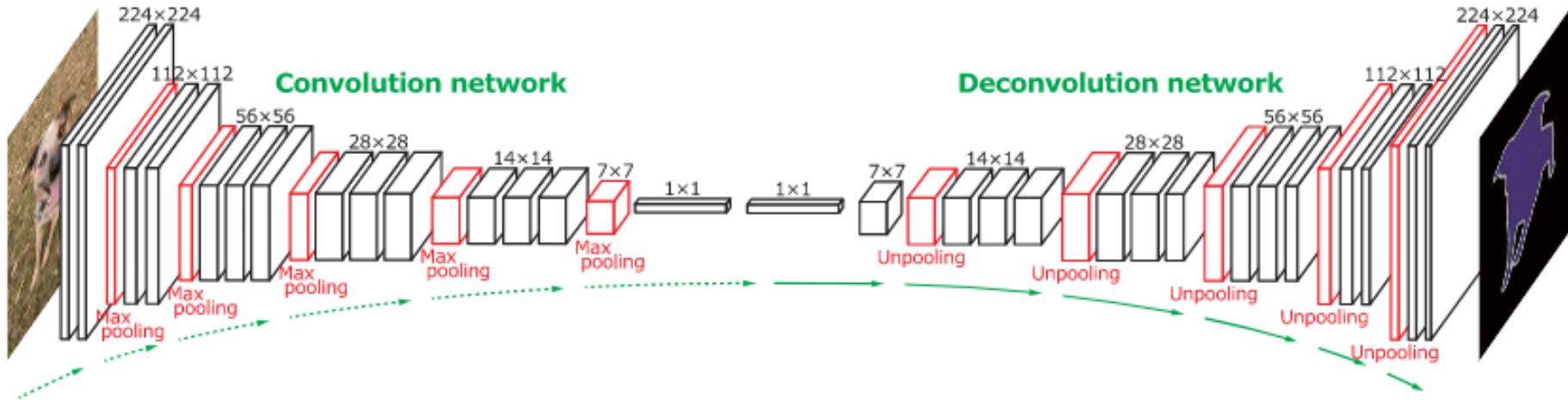
# Contribuições principais

- Popularizou o uso de redes convolucionais end-to-end para segmentação semântica
- Adaptou redes pré-treinadas com o ImageNet para segmentação semântica
- Upsampling usando convolução transposta
- Introduziu skip connections para aumentar a granularidade do upsampling

# *Segmentação Semântica*

Deconvolution Networks  
(DeconvNets)

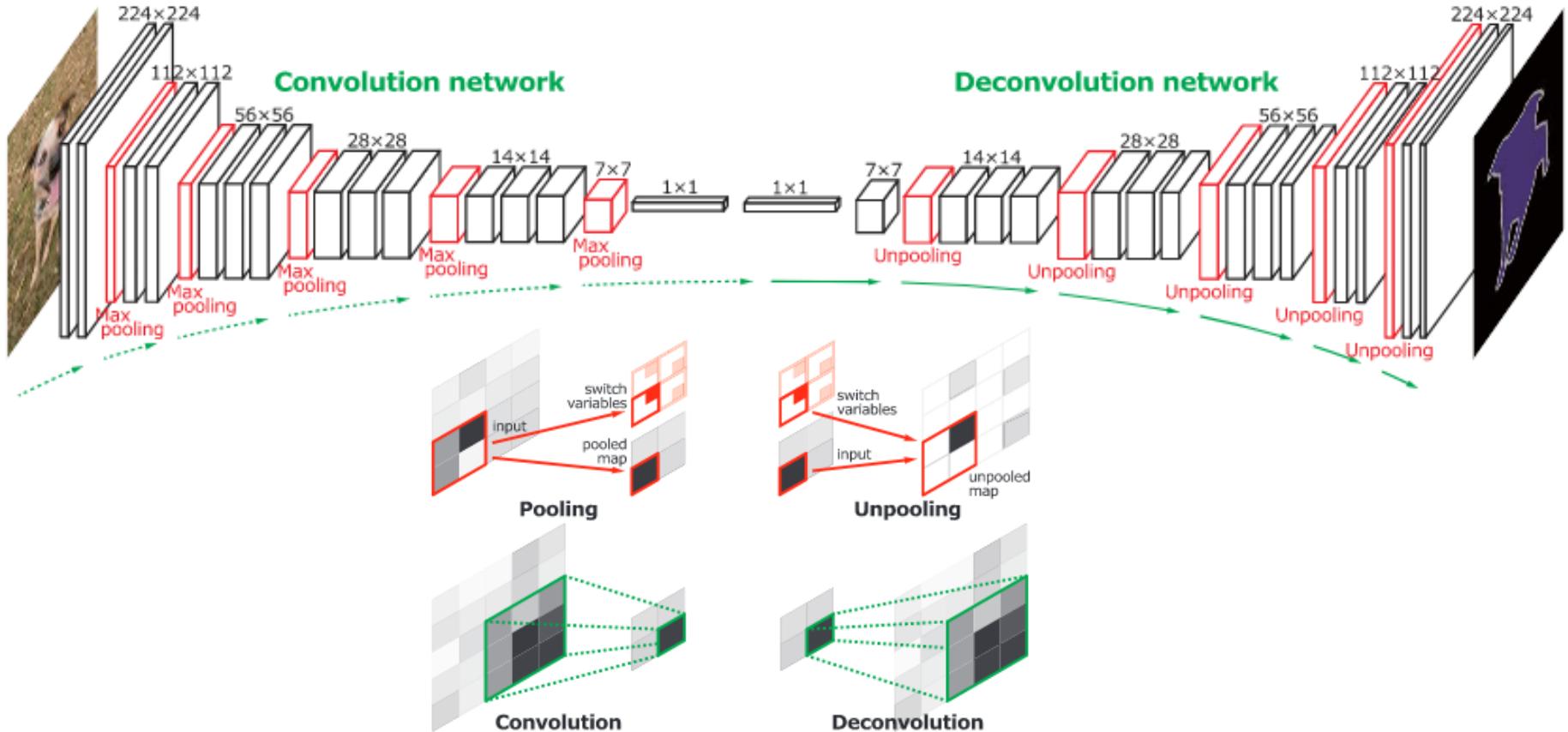
# DeconvNets



Arquitetura “espelhada”!

Noh, H., Hong, S., & Han, B. (2015). Learning deconvolution network for semantic segmentation. In Proceedings of the IEEE international conference on computer vision (pp. 1520-1528).

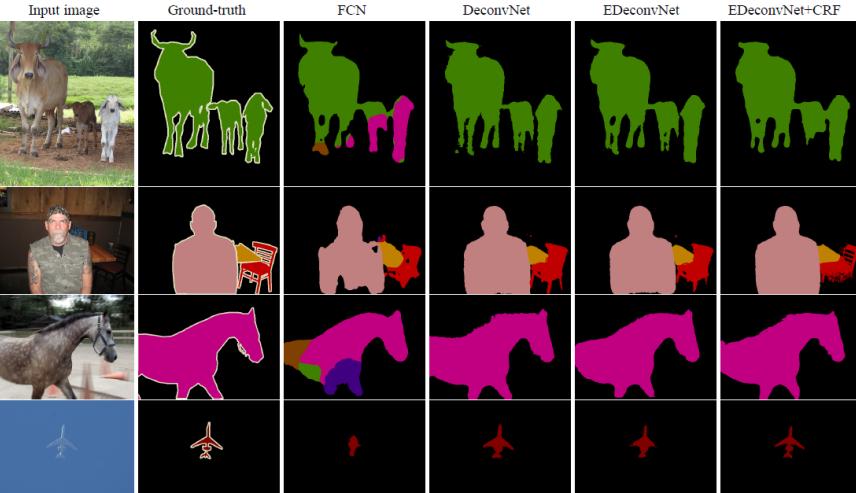
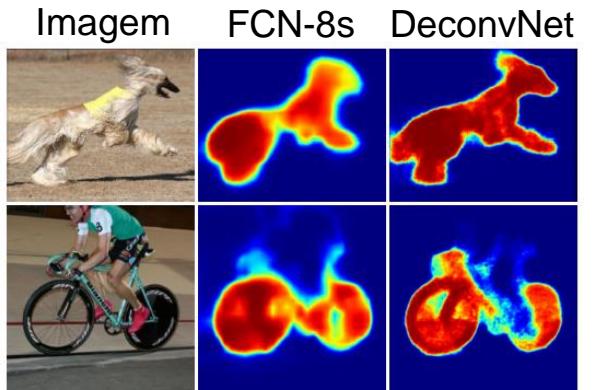
# DeconvNets



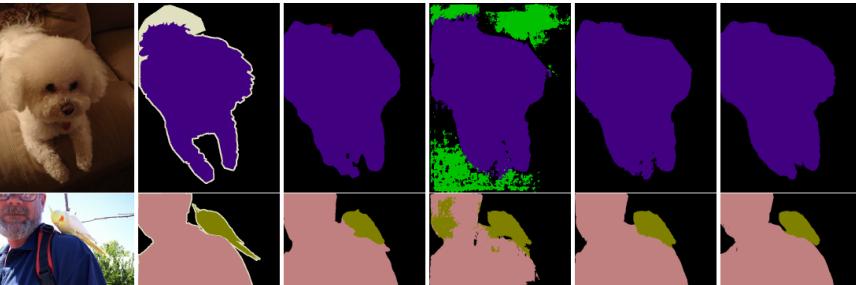
# DeconvNets

Problemas:

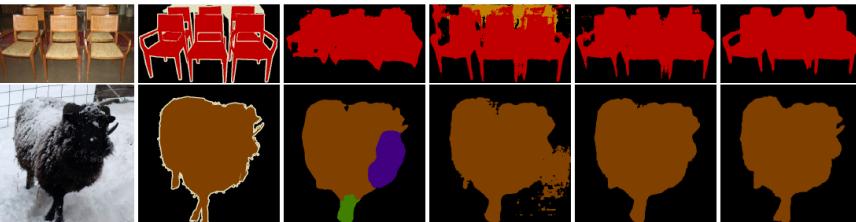
- Dificuldade de convergência
- Muitos parâmetros



(a) Examples that our method produces better results than FCN [17].



