

Hand-on Deep Learning - Aula 3

Padrões em Imagens: Classificação, Segmentação e Detecção

Camila Laranjeira¹, Hugo Oliveira¹², Keiller Nogueira¹²

¹Programa de Pós-Graduação em Ciência da Computação (PPGCC)
Universidade Federal de Minas Gerais

²Interest Group in Pattern Recognition and Earth Observation (PATREO)
Universidade Federal de Minas Gerais

03 de Agosto, 2018





Agenda

- 1 Introdução**
- 2 Redes Neurais Convolucionais**
- 3 Arquitetura Famosas**
- 4 Estratégias de Treino**
- 5 Segmentação de Imagens**
- 6 Detecção em Imagens**

Agenda



- 1 Introdução
- 2 Redes Neurais Convolucionais
- 3 Arquitetura Famosas
- 4 Estratégias de Treino
- 5 Segmentação de Imagens
- 6 Detecção em Imagens

MultiLayer Perceptrons



- Quantidade de pesos:
 - #Neurônios de saída da camada anterior
 - #Neurônios na camada atual

MLP

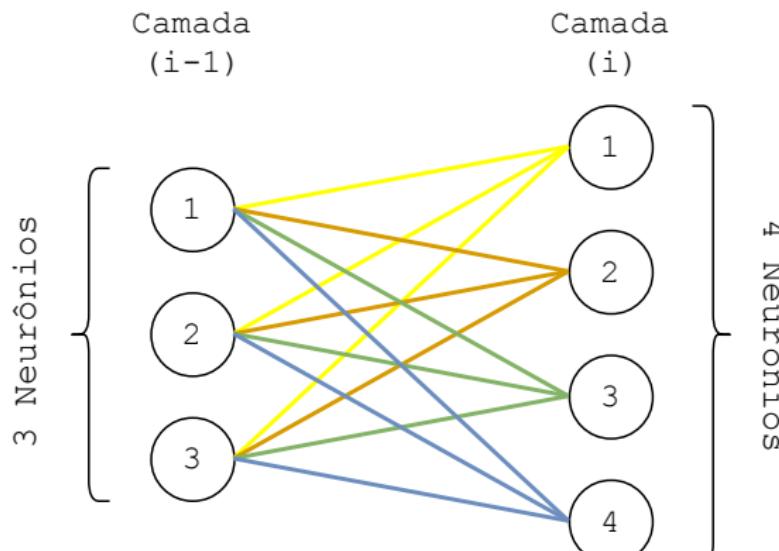


Figura: MLP.



MultiLayer Perceptrons

Número de Pesos da Camada i

$$\# \text{Pesos} = \# \text{Neuronios}_{(i-1)} \times \# \text{Neuronios}_{(i)}$$

MLP para Imagens

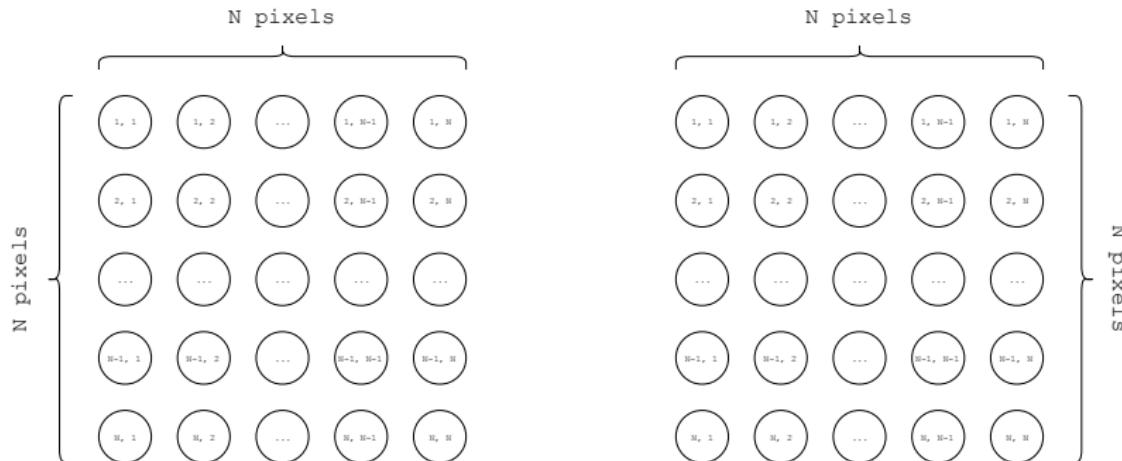


Figura: MLP em pixels de imagens.



MLP para Imagens

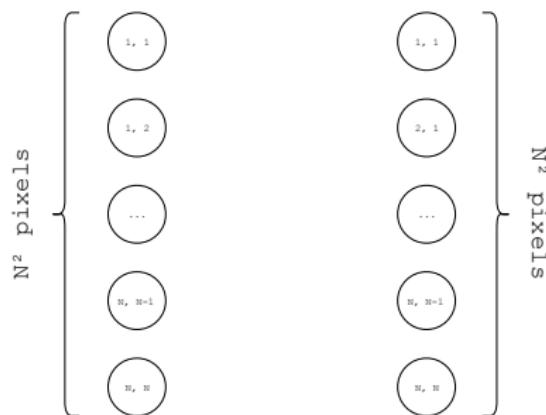


Figura: MLP em pixels de imagens.



MLP para Imagens

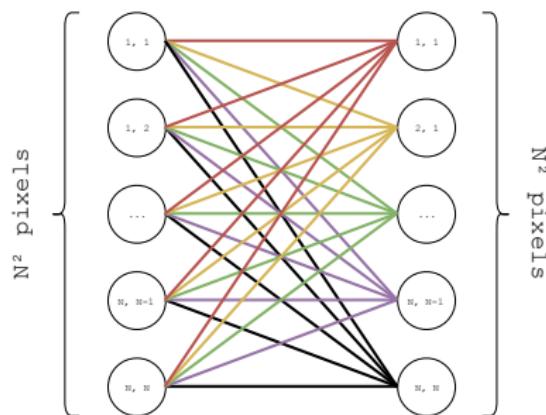


Figura: MLP em pixels de imagens.

MLP para Imagens

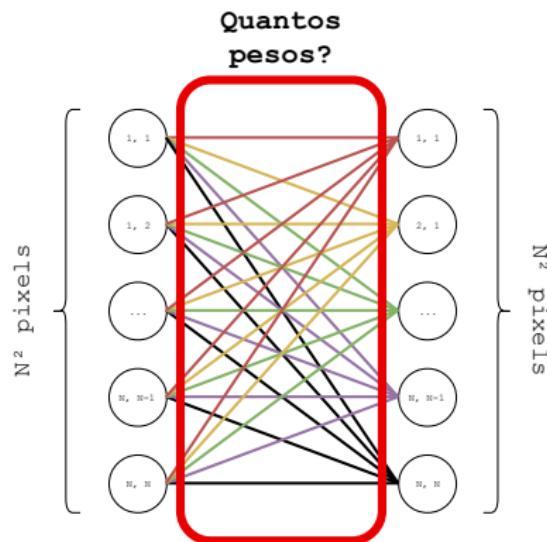


Figura: MLP em pixels de imagens.

MLP para Imagens

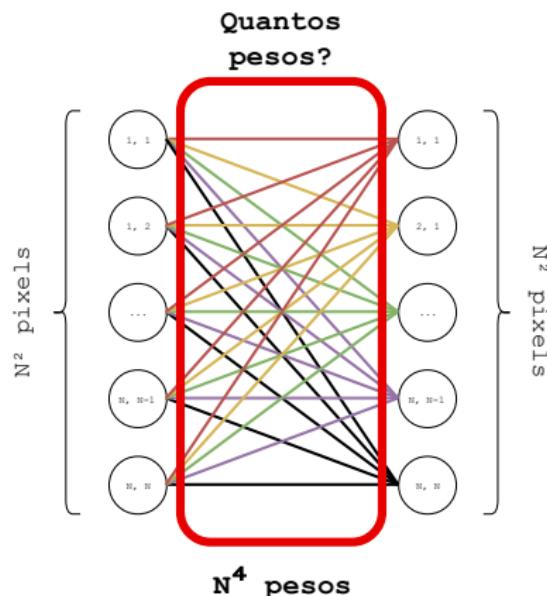


Figura: MLP em pixels de imagens.



MLP para Imagens

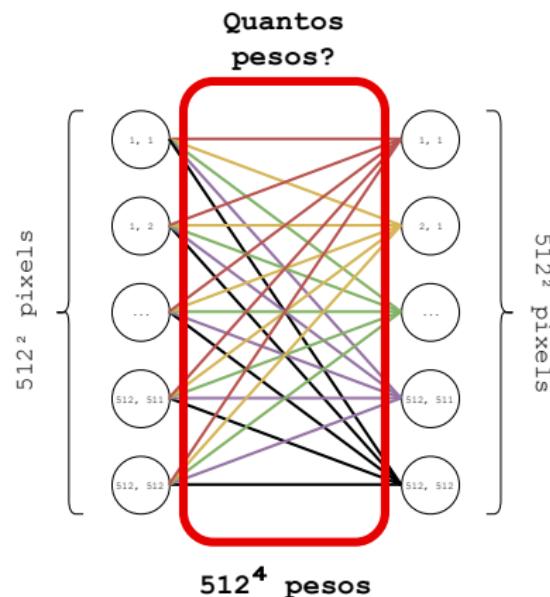


Figura: MLP em pixels de imagens.



MLP para Imagens

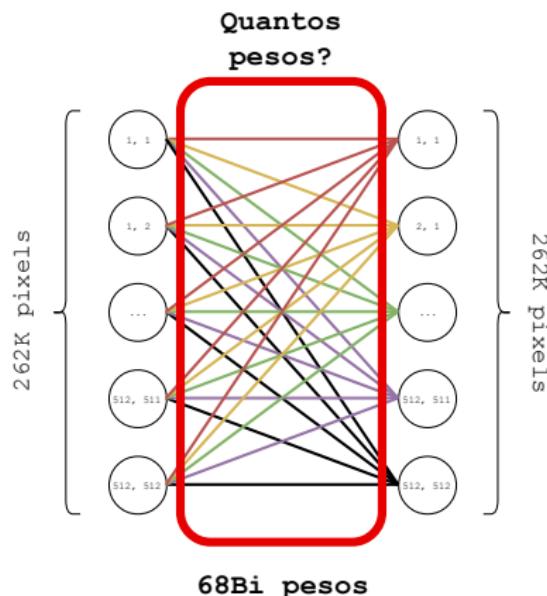


Figura: MLP em pixels de imagens.

MLP para Imagens



Quanto de
Memória?

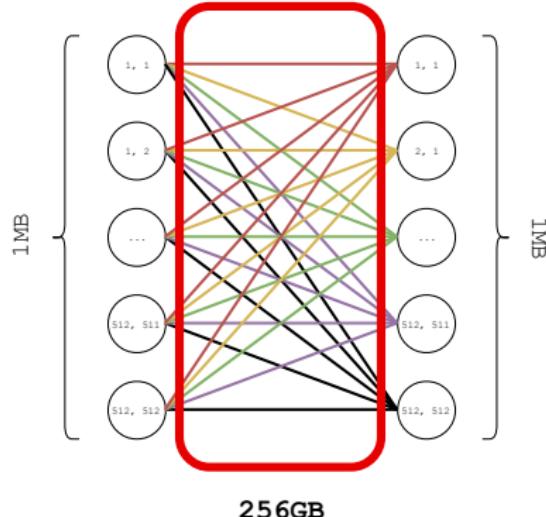


Figura: MLP em pixels de imagens.



Agenda

1 Introdução

2 Redes Neurais Convolucionais

3 Arquitetura Famosas

4 Estratégias de Treino

5 Segmentação de Imagens

6 Detecção em Imagens



Convoluçãoes 2D

- Operação que mede a “semelhança” entre dois sinais
- Imagem: $f(\cdot)$
- Kernel: $h(\cdot)$
- $f(x, y) * h(x, y) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m, n)h(x - m, y - n),$
para $x = 0, 1, \dots, X - 1$ e $y = 0, 1, \dots, Y - 1$



Convolução 1D

Image



Figura: Imagem.

Kernel



Figura: Kernel.



Convolução 1D

Image



Figura: Imagem.

Kernel



Figura: Kernel rebatido.



Convolução 1D

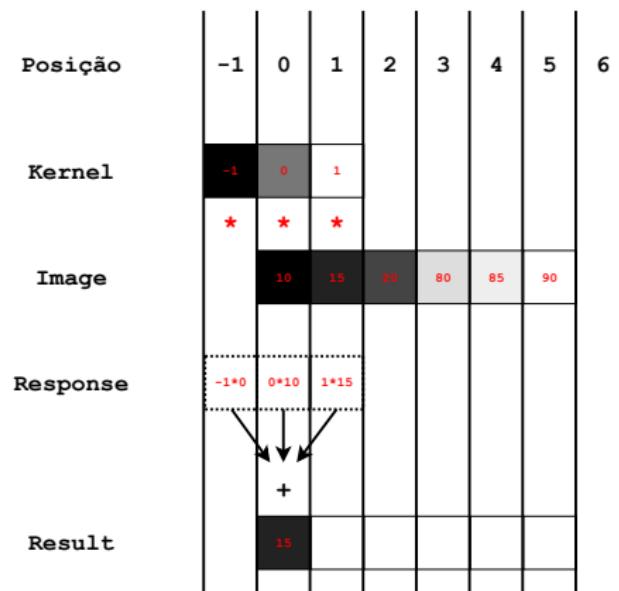


Figura: Cálculo de uma convolução 1D.



Convolução 1D

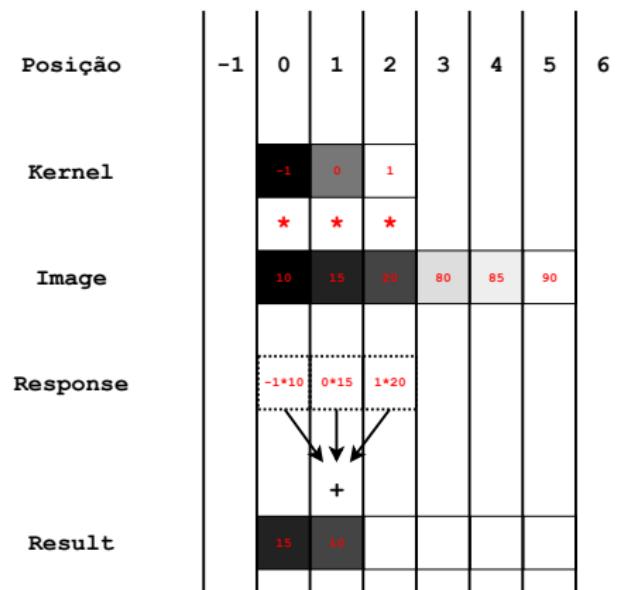


Figura: Cálculo de uma convolução 1D.



Convolução 1D

Posição	-1	0	1	2	3	4	5	6
Kernel			-1 0		1			
Image			*	*	*			
Response			-1*15 0*20 1*80					
Result		15 10 65						

Figura: Cálculo de uma convolução 1D.



Convolução 1D

Posição	-1	0	1	2	3	4	5	6
Kernel				-1 *	0 *	1 *		
Image	10	15	20	80	85	90		
Response				-1*20 0*80 1*85				
Result	15	10	65	65				

Figura: Cálculo de uma convolução 1D.



Convolução 1D

Posição	-1	0	1	2	3	4	5	6
Kernel					-1	0	1	
Image	10	15	20	80	85	90		
Response				-1*80	0*85	1*90		
Result	15	10	65	65	10			

Figura: Cálculo de uma convolução 1D.



Convolução 1D

Posição	-1	0	1	2	3	4	5	6
Kernel					-1	0	1	
Image	10	15	20	80	85	90		
Response					-1*85	0*90	1*0	
Result	15	10	65	65	10	-85		

Figura: Cálculo de uma convolução 1D.



Convolução 2D

10	15	20	80	85	90
10	15	20	80	85	90
10	15	20	80	85	90
10	15	20	80	85	90
10	15	20	80	85	90
10	15	20	80	85	90

Figura: Imagem.

-1	0	1
-1	0	1
-1	0	1

Figura: Kernel.



Convolução 2D

10	15	20	80	85	90
10	15	20	80	85	90
10	15	20	80	85	90
10	15	20	80	85	90
10	15	20	80	85	90
10	15	20	80	85	90

Figura: Imagem.

1	0	-1
1	0	-1
1	0	-1

Figura: Kernel rebatido.



Convolução 2D

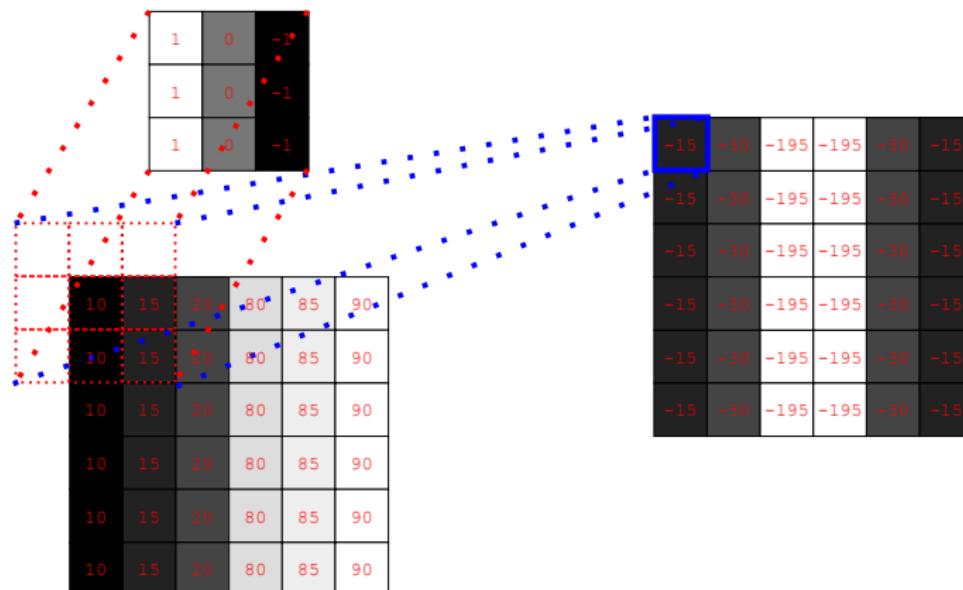


Figura: Cálculo de uma convolução 2D.



