Hand-on Deep Learning - Aula 3

Padrões em Imagens: Classificação, Segmentação e Detecção

Camila Laranjeira[1], Hugo Oliveira[12], Keiller Nogueira[12]

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

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

03 de Agosto, 2018

DCC
DEPARTAMENTO DE
CIÊNCIA DA COMPUTAÇÃO

UFMG

Agenda

Agenda

MultiLayer Perceptrons

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

MLP

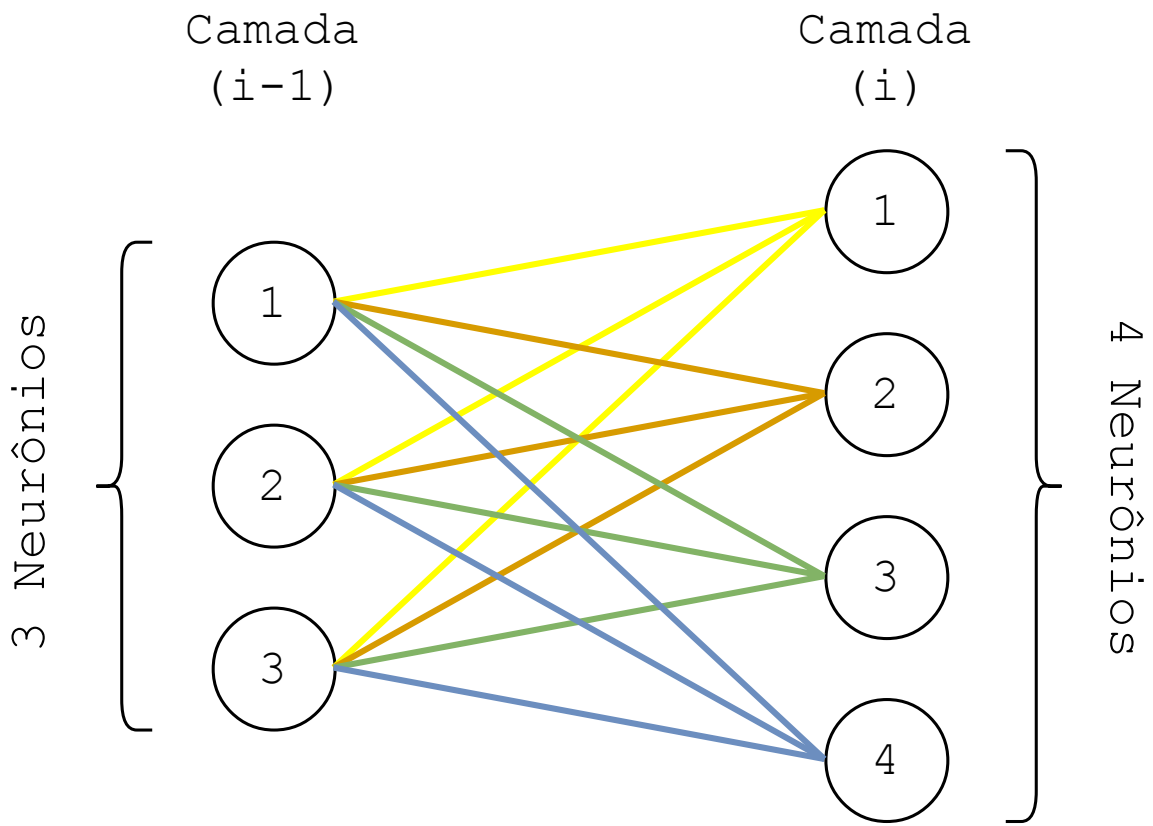


Figura: MLP.

MultiLayer Perceptrons

Número de Pesos da Camada i

$$\#Pesos = \#Neuronios_{(i-1)} \times \#Neuronios_{(i)}$$

MLP para Imagens

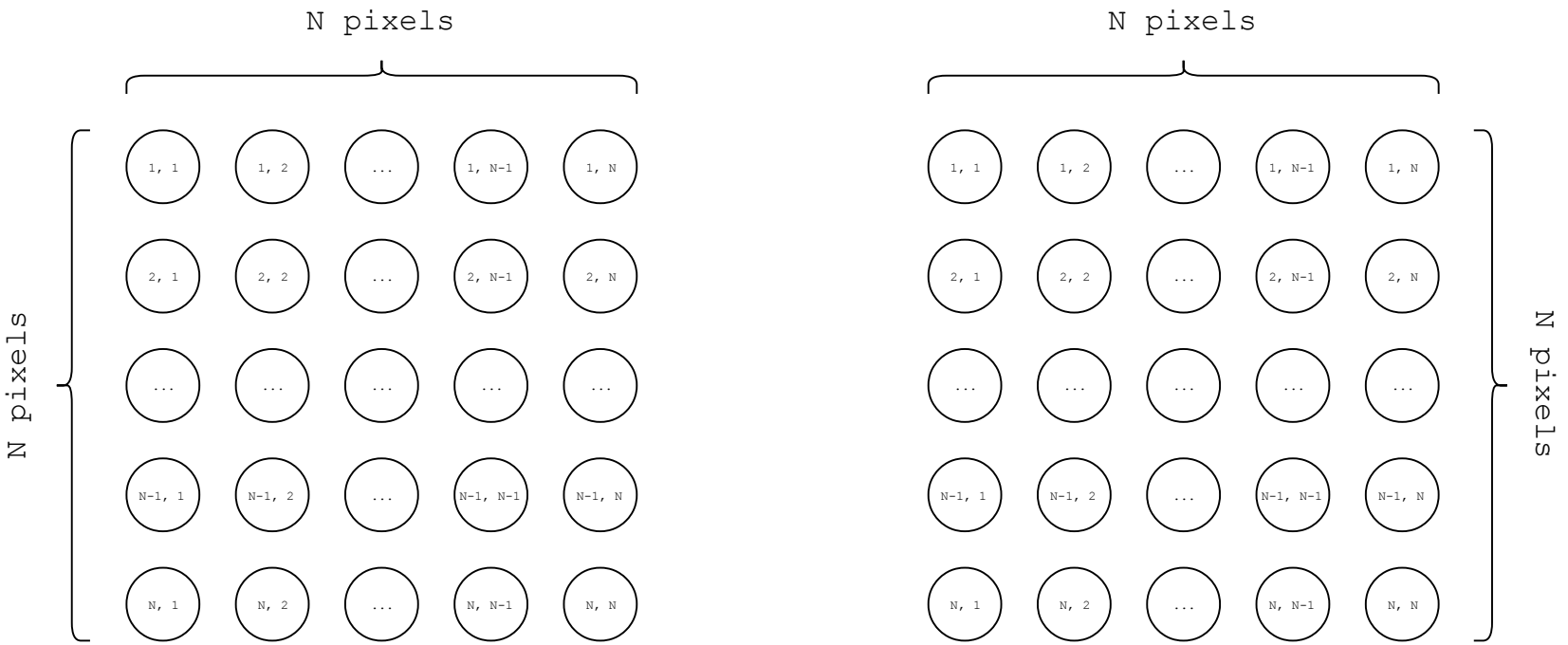


Figura: MLP em pixels de imagens.

MLP para Imagens

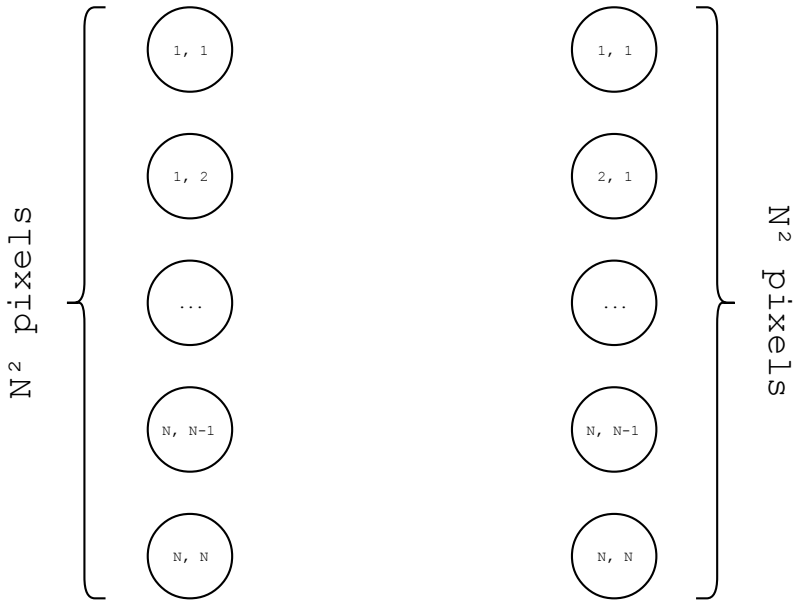


Figura: MLP em pixels de imagens.

MLP para Imagens

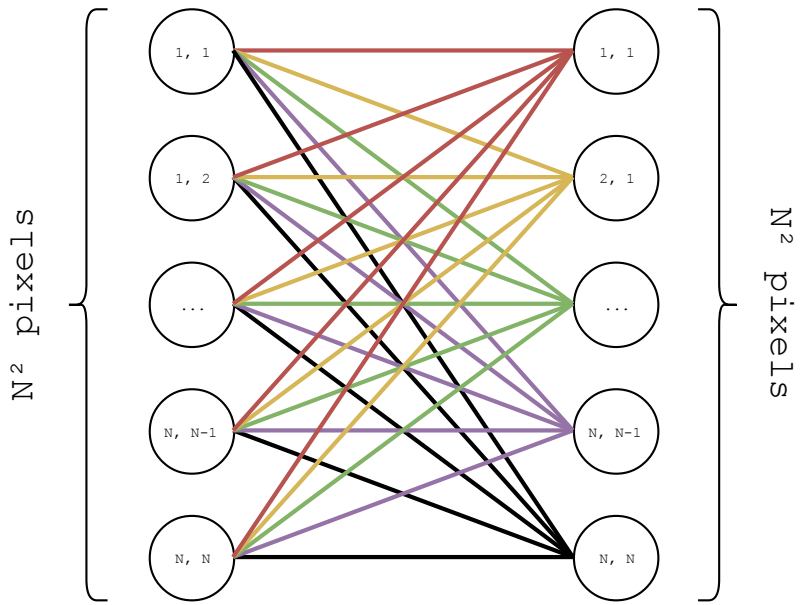


Figura: MLP em pixels de imagens.

MLP para Imagens

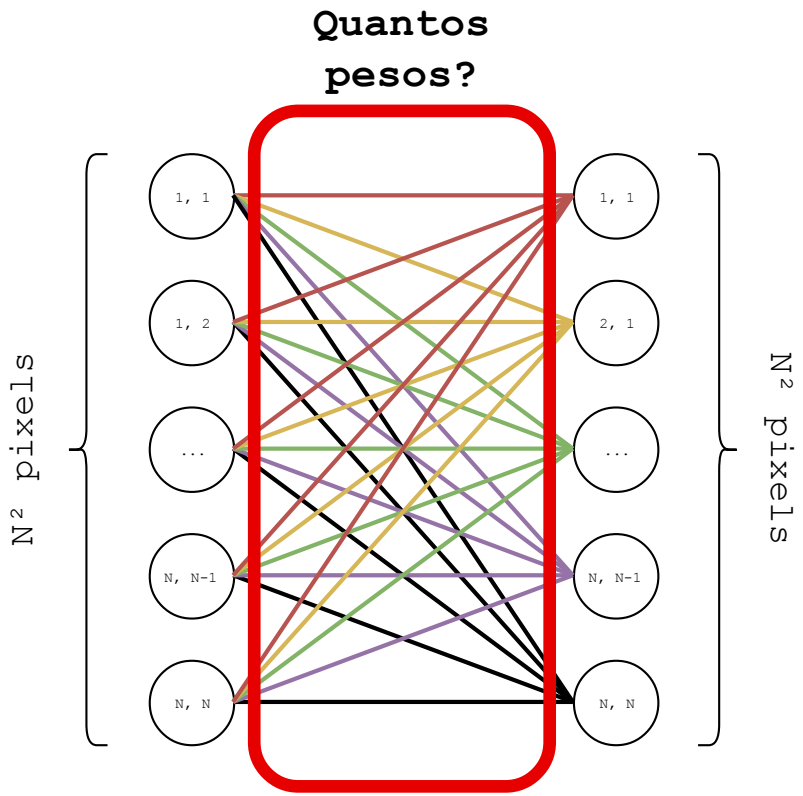


Figura: MLP em pixels de imagens.

MLP para Imagens

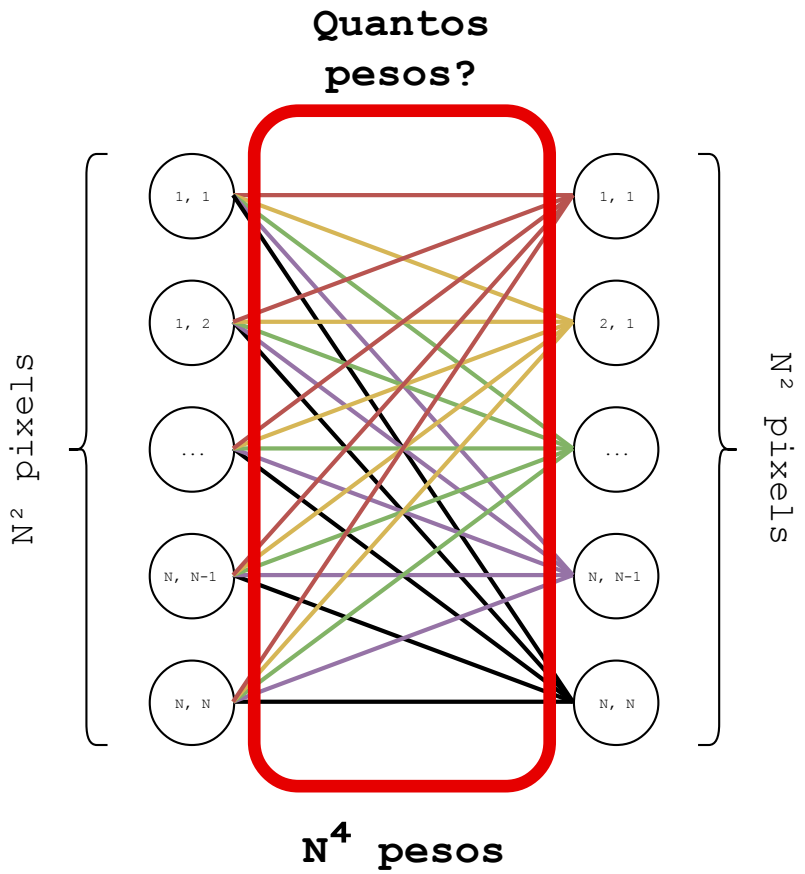


Figura: MLP em pixels de imagens.

MLP para Imagens

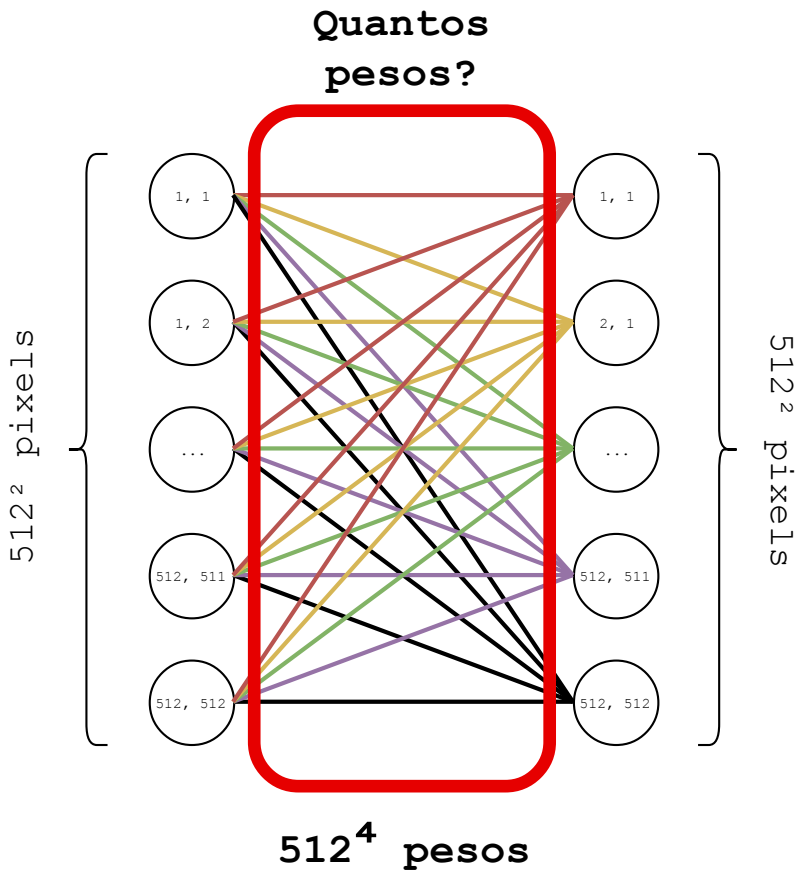


Figura: MLP em pixels de imagens.

MLP para Imagens

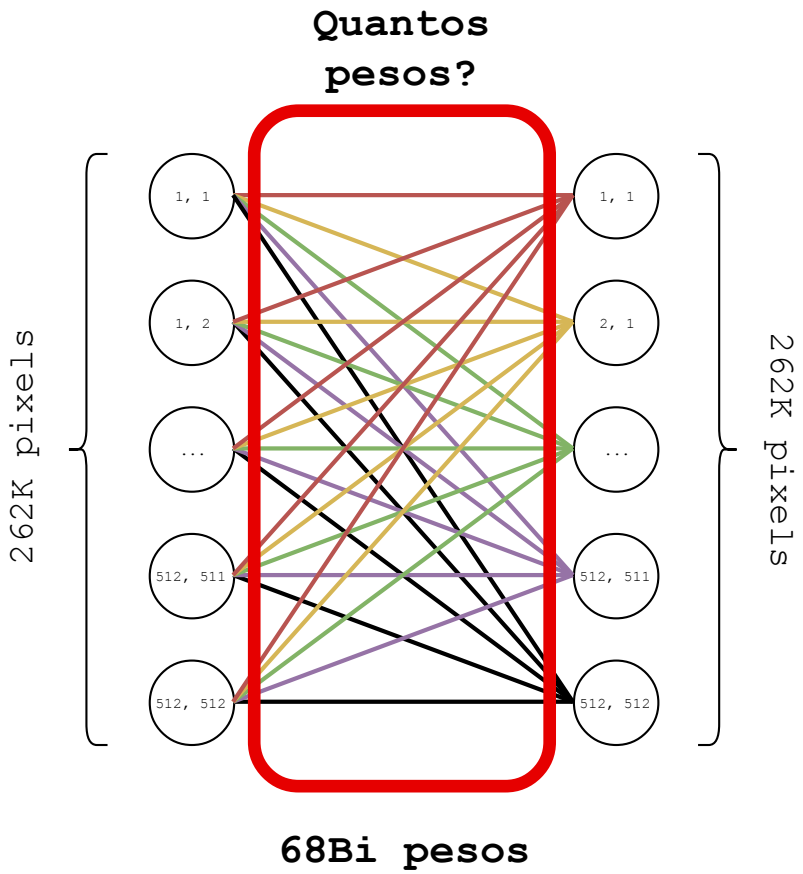


Figura: MLP em pixels de imagens.