(b) Examples that FCN produces better results than our method.



(c) Examples that inaccurate predictions from our method and FCN are improved by ensemble.

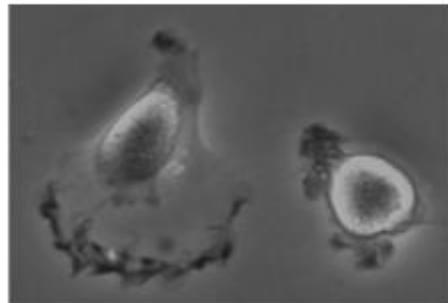
# *Segmentação Semântica*

U-Nets

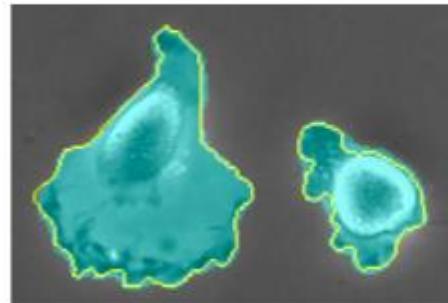
# U-Net

- Proposto no contexto de processamento de imagens biomédicas
  - ISBI challenge for segmentation of neuronal structures
  - Necessidade de se fazer classificação densa (por pixel)
  - Poucos exemplos de treinamento (~30 por aplicação)
  - Células que se tocam; espaço muito pequeno entre objetos
- Solução reutilizável em diferentes domínios

a

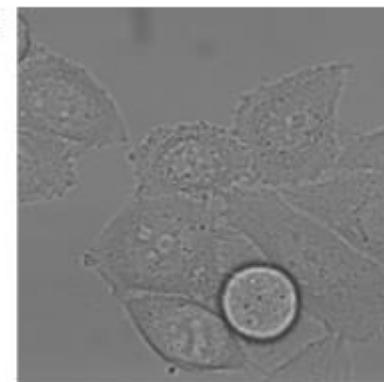


b



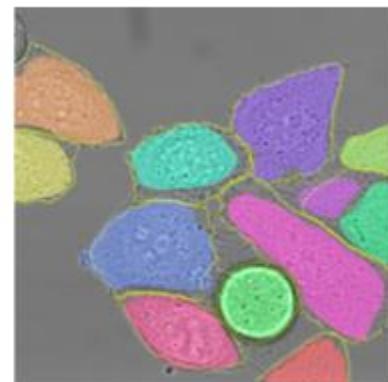
PhC-U373: 35 imagens

c



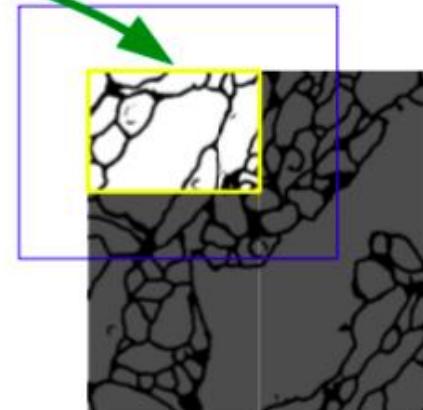
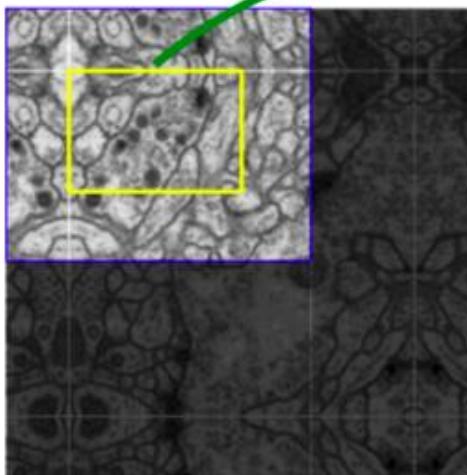
PhC-U373: 20 imagens

d



# U-Net: principais ideias

- Arquitetura encoder-decoder
- Augumentação de imagens
- Usa apenas "valid padding": em vez de completar com 0's, espelhar a imagem nas bordas. Técnica utilizada na imagem que entra na rede.



# U-Net: principais ideias

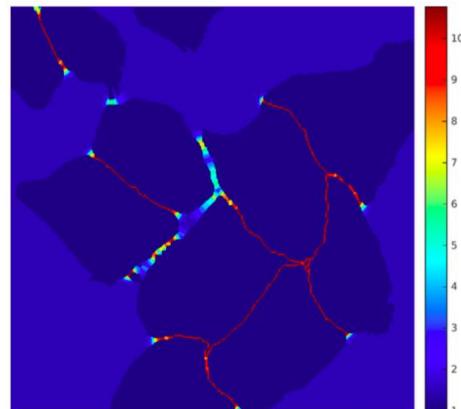
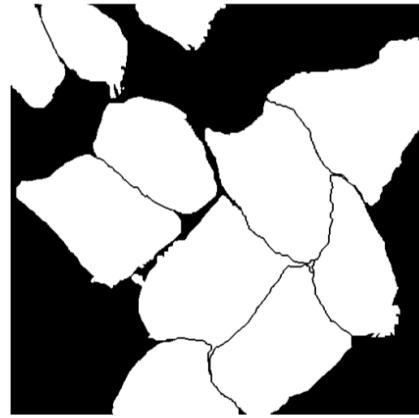
- Função de perda que dá pesos aos diferentes pixels
  - Balancear as diferentes frequências das classes (nas imagens há mais exemplos de uma classe que de outras)
  - Forçar a rede a aprender pequenas bordas de separação presentes entre células que se tocam

$$w(\mathbf{x}) = w_c(\mathbf{x}) + w_0 \cdot \exp\left(-\frac{(d_1(\mathbf{x}) + d_2(\mathbf{x}))^2}{2\sigma^2}\right)$$