Convolução 2D

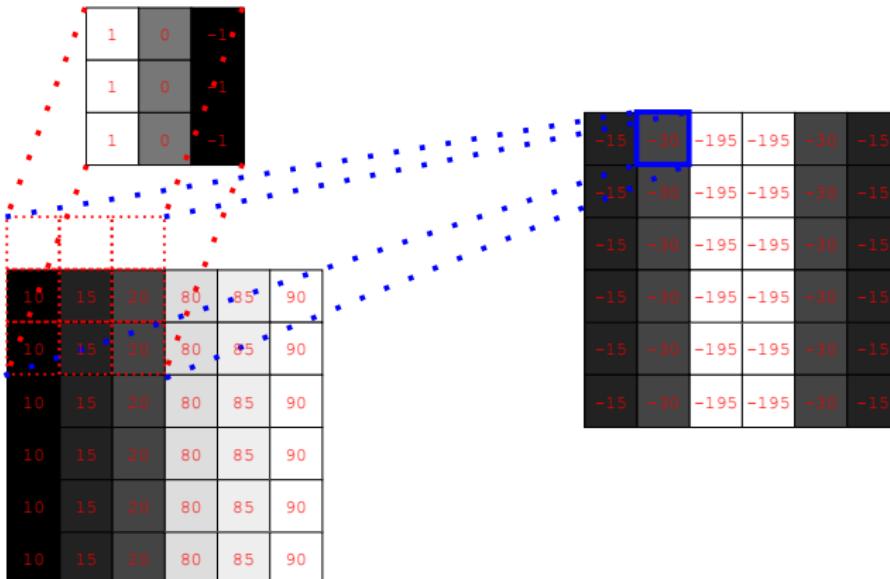


Figura: Cálculo de uma convolução 2D.



Convolução 2D

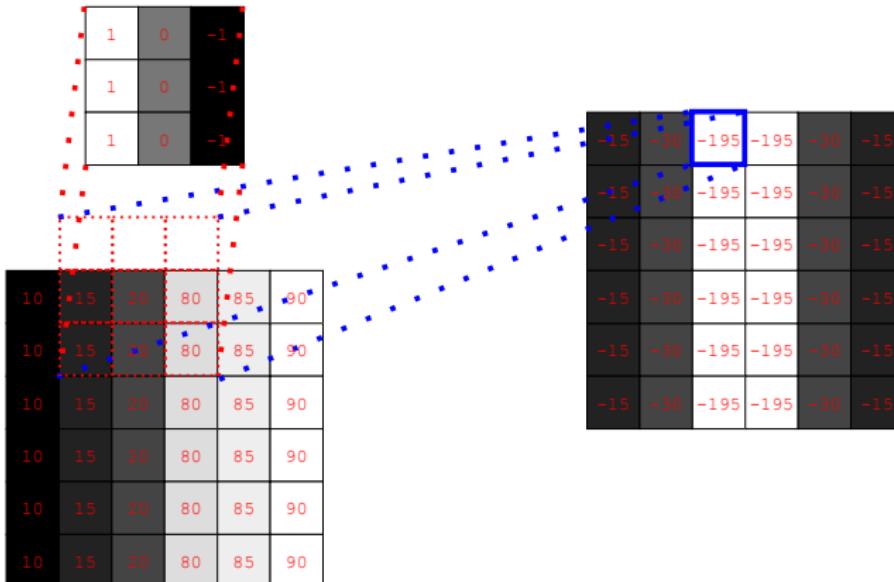


Figura: Cálculo de uma convolução 2D.



Convolução 2D

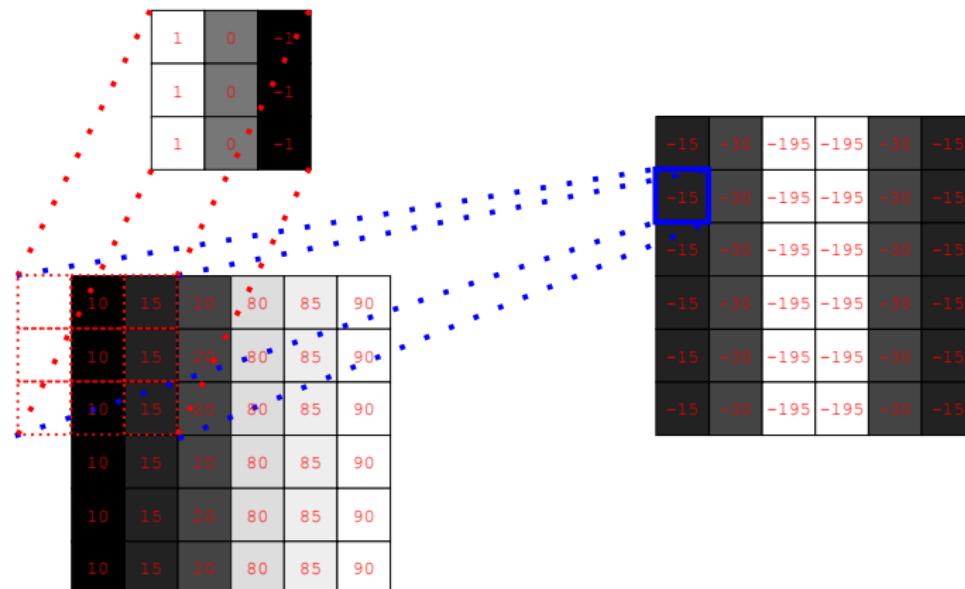


Figura: Cálculo de uma convolução 2D.



Convolução 2D

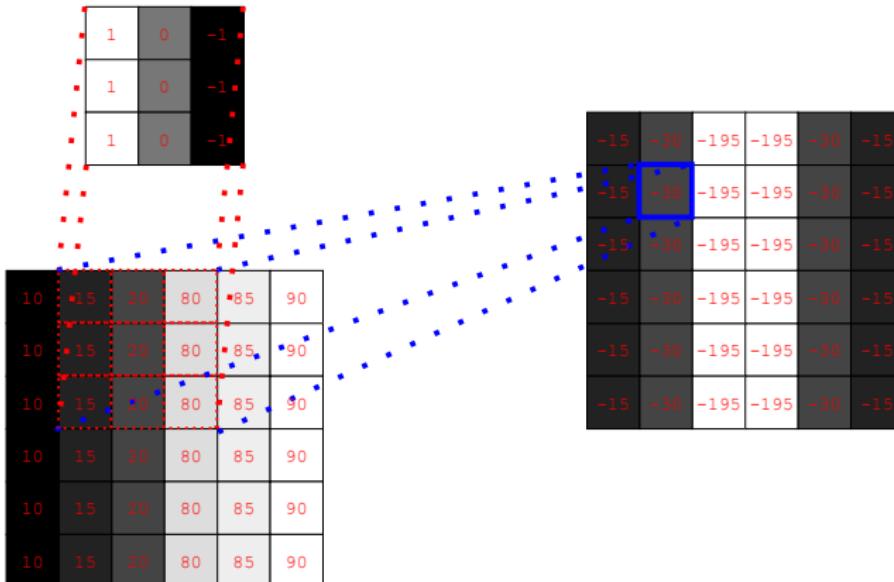


Figura: Cálculo de uma convolução 2D.



Convolução 2D

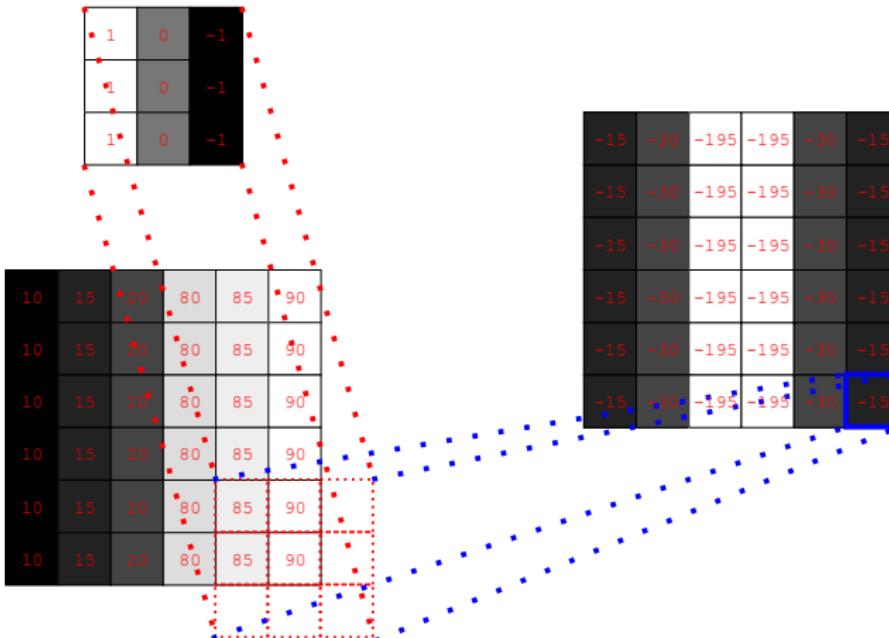


Figura: Cálculo de uma convolução 2D.

Convoluçãoções e Correlações



Convoluçãoções e Correlações

Convoluçãoções e Correlações podem ser usadas para **armazenar informação visual**.

Convoluçãoções e Correlações



Convoluçãoções e Correlações

Convoluçãoções e Correlações podem ser usadas para fazer **casamento de padrões visuais**.

Convoluçãoções e Correlações



Convoluçãoções e Correlações

As respostas geradas por Convoluçãoções e Correlações serão mais fortes nas áreas da imagem que casarem melhor com o conteúdo do kernel.

Filtros Convolucionais



Demo - Convolution Filters

Convolution_Filters.ipynb

Convoluçãoes e Correlações



Demo - Correlation and Convolution

Correlation_Convolution.ipynb

Convoluçãoções e Correlações



Convoluçãoções e Correlações

O uso de convoluções em redes neurais para imagens permite que o **número de parâmetros seja definido apenas pelos tamanhos dos kernels**.

Convoluçãoções e Correlações



Convoluçãoções e Correlações

O número de parâmetros da rede não está mais vinculado ao tamanho da imagem.



CNNs

CNNs

Redes Neurais Convolucionais (CNNs) funcionam **otimizando os pesos do kernel convolucional**.



Camadas Convolucionais

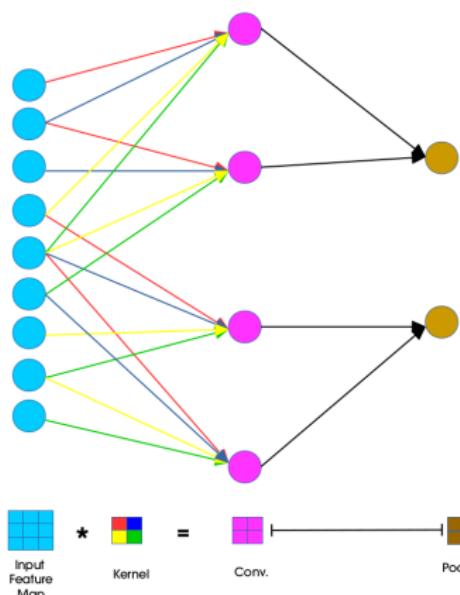


Figura: Exemplo [1] de feed-forward para uma CNN.



Camadas Convolucionais

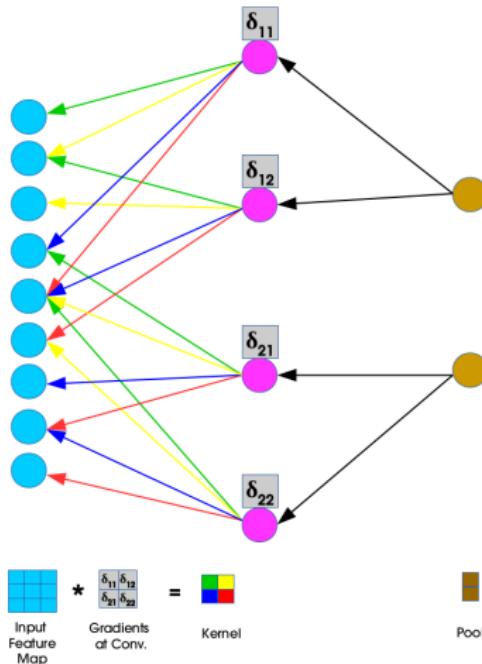


Figura: Exemplo [1] de backpropagation para uma CNN.



CNNs

- Camadas comuns em CNNs:
 - Camadas Convolucionais
 - Camadas de Pooling
 - Camadas Totalmente Conectadas (“Fully Connected”, “Linear”, ou “Dense”)

Camadas Convolucionais

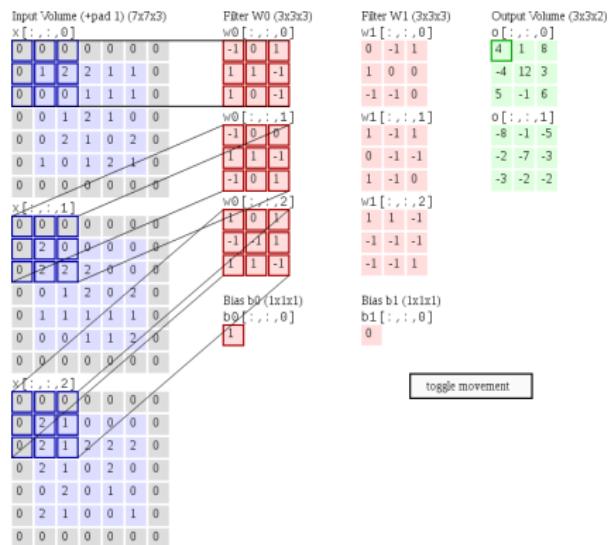


Figura: Convoluçãoes em uma CNN.



Camadas Convolucionais

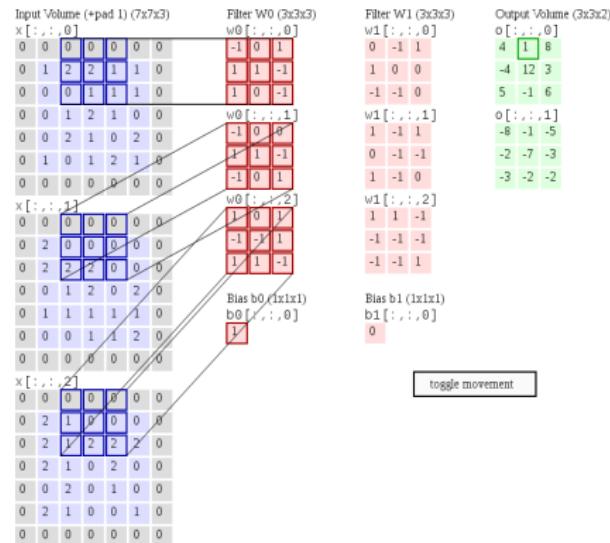


Figura: Convoluçãoções em uma CNN.



Camadas Convolucionais

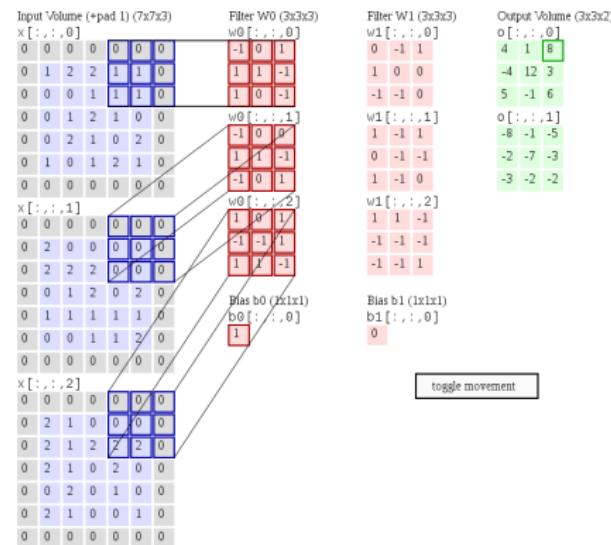


Figura: Convoluçãoções em uma CNN.



Camadas Convolucionais

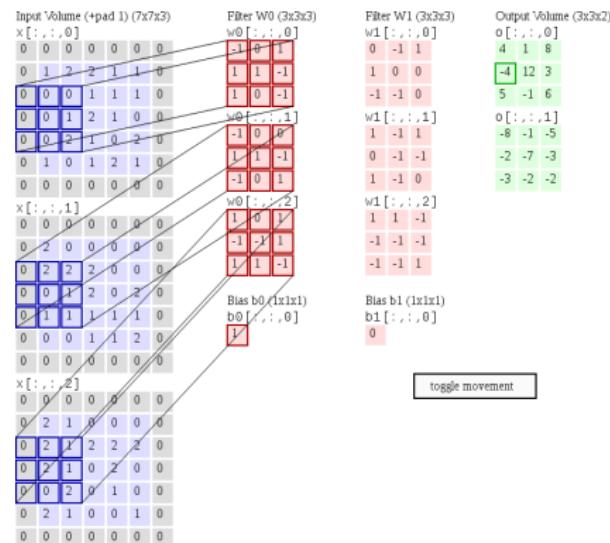


Figura: Convoluçãoções em uma CNN.



Camadas Convolucionais

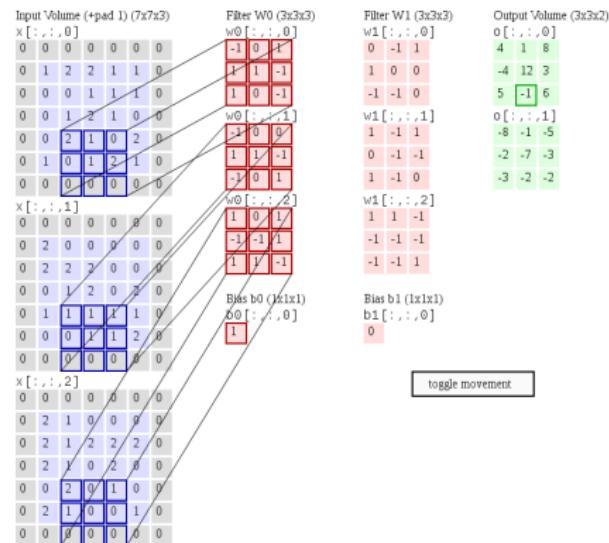


Figura: Convoluçãoes em uma CNN.