MLP para Imagens

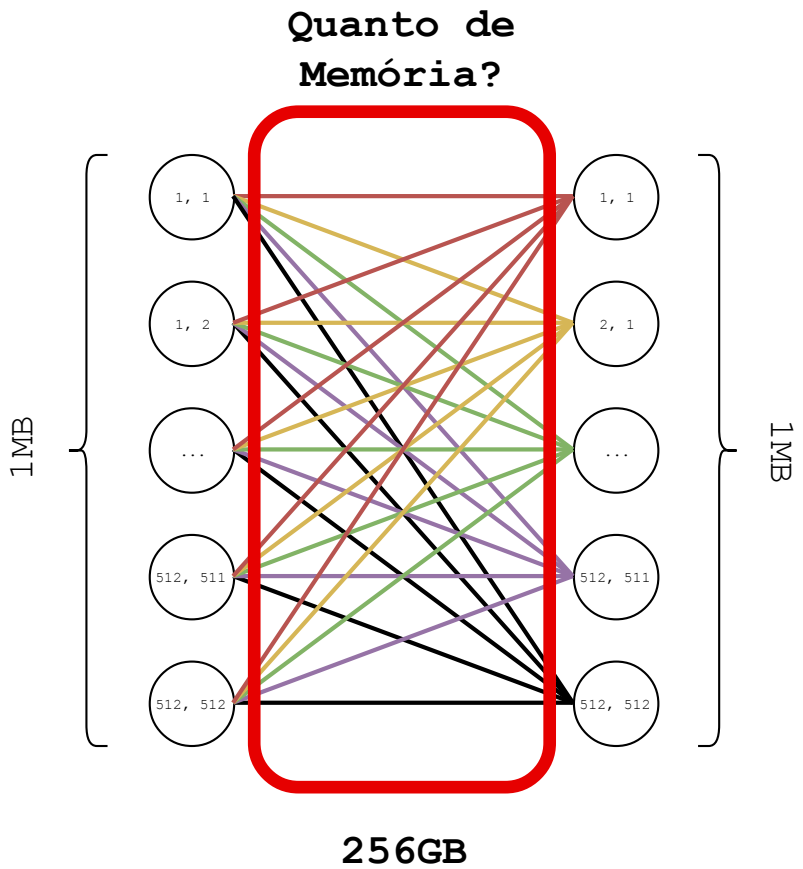


Figura: MLP em pixels de imagens.

Agenda

Convoluções 2D

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

Convolução 1D

Image

Figura: Imagem.

Kernel

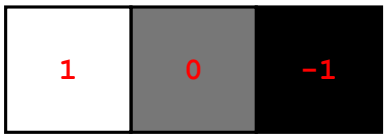


Figura: Kernel.

Convolução 1D

Figura: Imagem.

Figura: Kernel rebatido.

Convolução 1D

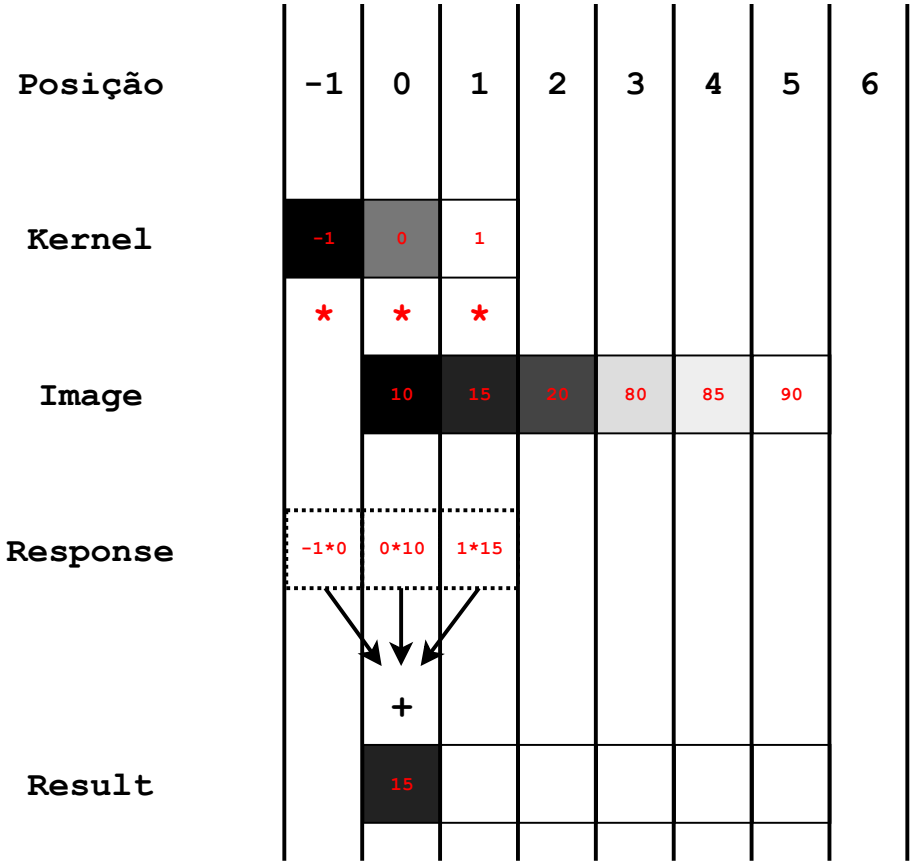


Figura: Cálculo de uma convolução 1D.

Convolução 1D

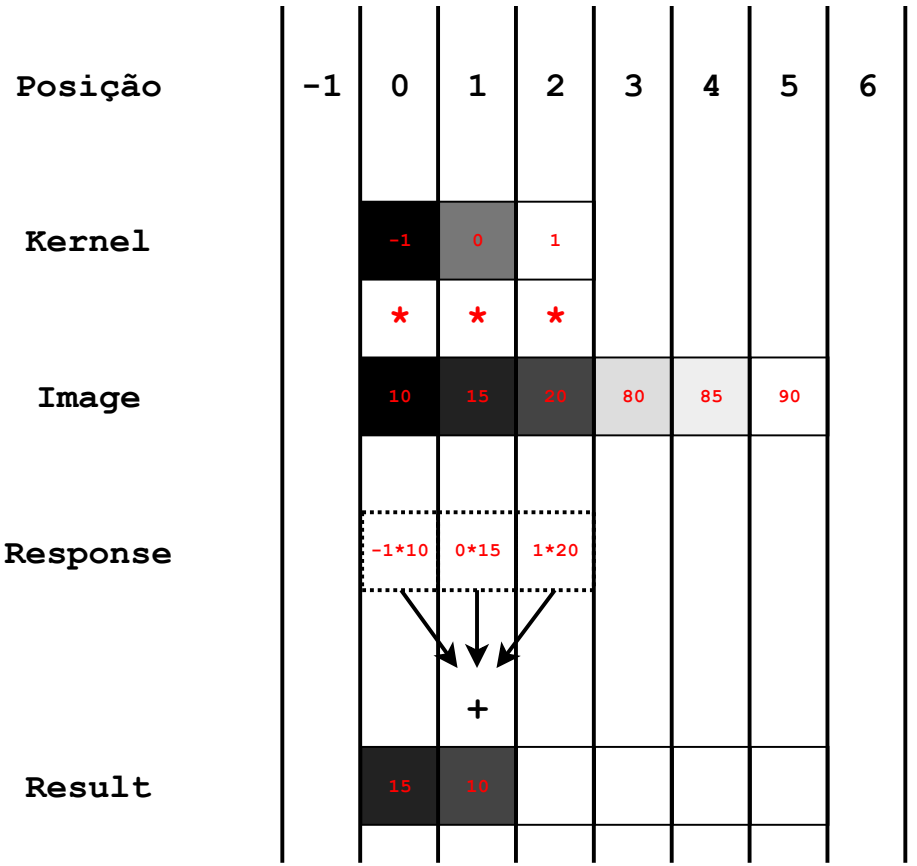


Figura: Cálculo de uma convolução 1D.

Convolução 1D

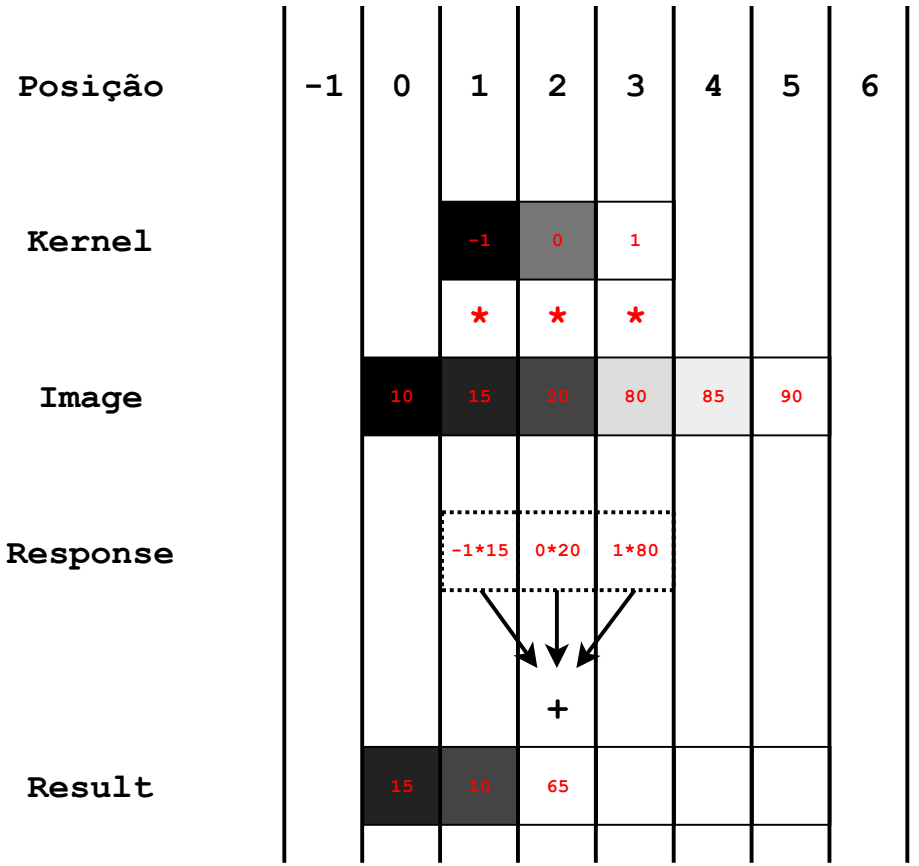


Figura: Cálculo de uma convolução 1D.

Convolução 1D

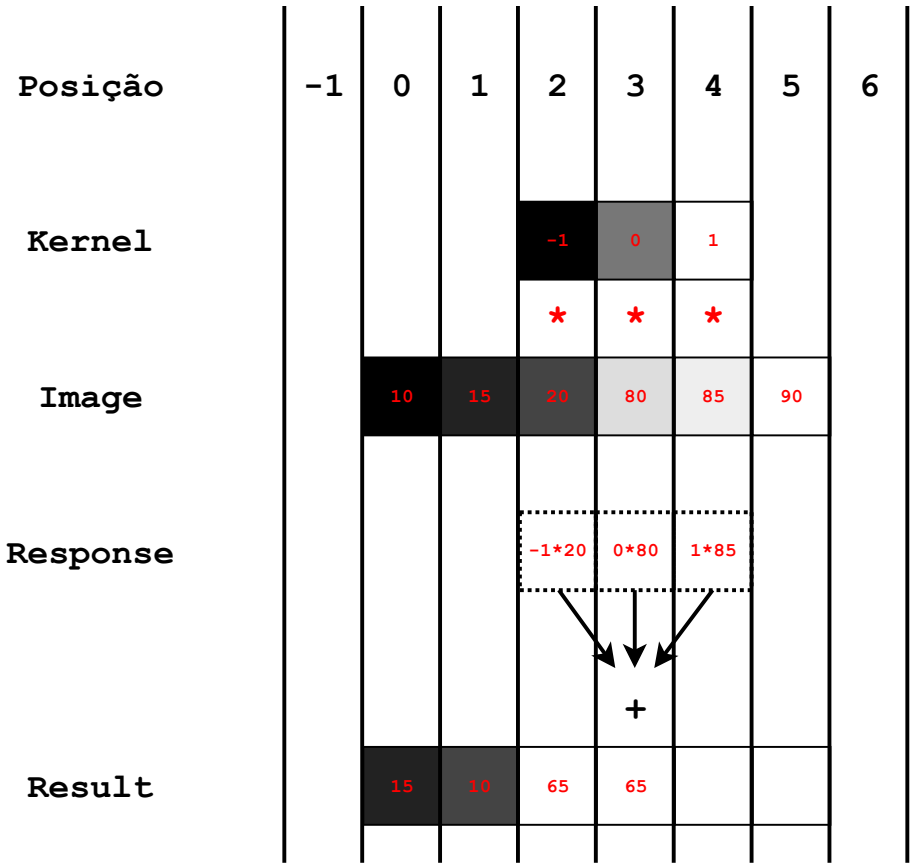


Figura: Cálculo de uma convolução 1D.

Convolução 1D

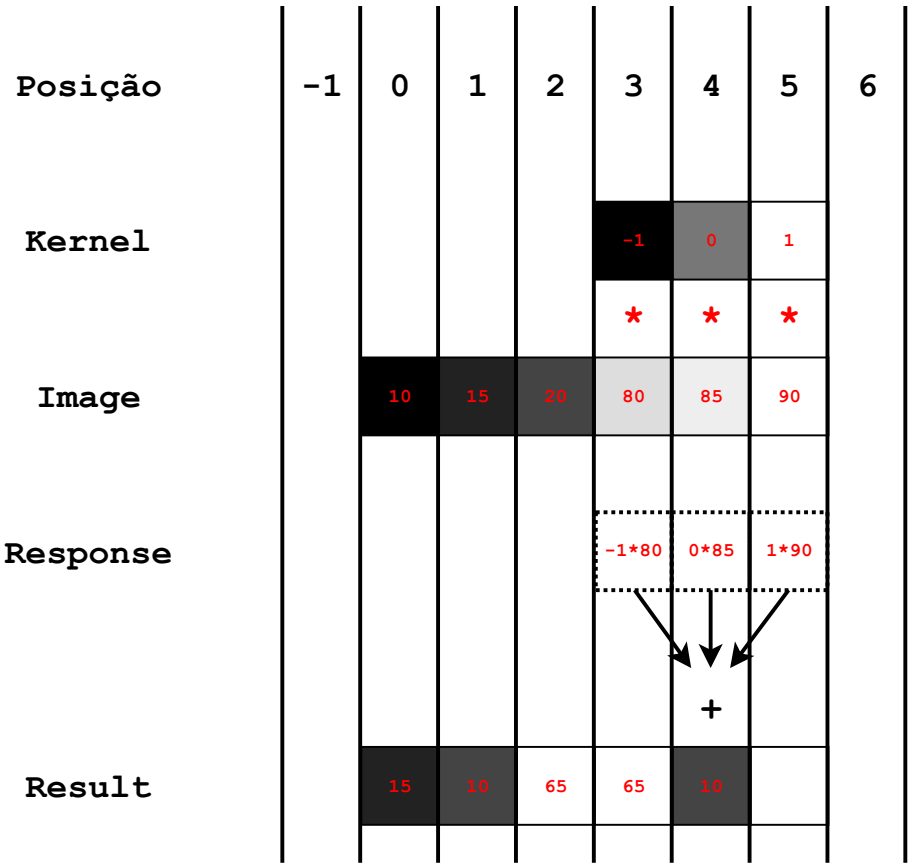


Figura: Cálculo de uma convolução 1D.

Convolução 1D

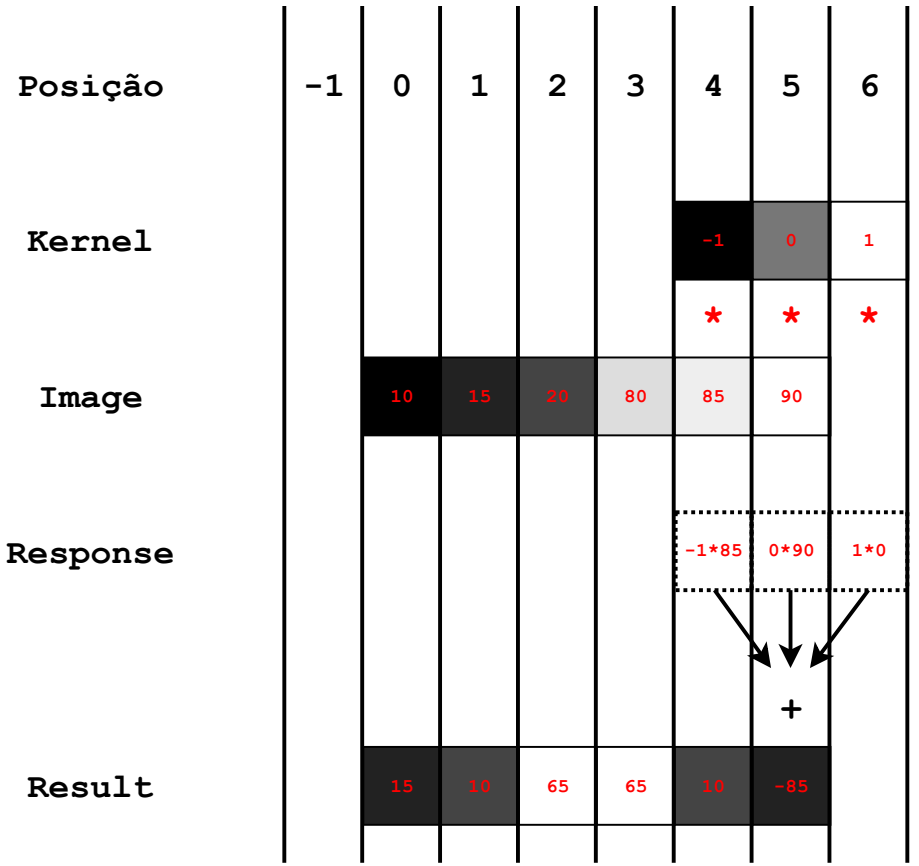


Figura: Cálculo de uma convolução 1D.