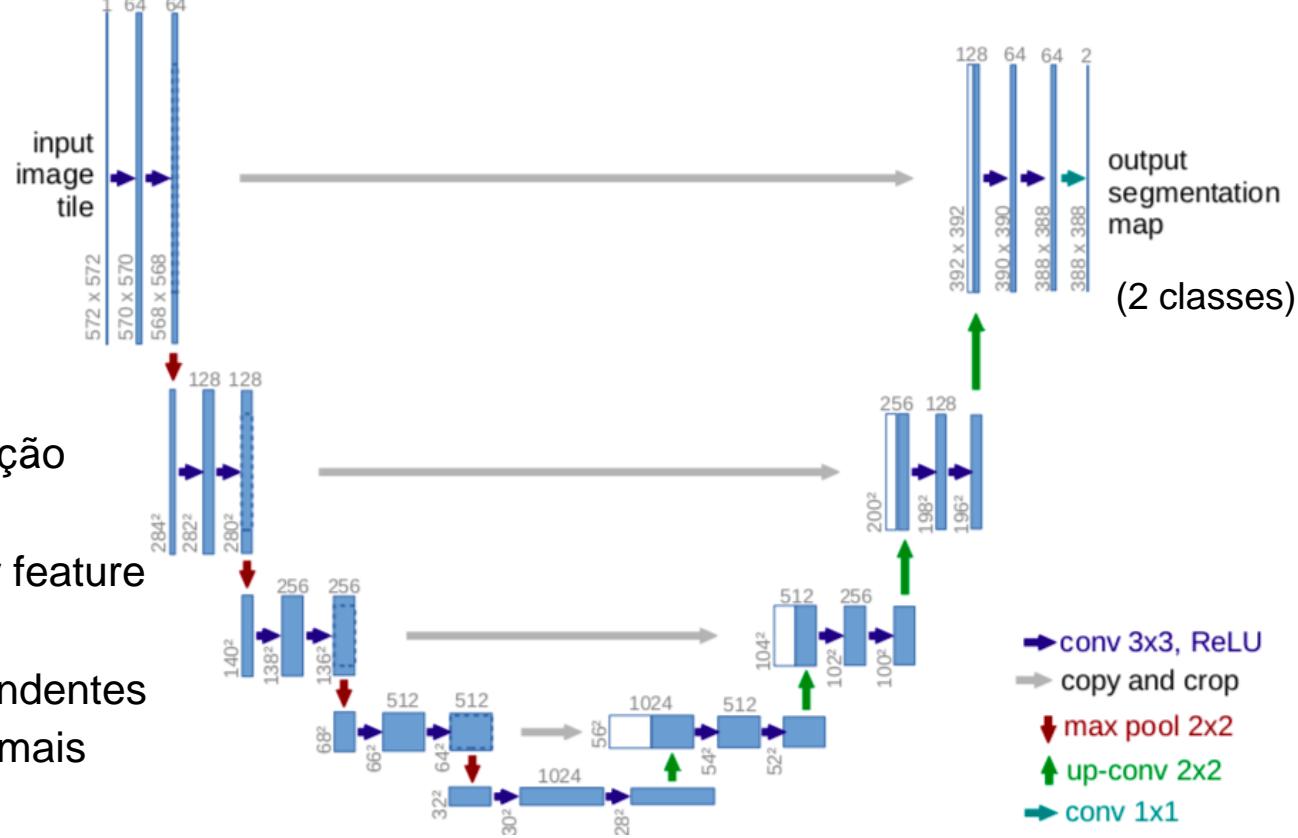
compensação de frequências das classes

Distância até borda mais próxima

Distância até 2a. borda mais próxima

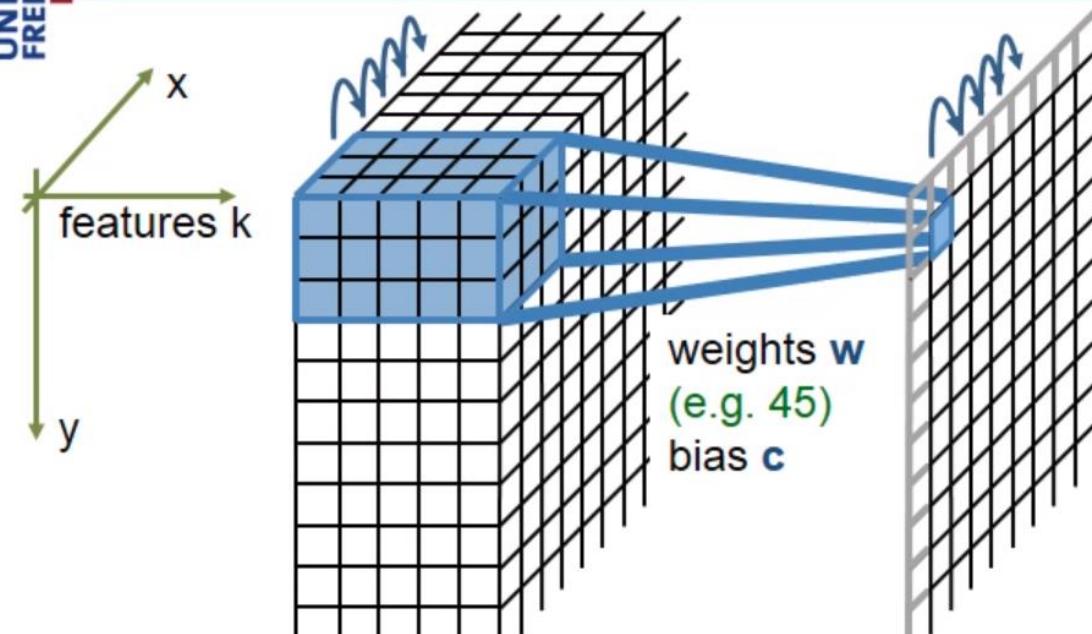


# U-Net: principais ideias



- Camadas dedicadas à recuperação da informação espacial
- No decoder: concatenar feature maps das camadas de downsampling correspondentes para obter localizações mais precisas

## 3x3 convolution + ReLU

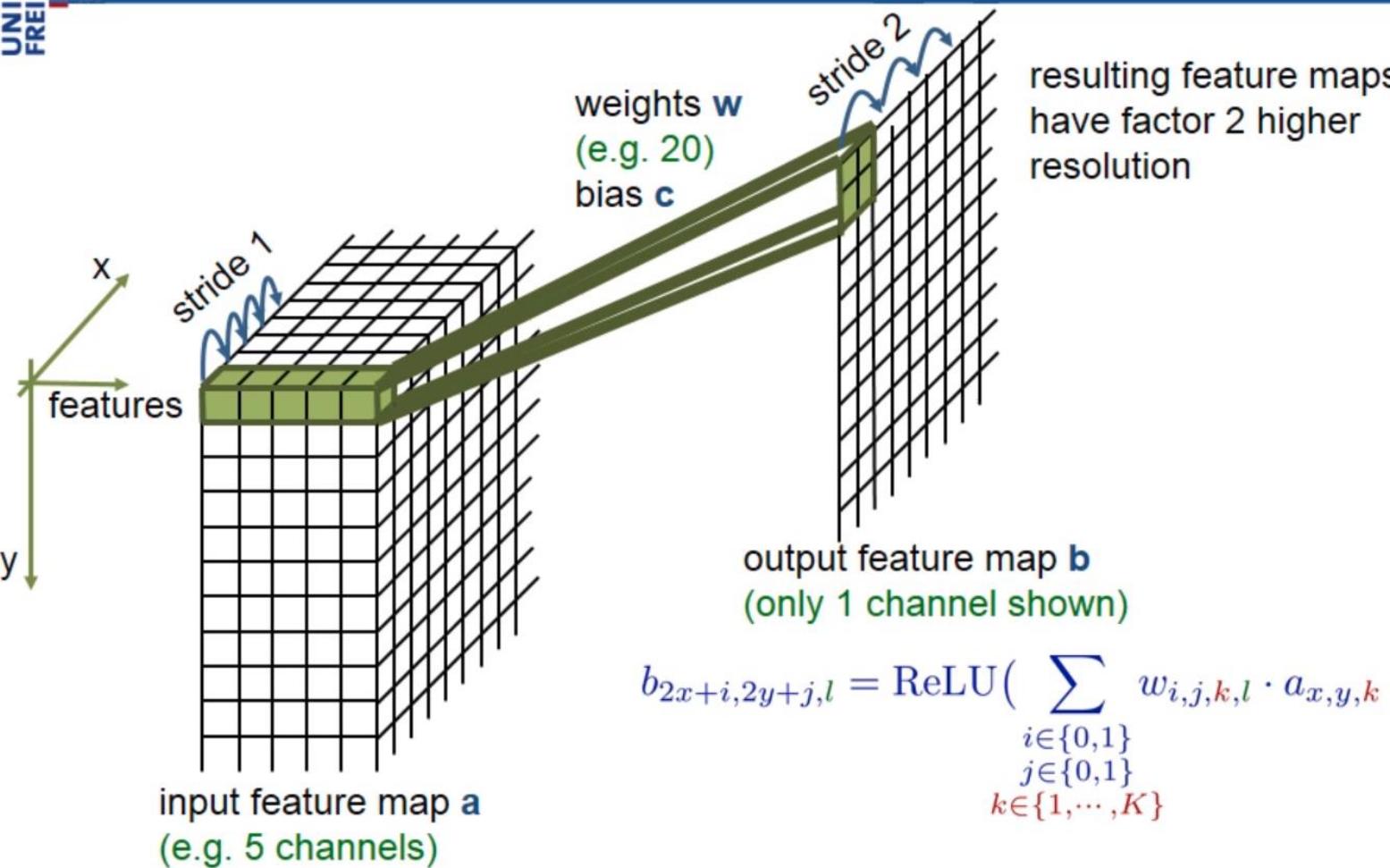


input feature map **a**  
(e.g. 5 channels)

output feature map **b**  
(only 1 channel shown)

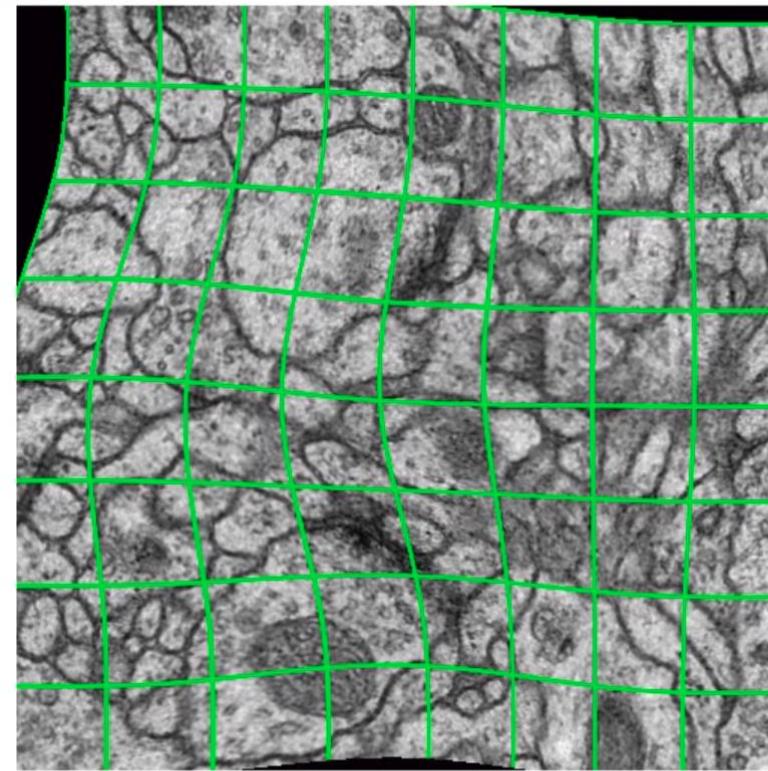
$$b_{x,y,l} = \text{ReLU} \left( \sum_{\substack{i \in \{-1,0,1\} \\ j \in \{-1,0,1\} \\ k \in \{1, \dots, K\}}} w_{i,j,k,l} \cdot a_{x+i, y+j, k} + c_l \right)$$