Camadas Convolucionais

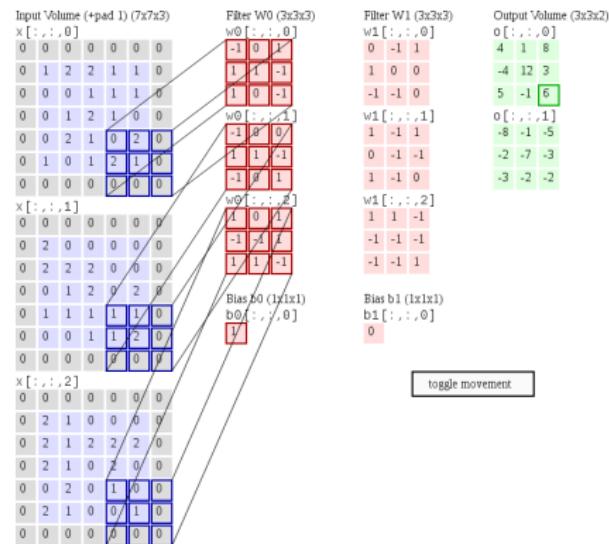


Figura: Convoluçãoções em uma CNN.



Camadas Convolucionais

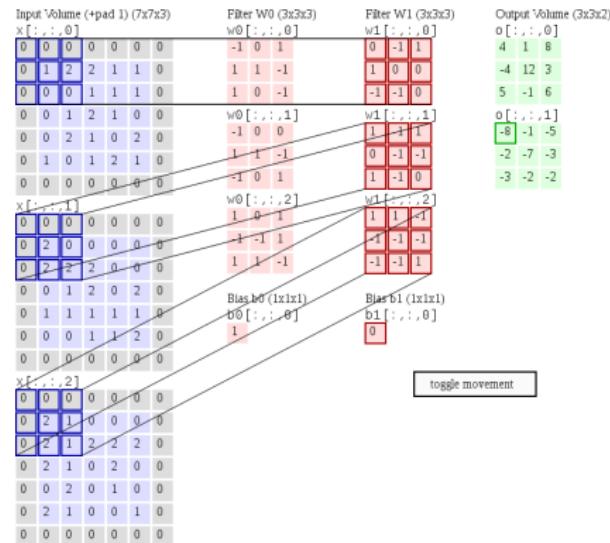


Figura: Convoluçãoções em uma CNN.



Camadas Convolucionais

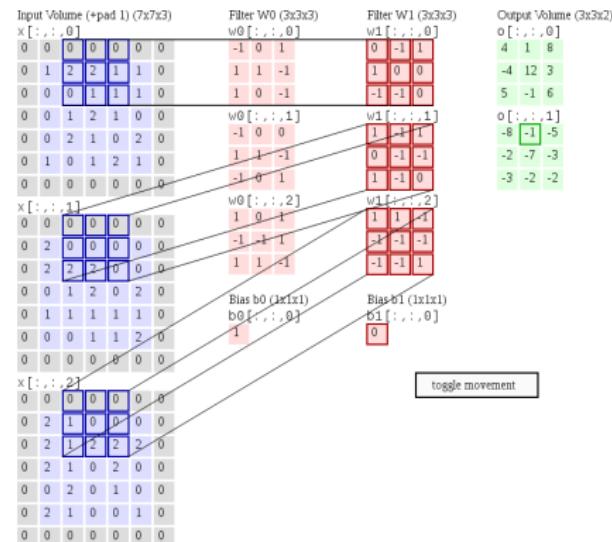


Figura: Convoluçãoções em uma CNN.



Camadas Convolucionais

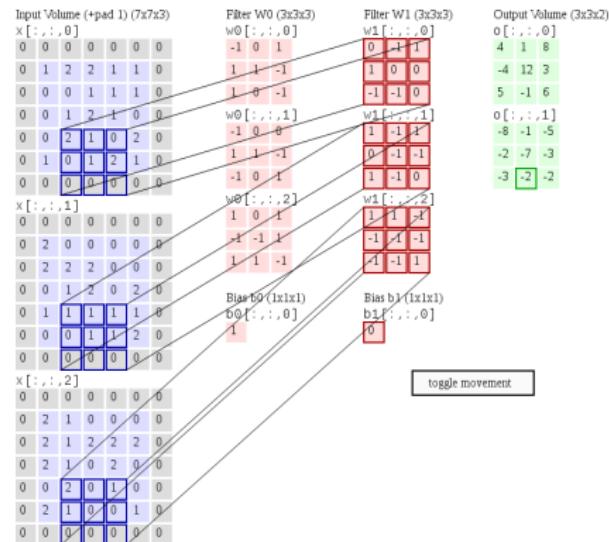


Figura: Convoluções em uma CNN.



Camadas Convolucionais

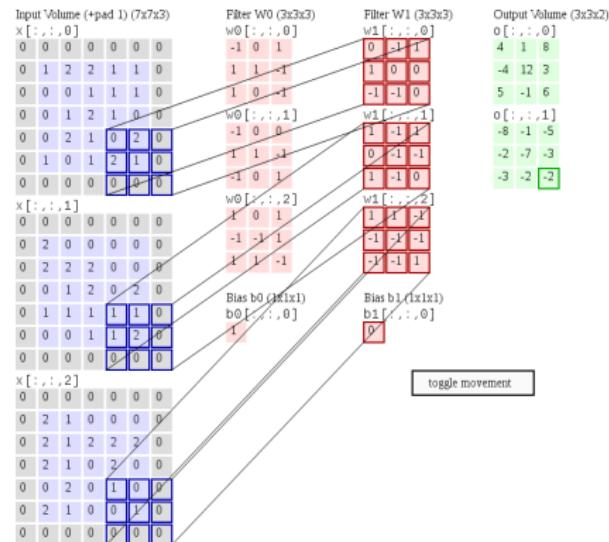


Figura: Convoluçãoções em uma CNN.

Camadas Convolucionais



- Parâmetros de Camadas Convolucionais:
 - Field of View (F)
 - Stride (S)
 - Padding (P)



Field of View

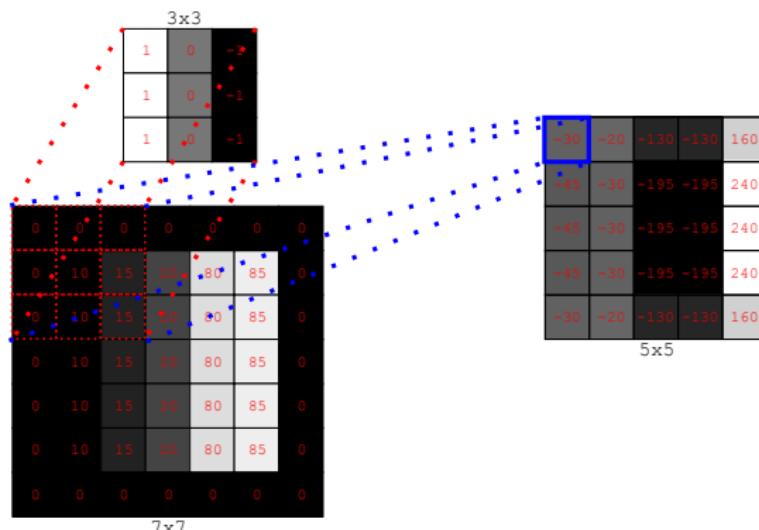


Figura: Field of view de tamanho 3×3 .



Field of View

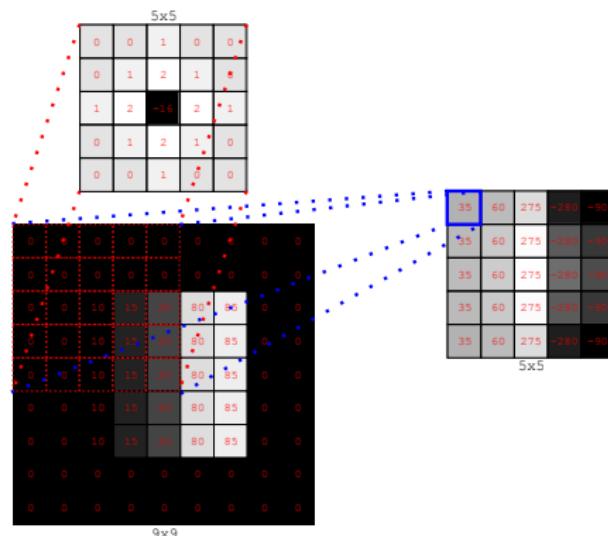


Figura: Field of view de tamanho 5x5.



Stride

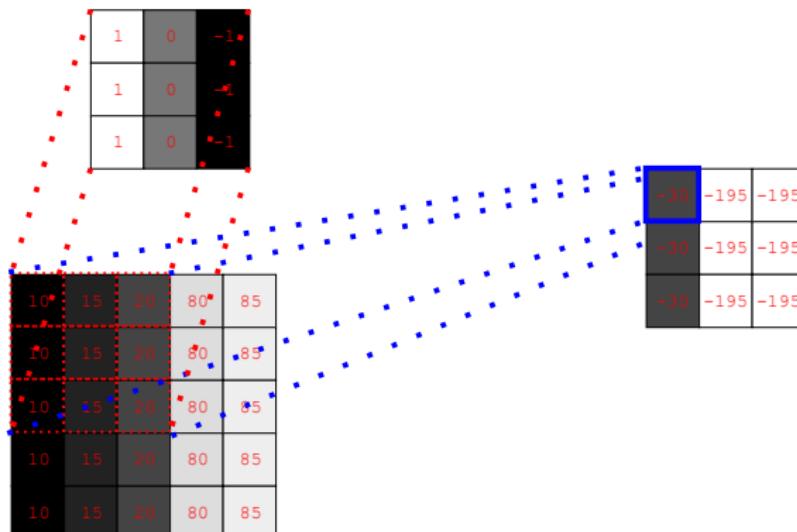


Figura: Stride de tamanho 1.



Stride

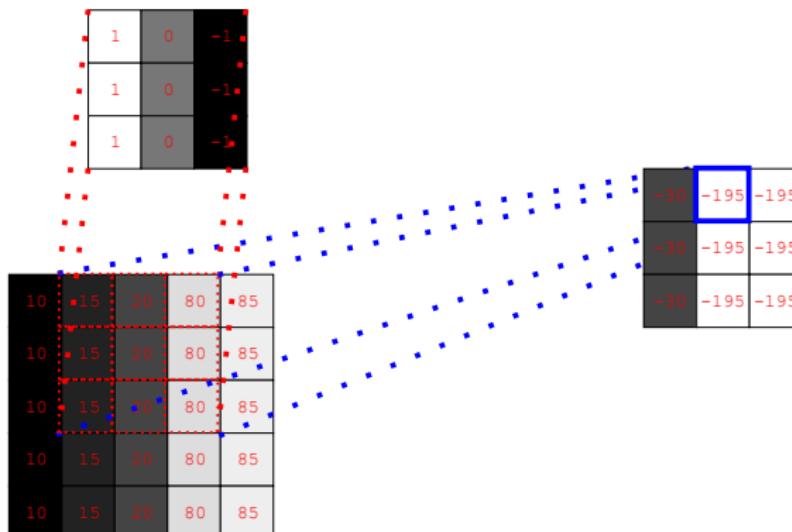


Figura: Stride de tamanho 1.



Stride

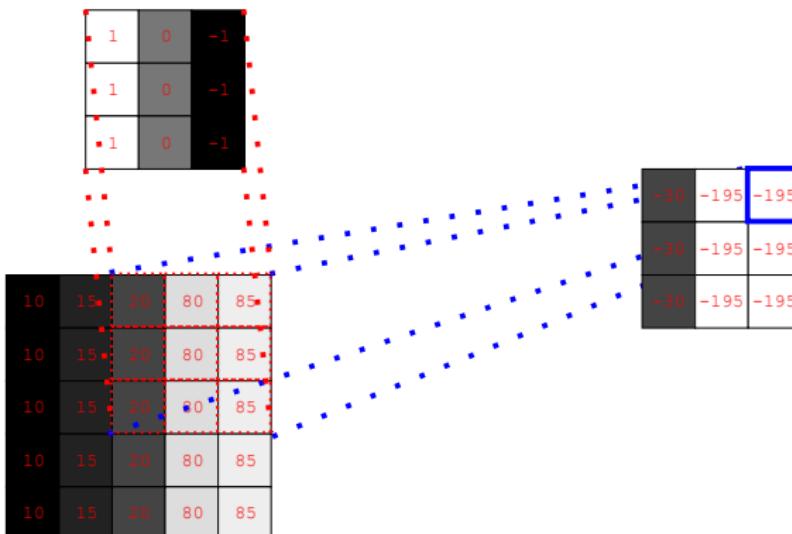


Figura: Stride de tamanho 1.



Stride

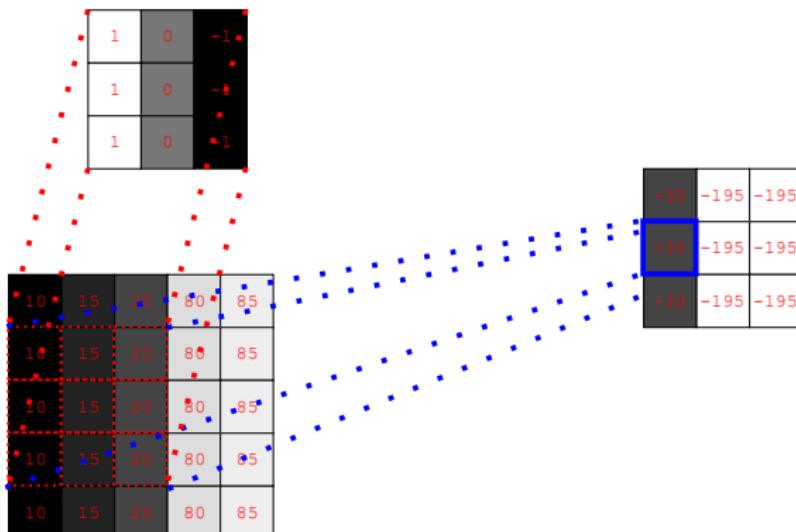


Figura: Stride de tamanho 1.



Stride

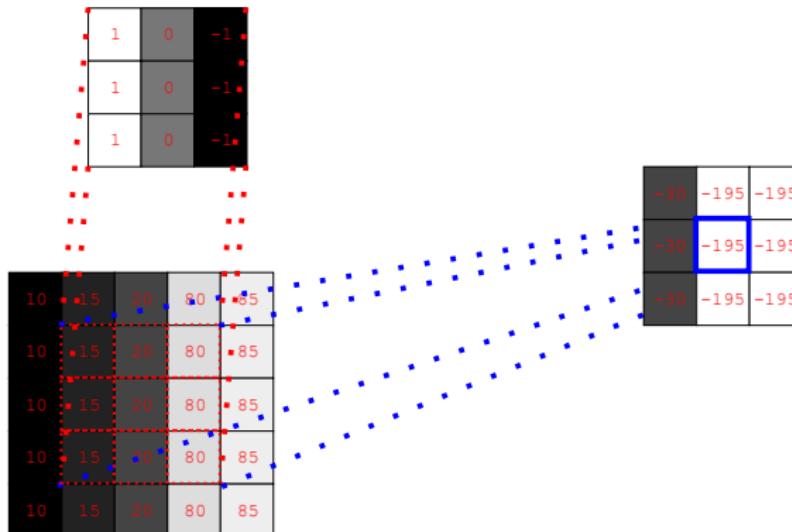


Figura: Stride de tamanho 1.



Stride

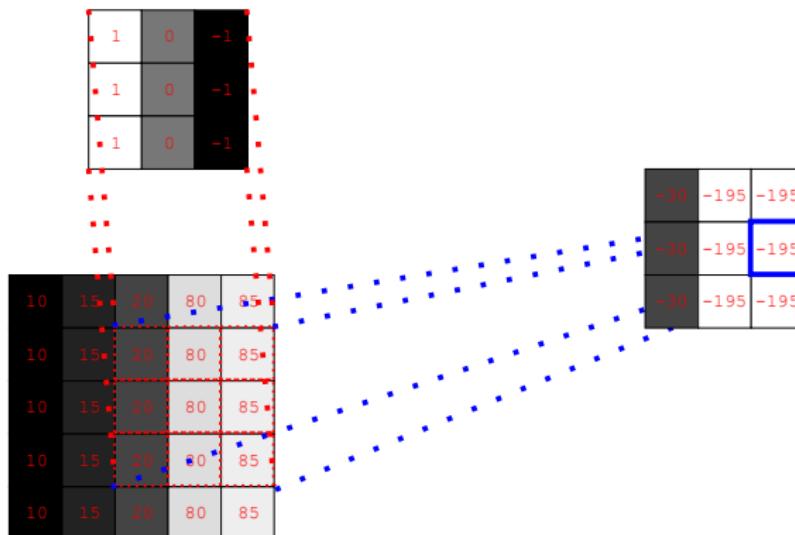


Figura: Stride de tamanho 1.



Stride

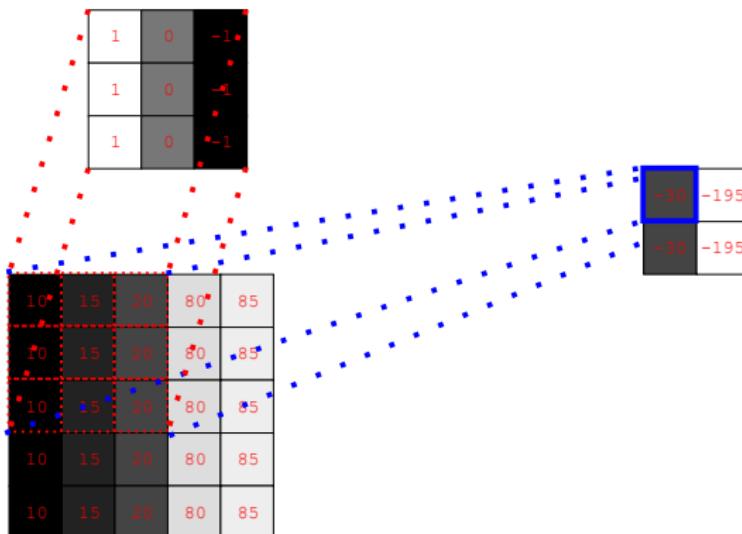


Figura: Stride de tamanho 2.