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

Figura: Imagem.

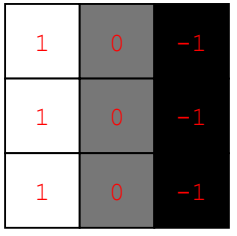


Figura: Kernel rebatido.

Convolução 2D

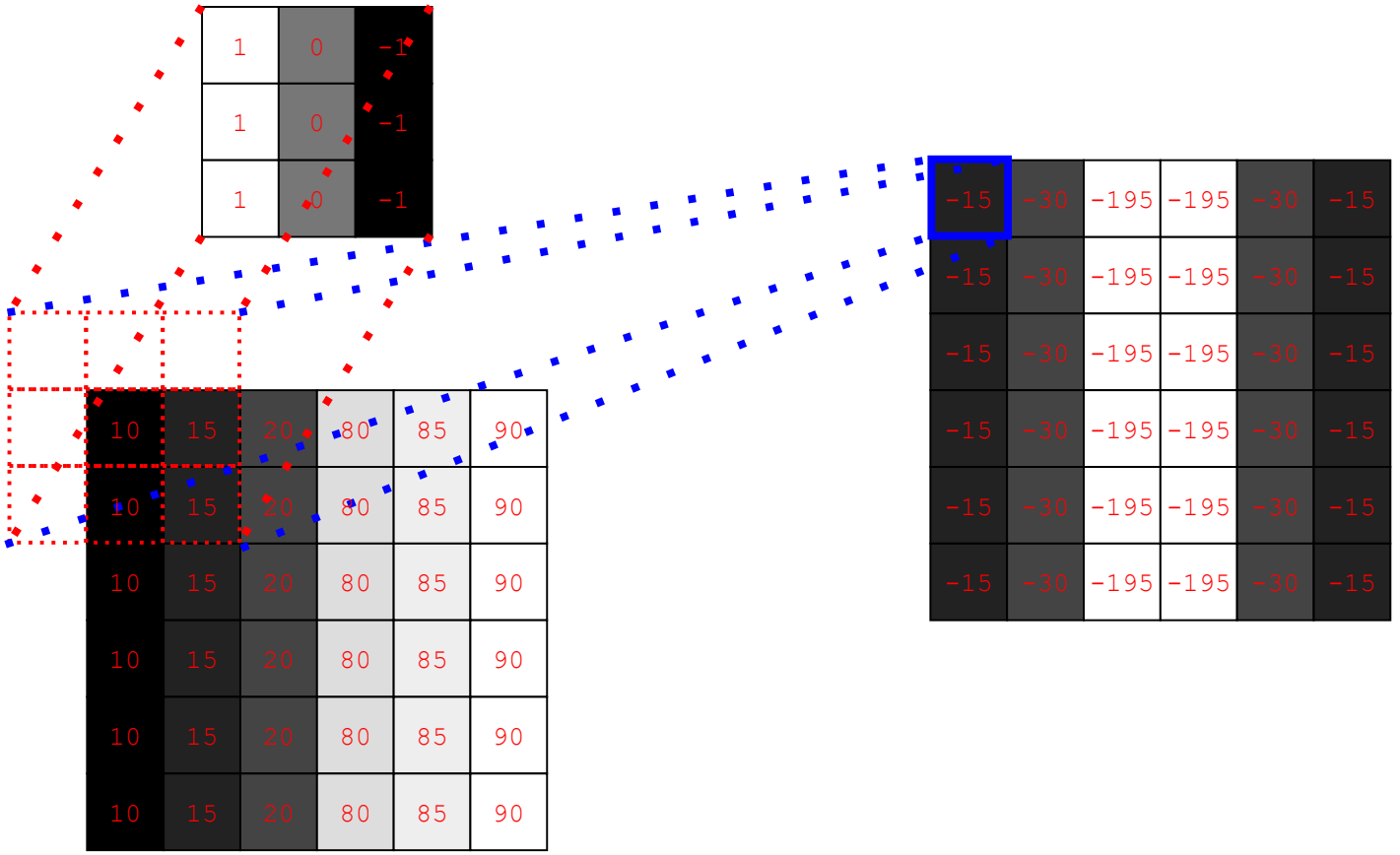


Figura: Cálculo de uma convolução 2D.

Convolução 2D

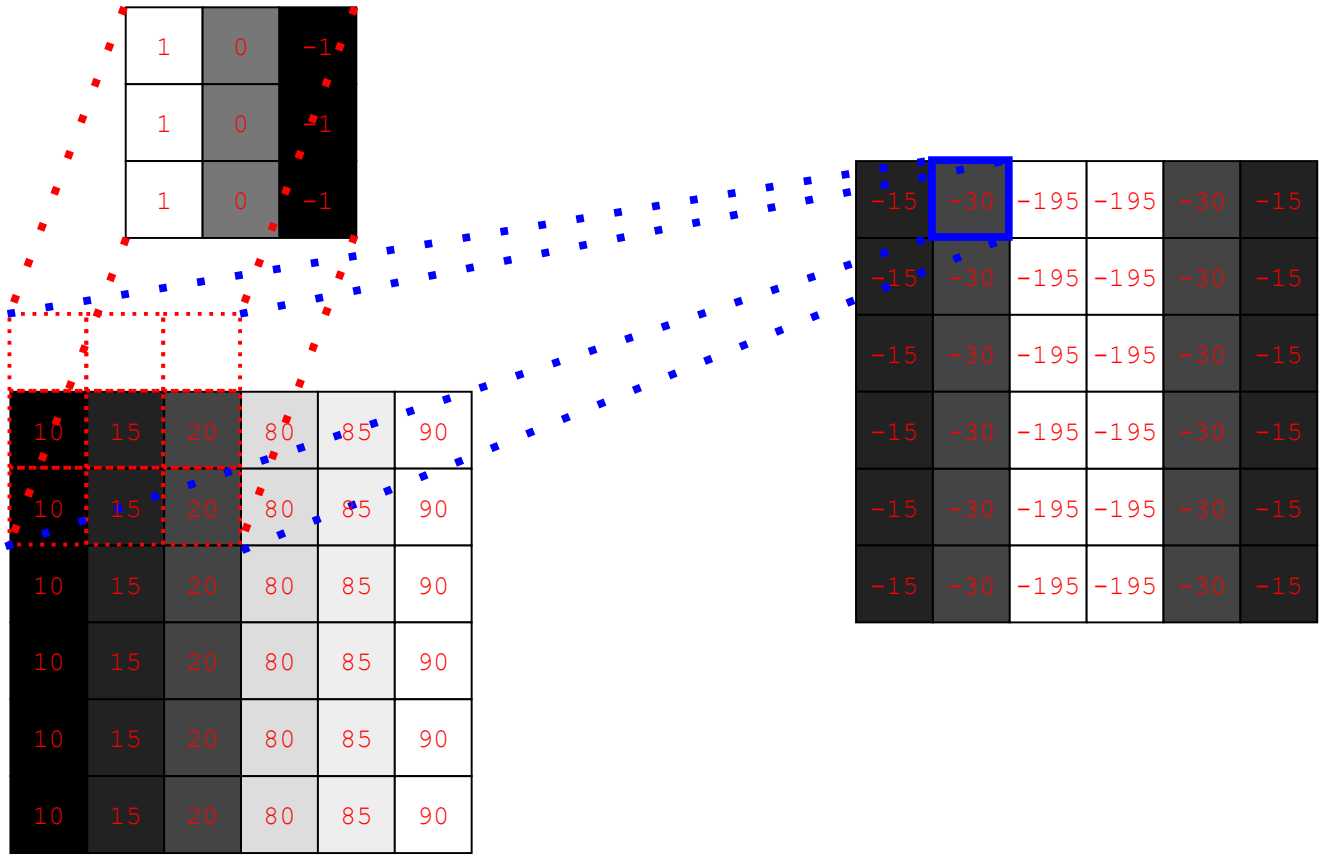


Figura: Cálculo de uma convolução 2D.

Convolução 2D

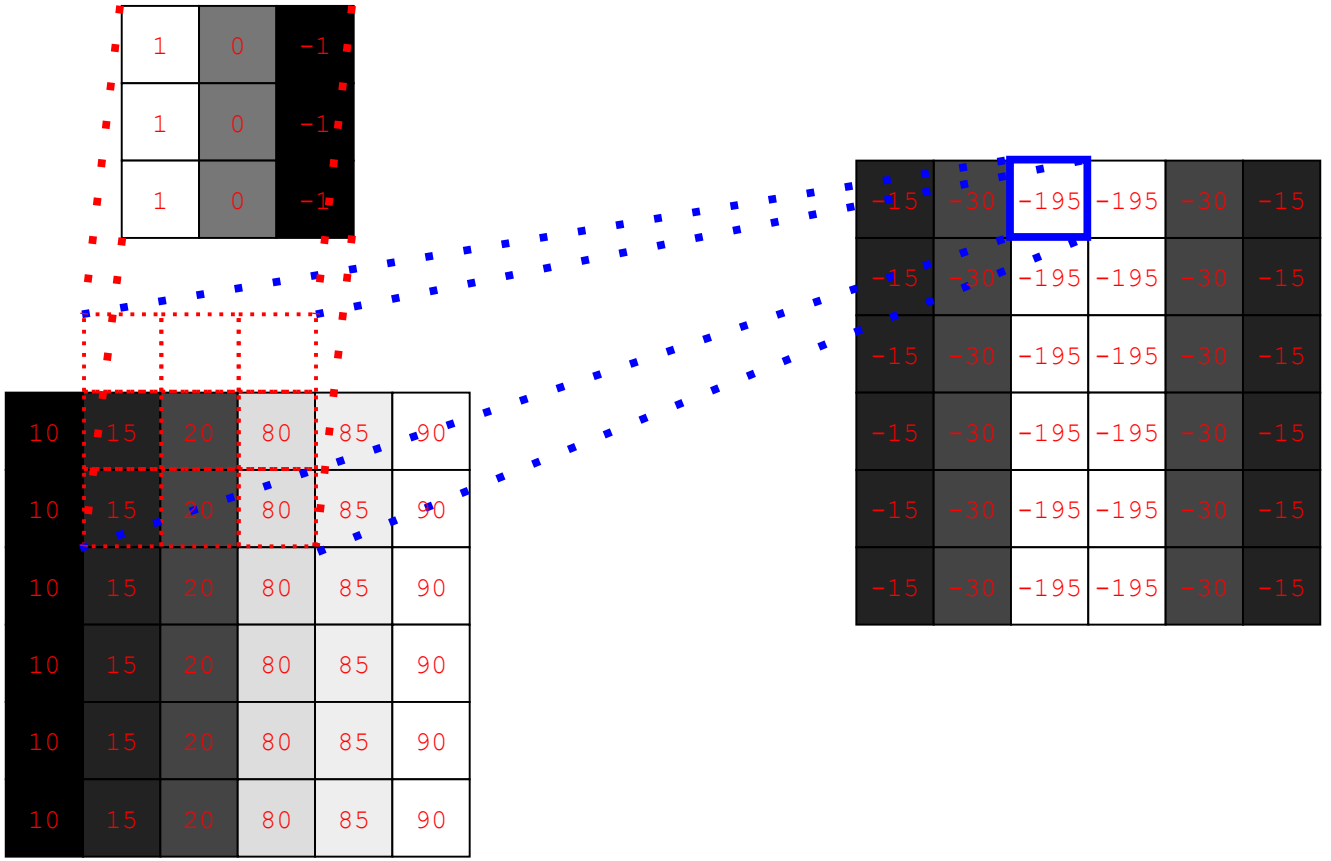


Figura: Cálculo de uma convolução 2D.

Convolução 2D

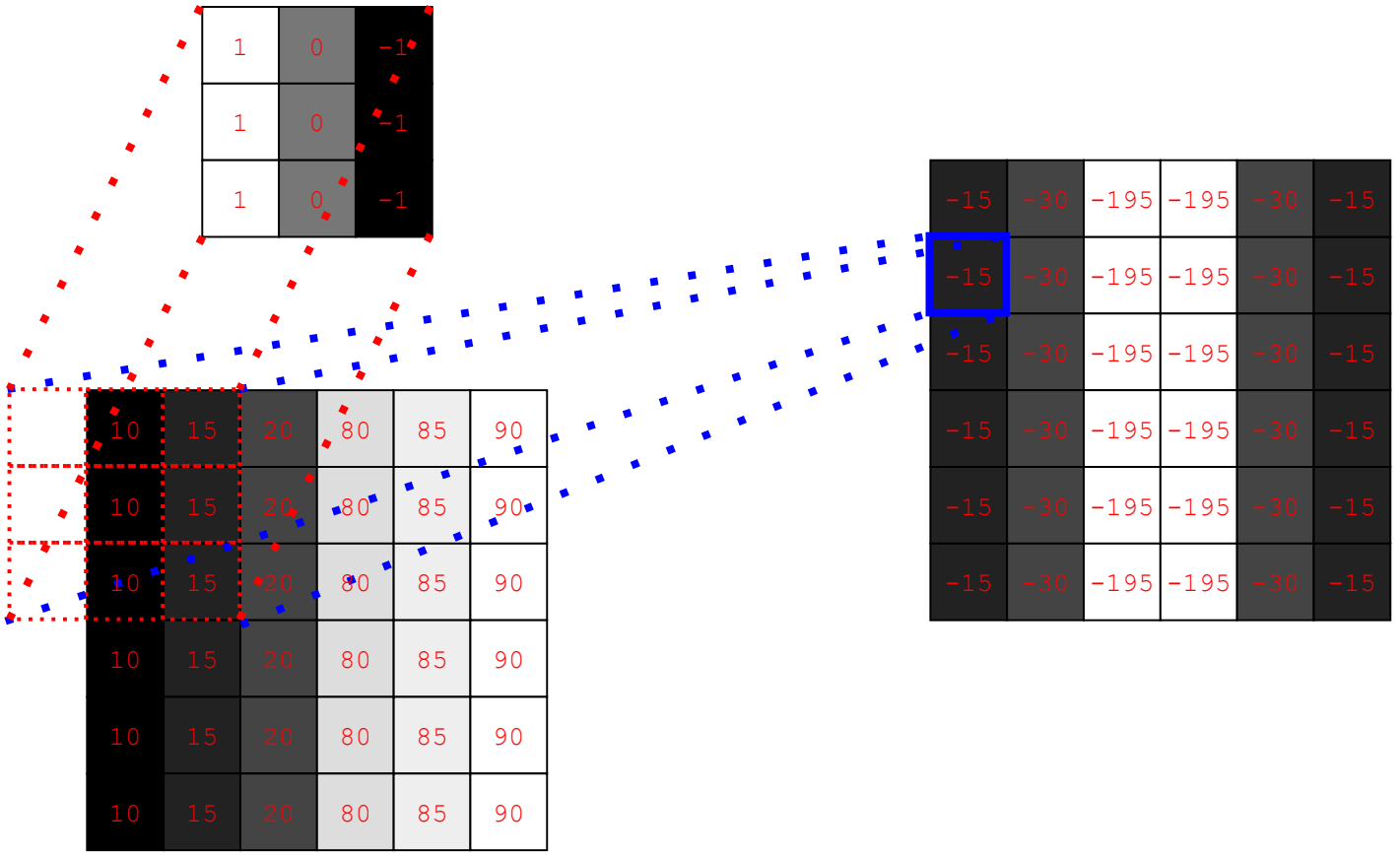


Figura: Cálculo de uma convolução 2D.

Convolução 2D

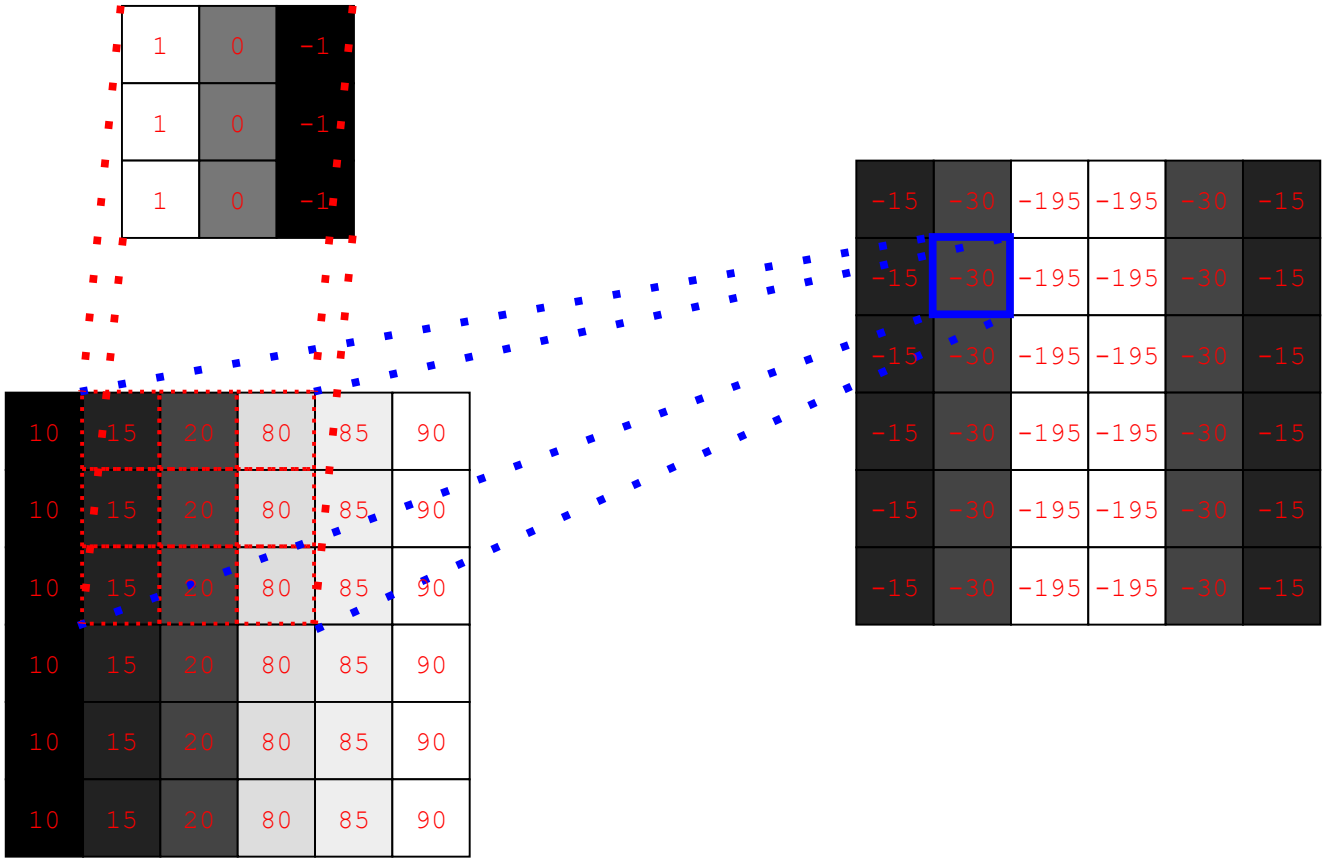


Figura: Cálculo de uma convolução 2D.