# 2x2 up-convolution

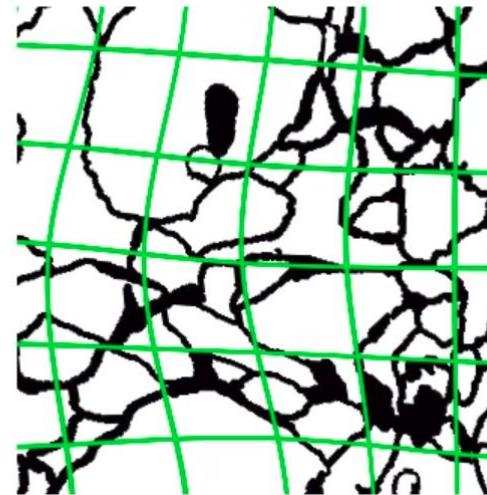


$$b_{2x+i, 2y+j, l} = \text{ReLU}\left(\sum_{\substack{i \in \{0,1\} \\ j \in \{0,1\}}} w_{i,j,k,l} \cdot a_{x,y,k} + c_l\right)$$
$$k \in \{1, \dots, K\}$$

# Augment Training Data using Deformations



resulting deformed image  
(for visualization: no rotation, no shift, no extrapolation)



correspondingly deformed  
manual labels

# Contribuições principais da U-Net

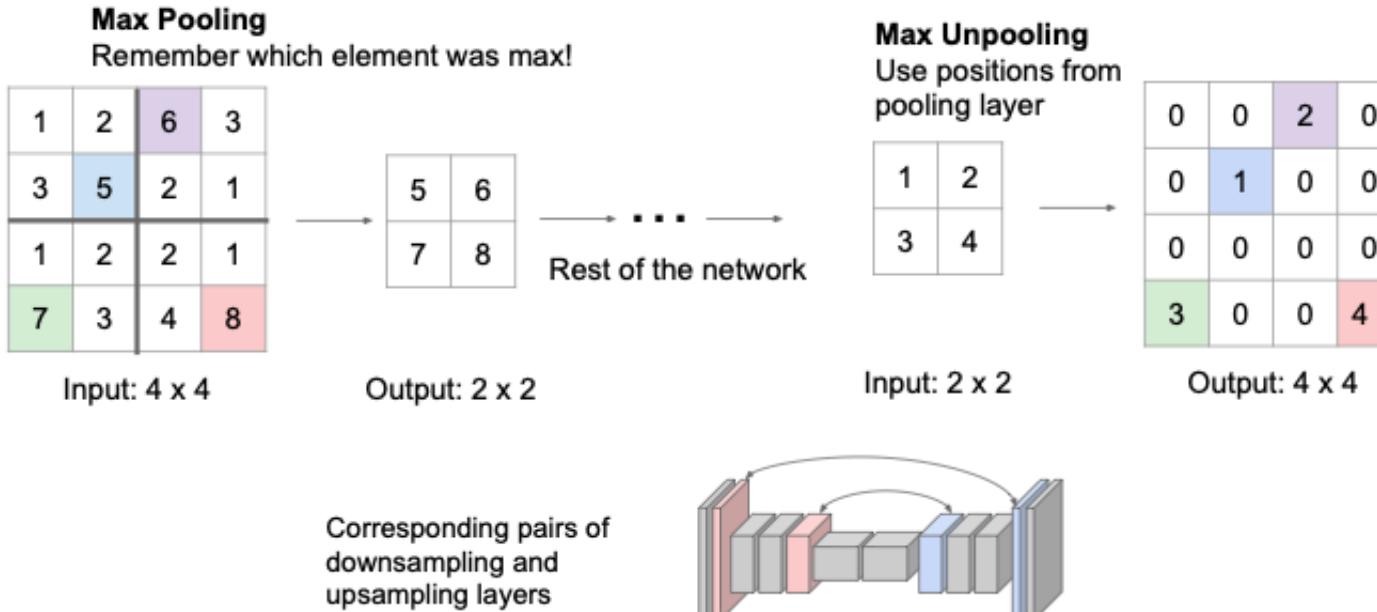
- Camadas dedicadas à recuperação da informação espacial
- No decoder: concatenar feature maps das camadas de downsampling correspondentes para obter localizações mais precisas
- Augumentação de dados para imagens microscópicas
- Função de custo que permite aprender pequenas bordas de separação entre objetos

# *Segmentação Semântica*

SegNet

# Como fazer upsampling in-network (unpooling)?

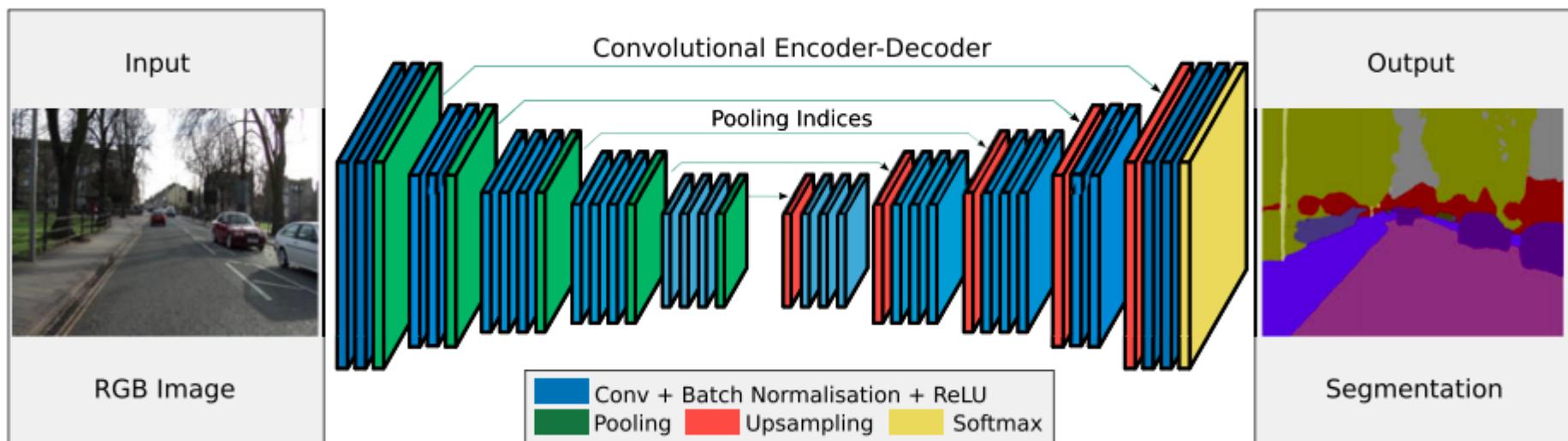
Quando fazemos max pooling, perdemos a informação de que elemento da camada foi utilizado. Podemos guardar estas posições e utilizá-las posteriormente:



Semelhante à cama de pregos, mas coloca os elementos correspondentes nas posições originais

# SegNet

- Arquiteturas Encoder-Decoder
- Uso de índices de pooling do Encoder para a reconstrução do



# SegNet: resultados no CamVid dataset

Method	Building	Tree	Sky	Car	Sign-Symbol	Road	Pedestrian	Fence	Column-Pole	Side-walk	Bicyclist	Class avg.	Global avg.
SfM+Appearance [28]	46.2	61.9	89.7	68.6	42.9	89.5	53.6	46.6	0.7	60.5	22.5	53.0	69.1
Boosting [29]	61.9	67.3	91.1	71.1	58.5	92.9	49.5	37.6	25.8	77.8	24.7	59.8	76.4
Dense Depth Maps [32]	85.3	57.3	95.4	69.2	46.5	<b>98.5</b>	23.8	44.3	22.0	38.1	28.7	55.4	82.1
Structured Random Forests [31]						n/a						51.4	72.5
Neural Decision Forests [64]						n/a						56.1	82.1
Local Label Descriptors [65]	80.7	61.5	88.8	16.4	n/a	98.0	1.09	0.05	4.13	12.4	0.07	36.3	73.6
Super Parsing [33]	87.0	67.1	96.9	62.7	30.1	95.9	14.7	17.9	1.7	70.0	19.4	51.2	83.3
SegNet (3.5K dataset training - 140K)	<b>89.6</b>	<b>83.4</b>	96.1	<b>87.7</b>	52.7	96.4	<b>62.2</b>	<b>53.45</b>	<b>32.1</b>	<b>93.3</b>	<b>36.5</b>	<b>71.20</b>	<b>90.40</b>
CRF based approaches													
Boosting + pairwise CRF [29]	70.7	70.8	94.7	74.4	55.9	94.1	45.7	37.2	13.0	79.3	23.1	59.9	79.8
Boosting+Higher order [29]	84.5	72.6	<b>97.5</b>	72.7	34.1	95.3	34.2	45.7	8.1	77.6	28.5	59.2	83.8
Boosting+Detectors+CRF [30]	81.5	76.6	96.2	78.7	40.2	93.9	43.0	47.6	14.3	81.5	33.9	62.5	83.8