Stride

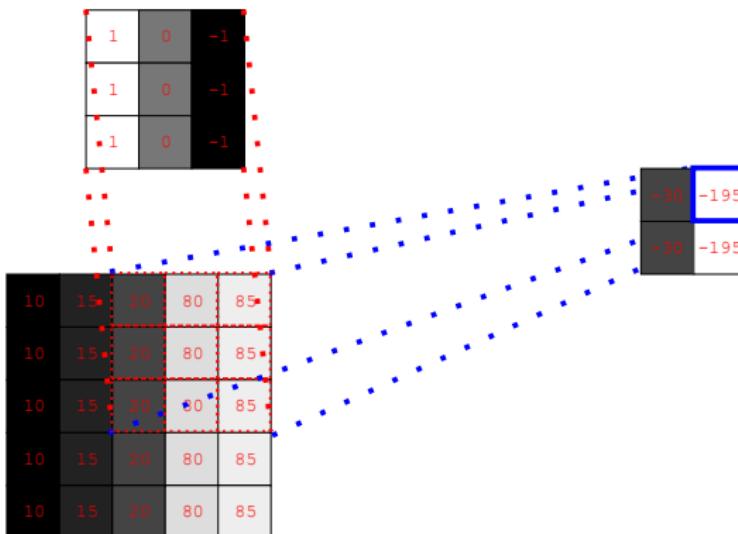


Figura: Stride de tamanho 2.



Stride

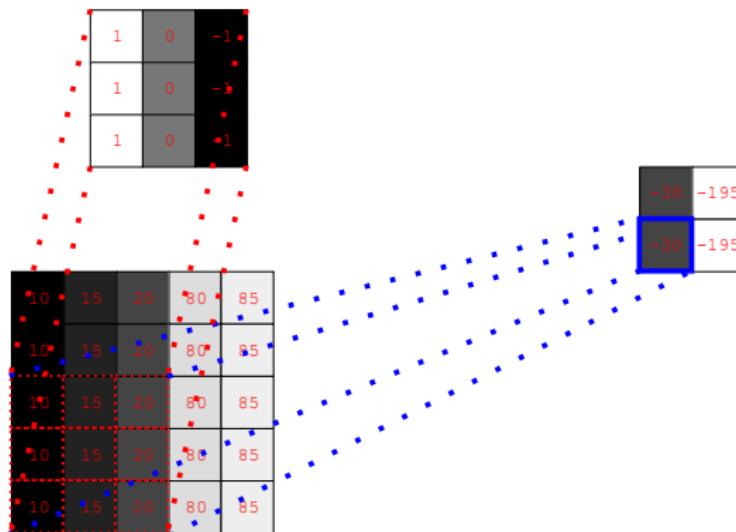


Figura: Stride de tamanho 2.



Stride

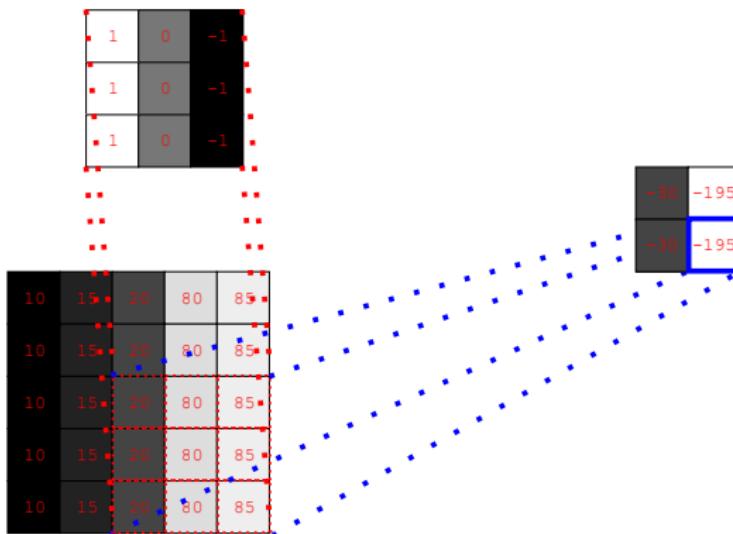


Figura: Stride de tamanho 2.



Padding

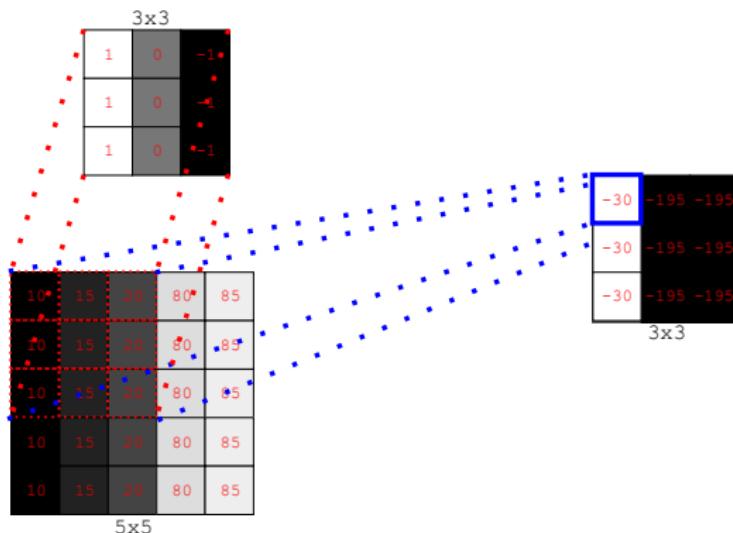


Figura: Padding de tamanho 0.



Padding

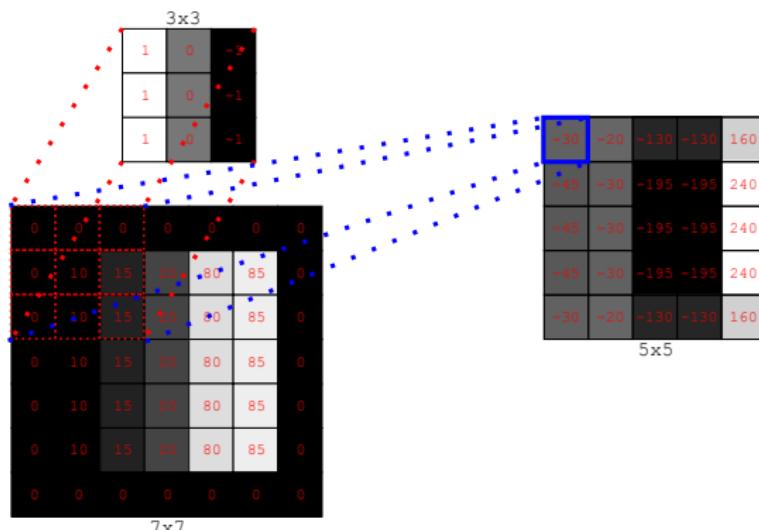


Figura: Padding de tamanho 1.



Padding

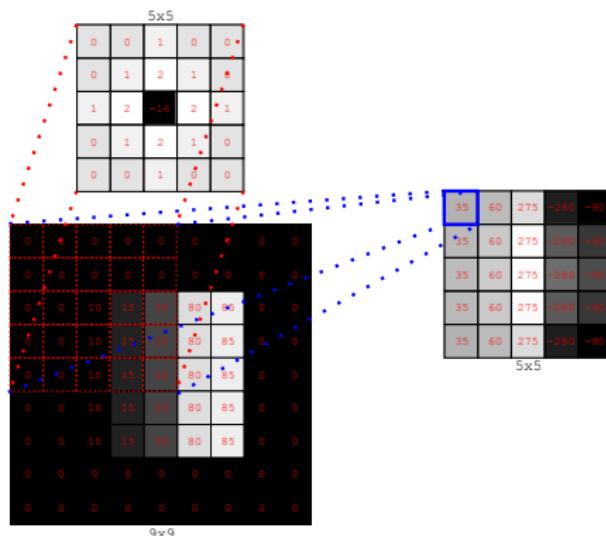


Figura: Padding de tamanho 2.

Resolução de Saída de Convolução



Calculando a Resolução Espacial de Saída

$$X_{out} = ((X_{in} - F + 2P)/S) + 1$$



Resolução de Saída de Convolução

Calculando a Resolução Espacial de Saída

$$X_{out} = ((X_{in} - F + 2P)/S) + 1$$

Calculando a Resolução Espacial de Saída

Qual a resolução de saída de uma camada convolucional se a imagem de entrada for 224×224 , o Field of View (**F**) for 3×3 , o Stride (**S**) for igual a 1 e o Padding (**P**) for igual a 1?



Resolução de Saída de Convolução

Calculando a Resolução Espacial de Saída

$$X_{out} = ((X_{in} - F + 2P)/S) + 1$$

Calculando a Resolução Espacial de Saída

Qual a resolução de saída de uma camada convolucional se a imagem de entrada for 224×224 , o Field of View (**F**) for 7×7 , o Stride (**S**) for igual a 7 e o Padding (**P**) for igual a 0?

Camadas de Pooling



- Parâmetros de Camadas de Pooling:
 - Field of View (F)
 - Stride (S)



Pooling

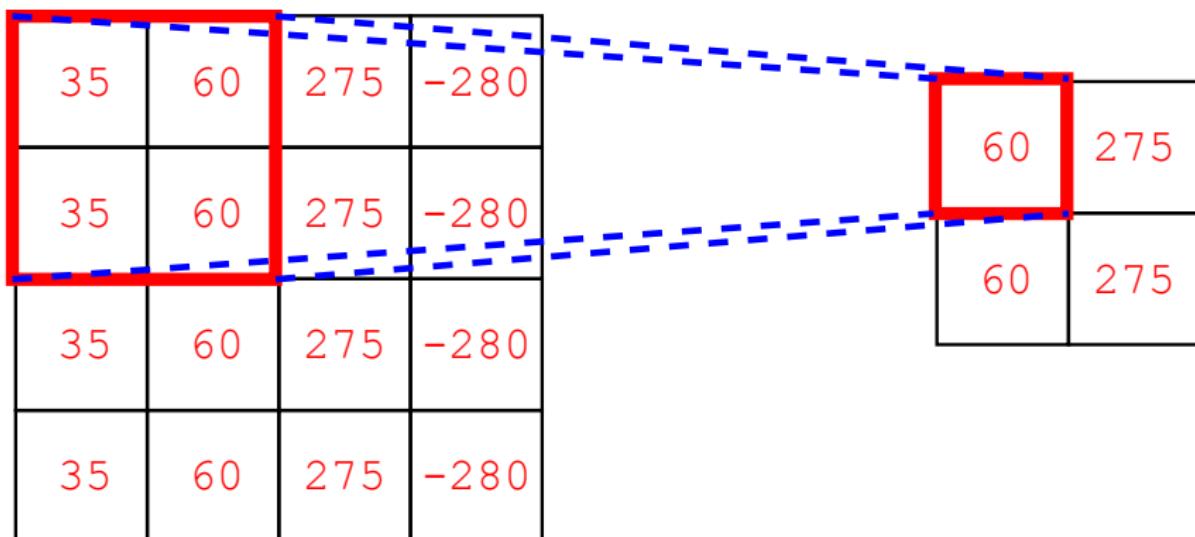


Figura: Exemplo de Max-Pooling.



Pooling

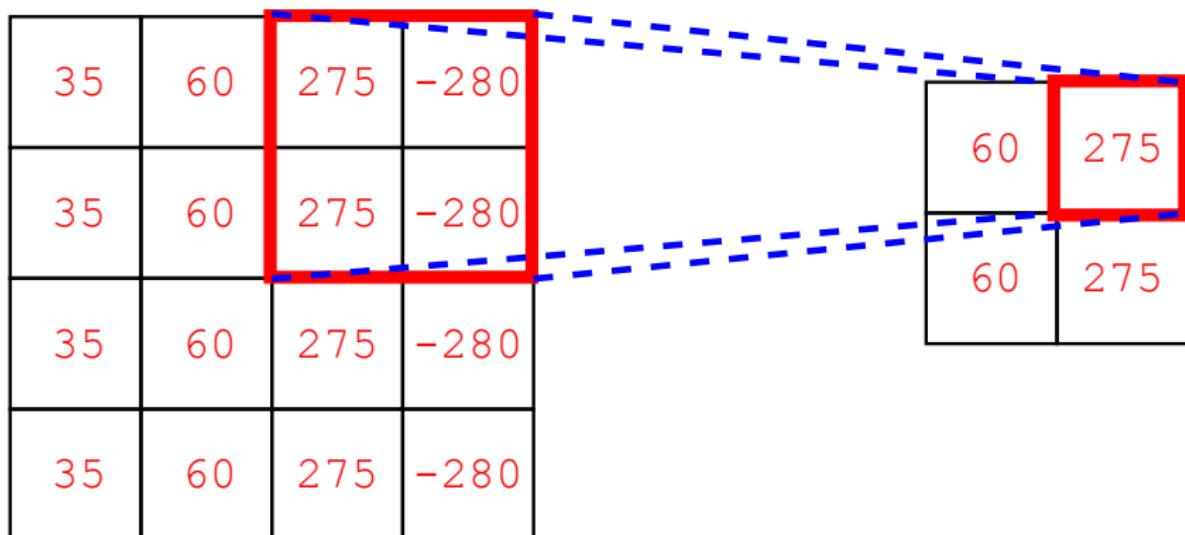


Figura: Exemplo de Max-Pooling.



Pooling

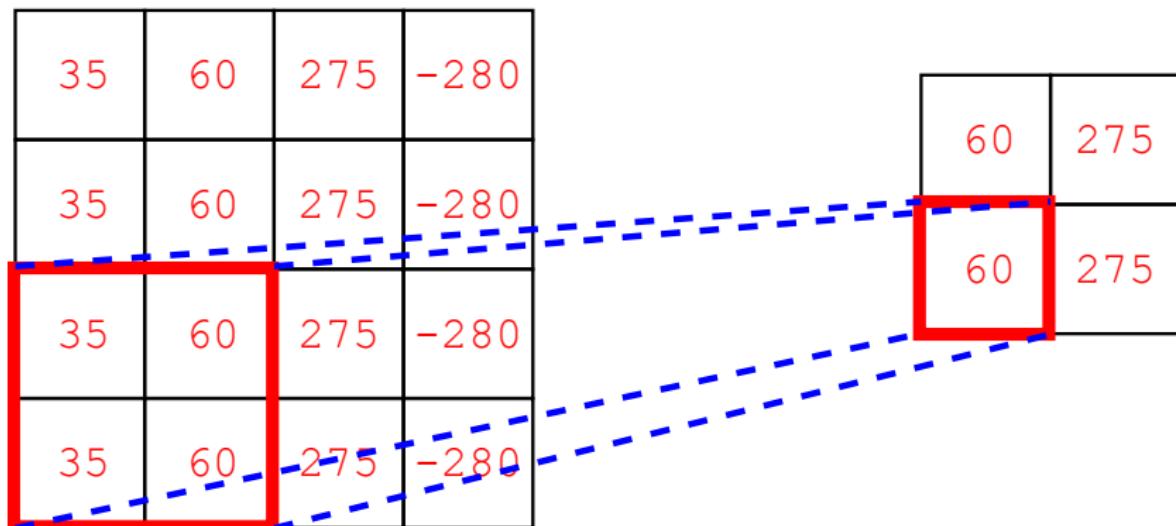


Figura: Exemplo de Max-Pooling.



Pooling

35	60	275	-280
35	60	275	-280
35	60	275	-280
35	60	275	-280

60	275
60	275

Figura: Exemplo de Max-Pooling.



Field of View do Pooling

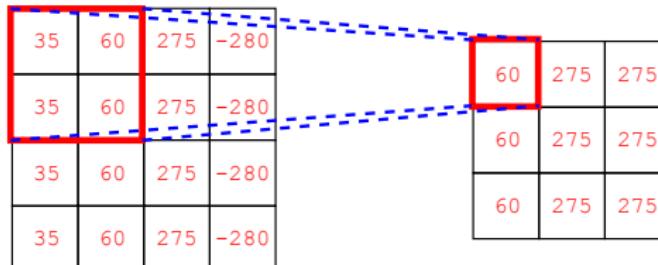


Figura: FoV de 2x2 de uma camada de Max-Pooling.

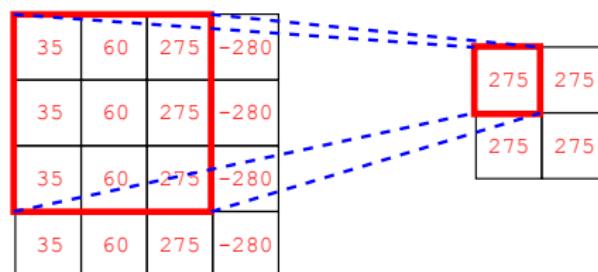


Figura: FoV de 3x3 de uma camada de Max-Pooling.



Stride do Pooling

35	60	275	-280
35	60	275	-280
35	60	275	-280
35	60	275	-280

60	275	275
60	275	275
60	275	275

(a)

35	60	275	-280
35	60	275	-280
35	60	275	-280
35	60	275	-280

60	275	275
60	275	275
60	275	275

(b)

Figura: Stride de tamanho 1.

35	60	275	-280
35	60	275	-280
35	60	275	-280
35	60	275	-280

60	275
60	275

(a)

35	60	275	-280
35	60	275	-280
35	60	275	-280
35	60	275	-280

60	275
60	275

(b)

Figura: Stride de tamanho 2.