Convolução 2D

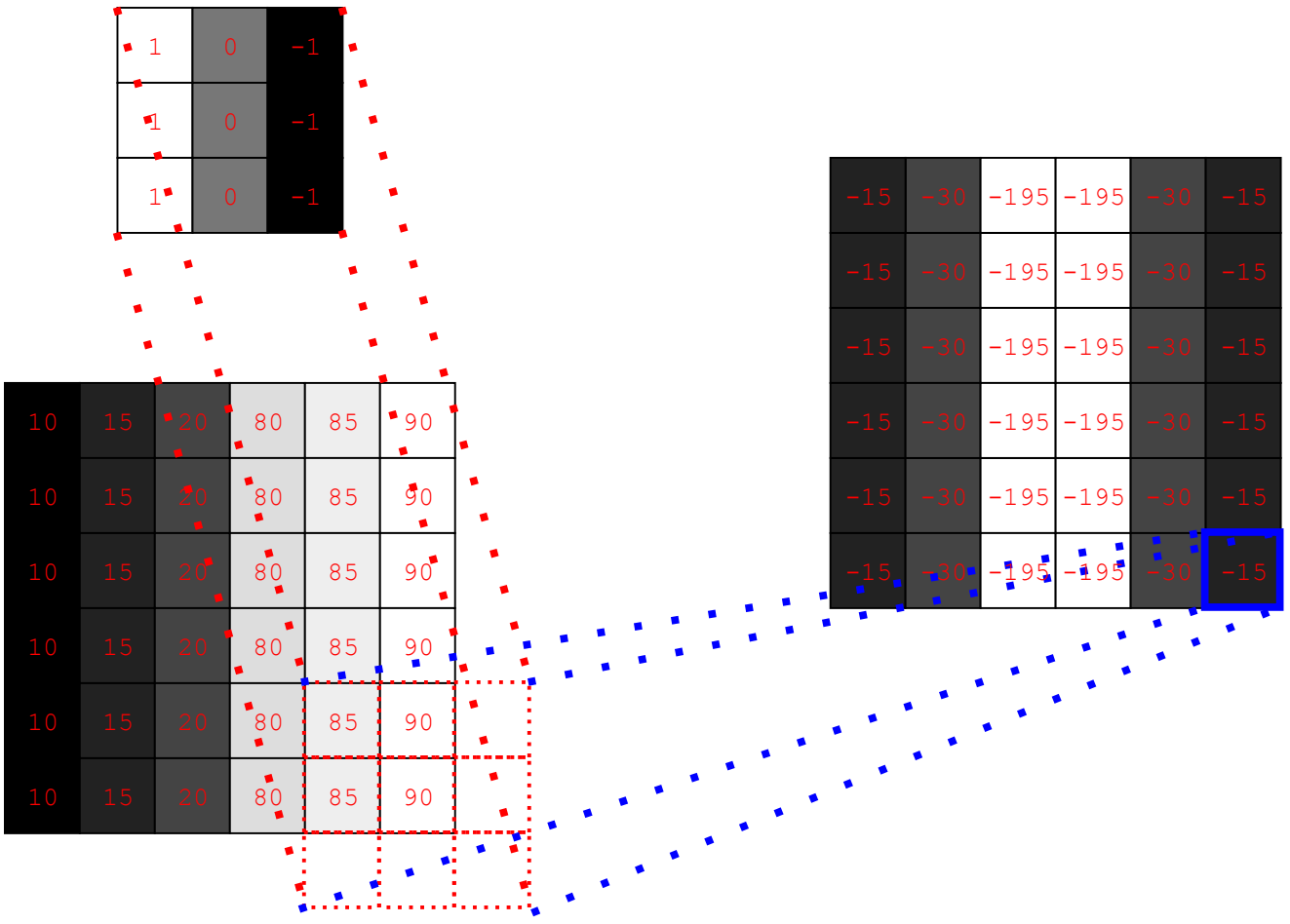


Figura: Cálculo de uma convolução 2D.

Convoluções e Correlações

Convoluções e Correlações

Convoluções e Correlações podem ser usadas para armazenar informação visual.

Convoluções e Correlações

Convoluções e Correlações

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

Convoluções e Correlações

Convoluções e Correlações

As respostas geradas por Convoluções 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

Convoluções e Correlações

Demo - Correlation and Convolution

Correlation_Convolution.ipynb

Convoluções e Correlações

Convoluções e Correlações

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

Convoluções e Correlações

Convoluções e Correlações

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

CNNs

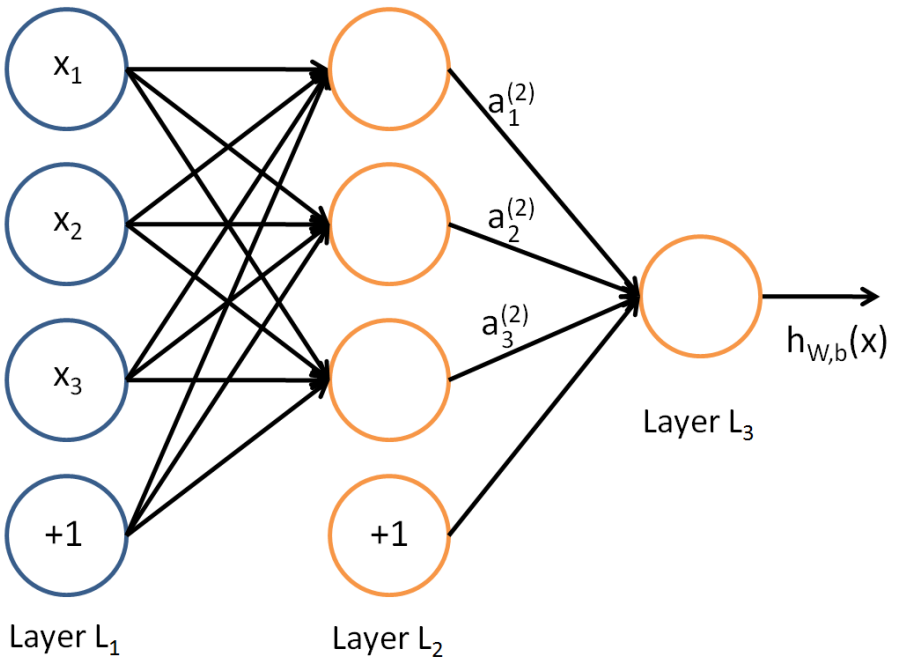
CNNs

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

CNNs

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

Camadas Totalmente Conectadas (MLPs)



Arquitetura Padrão de uma CNN

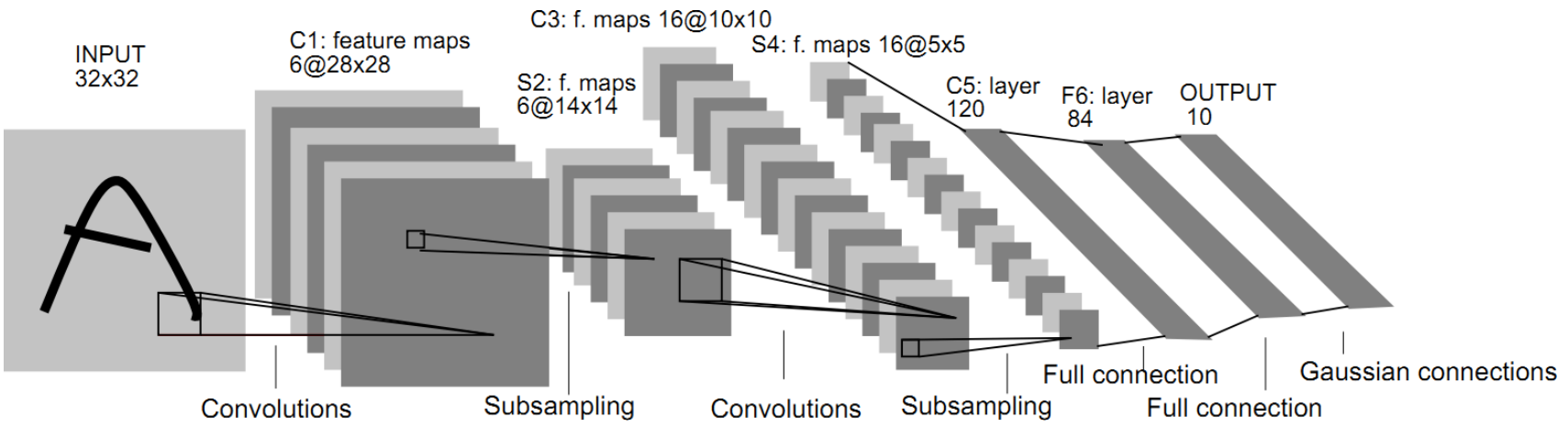
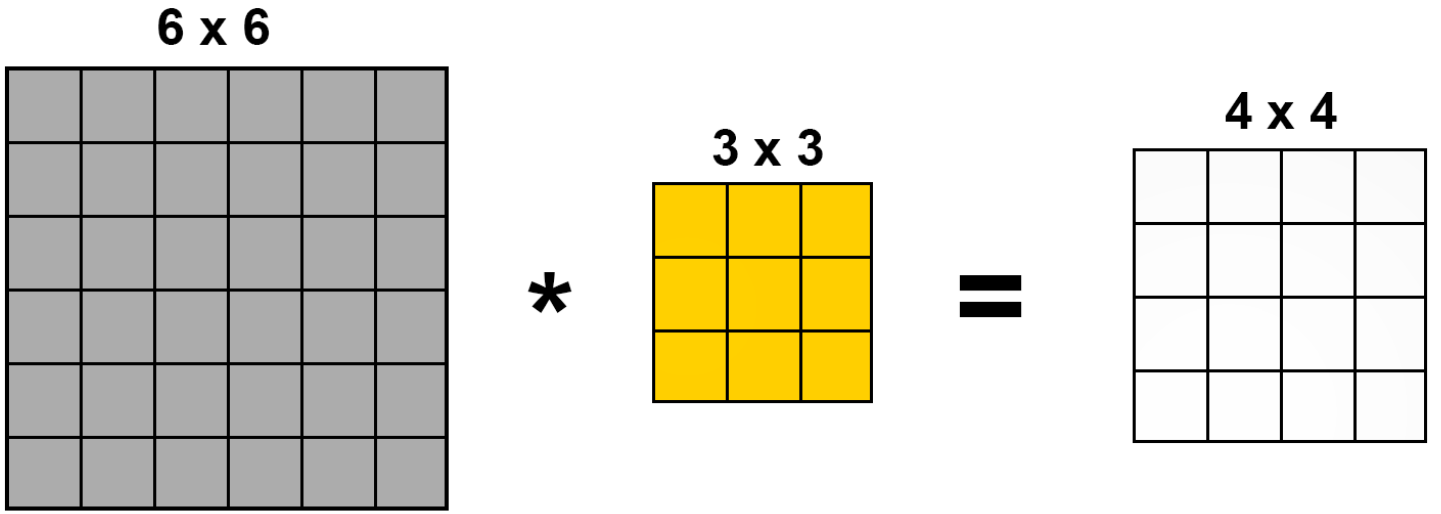


Figura: Arquitetura da LeNet [1].

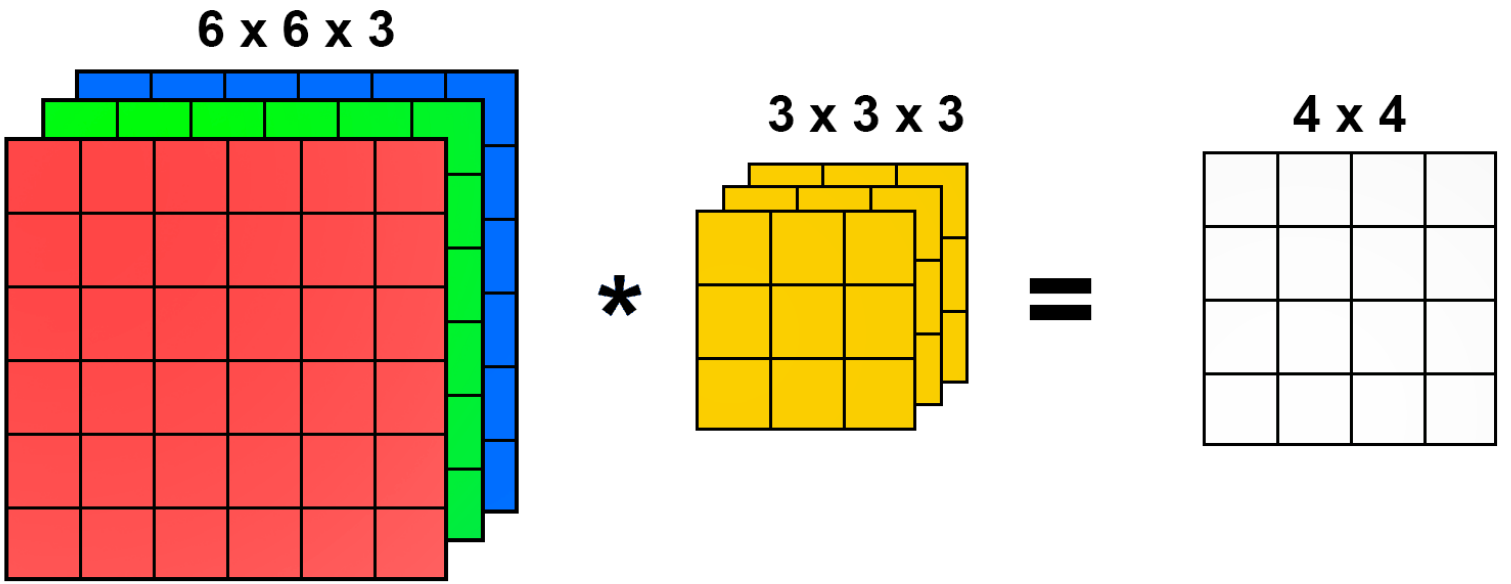
Camadas Convolucionais

- Convoluções com imagens de 1 canal (preta e branca).



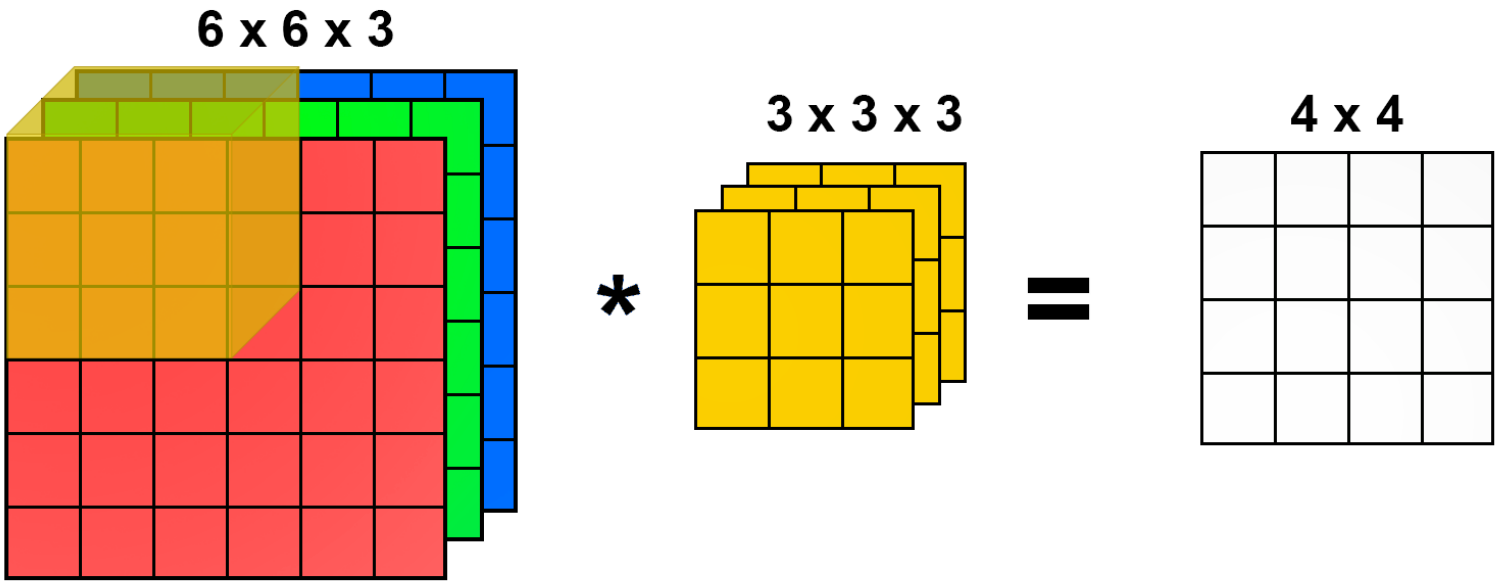
Camadas Convolucionais

- Convoluções com imagens de 3 canais (RGB).



Camadas Convolucionais

- Convoluções com imagens de 3 canais (RGB).