# Contribuições principais da SegNet

- Índices do maxpooling transferidos para o decoder para aumentar a resolução da segmentação
  - Como as features do encoder não são copiadas (como na FCN), SegNet é mais eficiente no uso de memória

Network	Forward pass(ms)	Backward pass(ms)	GPU training memory (MB)	GPU inference memory (MB)	Model size (MB)
SegNet	422.50	488.71	6803	<b>1052</b>	117
DeepLab-LargeFOV [3]	<b>110.06</b>	<b>160.73</b>	<b>5618</b>	1993	83
FCN (learnt deconv) [2]	317.09	484.11	9735	1806	539
DeconvNet [4]	474.65	602.15	9731	1872	877

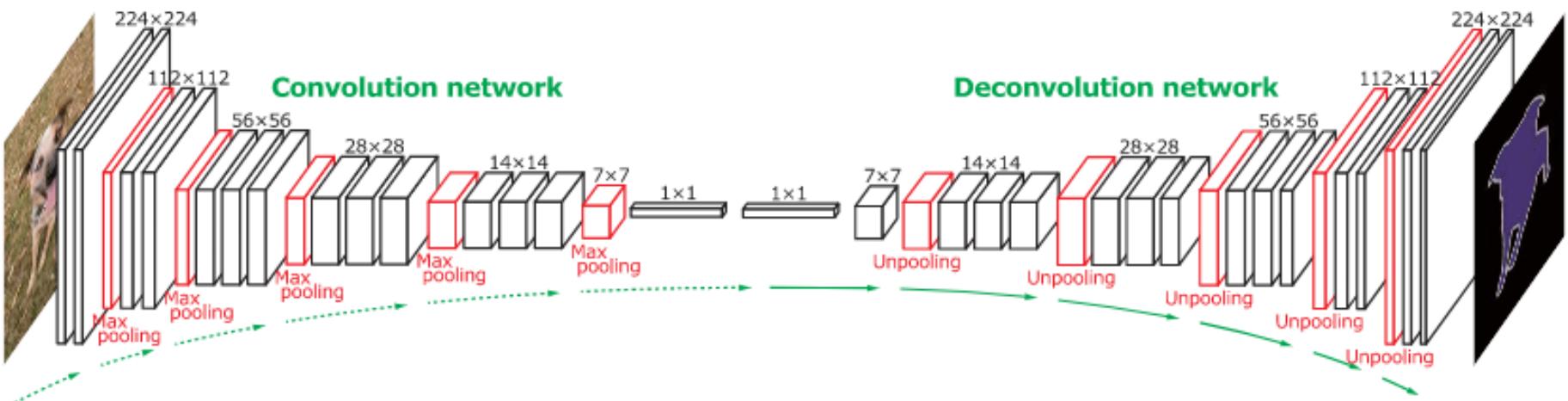
# *Segmentação Semântica*

Convolução Dilatada  
(a.k.a. *Atrous Convolution*)

# Segmentação e CNNs

## Problema:

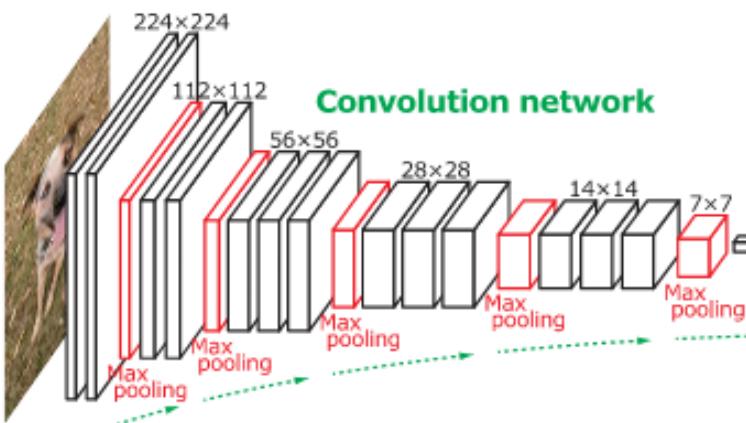
- ✗ Down-sampling causa perda de informação espacial.



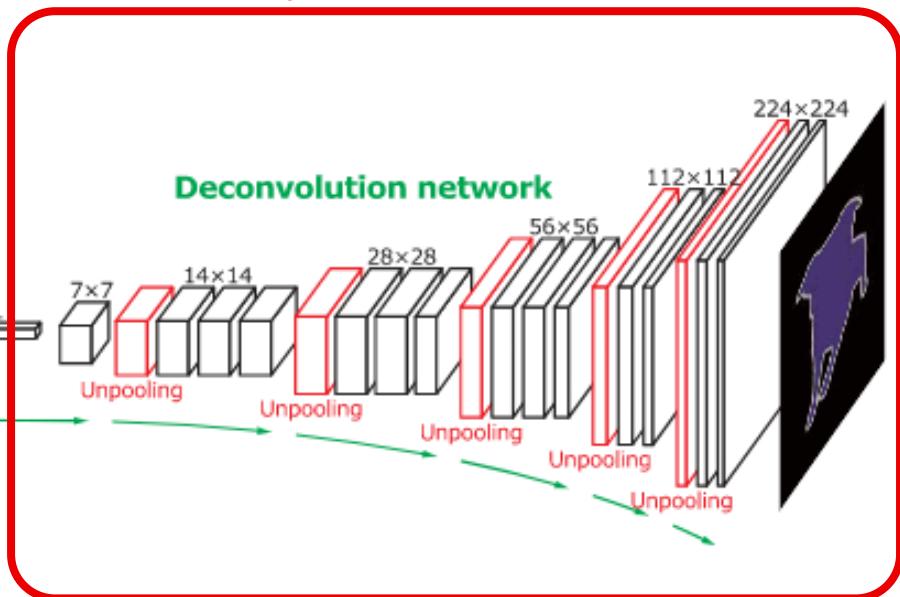
# Segmentação e CNNs

## Problema:

- ✗ Down-sampling causa perda de informação espacial.



Possível solução



# Segmentação e CNNs

## Problema:

- ✗ **Down-sampling** causa perda de informação espacial.

## Outra solução:

- Convolução dilatada ('Holes' algorithm).

# Convolução Dilatada

Introduz um parâmetro para as camadas convolucionais: a taxa de dilatação.

- Convolução dilatada para o sinal 1-D:

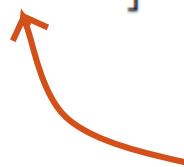
$$y[i] = \sum_{k=1}^K x[i + r \cdot k]w[k]$$

$x[i]$  1-D input signal

$w[k]$  filter of length  $K$

$r$  rate parameter corresponds to the stride  
with which we sample the input signal.

$y[i]$  output of atrous convolution.

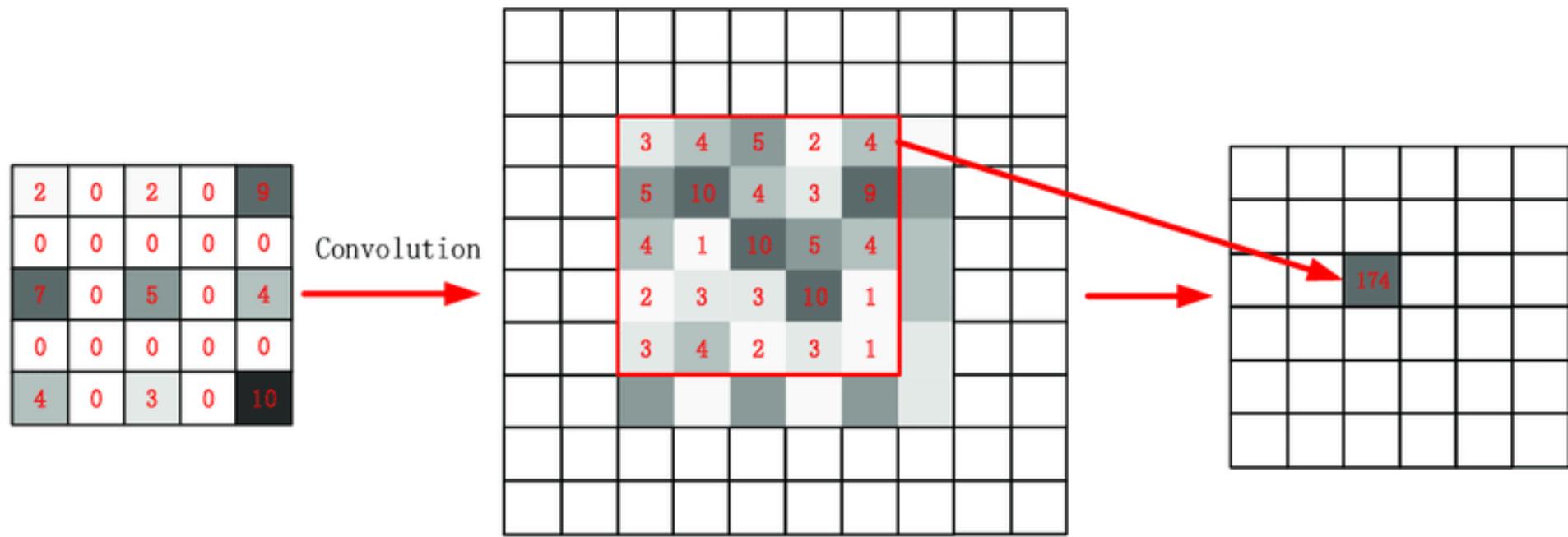


Introduz zeros entre os  
valores do filtro

Nota: convolução padrão é um caso especial em que a taxa é  $r = 1$ .