Resolução de Saída do Pooling



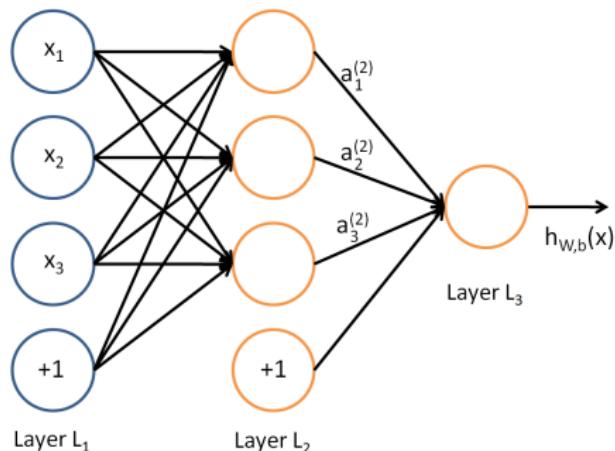
Calculando a Resolução Espacial de Saída

$$W_{out} = ((W_{in} - F_w)/S_w) + 1$$

$$H_{out} = ((H_{in} - F_h)/S_h) + 1$$



Camadas Totalmente Conectadas (MLPs)





Arquitetura Padrão de uma CNN

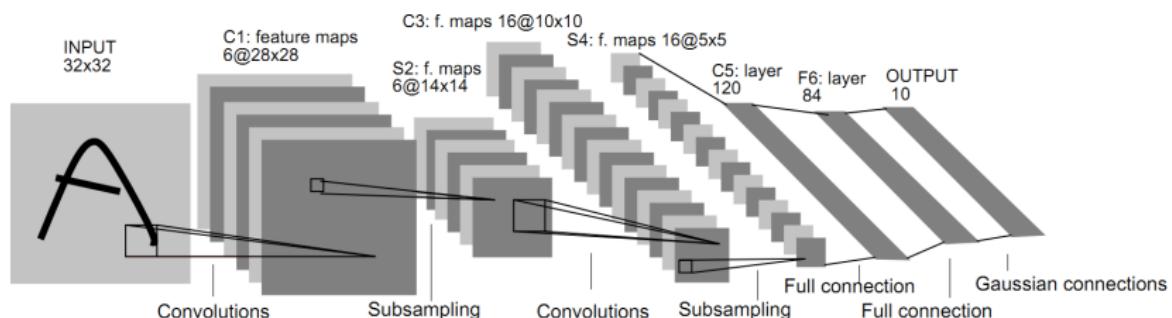


Figura: Arquitetura da LeNet [2].



CNNs no Pytorch

- Detalhes de implementação:

- Camadas convolucionais recebem tensores de dimensões: (BATCH_SIZE, INPUT_CHANNELS, WIDTH, HEIGHT)
- Não é preciso especificar o número de canais de entrada nas camadas de Max-Pooling e ReLU
- É preciso linearizar os tensores manualmente antes da primeira camada Fully Connected usando a função `view()`
- É preciso passar o número de canais da Camada Convolucional anterior para as camadas de Batch Normalization 2D.
- Leiam as documentações das funções antes de implementar



CNNs no Pytorch

- Detalhes de implementação:

- Camadas Convolucionais (`torch.nn.Conv2d`)¹
- Camadas de Pooling (`torch.nn.MaxPool2d`)²
- Camadas de Batch Normalization (`torch.nn.BatchNorm2d`)³
- Ativação ReLU (`torch.nn.ReLU`)⁴
- Camadas Fully Connected (`torch.nn.Linear`)⁵

¹ <https://pytorch.org/docs/stable/nn.html#conv2d>

² <https://pytorch.org/docs/stable/nn.html#maxpool2d>

³ <https://pytorch.org/docs/stable/nn.html#batchnorm2d>

⁴ <https://pytorch.org/docs/stable/nn.html#relu>

⁵ <https://pytorch.org/docs/stable/nn.html#linear>

Procedimento de Treinamento de CNNs



Exercício Prático - Arquitetura

Arquitetura (parte 1):

- Camada Convolucional com $X_{in} = 1$, $X_{out} = 32$, $F = 3$, $S = 1$ e $P = 1$
- Camada de Batch Normalization 2D
- Camada Convolucional com $X_{in} = 32$, $X_{out} = 32$, $F = 3$, $S = 1$ e $P = 1$
- Camada de Batch Normalization 2D
- Camada ReLU
- Camada de Max-Pooling com $F = 2$ e $S = 2$
- #####A saída aqui deve ter 32 canais e resolução 14×14
- Camada Convolucional com $X_{in} = 32$, $X_{out} = 64$, $F = 3$, $S = 1$ e $P = 1$
- Camada de Batch Normalization 2D
- Camada Convolucional com $X_{in} = 64$, $X_{out} = 64$, $F = 3$, $S = 1$ e $P = 1$
- Camada de Batch Normalization 2D
- Camada ReLU
- Camada de Max-Pooling com $F = 2$ e $S = 2$
- #####A saída aqui deve ter 64 canais e resolução 7×7

Procedimento de Treinamento de CNNs



Exercício Prático - Arquitetura

Arquitetura (parte 2):

- #####Nesse ponto é preciso linearizar a saída da última camada para entrar na Fully Connected
- Camada Fully Connected ($7 \times 7 \times 64$) -> 200
- Camada Dropout
- Camada ReLU
- Camada Fully Connected 200 -> 50
- Camada Dropout
- Camada ReLU
- Camada Fully Connected 50 -> 10



Procedimento de Treinamento de CNNs

Exercício Prático - Training Procedure

Training_Procedure.ipynb



Agenda

- 1** Introdução
- 2** Redes Neurais Convolucionais
- 3** Arquitetura Famosas
- 4** Estratégias de Treino
- 5** Segmentação de Imagens
- 6** Detecção em Imagens



Arquitetura Famosas

- LeNet [2] (1998)
- AlexNet [3] (2012)
- VGG [4] (2014)
- GoogLeNet [5] (2015)
- ResNets [6] (2016)
- DenseNets [7] (2017)



Arquitetura Famosas

- AlexNet [3] – primeiro uso de CNN em datasets desafiadores
- *ImageNet Large Scale Visual Recognition Competition* (ILSVRC)
- Inovações:
 - Novas funções de ativação (ReLU)
 - Implementação em GPU
 - Regularização via Dropout

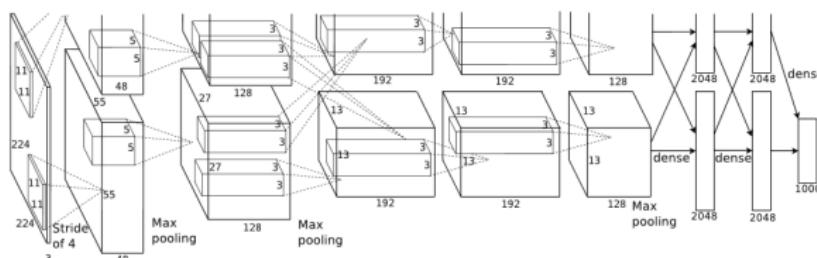


Figura: Arquitetura da AlexNet. Fonte: [3].

Arquitetura Famosas



VGG, 19 layers
(ILSVRC 2014)

- VGGs [4] – ideia de que redes mais profundas geram representações semânticas mais ricas
- Convoluçãoes 3x3 (Padding 1) + Max-Pooling



Figura: Arquitetura da VGG.
Fonte: [6].

Arquitetura Famosas



- GoogLeNet [5] – módulos de convoluções paralelas (Inception)
- Combinam convoluções com diferentes tamanhos de kernel
- Redes mais profundas possuem representação semântica de mais alto nível

GoogleNet, 22 layers
(ILSVRC 2014)



Figura: Arquitetura da GoogLeNet.
Fonte: [5].



Arquitetura Famosas

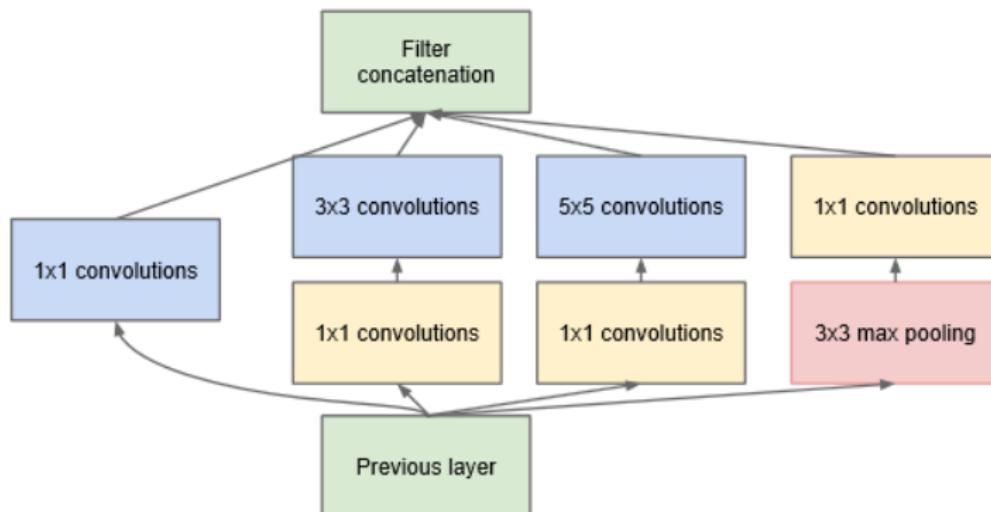


Figura: Módulo Inception da GoogLeNet [5]



Arquitetura Famosas

Nível Semântico de Redes Deep

VGG e GoogLeNet partem da premissa que mais camadas conseguem extrair informações de nível semântico mais alto. **Como explicar o seguinte gráfico então?**

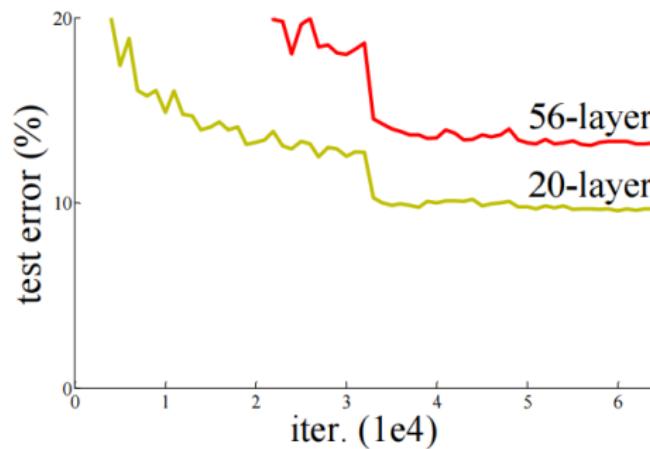


Figura: Número de Camadas vs. Erro no conjunto de teste. Fonte: [6].



Vanishing Gradient

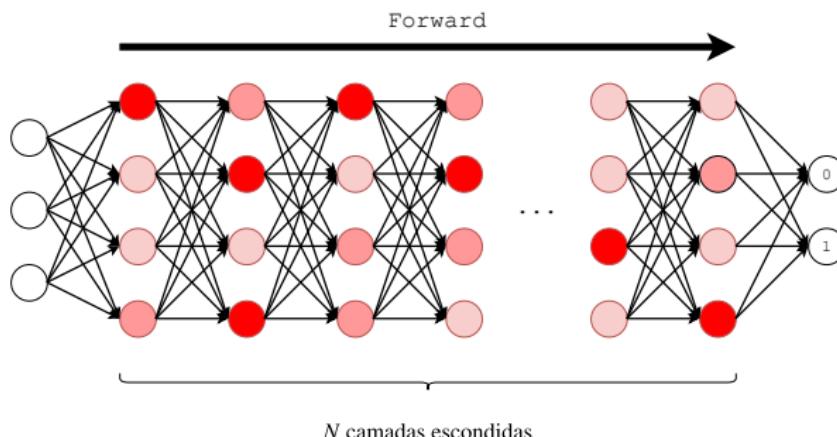


Figura: Exemplo do problema do Vanishing Gradient.



Vanishing Gradient

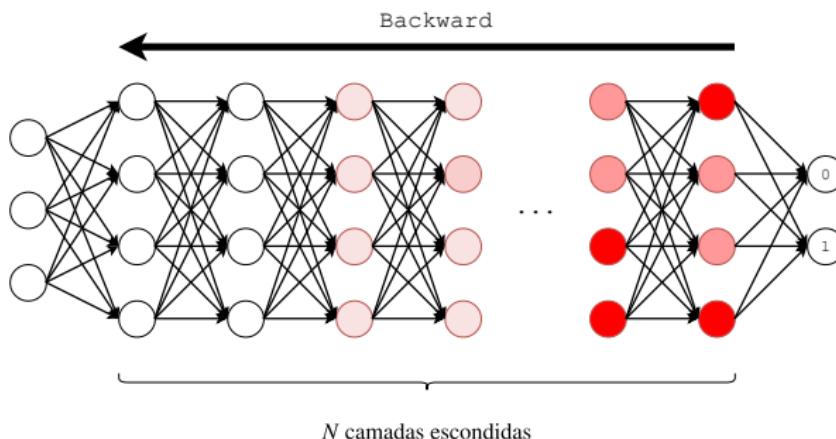


Figura: Exemplo do problema do Vanishing Gradient.

Arquitetura Famosas



- Deep Residual Networks (ResNets)
- Nem sempre redes mais profundas representam melhor o dado
- Como contornar o problema do Vanishing Gradient?



Arquitetura Famosas

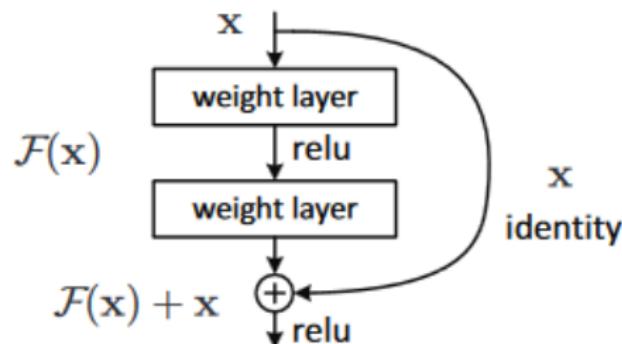
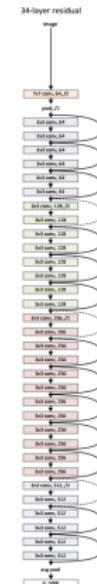


Figura: Conexões residuais das ResNets. Fonte: [6].

Arquitetura Famosas



(a)



(b)

Figura: Arquitetura de (a) uma CNN comum e (b) uma ResNet com 34 camadas. Fonte: [6].



Arquitetura Famosas

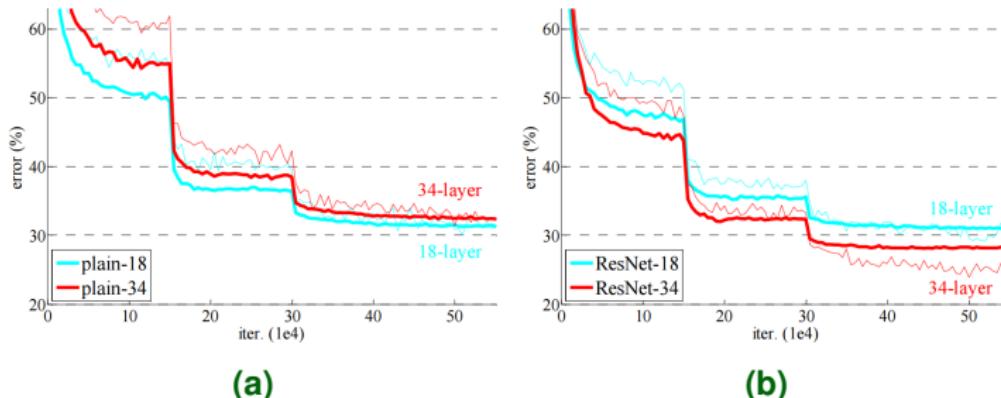


Figura: CNNs comuns (a) vs. ResNets (b). Fonte: [6].

Arquitetura Famosas



Eficiência de Parâmetros

Foi observado [6] que grande parte dos filtros das ResNets não contribuem em nada na inferência feita por essas redes.

Arquitetura Famosas



Eficiência de Parâmetros

Arquiteturas com melhor eficiência de parâmetros [8, 9, 10] começaram a ser propostas.



Arquitetura Famosas

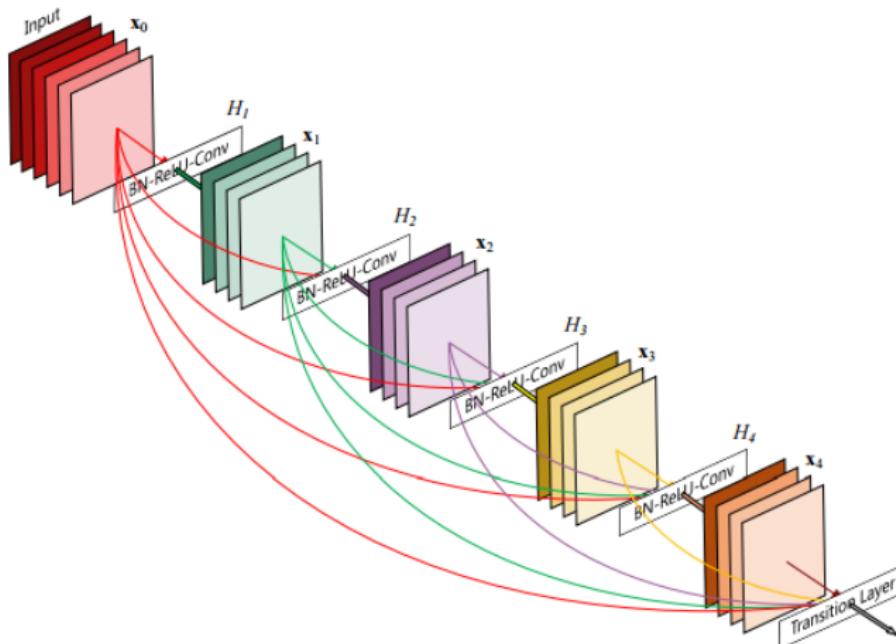


Figura: Bloco denso de uma DenseNet. Fonte: [7].



Arquitetura Famosas

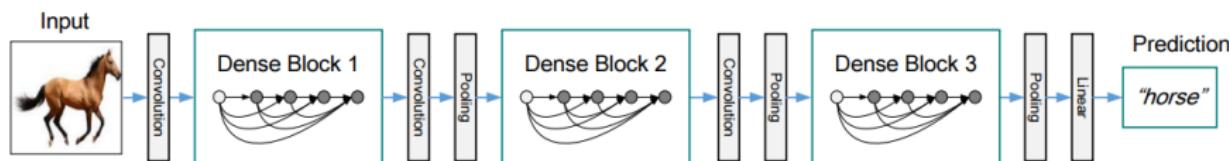


Figura: Arquitetura de uma DenseNet. Fonte: [7].