Camadas Convolucionais

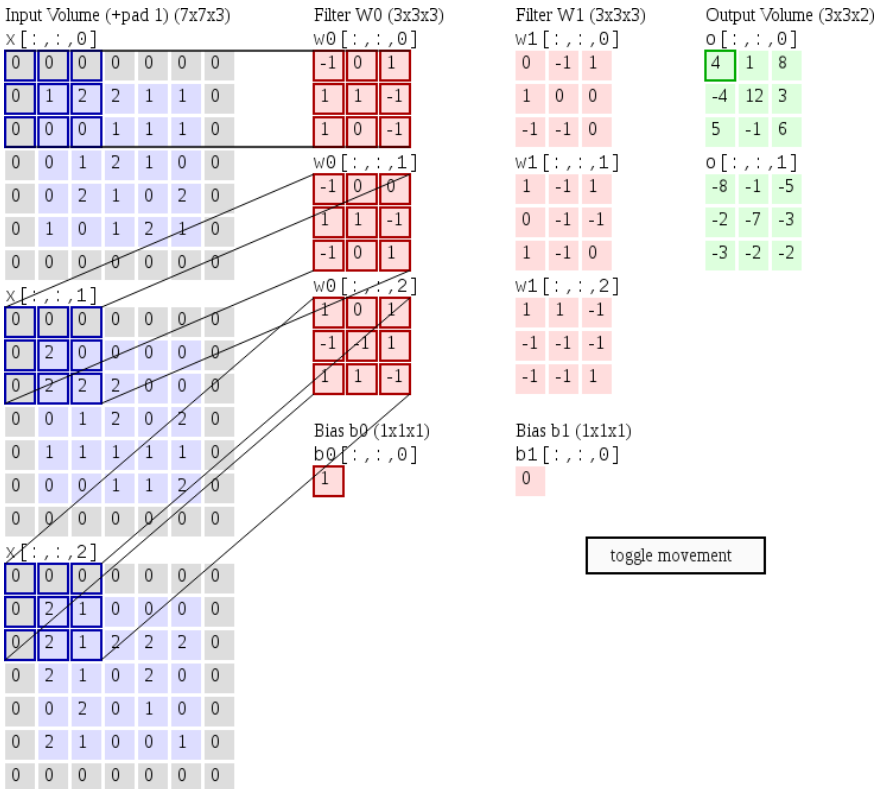


Figura: Convoluções em uma CNN.
https://cs231n.github.io/assets/conv-demo/index.html

Camadas Convolucionais

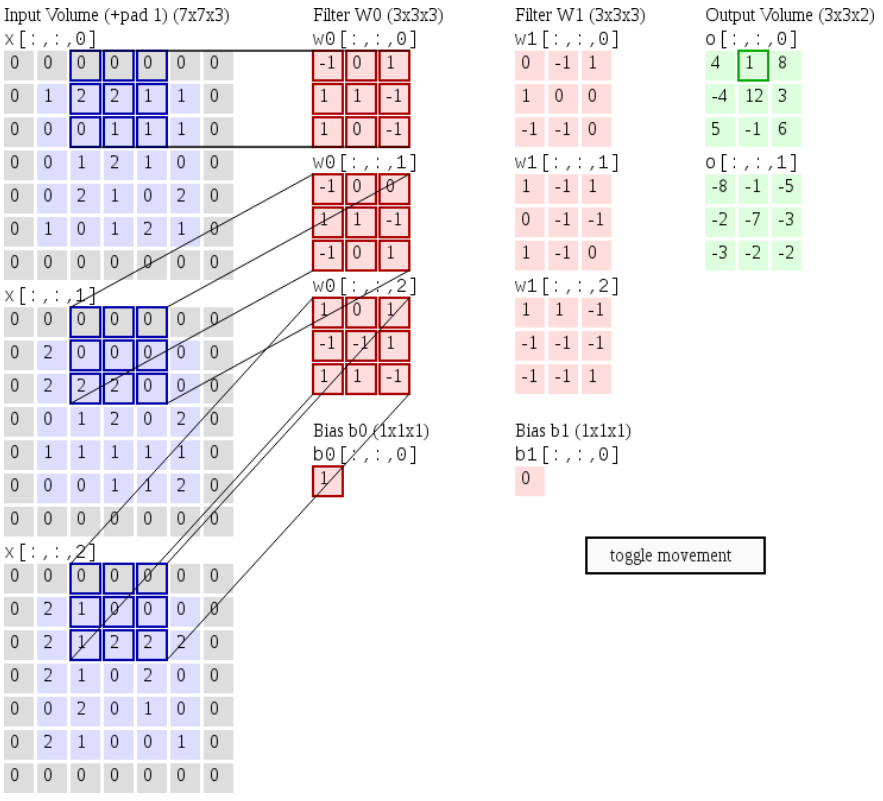


Figura: Convoluções em uma CNN.

Camadas Convolucionais

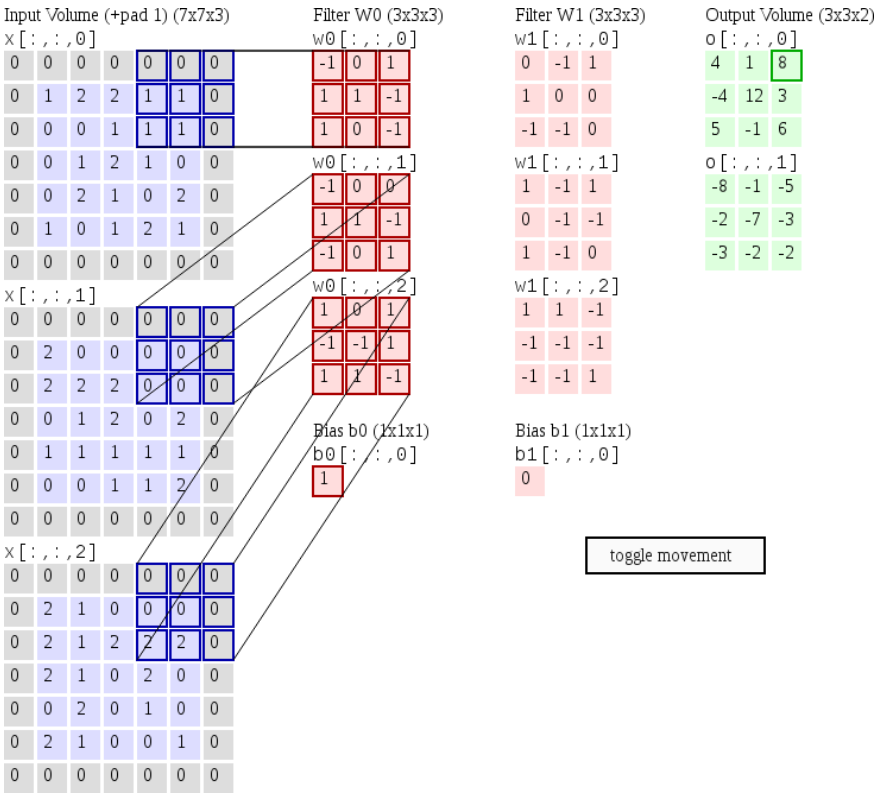


Figura: Convoluções em uma CNN.

Camadas Convolucionais

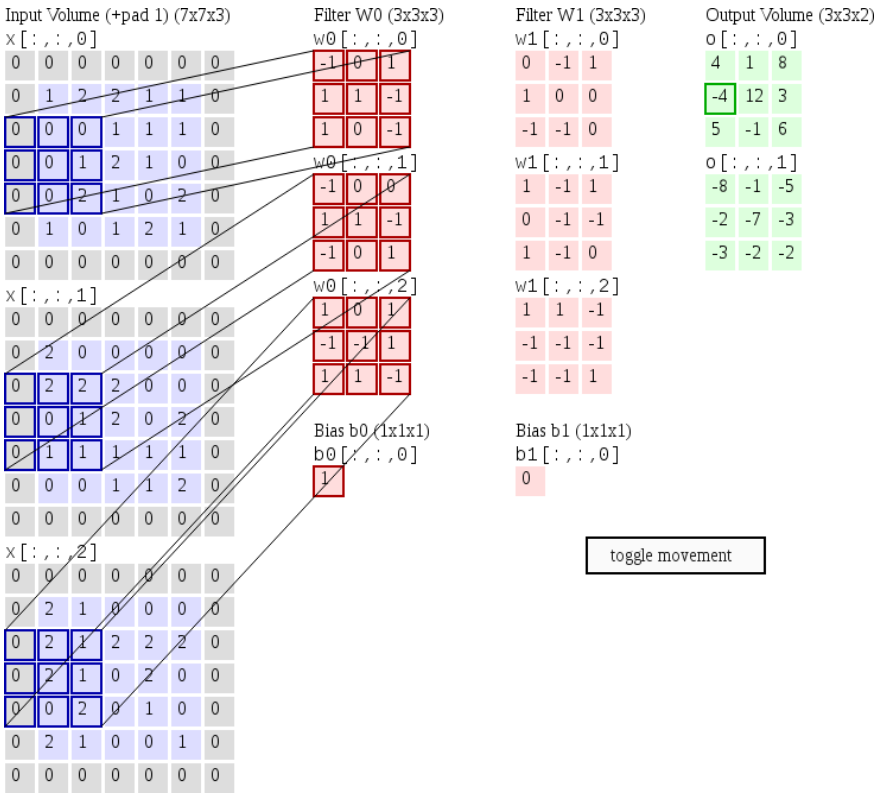


Figura: Convoluções em uma CNN.

Camadas Convolucionais

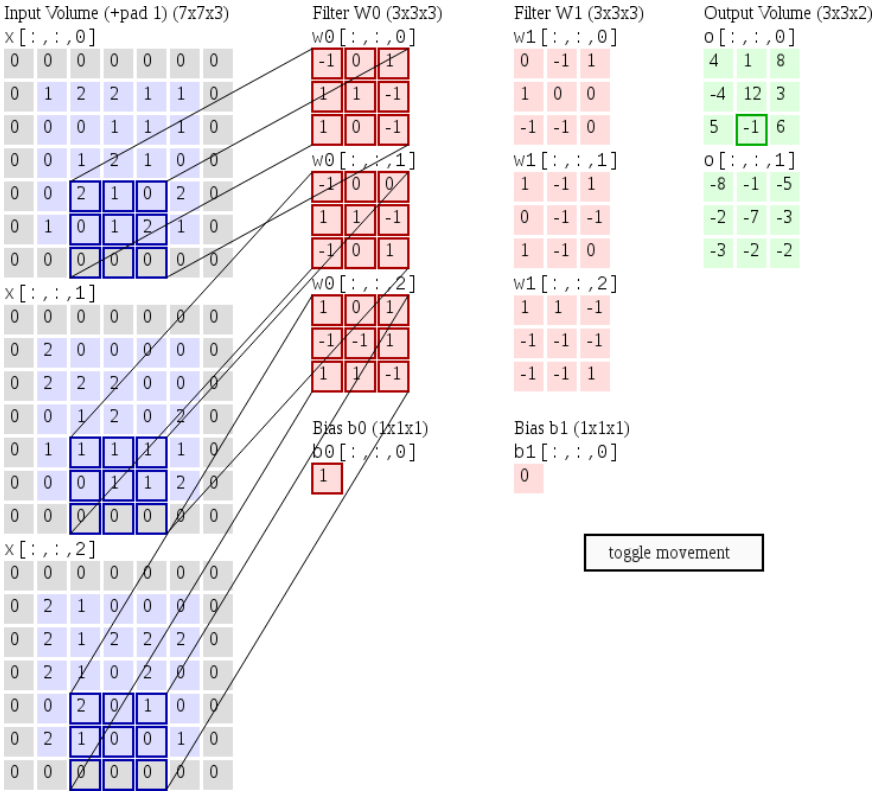


Figura: Convoluções em uma CNN.

Camadas Convolucionais

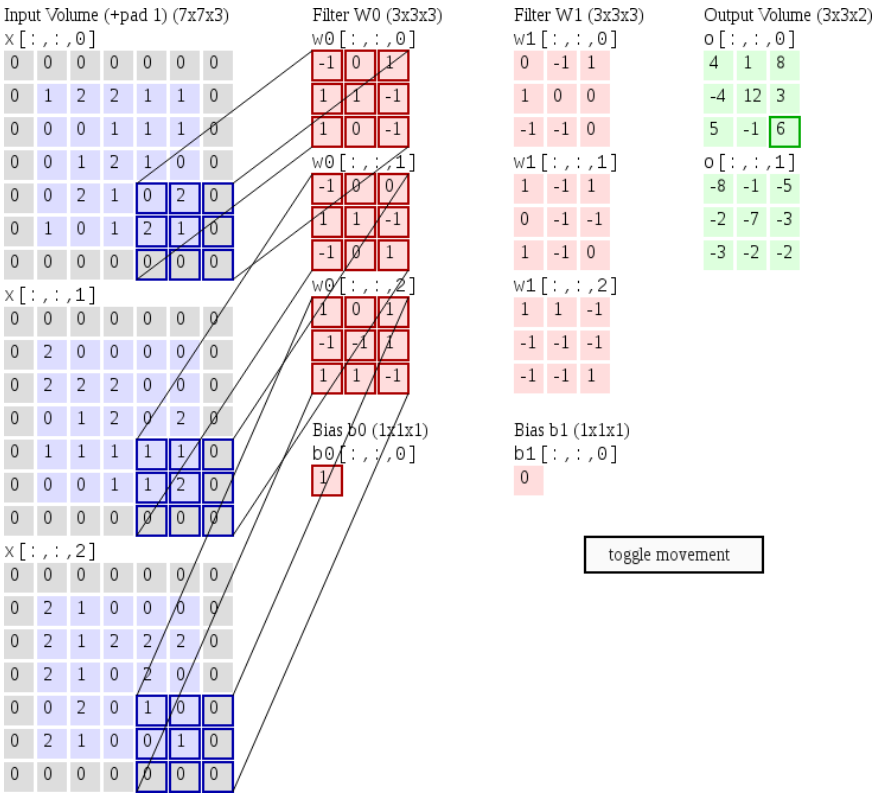


Figura: Convoluções em uma CNN.

Camadas Convolucionais

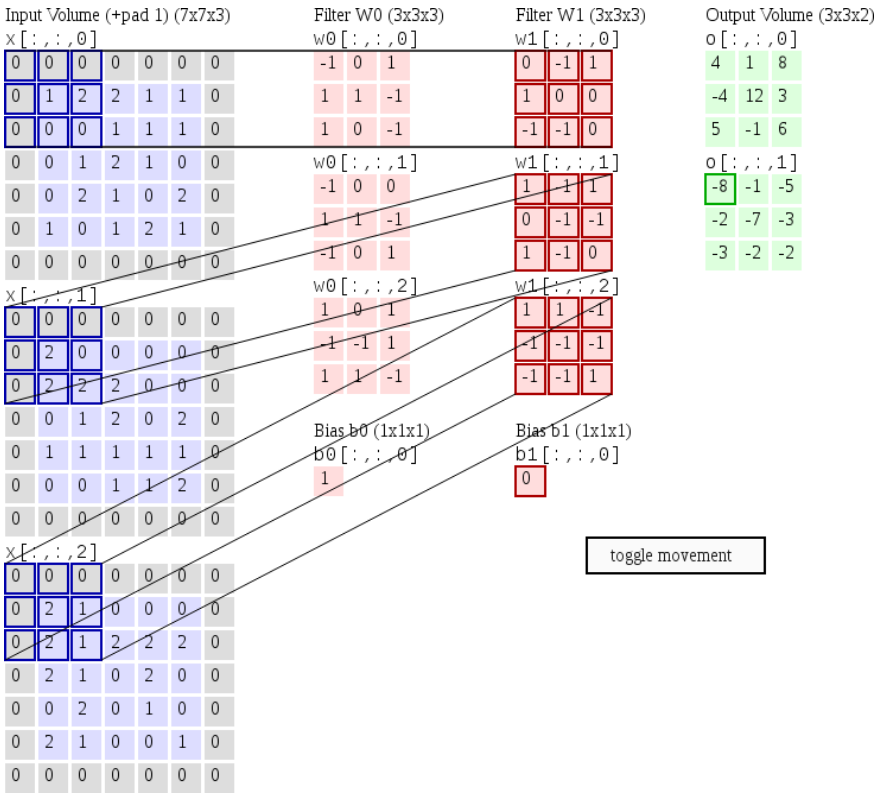


Figura: Convoluções em uma CNN.

Camadas Convolucionais

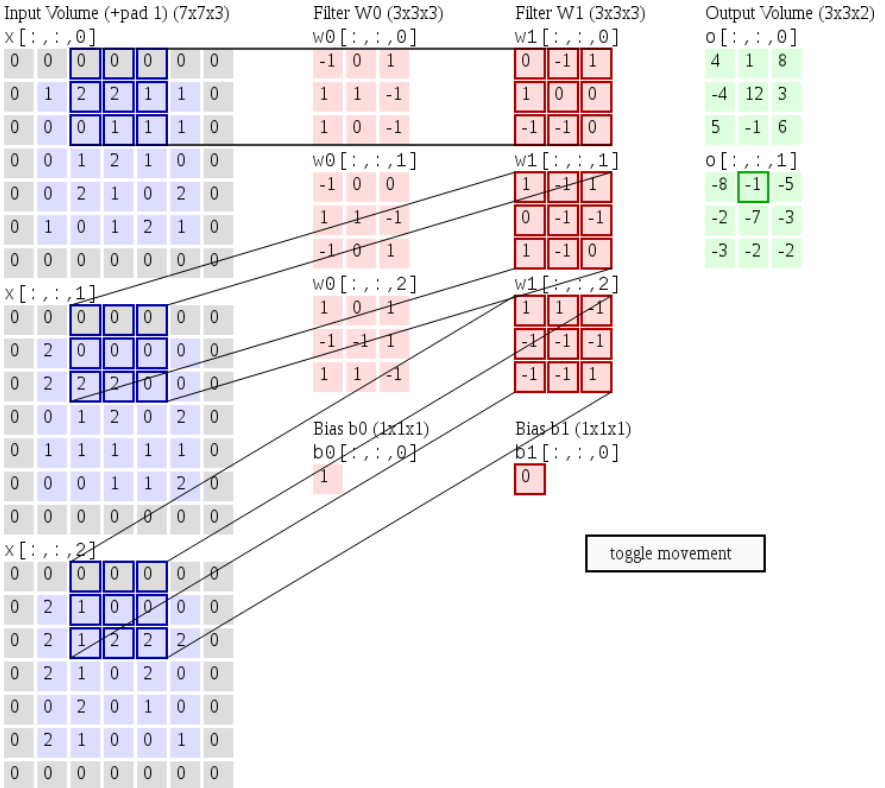


Figura: Convoluções em uma CNN.