# Convolução Dilatada

Introduz um parâmetro para as camadas convolucionais: a taxa de dilatação.

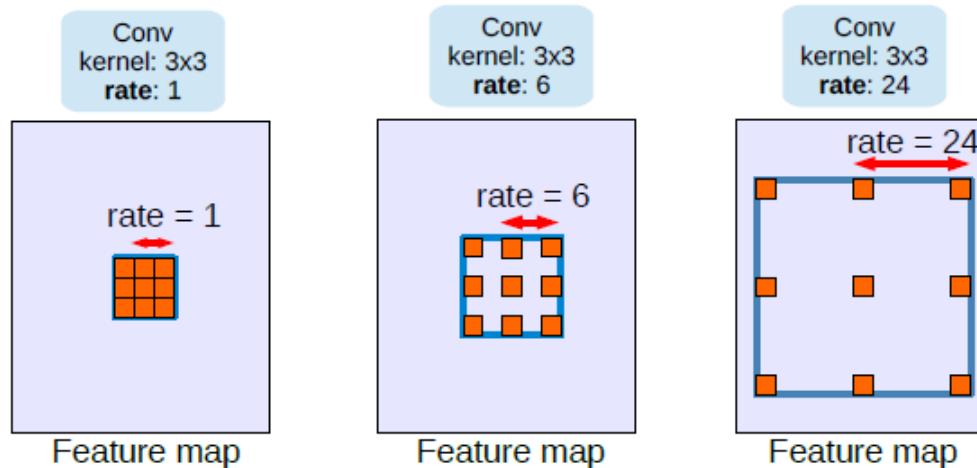


# Convolução Dilatada

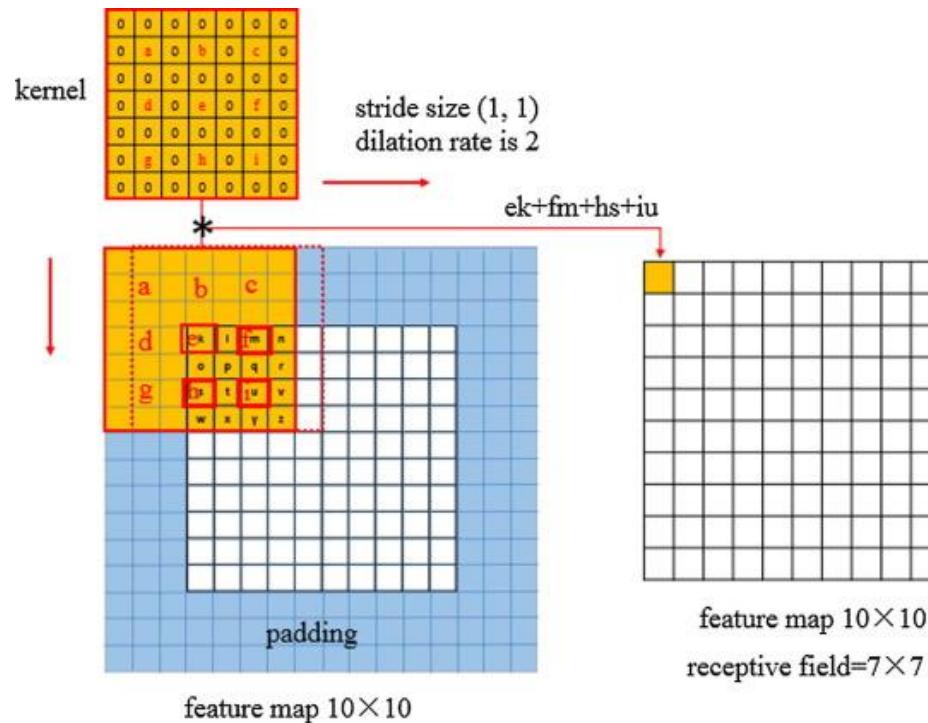
Introduz um parâmetro para as camadas convolucionais: a taxa de dilatação.

- Efeito:

- um kernel 3x3 com uma taxa de dilatação de 2 terá o mesmo campo de visão que um kernel 5x5, enquanto usa apenas 9 parâmetros.