Arquitetura Famosas

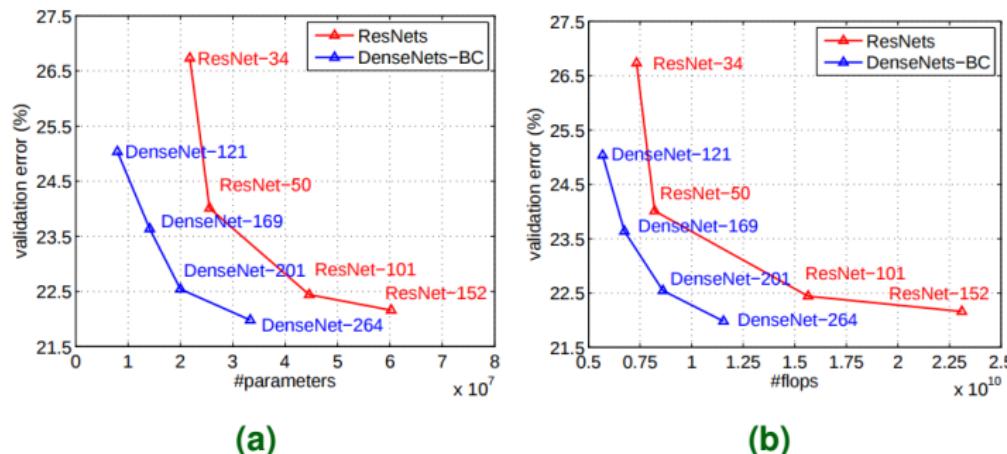


Figura: ResNets vs. DenseNets. Fonte: [7].

Arquitetura Famosas



Modelos Pré-Definidos no *torchvision*

Praticamente todas as arquiteturas clássicas estão disponíveis no pacote *torchvision*⁶. Versões não treinadas e pré-treinadas das arquiteturas estão disponíveis e, para o caso das versões pré-treinadas, o download e carregamento dos pesos é feito automaticamente, caso o arquivo ainda não tenha sido baixado. Basta importar o subpacote *models* e instanciar as redes, como explicado no link no rodapé deste slide.

⁶<https://pytorch.org/docs/stable/torchvision/models.html>

Agenda



- 1 Introdução**
- 2 Redes Neurais Convolucionais**
- 3 Arquitetura Famosas**
- 4 Estratégias de Treino**
- 5 Segmentação de Imagens**
- 6 Detecção em Imagens**

Estratégias de Treino



- Treinamento completo
- Extração de features
- Fine-Tuning

Treinamento Completo

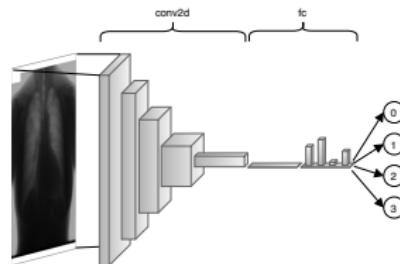


Figura: Treinamento completo de uma CNN.



Extração de Features

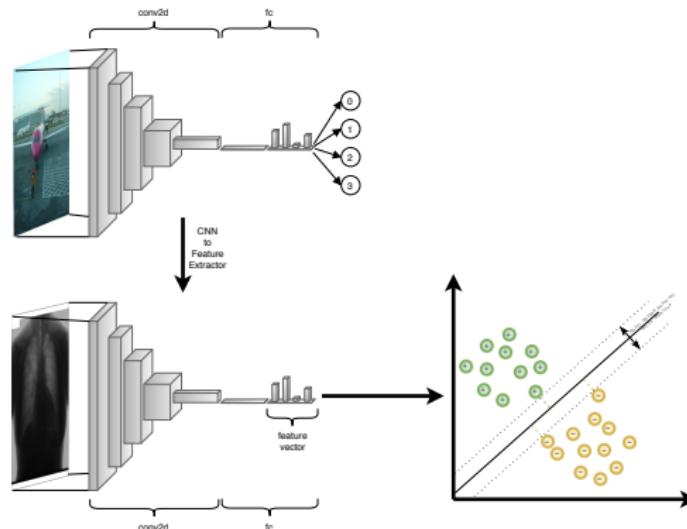


Figura: Extração de features usando uma CNN.

Fine-Tuning

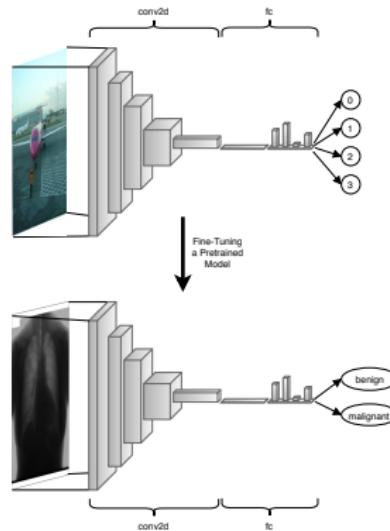


Figura: Fine-tuning em uma CNN.



Fine-Tuning

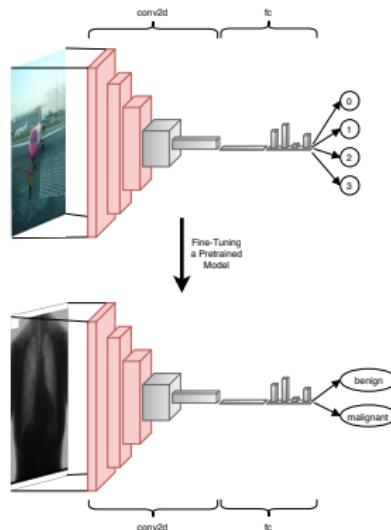


Figura: Fine-tuning em uma CNN.



Fine-Tuning

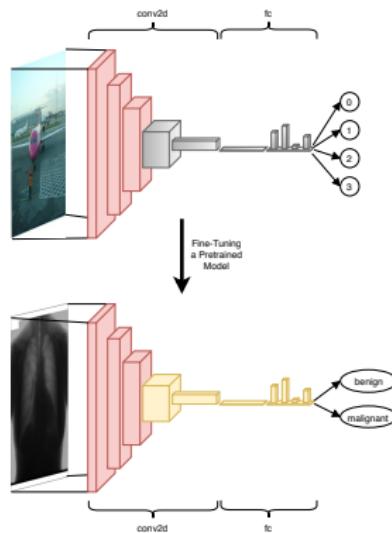


Figura: Fine-tuning em uma CNN.

Transfer Learning



Demo - Transfer Learning

Transfer_Learning.ipynb



Agenda

- 1** Introdução
- 2** Redes Neurais Convolucionais
- 3** Arquitetura Famosas
- 4** Estratégias de Treino
- 5** Segmentação de Imagens
- 6** Detecção em Imagens

Predição Esparsa



- Convolutional Neural Networks (CNNs)
 - Arquitetura mais usada em imagens
- Predição Esparsa
 - 1 label/predição por imagem

Predição Esparsa



- Principais Camadas

- Camadas Convolucionais
- Camadas de Max-Pooling
- Camadas Fully Connected (FC)

- Outras Camadas

- Rectified Linear Unit (ReLU)
- Batch Normalization
- Softmax para Classificação
- Sigmóide para Regressão

Predição Esparsa

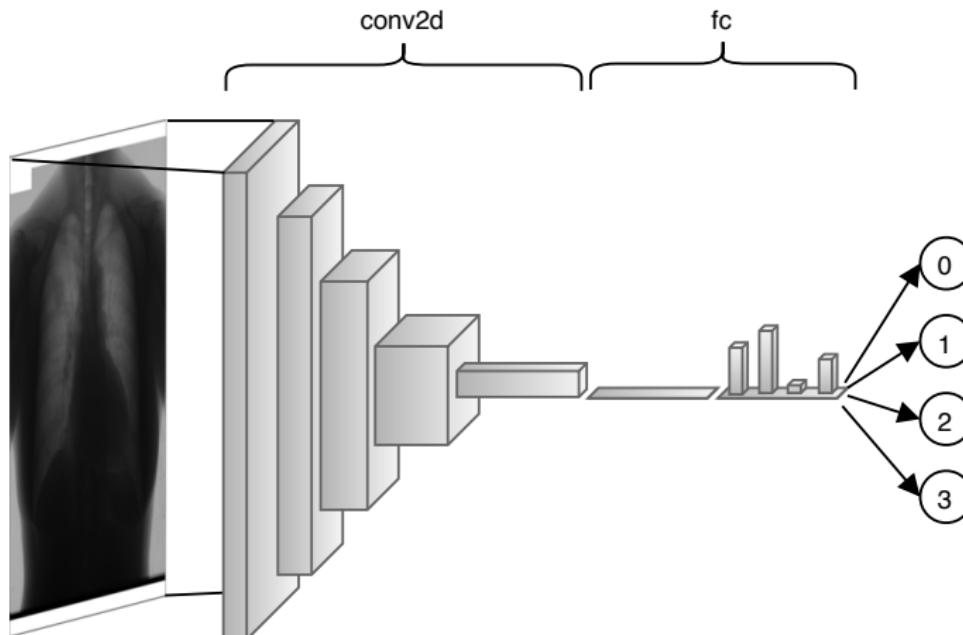


Figura: Arquitetura de uma CNN.

Rotulação Esparsa → Rotulação Densa



- Rotulação Esparsa
 - 1 classe por imagem
- Rotulação Densa
 - 1 classe por pixel

Rotulação Esparsa → Rotulação Densa



- Rotulação Esparsa

- Classificação de objetos
- Regressão da probabilidade de malignidade em imagens radiológicas
- Classificação entre imagens de plantações de café e de imagens urbanas

- Rotulação Densa

- Segmentação de objetos
- Segmentação de tumores
- Segmentação de regiões de plantação



Métodos Clássicos de Segmentação

- Métodos de Processamento de Imagens
 - Watershed, Active Contour, Thresholding, Operações Morfológicas (i.e. Top-Hat), Region Growing, Image Foresting Transform (IFT)...
- Métodos com uso de Learning
 - Regiões de contexto, Superpixels, Clustering
 - Extração de features
 - Classificação/clusterização do pixel ou superpixel



Watershed

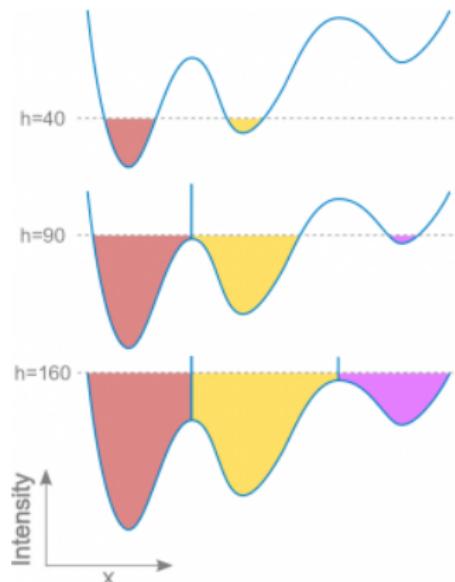


Figura: Flooding do algoritmo do Watershed em 1D. A ideia básica do algoritmo é considerar a imagem como um mapa topográfico e colocar uma fonte de água em cada mínimo local. Em seguida, barragens são adicionadas nas regiões em que as fontes se encontram⁷.



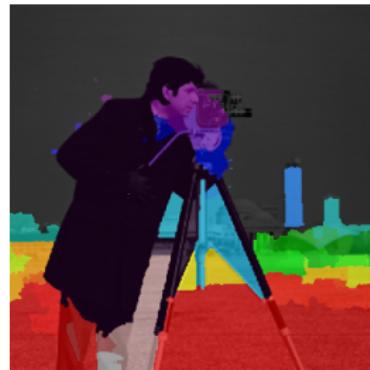
Watershed



(a)



(b)



(c)

Figura: Segmentação de imagem via Watershed⁷.

⁷ http://scikit-image.org/docs/dev/auto_examples/segmentation/plot_marked_watershed.html

Métodos Clássicos de Segmentação



- Problemas com os métodos clássicos
 - Processamentos locais, geralmente envolvendo features muito rasos
 - Generalidade
 - Falta de informação semântica

Métodos Deep de Segmentação



- Abordagens Deep para Rotulação Densa
 - Classificação de Pixels
 - Fully Convolutional Networks (FCNs) [11]
 - Redes de Deconvolução
 - U-Nets [12]
 - SegNets [13]

Classificação de Patches



- Transformar um problema de segmentação em um problema de classificação
- Cada pixel é passado para uma CNN comum com uma janela de contexto ao redor



(a)



(b)

Figura: Pixel central da imagem original e da máscara de labels circundados por janela de contexto.

Classificação de Patches

- Abordagem extremamente ineficiente
 - Imagem 256×256 possui 65536 pixels
 - Alta redundância nos cálculos das convoluções para janelas de contexto sobrepostas

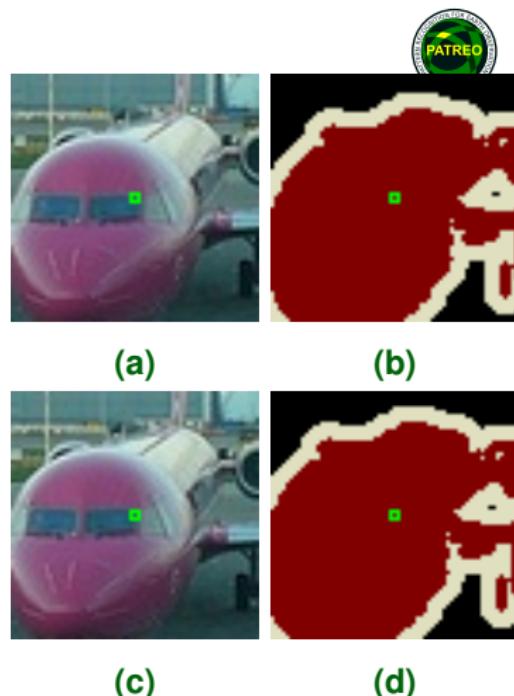


Figura: Pixel central da imagem original e da máscara de labels circundados por janela de contexto.

CNN → FCN



- Fully Convolutional Networks (FCNs)
- Camadas convolucionais reaproveitadas
- Uso de camadas de CNNs pré-treinadas

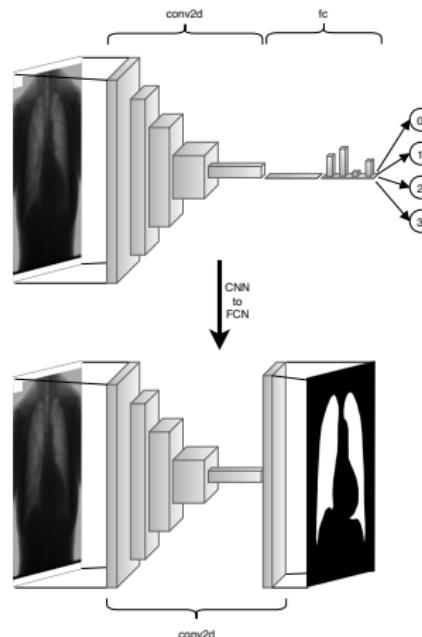


Figura: Transformando uma CNN [3] em uma FCN [11].

Skip Connections



- Propostos por [11]
- Fazem a ligação entre as primeiras camadas da rede e as últimas camadas
- Diversas vantagens
 - Fusão de informação de alto nível semântico com informação de baixo nível semântico
 - Mitigação do problema do Vanishing Gradient em redes maiores

Convoluçãoes Transpostas



- Implementação de Deconvolução
- Usada em arquiteturas Encoder-Decoder convolucionais



Convoluçãoes Normais

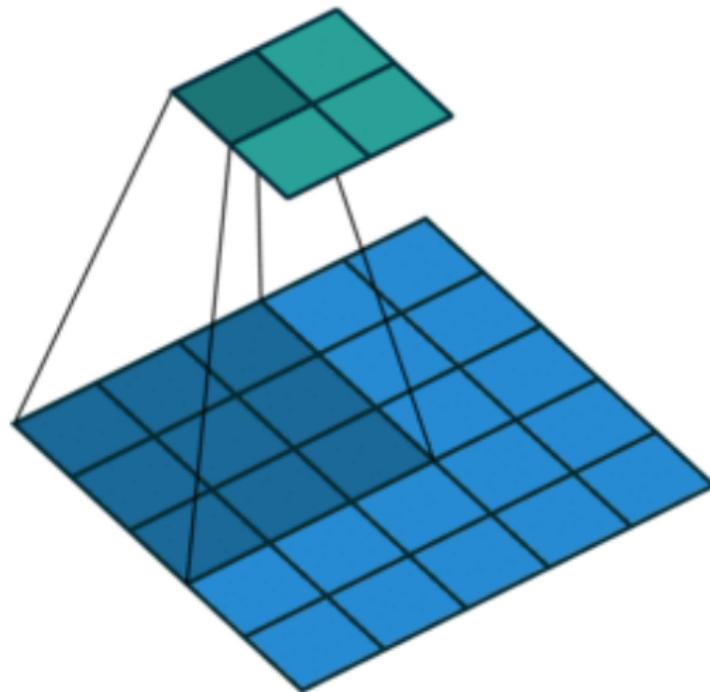


Figura: Convolução Normais⁷.



Convoluçãoes Normais

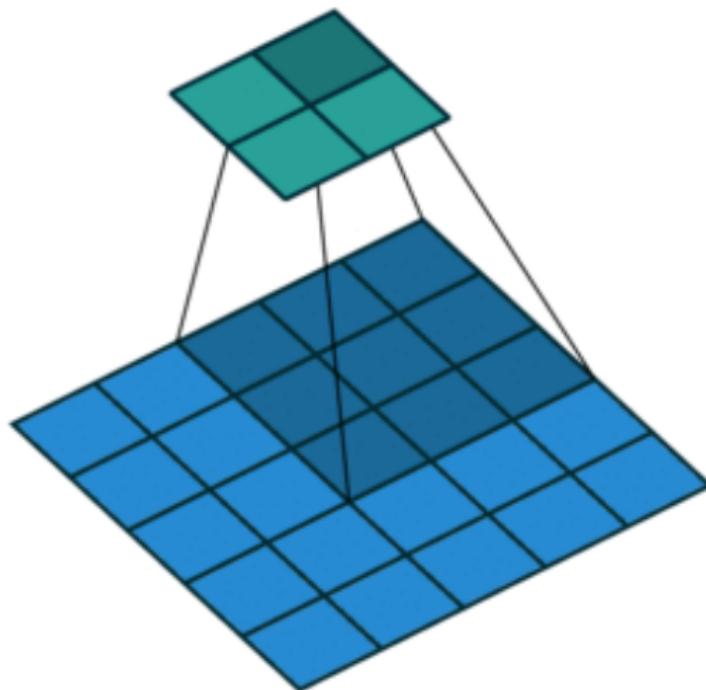


Figura: Convolução Normais⁷.