Camadas Convolucionais

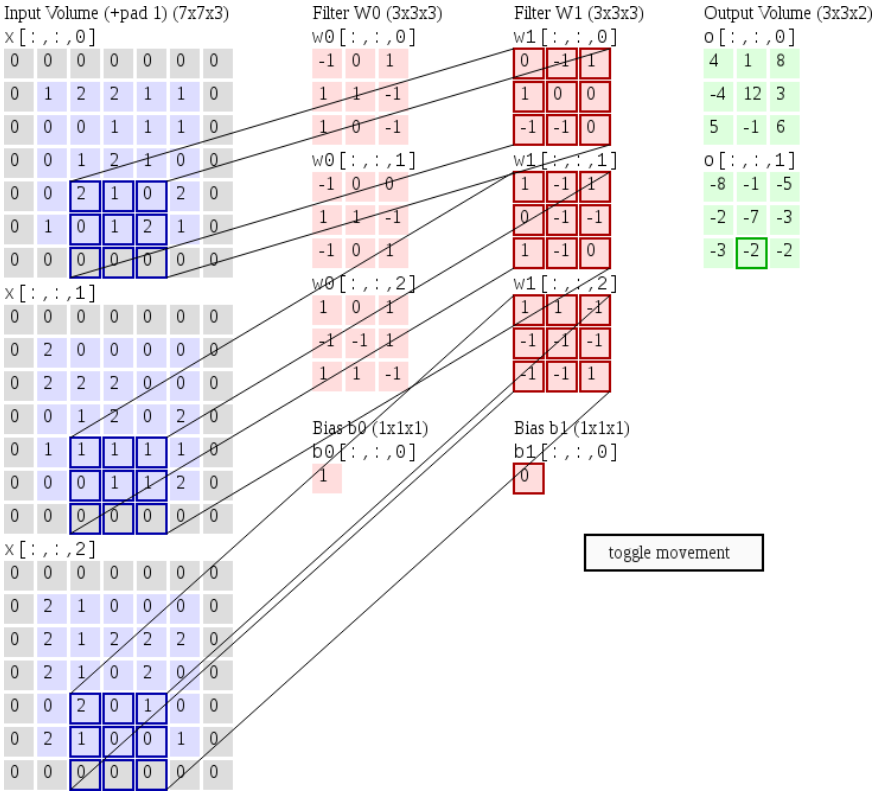


Figura: Convoluções em uma CNN.

Camadas Convolucionais

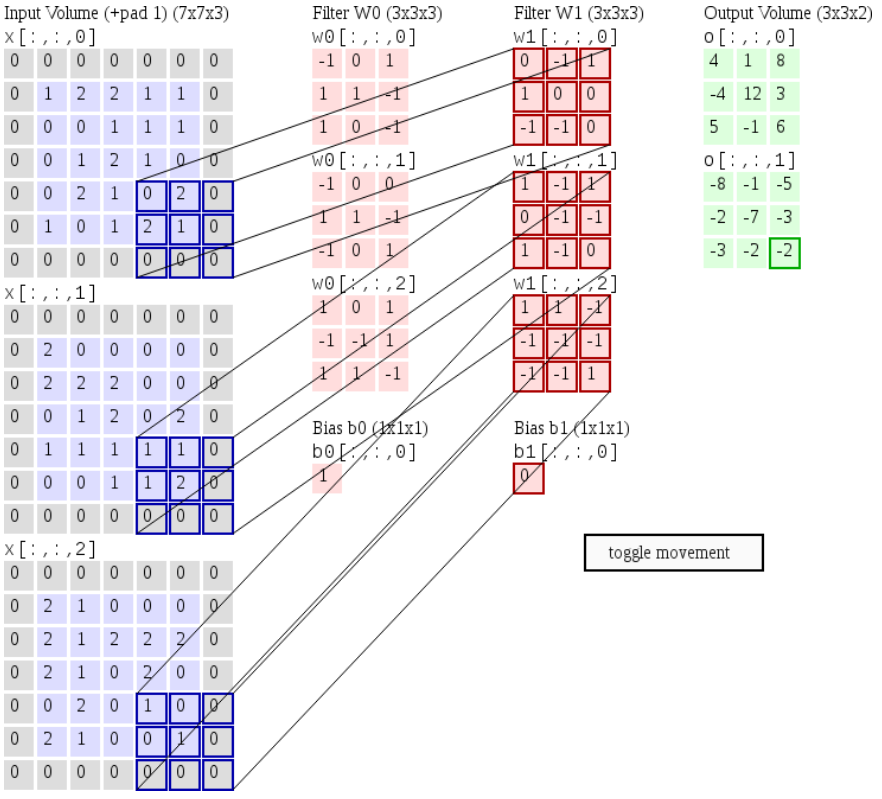


Figura: Convoluções em uma CNN.

Camadas Convolucionais

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

Field of View

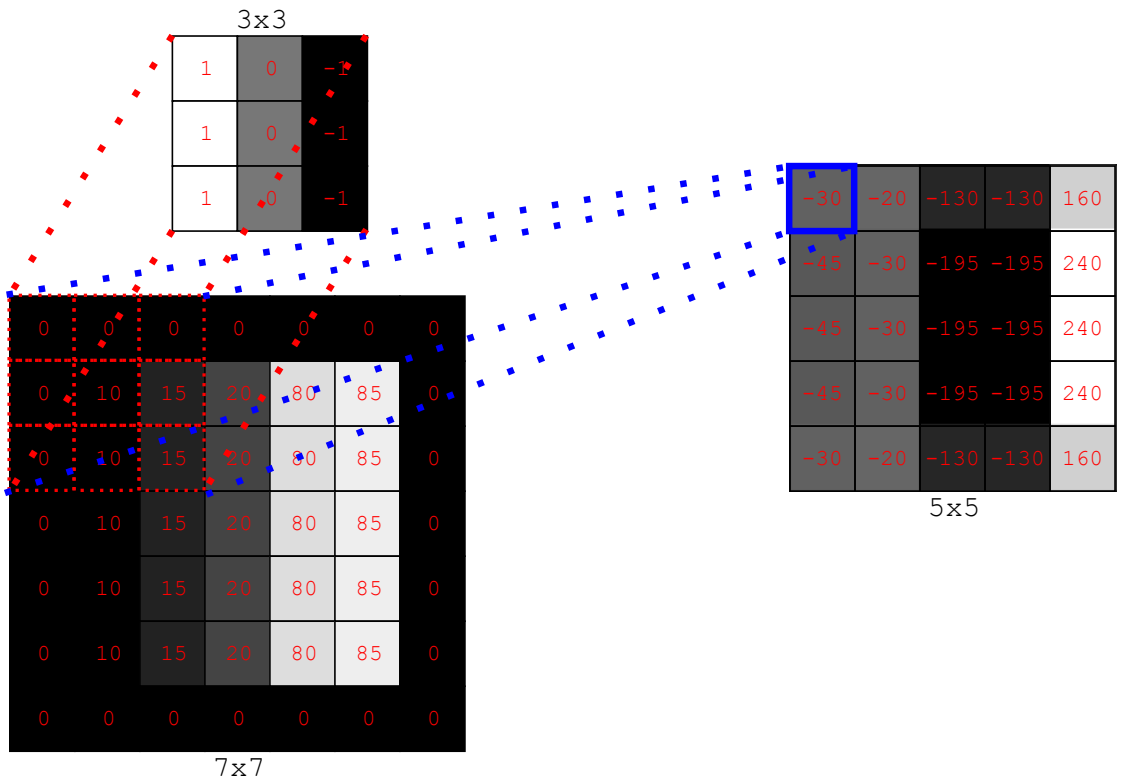


Figura: Field of view de tamanho 3x3.

Field of View

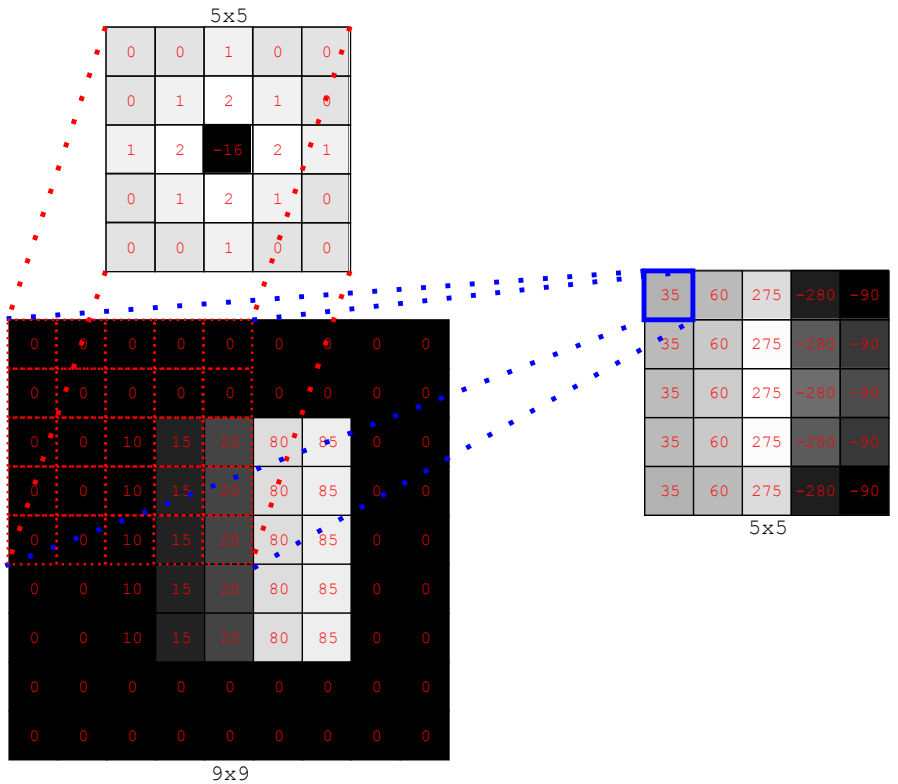


Figura: Field of view de tamanho 5x5.

Stride

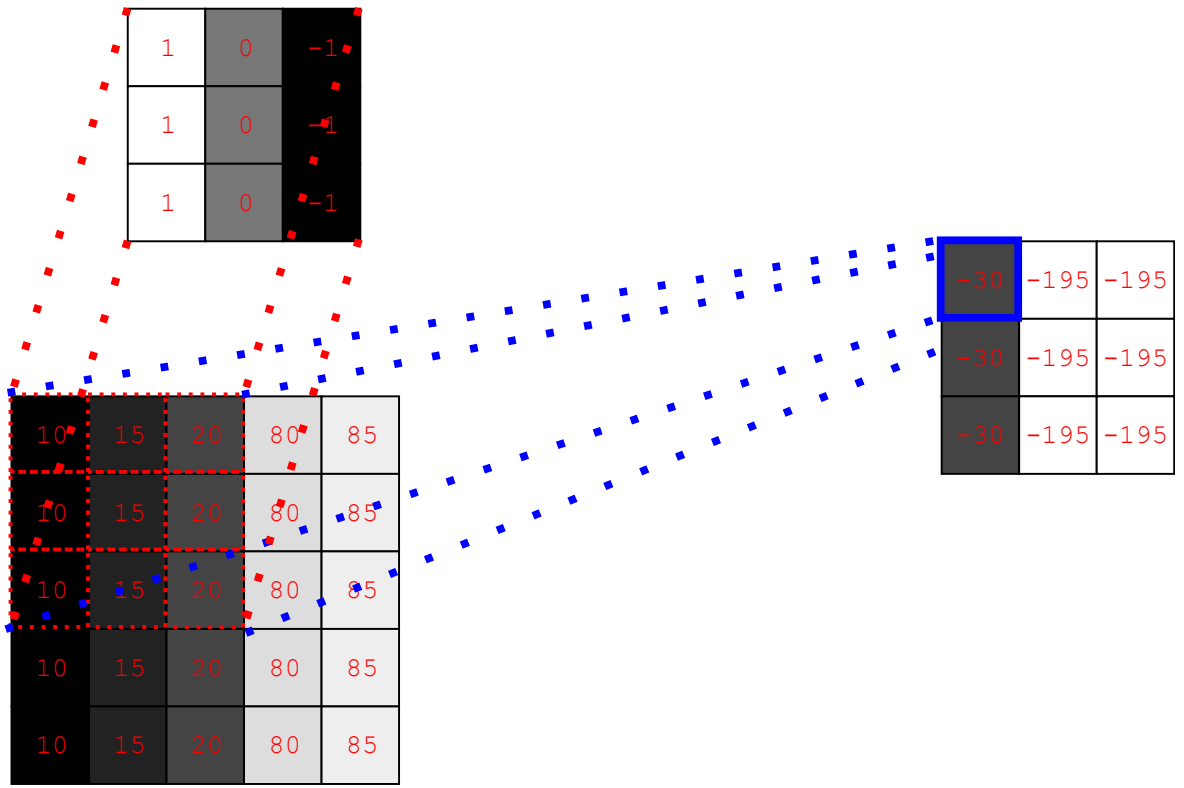


Figura: Stride de tamanho 1.

Stride

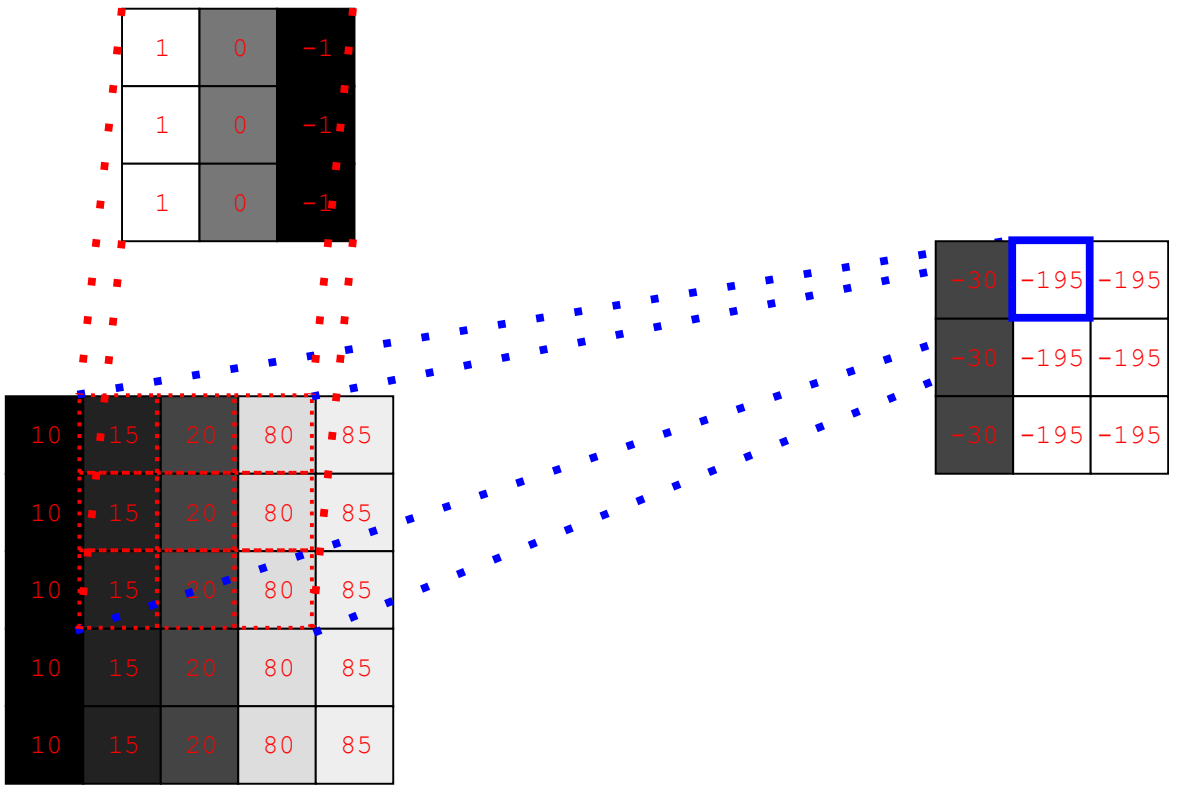


Figura: Stride de tamanho 1.

Stride

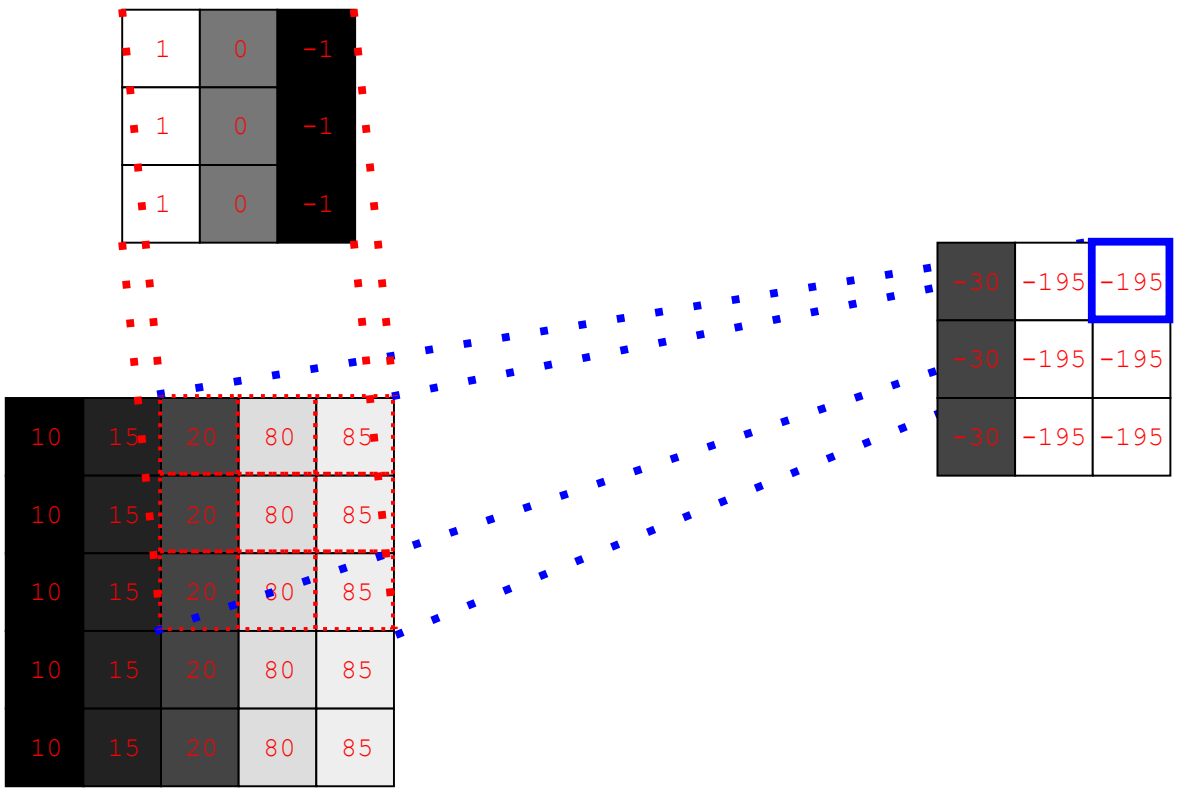


Figura: Stride de tamanho 1.