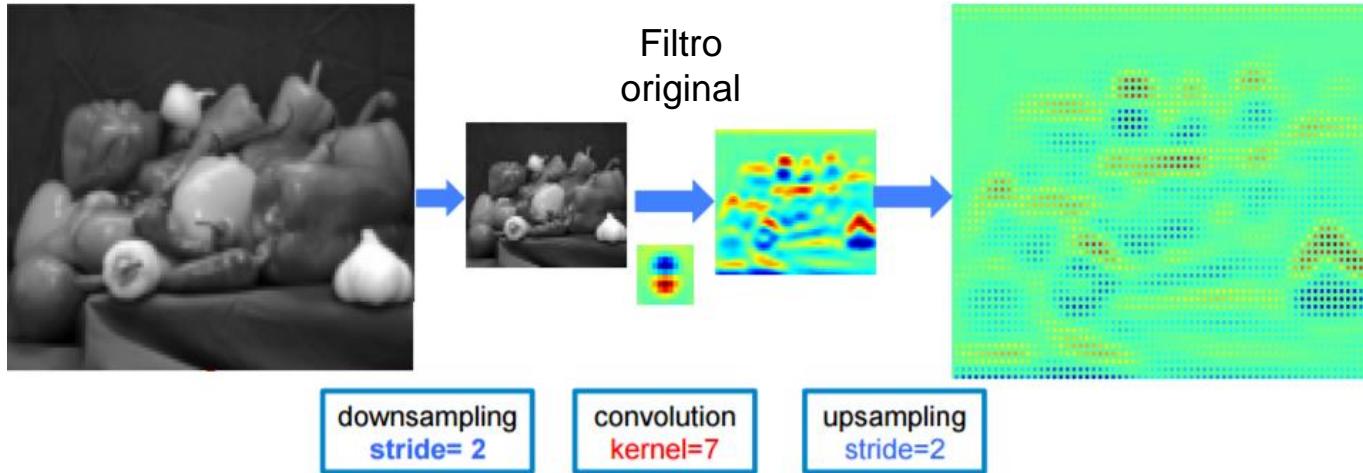


# Convolução Dilatada - Padding

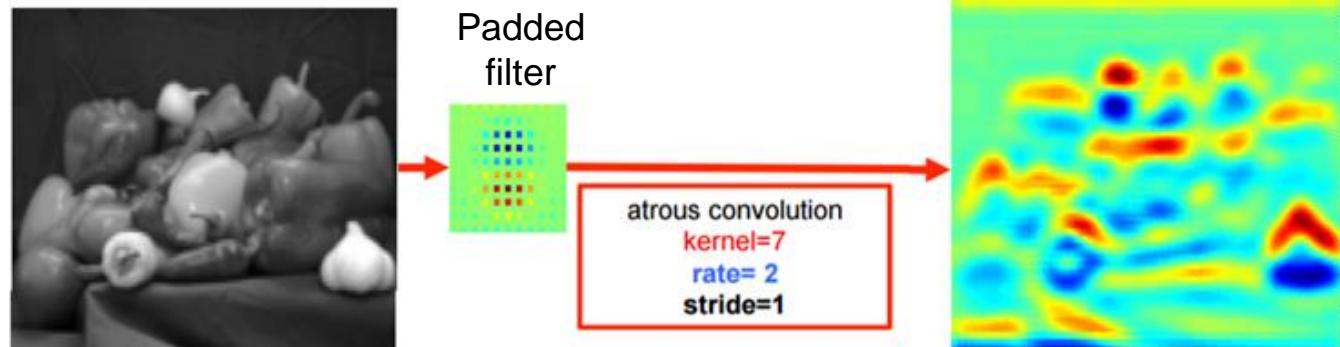


# Convolução Dilatada

Standard convolution

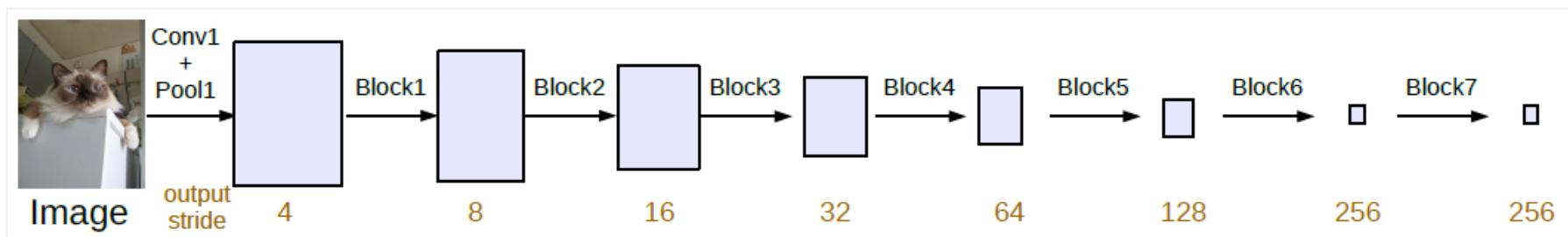


Atrous convolution

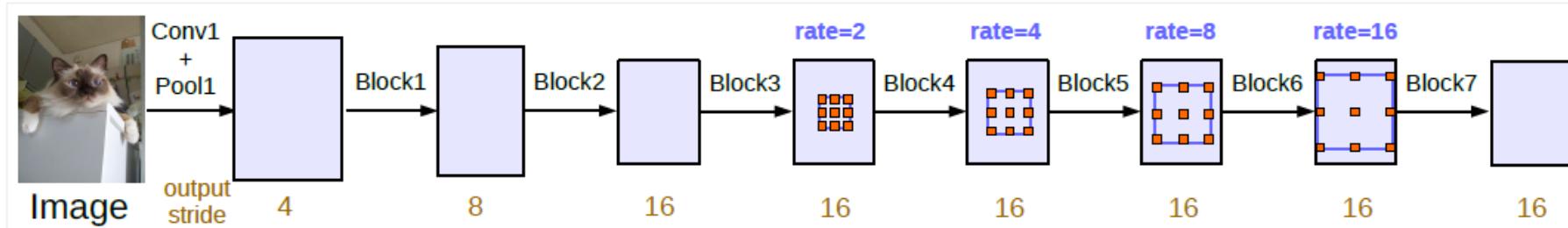


# Convolução Dilatada

- Podemos manter o *stride* constante, mas com um *field-of-view* maior sem aumentar o número de parâmetros ou a quantidade de computação.
- Temos um *feature-map* maior, o que é bom para a segmentação semântica.

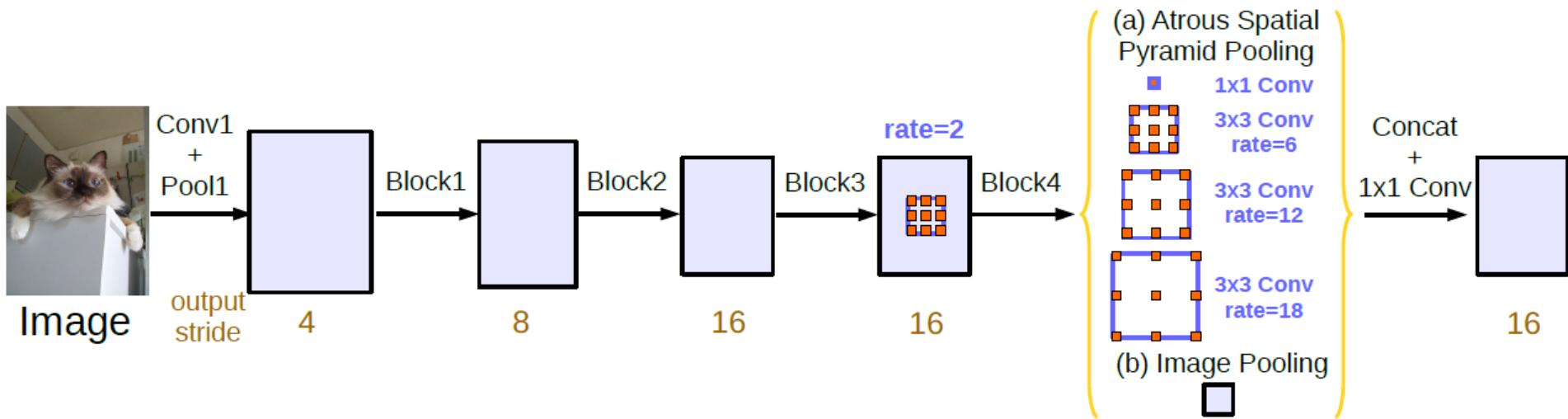


(a) Going deeper without atrous convolution.



(b) Going deeper with atrous convolution. Atrous convolution with  $rate > 1$  is applied after block3 when  $output\_stride = 16$ .

# Atrous Spatial Pyramid Pooling (ASPP)



Chen, L. C., Papandreou, G., Schroff, F., & Adam, H. (2017). Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587.

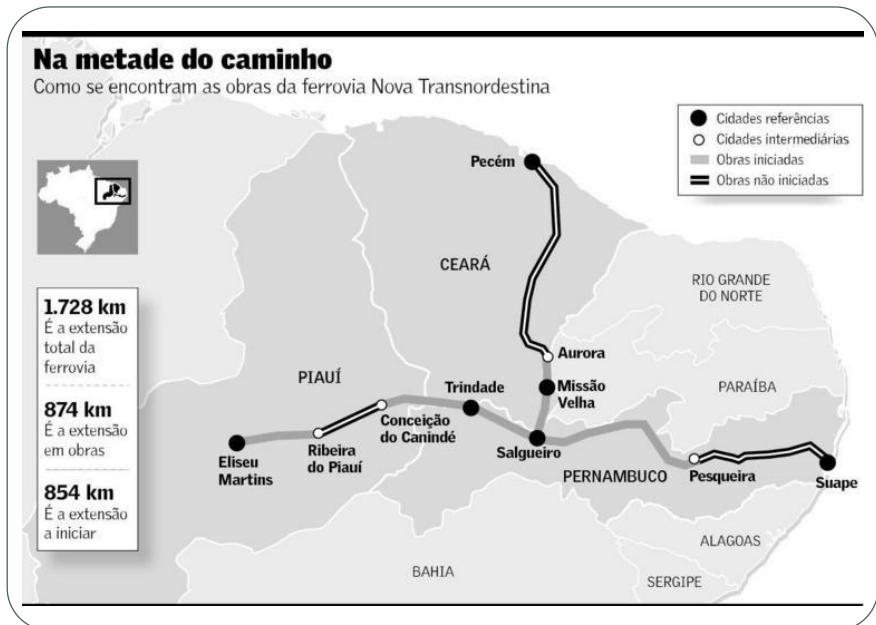
# *Segmentação Semântica*

Processamento de Grandes Imagens

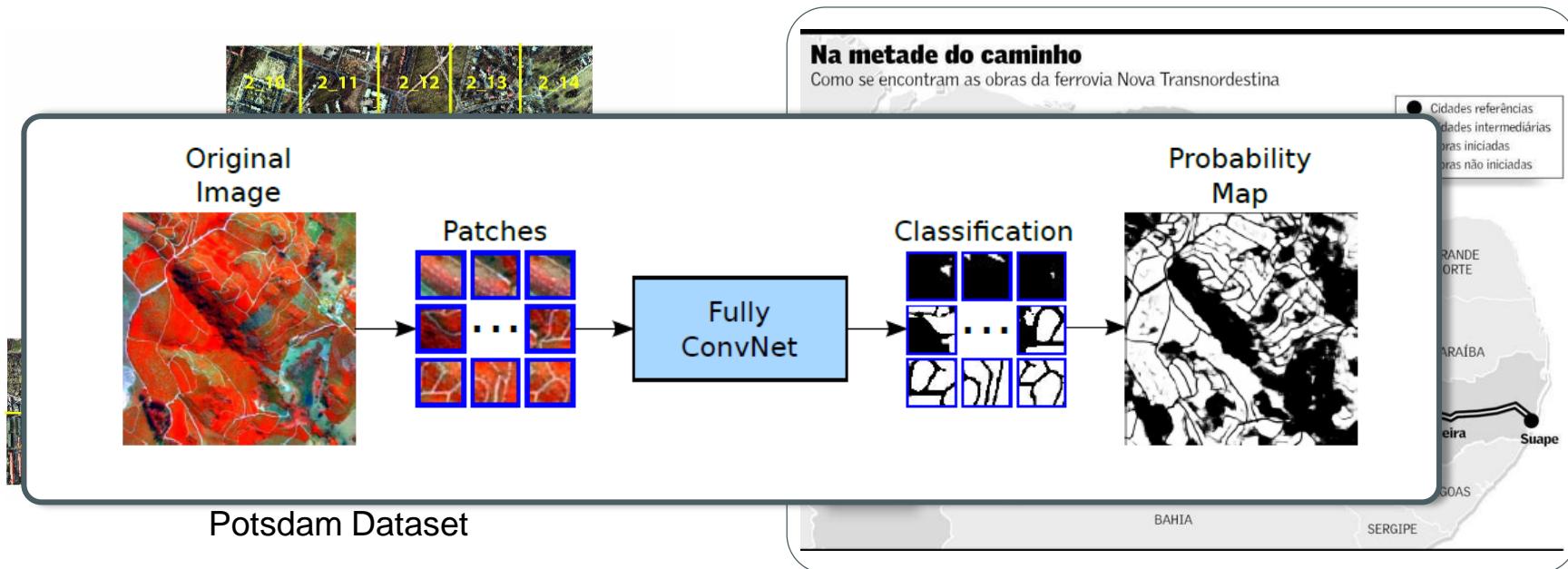
# Como processar grandes imagens aéreas?