Convoluçãoções Normais

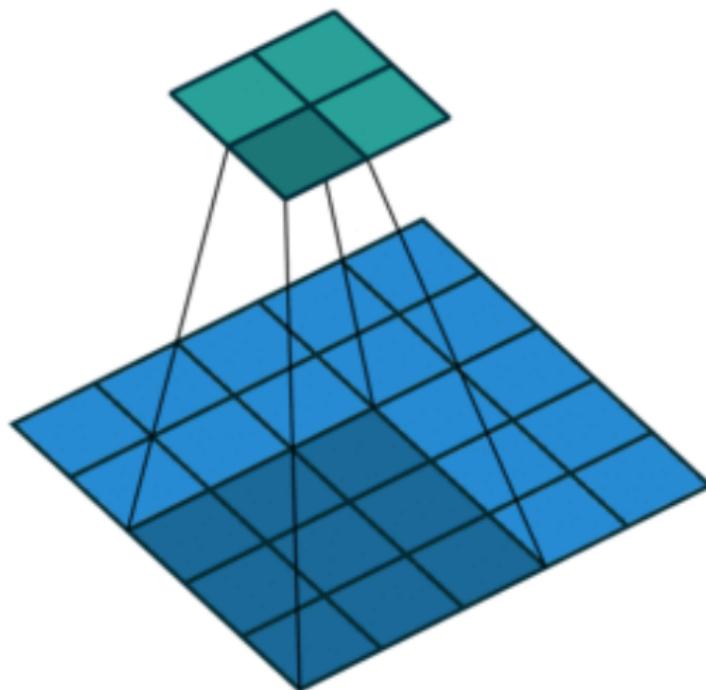


Figura: Convolução Normais⁷.



Convoluçãoções Normais

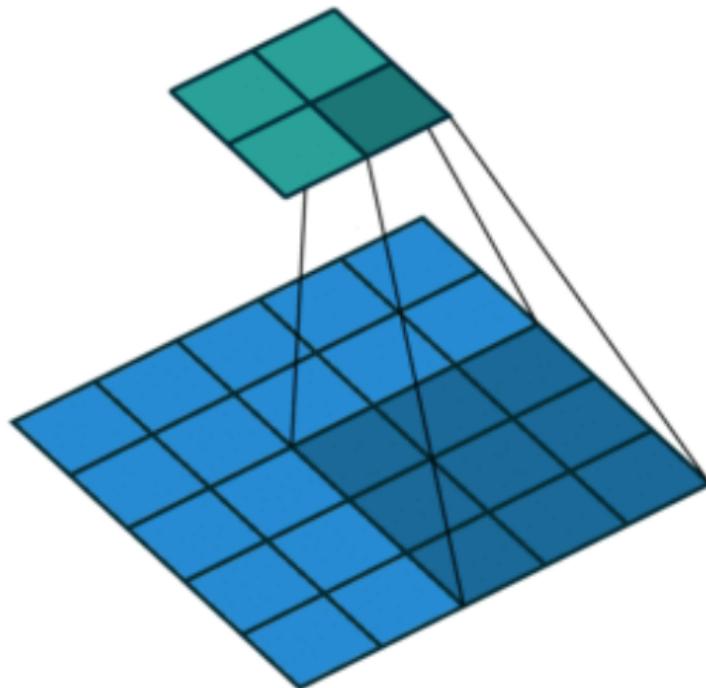


Figura: Convolução Normais⁷.



Convoluçãoes Transpostas

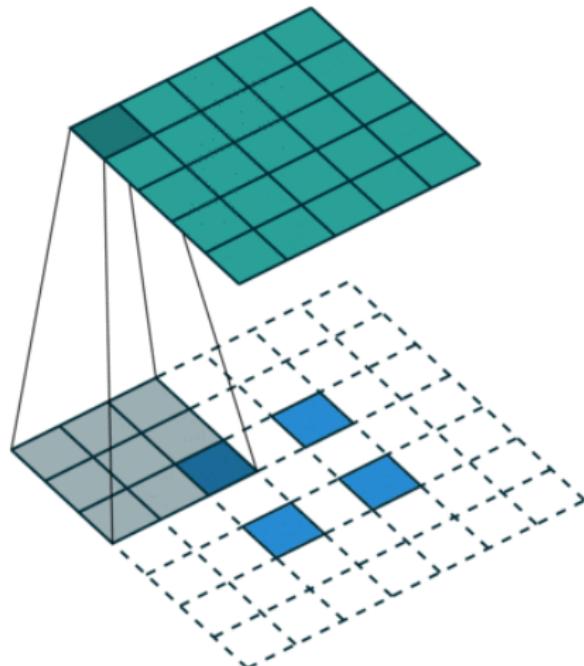


Figura: Convolução Transposta⁷.



Convoluçãoes Transpostas

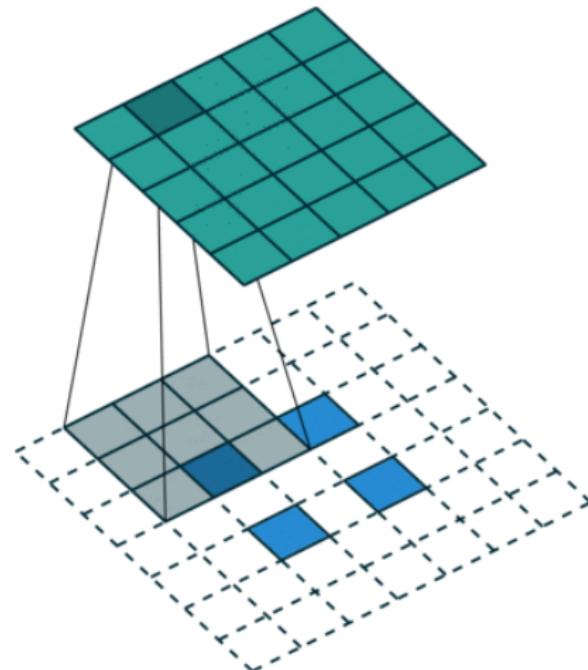


Figura: Convolução Transposta⁷.



Convoluçãoes Transpostas

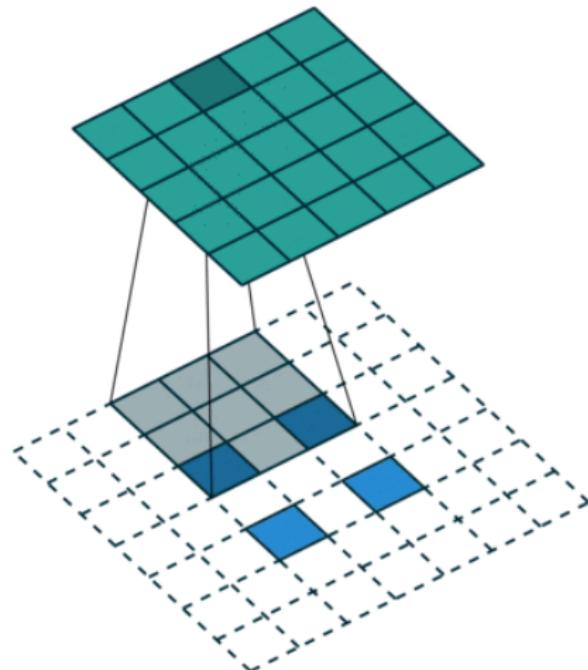


Figura: Convolução Transposta⁷.



Convoluçãoes Transpostas

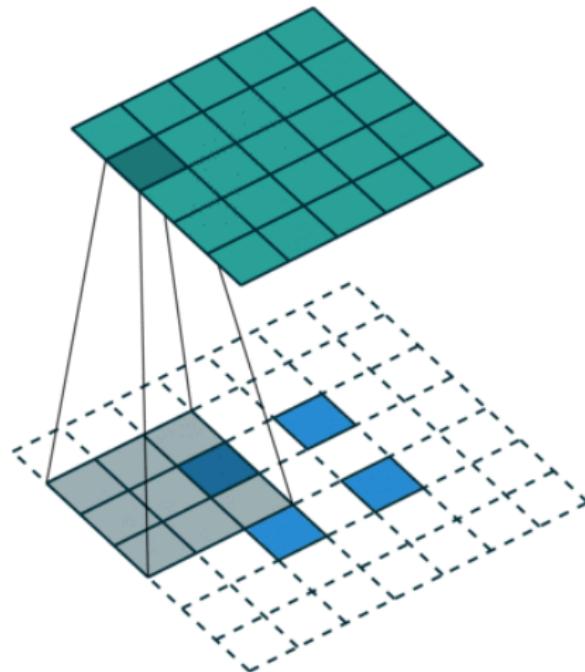


Figura: Convolução Transposta⁷.



Convoluçãoes Transpostas

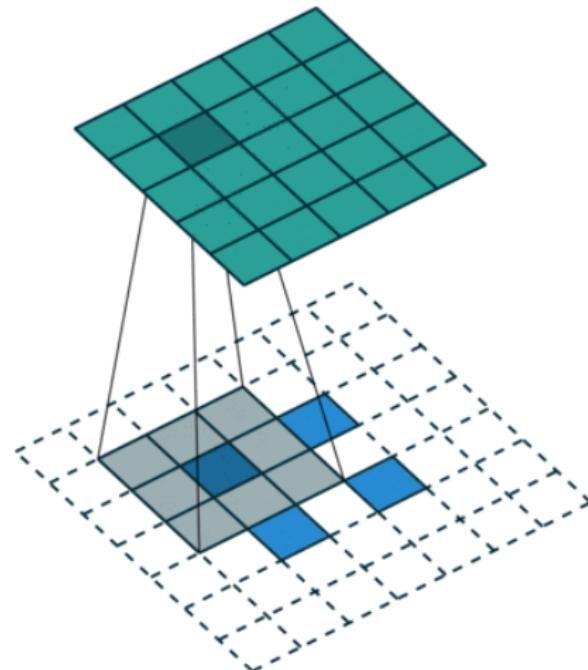


Figura: Convolução Transposta⁷.



Convoluçãoes Transpostas

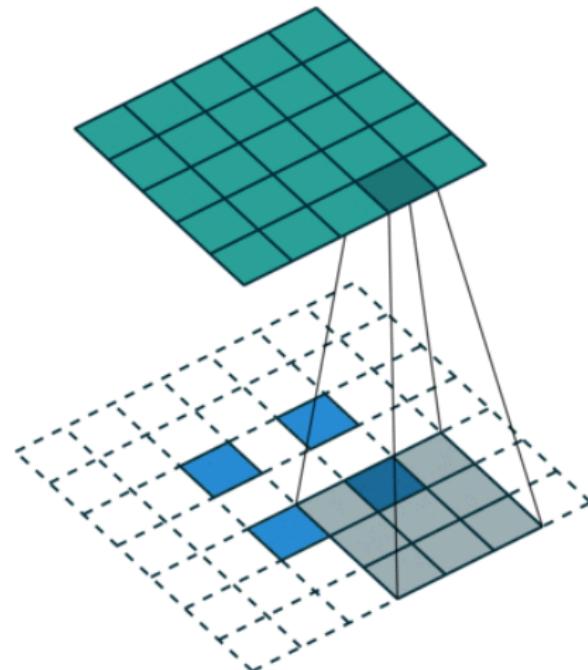


Figura: Convolução Transposta⁷.



Convoluçãoes Transpostas

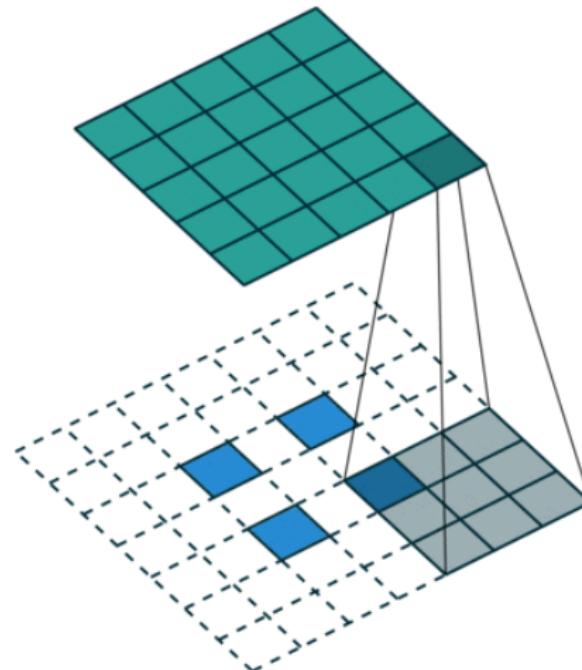


Figura: Convolução Transposta⁷.



Redes de Deconvolução

- U-nets [12]

- Arquiteturas Encoder-Decoder
- Mais camadas dedicadas à recuperação da informação espacial
- Mais canais nas camadas do Decoder que nas do Encoder

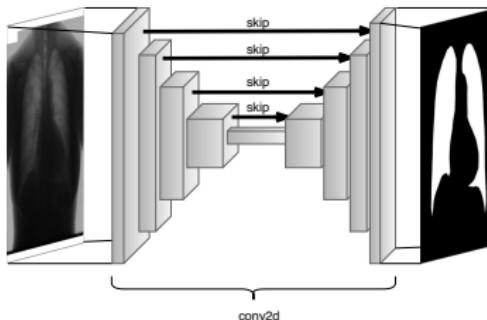


Figura: Arquitetura de U-net. Cada conv2d corresponde a convoluções seguidas de downsampling e upsampling. Setas denotam as skip connections entre camadas simétricas. Adaptado de [12].

Redes de Deconvolução



- SegNets [13]

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

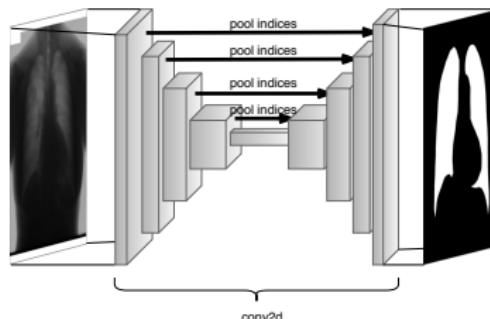


Figura: Ilustração de uma SegNet. Setas correspondem à passagem dos índices de pooling para camadas posteriores. Adaptado de [13].



Detalhes de Implementação

- A maioria das funções de perda (i.e. NLL Loss, Binary Cross Entropy Loss...) para classificação aceitam máscaras de segmentação como rótulos
 - É possível transformar uma arquitetura de classificação em uma de segmentação com pouco esforço

Agenda



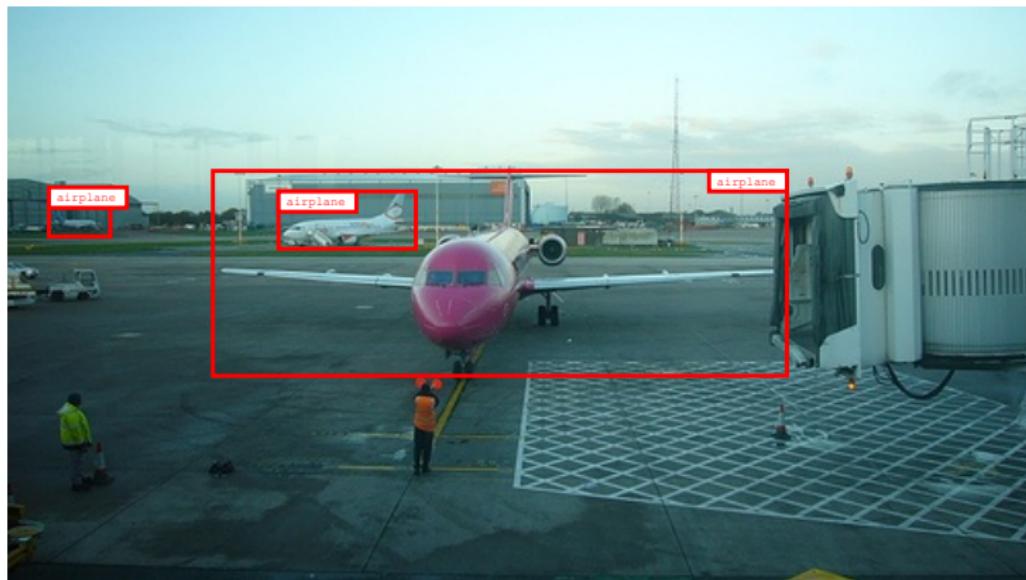
- 1 Introdução
- 2 Redes Neurais Convolucionais
- 3 Arquitetura Famosas
- 4 Estratégias de Treino
- 5 Segmentação de Imagens
- 6 Detecção em Imagens

Detecção em imagens

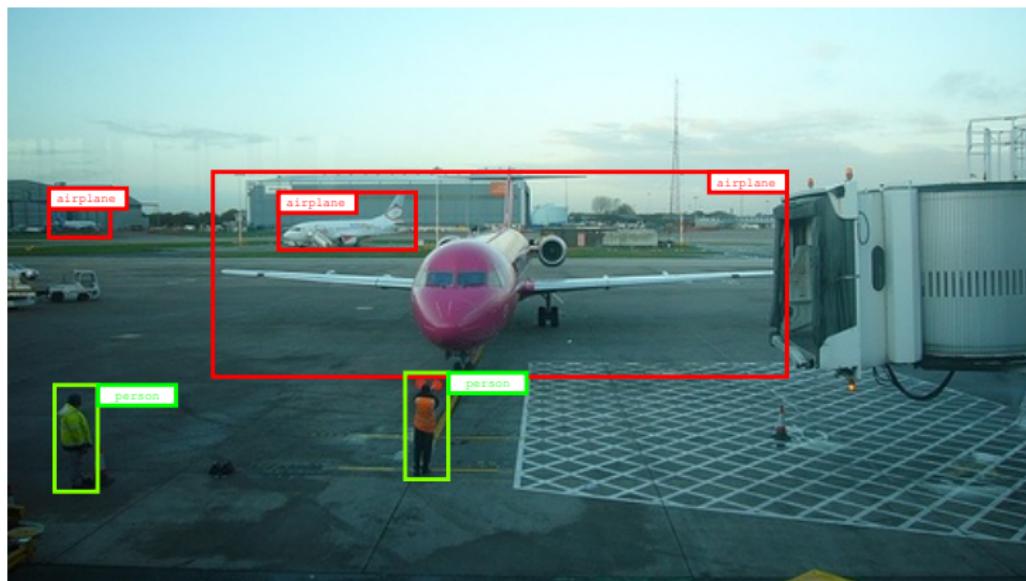


Figura: Exemplo de Detecção em imagens.