Stride

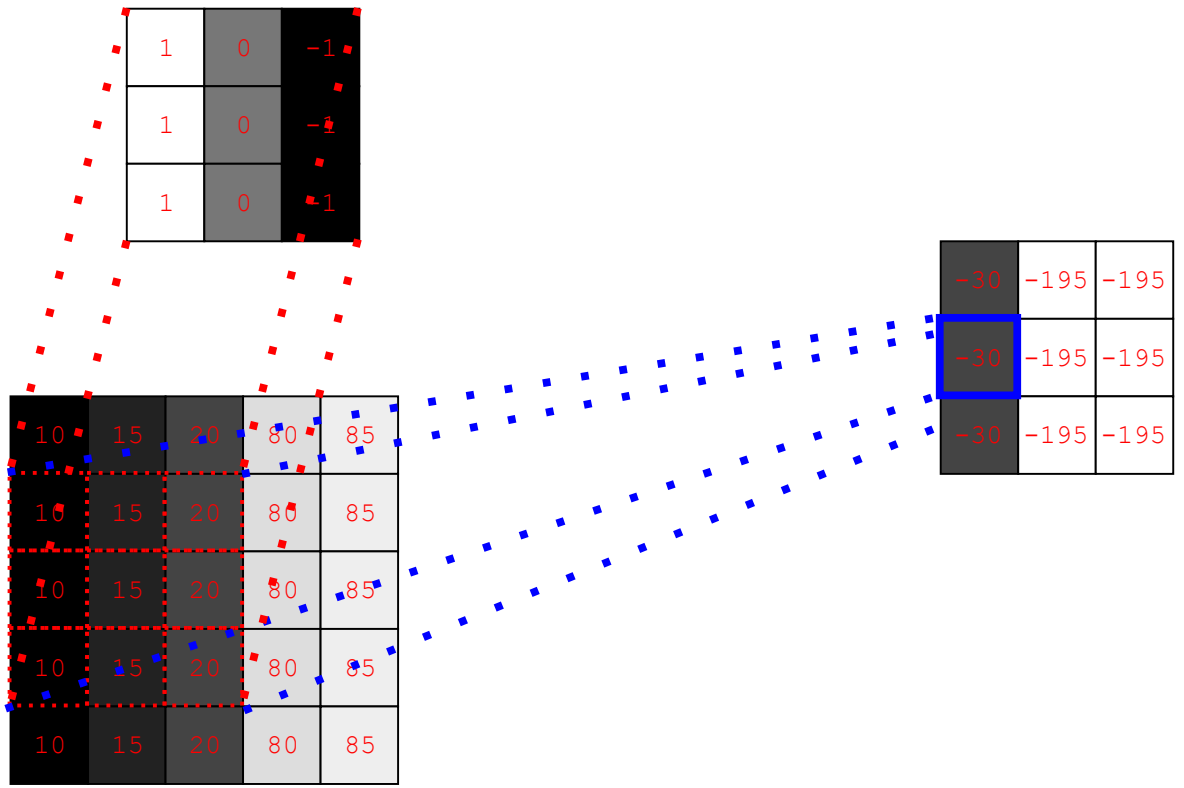


Figura: Stride de tamanho 1.

Stride

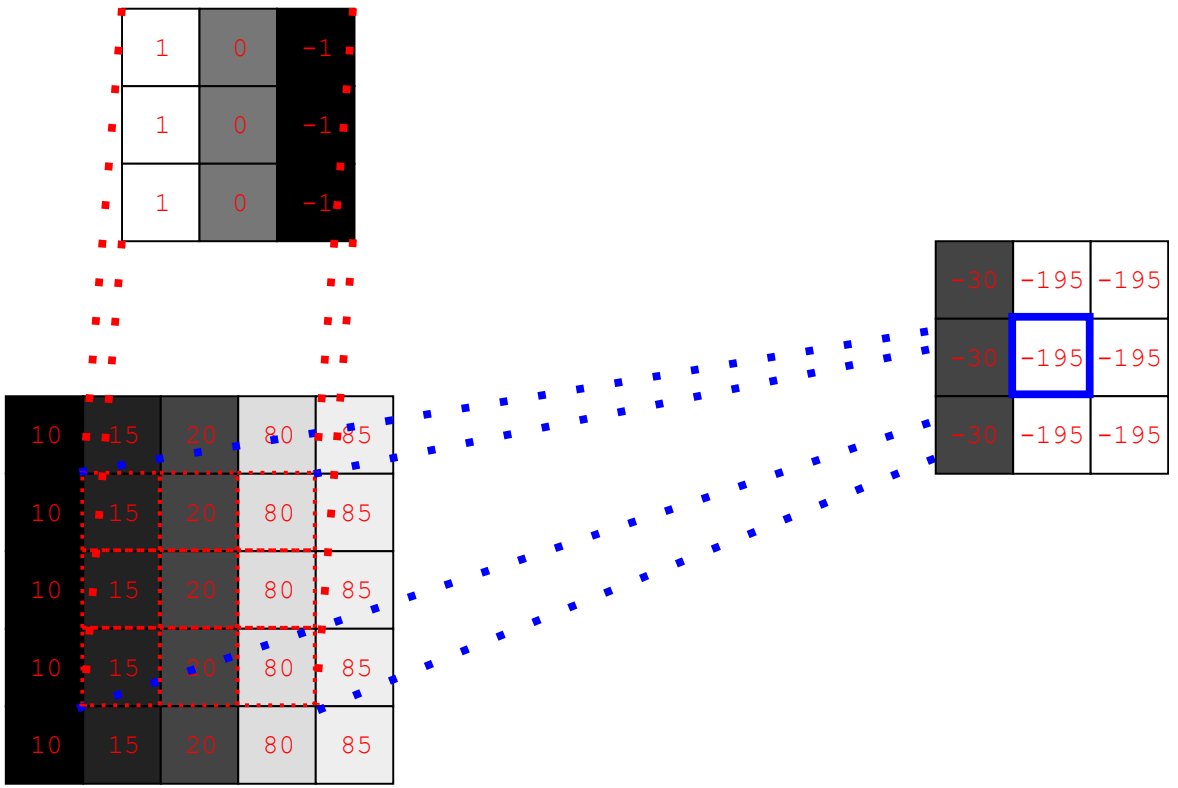


Figura: Stride de tamanho 1.

Stride

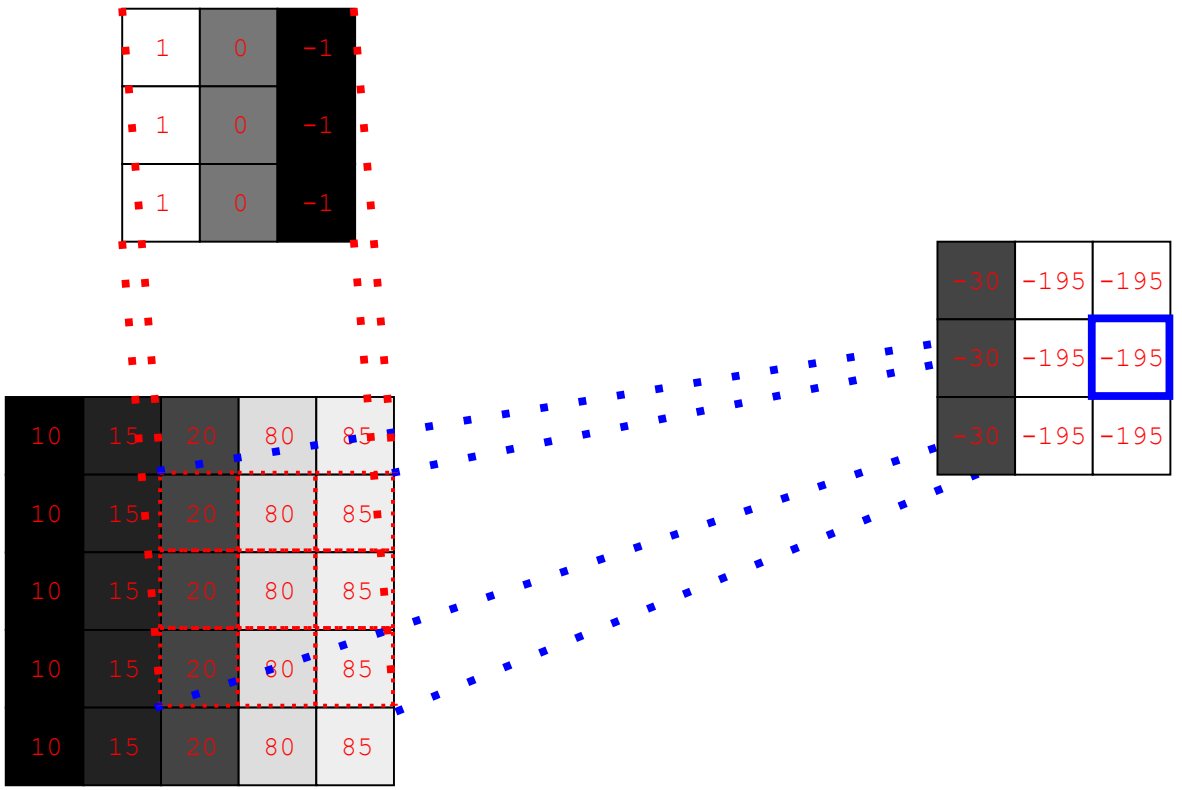


Figura: Stride de tamanho 1.

Stride

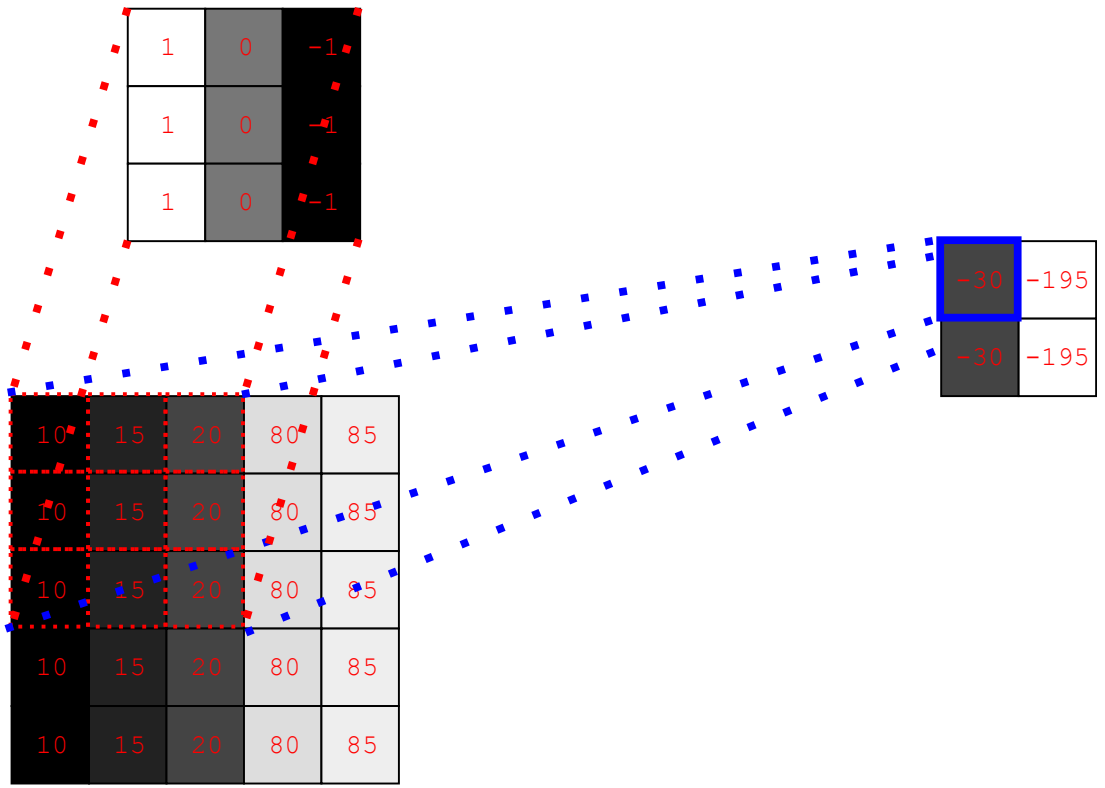


Figura: Stride de tamanho 2.

Stride

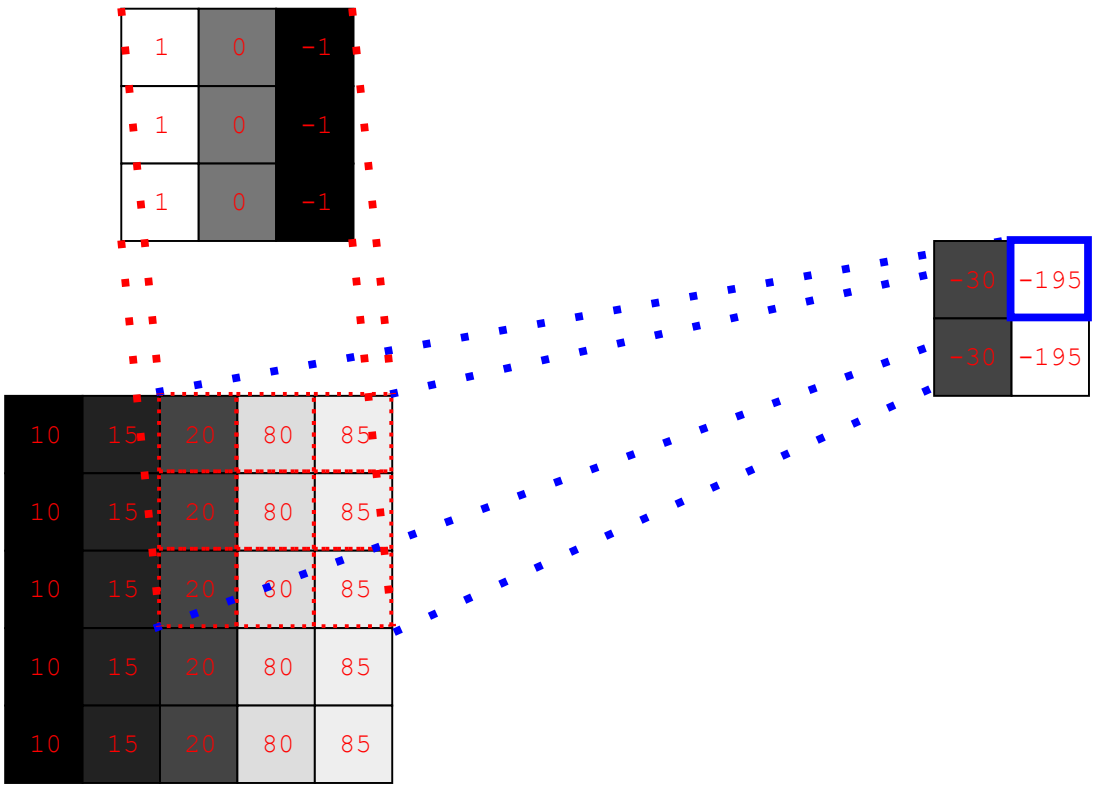


Figura: Stride de tamanho 2.

Stride

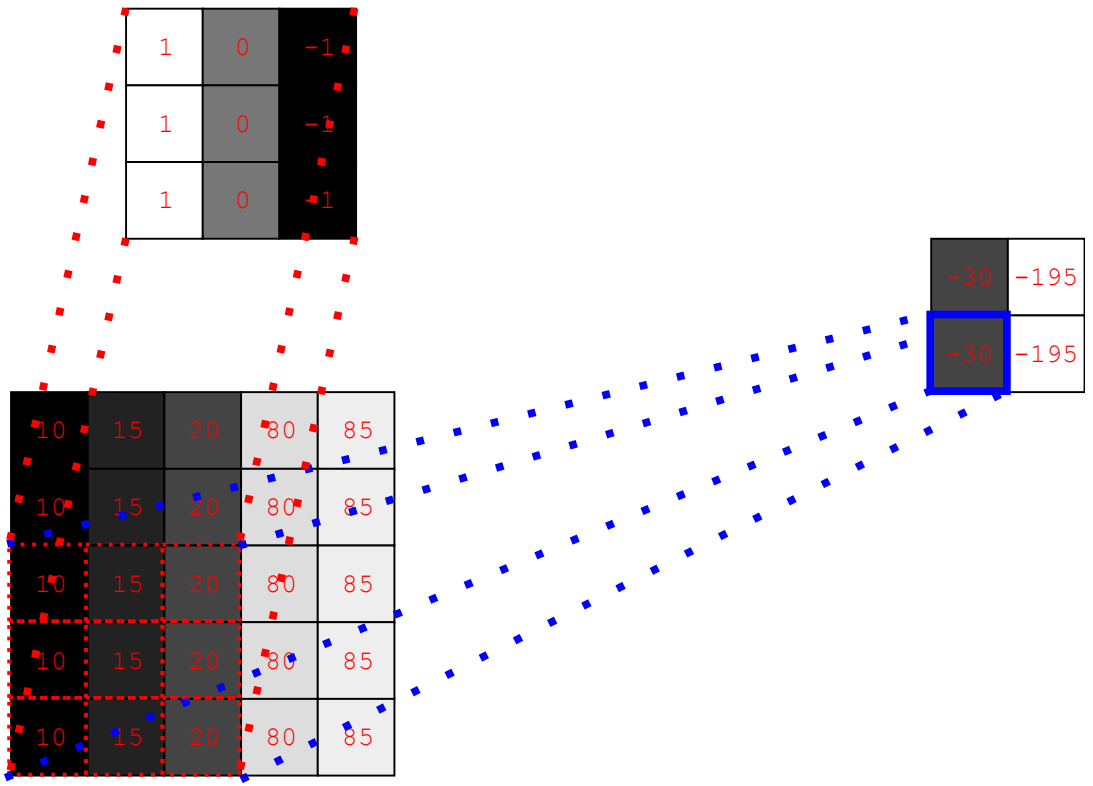


Figura: Stride de tamanho 2.