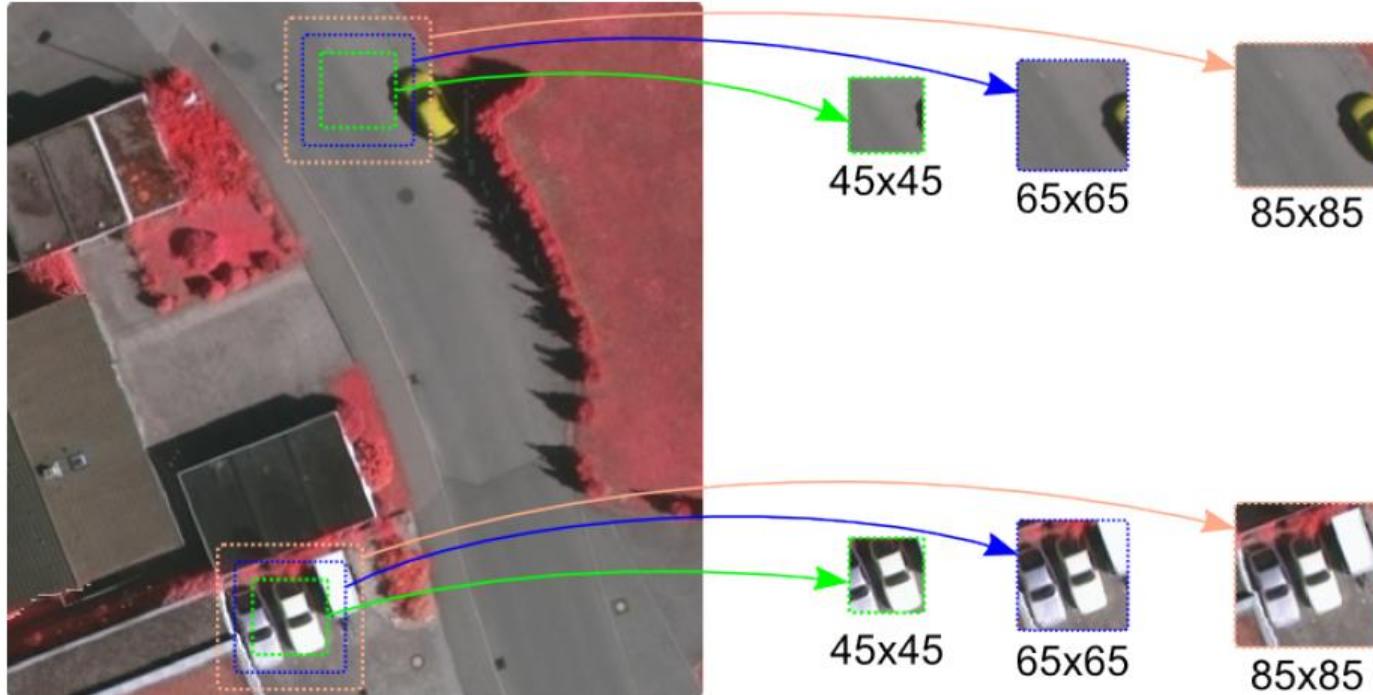
Potsdam Dataset



# Como processar grandes imagens aéreas?

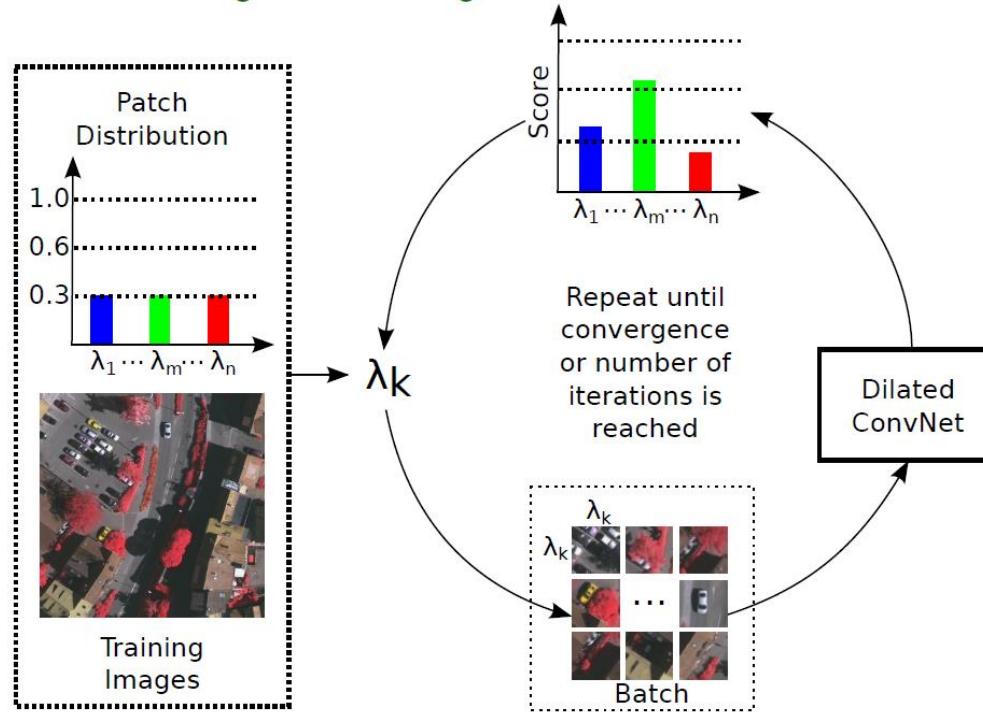


# Como processar grandes imagens aéreas?



# Como processar grandes imagens aéreas?

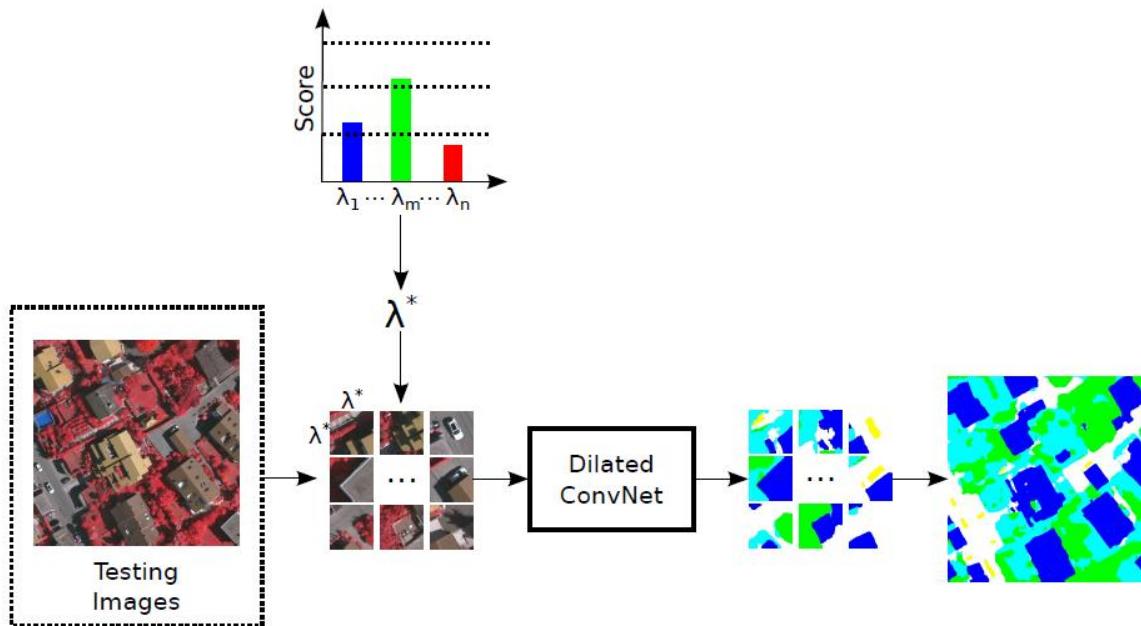
## Dynamic Multi-Context Algorithm: Training



K. Nogueira, M. Dalla Mura, J. Chanussot, W. R. Schwartz and J. A. dos Santos, "Dynamic Multicontext Segmentation of Remote Sensing Images Based on Convolutional Networks," in *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 10, pp. 7503-7520, Oct. 2019.

# Como processar grandes imagens aéreas?

## Dynamic Multi-Context Algorithm: Testing



K. Nogueira, M. Dalla Mura, J. Chanussot, W. R. Schwartz and J. A. dos Santos, "Dynamic Multicontext Segmentation of Remote Sensing Images Based on Convolutional Networks," in *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 10, pp. 7503-7520, Oct. 2019.