Detecção em imagens

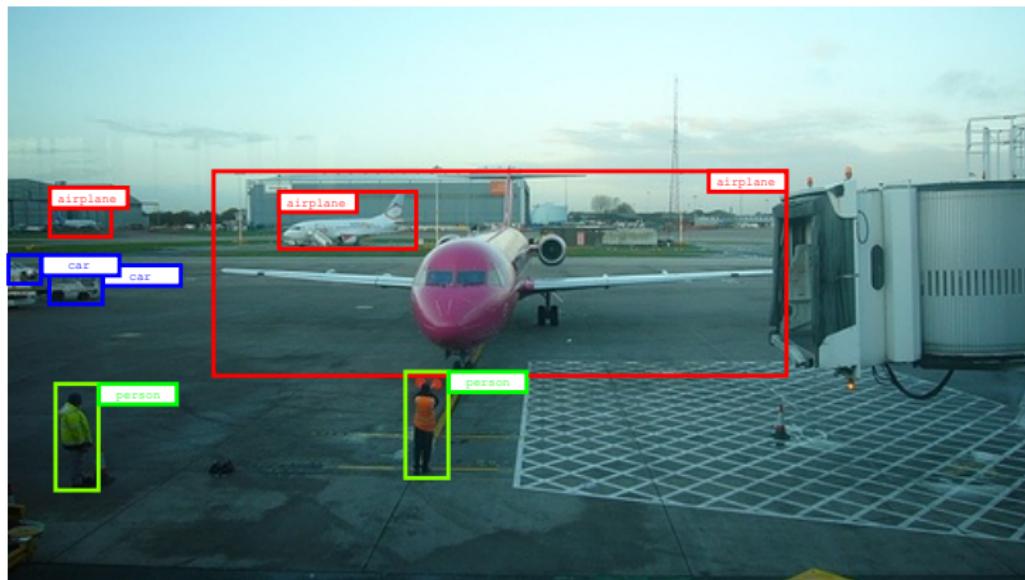


Detecção em imagens





Detecção em imagens



Detecção em Imagens



- Problema mais esparso que segmentação
- Rotulação menos exata que em segmentação
- Esquemas de detecção muito frequentemente precisam funcionar em **tempo real**



Detecção Shallow em Imagens

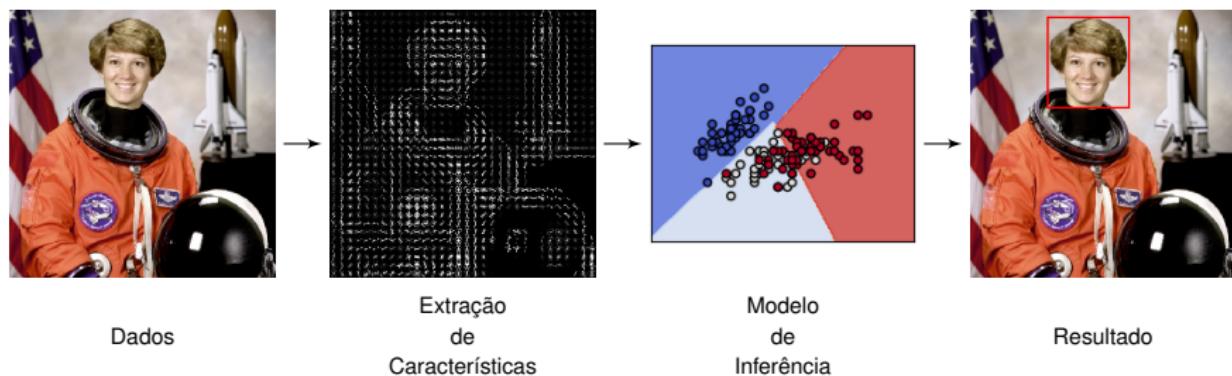


Figura: Pipeline padrão de um algoritmo de Machine Learning clássico.

Detecção Deep em Imagens



Detecção Deep

A maior parte dos algoritmos se baseia em CNNs que fazem regressão sobre imagens para determinar as coordenadas dos **Bounding Boxes**.



Detecção Deep

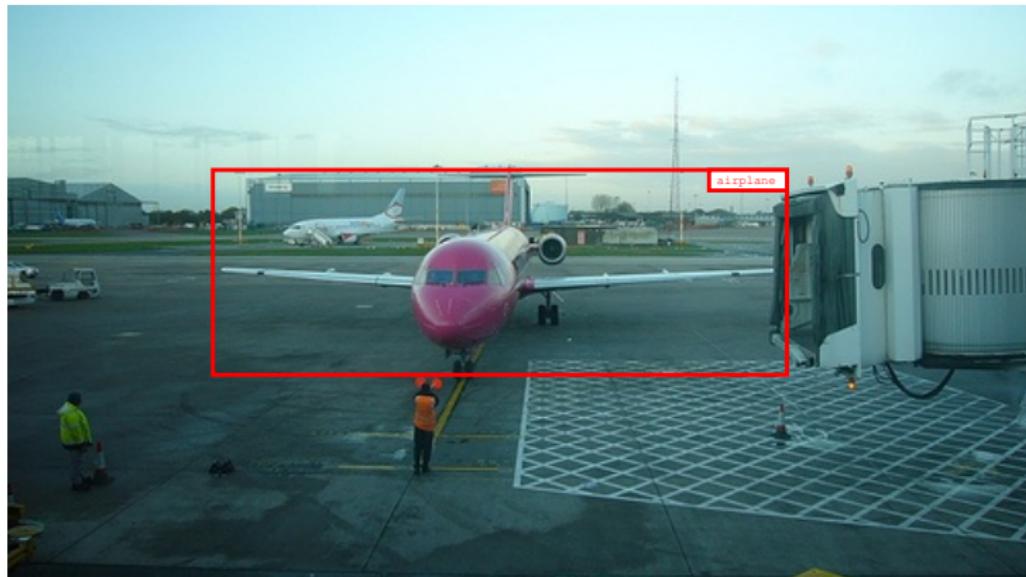


Figura: Parâmetros dos Bounding Boxes.



Detecção Deep

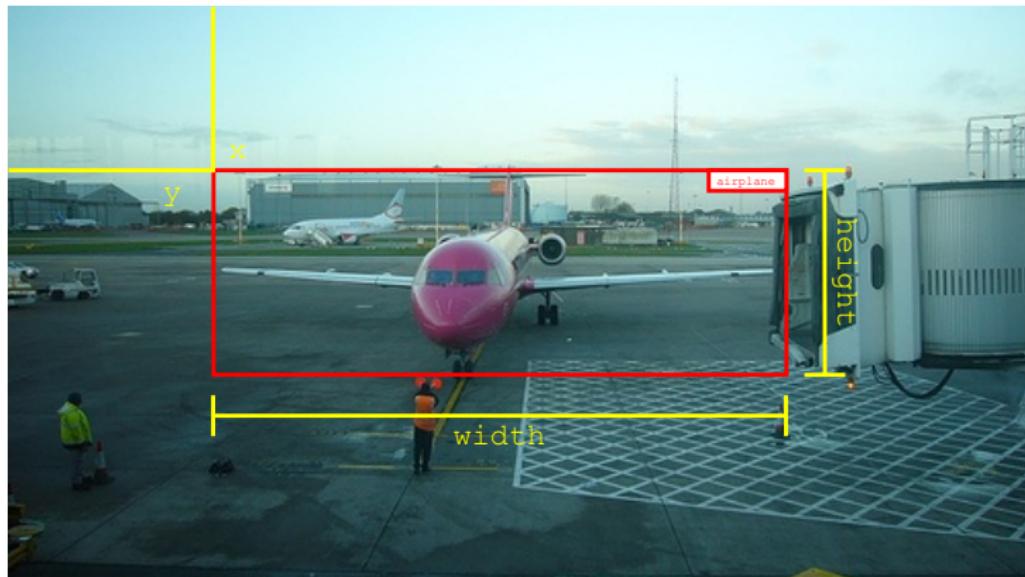


Figura: Parâmetros dos Bounding Boxes.



Detecção Deep

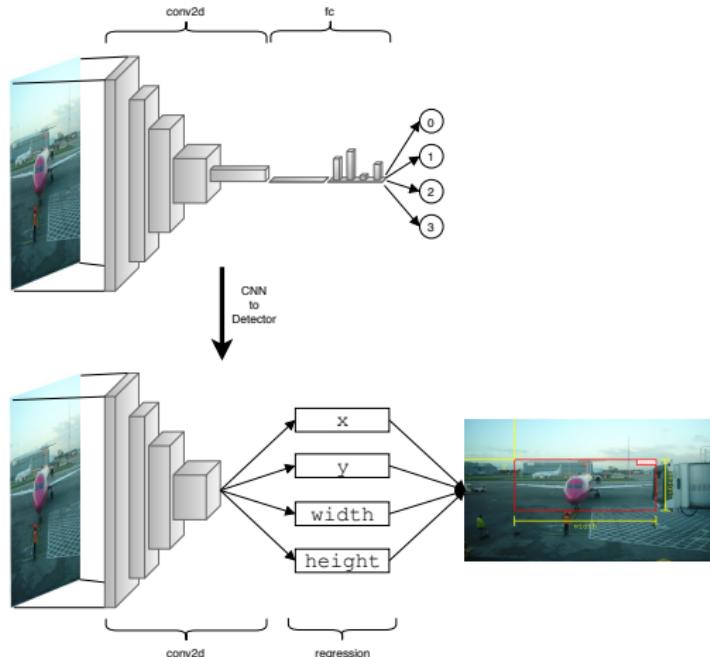


Figura: Pipeline básico de uma estratégia Deep para detecção em imagens.

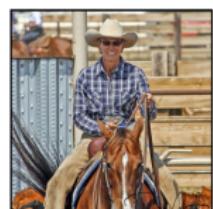
Detecção Deep em Imagens



- Regional CNNs (R-CNNs) [14]
- YOLO [15]



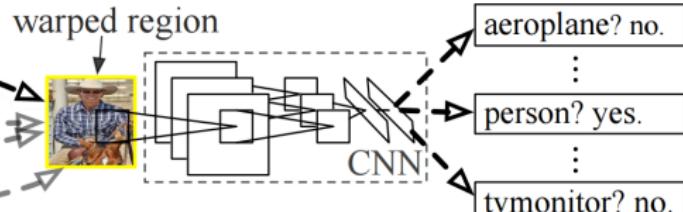
R-CNNs



1. Input image



2. Extract region proposals (~2k)



3. Compute CNN features

4. Classify regions

Figura: Pipeline de uma R-CNN. Fonte: [14].

R-CNNs



R-CNNs

rcnnns_2.png [14]

R-CNNs



- Treinamento do modelo de inferência é feito separadamente da extração de features
- Extremamente lento
- Pipeline de treinamento demasiadamente complexo
- Proposição de regiões ainda feito usando métodos shallow



Fast R-CNNs

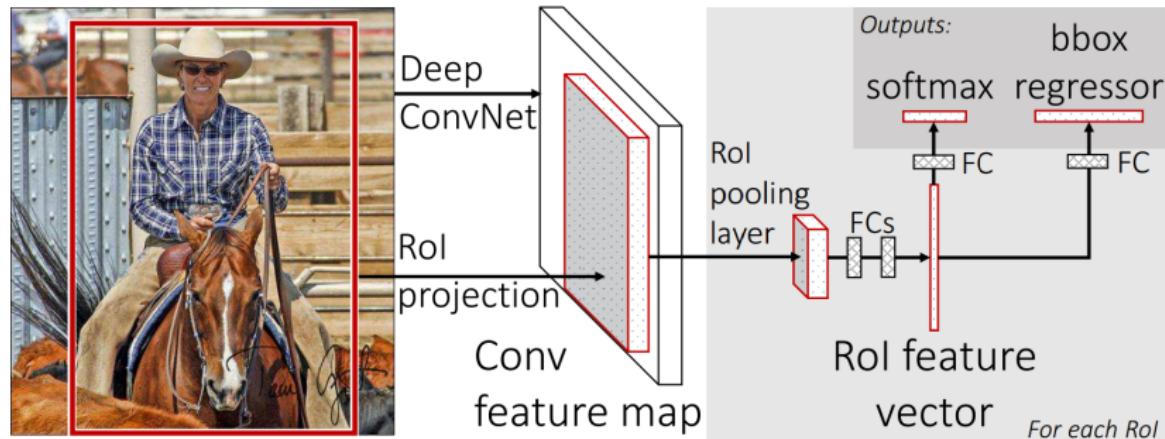


Figura: Pipeline de uma Fast R-CNN. Fonte: [16].

Fast R-CNNs



Fast R-CNNs

fast_rcnn_2.png [17]

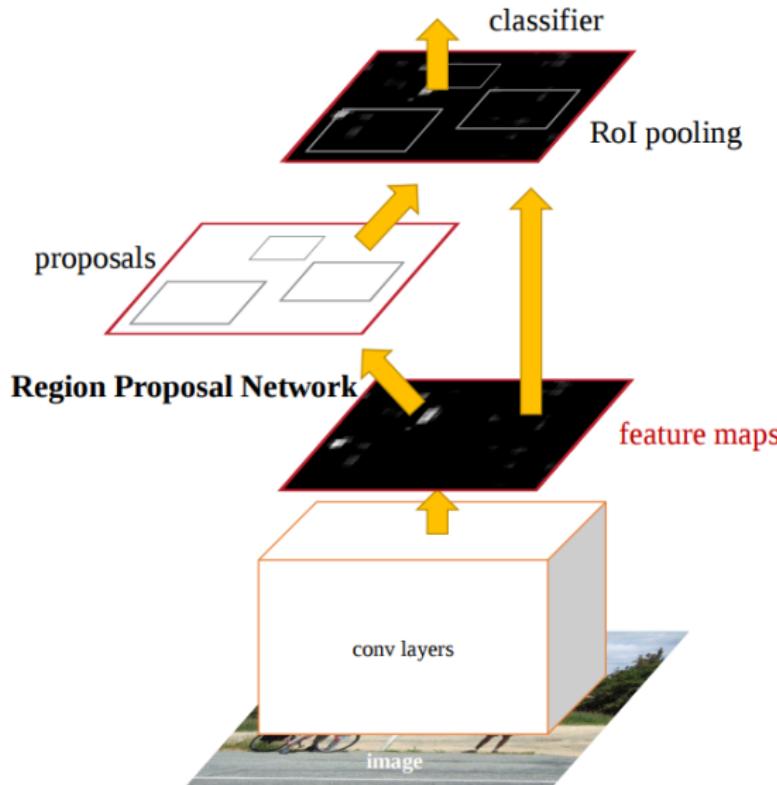
Fast R-CNNs



- Algoritmo de Proposição de Regiões muito lento
- Não é possível de ser executado em tempo real
- Proposição de regiões ainda é shallow, apesar de usar features de uma Rede Profunda



Faster R-CNNs



Faster R-CNNs



- Region Proposal Networks (RPNs)
- Já consegue rodar em frações de segundo para cada imagem (quase em tempo real)

References

- [1] Jefkine.
Backpropagation in convolutional neural networks, 2016.
- [2] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner.
Gradient-based learning applied to document recognition.
Proceedings of the IEEE, 86(11):2278–2324, 1998.
- [3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton.
Imagenet classification with deep convolutional neural networks.
In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *NIPS*, pages 1097–1105. Curran Associates, Inc., 2012.
- [4] Karen Simonyan and Andrew Zisserman.
Very deep convolutional networks for large-scale image recognition.
arXiv preprint arXiv:1409.1556, 2014.
- [5] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich.
Going deeper with convolutions.
In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.
- [6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun.
Deep residual learning for image recognition.
In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [7] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten.
Densely connected convolutional networks.
In *Proceedings of the IEEE conference on computer vision and pattern recognition*, volume 1, page 3, 2017.
- [8] Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber.
Training very deep networks.
In *Advances in neural information processing systems*, pages 2377–2385, 2015.
- [9] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger.
Deep networks with stochastic depth.
In *European Conference on Computer Vision*, pages 646–661. Springer, 2016.
- [10] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich.



References

- Fractalnet: Ultra-deep neural networks without residuals.
[arXiv preprint arXiv:1605.07648](https://arxiv.org/abs/1605.07648), 2016.
- [11] J. Long, E. Shelhamer, and T. Darrell.
Fully convolutional networks for semantic segmentation.
pages 3431–3440, June 2015.
- [12] Olaf Ronneberger, Philipp Fischer, and Thomas Brox.
U-net: Convolutional networks for biomedical image segmentation.
In [MICCAI](#), pages 234–241. Springer, 2015.
- [13] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla.
Segnet: A deep convolutional encoder-decoder architecture for image segmentation.
39(12):2481–2495, 2017.
- [14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik.
Rich feature hierarchies for accurate object detection and semantic segmentation.
In [Proceedings of the IEEE conference on computer vision and pattern recognition](#), pages 580–587, 2014.
- [15] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi.
You only look once: Unified, real-time object detection.
In [Proceedings of the IEEE conference on computer vision and pattern recognition](#), pages 779–788, 2016.
- [16] Ross Girshick.
Fast r-cnn.
In [Proceedings of the IEEE international conference on computer vision](#), pages 1440–1448, 2015.
- [17] Ross Girshick, Forrest Iandola, Trevor Darrell, and Jitendra Malik.
Deformable part models are convolutional neural networks.
In [Proceedings of the IEEE conference on Computer Vision and Pattern Recognition](#), pages 437–446, 2015.
- [18] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun.
Faster r-cnn: towards real-time object detection with region proposal networks.
[IEEE Transactions on Pattern Analysis & Machine Intelligence](#), (6):1137–1149, 2017.