Stride

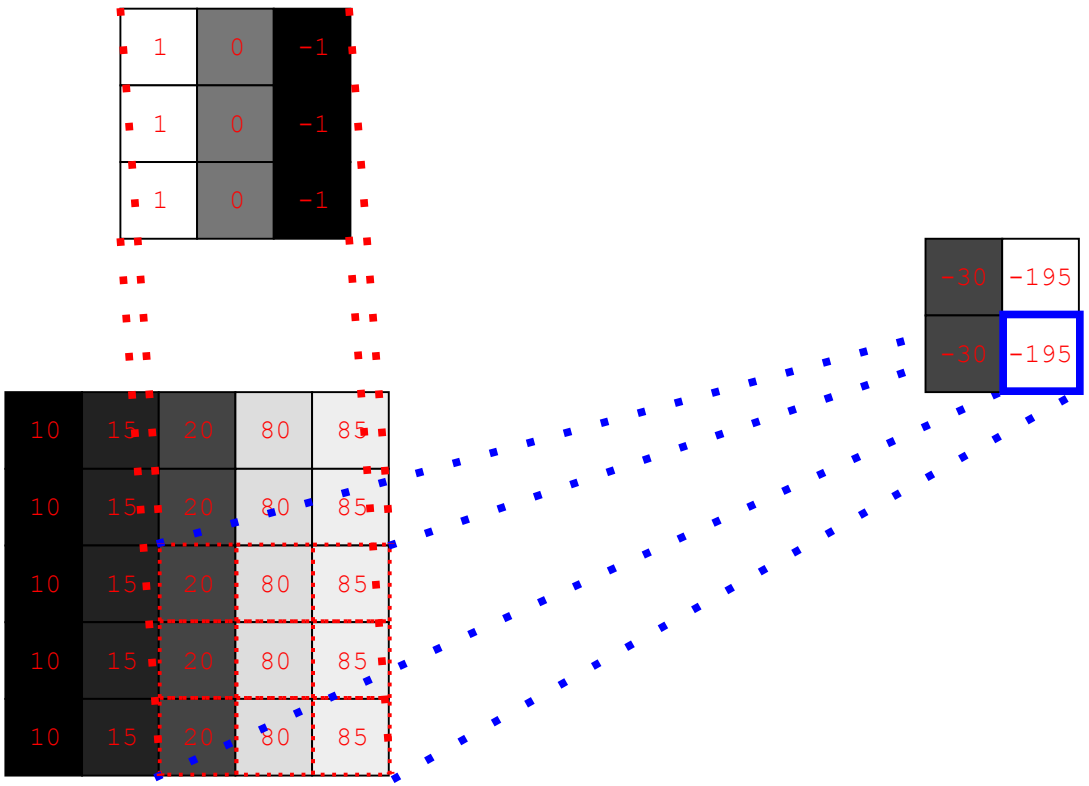


Figura: Stride de tamanho 2.

Padding

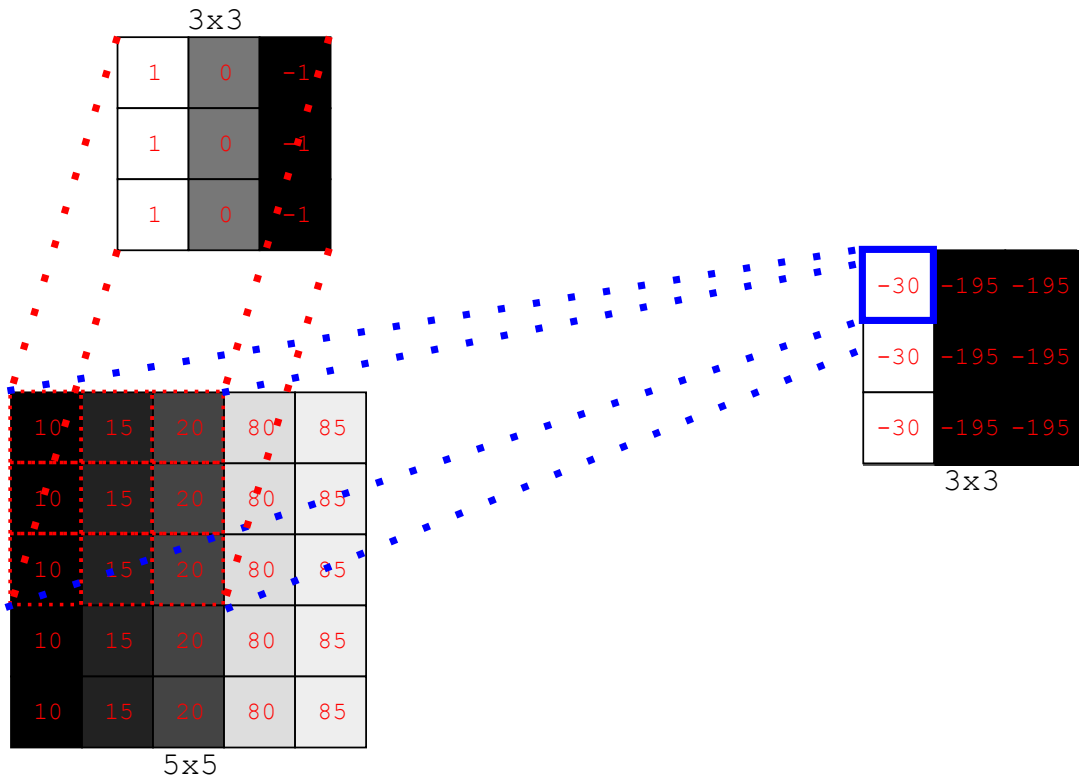


Figura: Padding de tamanho 0.

Padding

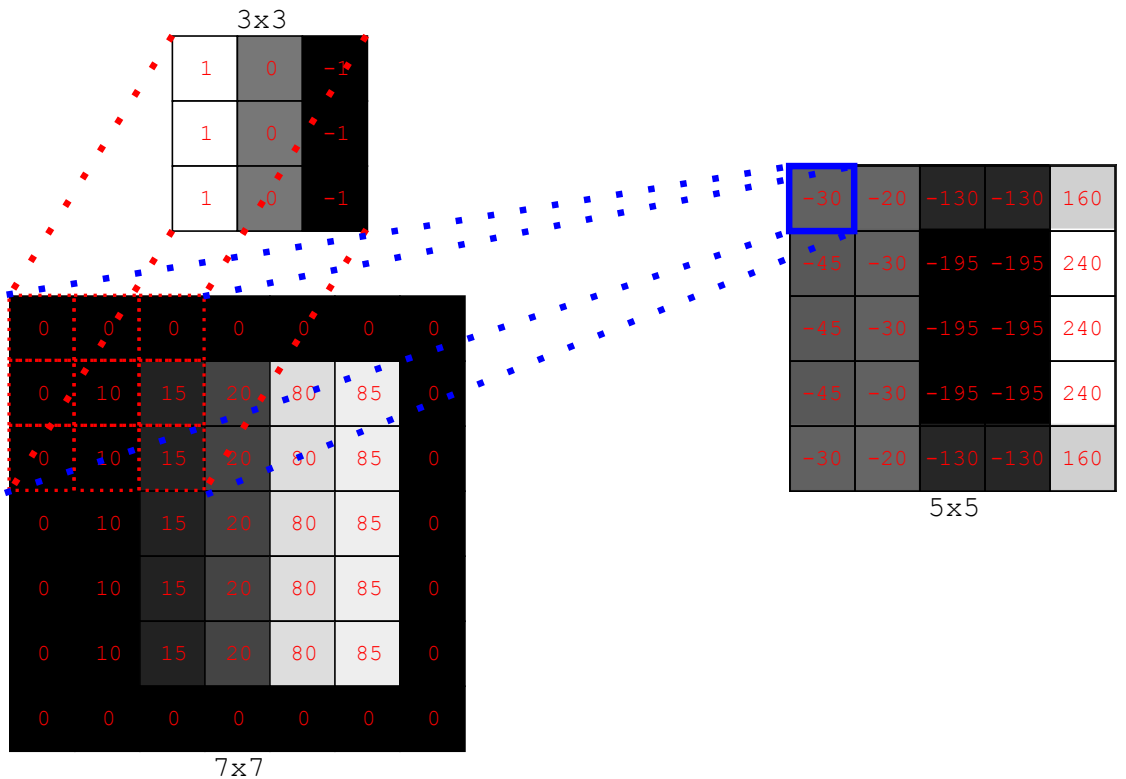


Figura: Padding de tamanho 1.

Padding

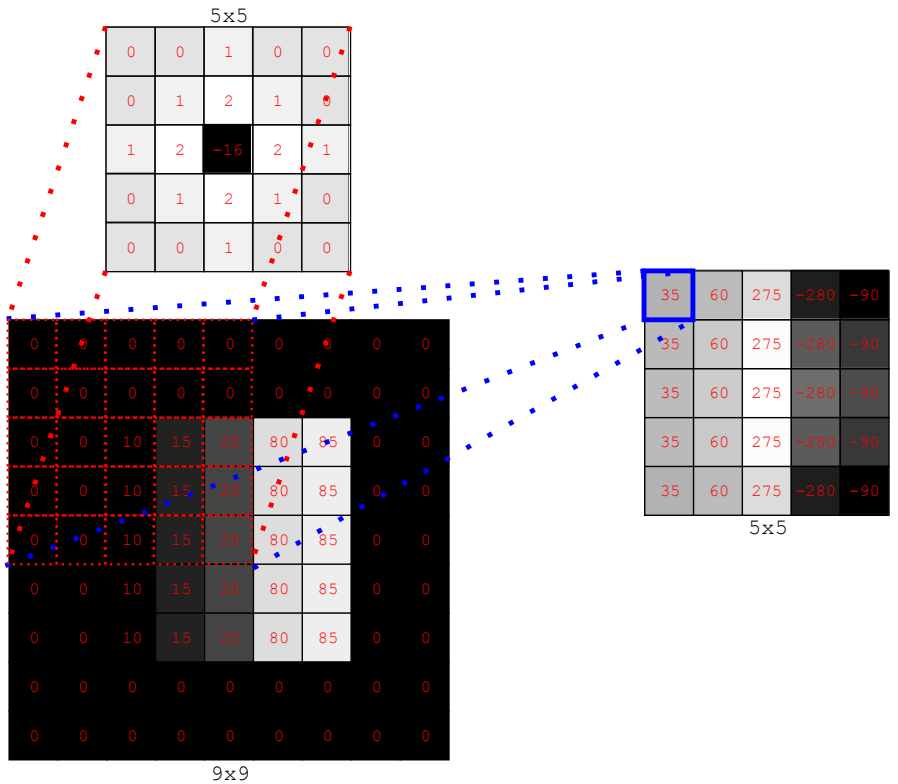


Figura: Padding de tamanho 2.

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 3x3, 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 7x7, o Stride (S) for igual a 7 e o Padding (P) for igual a 0?

Camada Convolucional - PyTorch

Demo Conv2D

Camadas-CNN.ipynb

Camadas de Pooling

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

Pooling

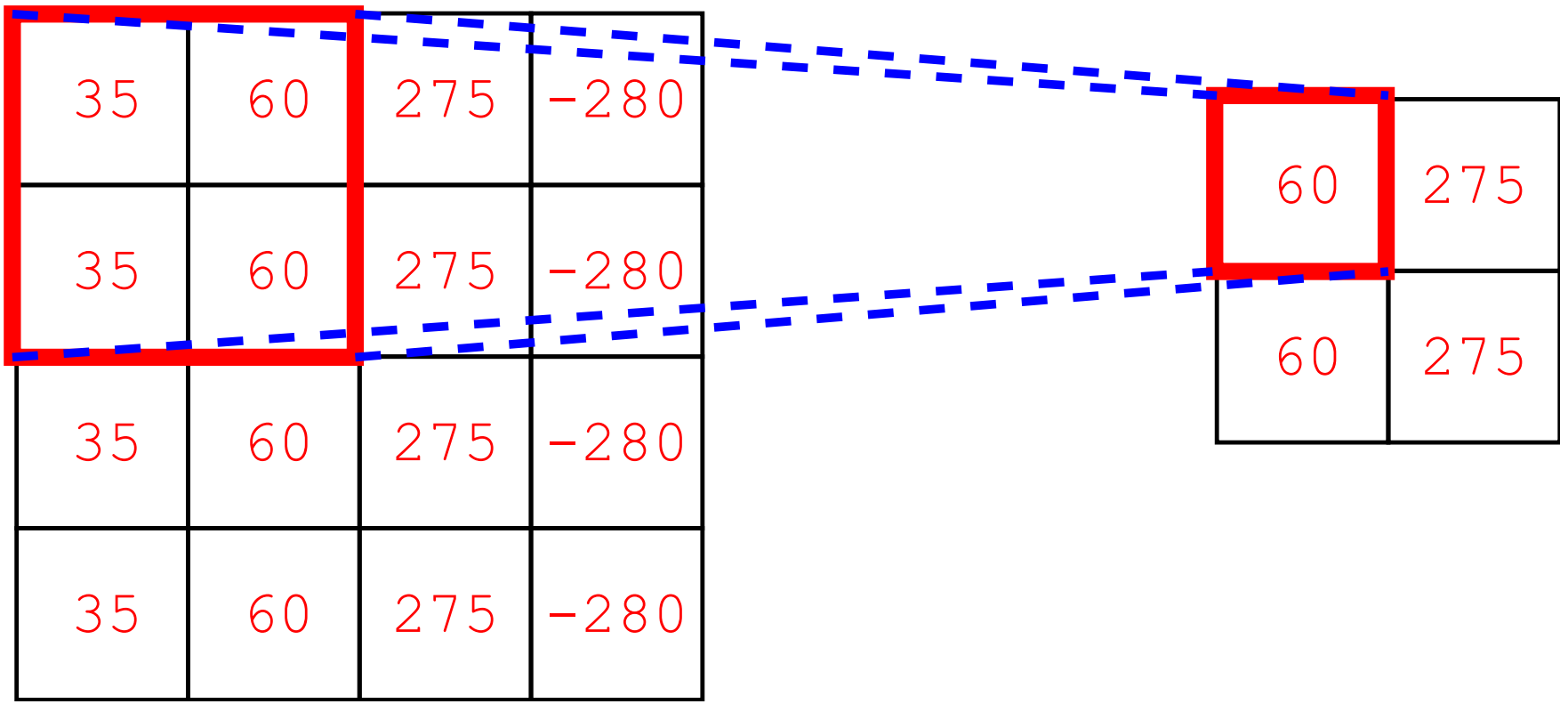


Figura: Exemplo de Max-Pooling.

Pooling

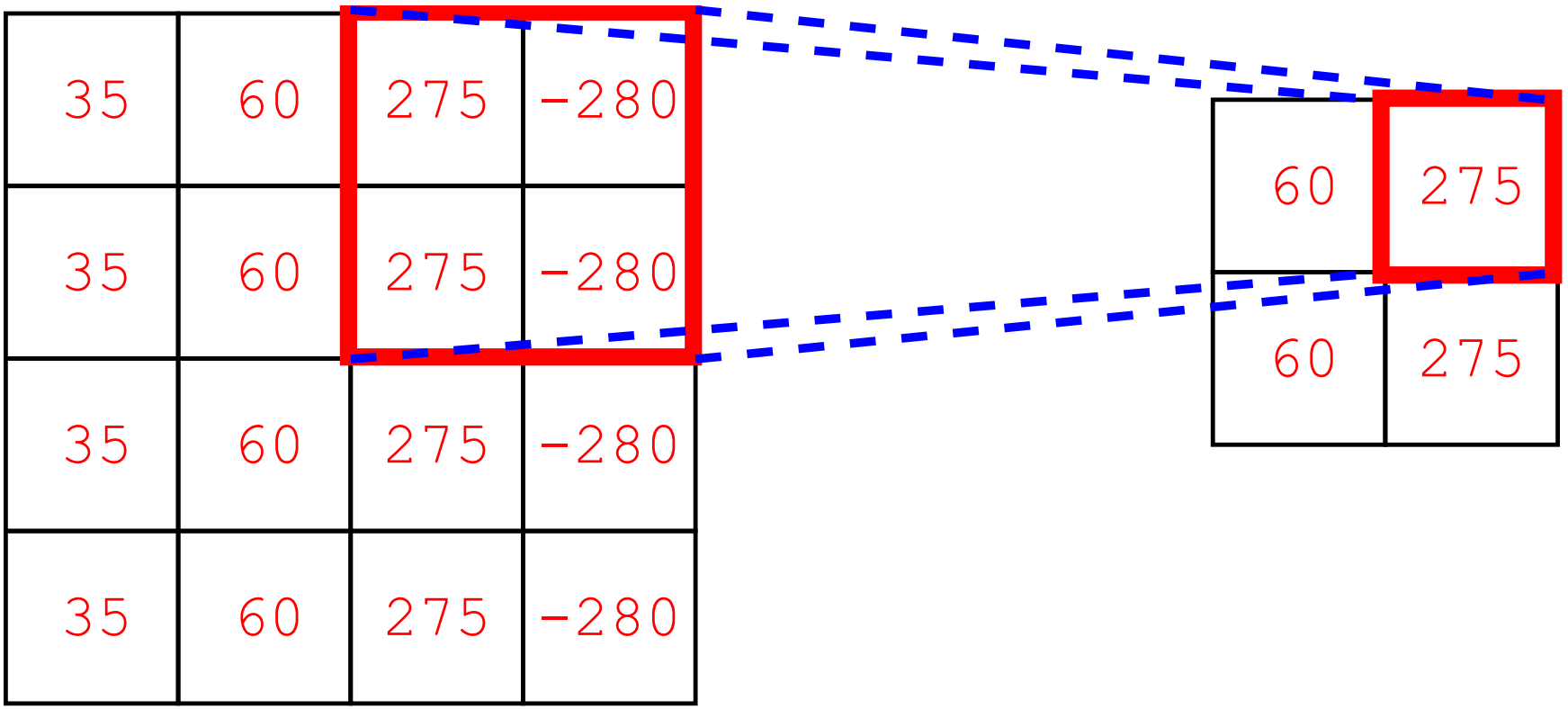


Figura: Exemplo de Max-Pooling.

Pooling

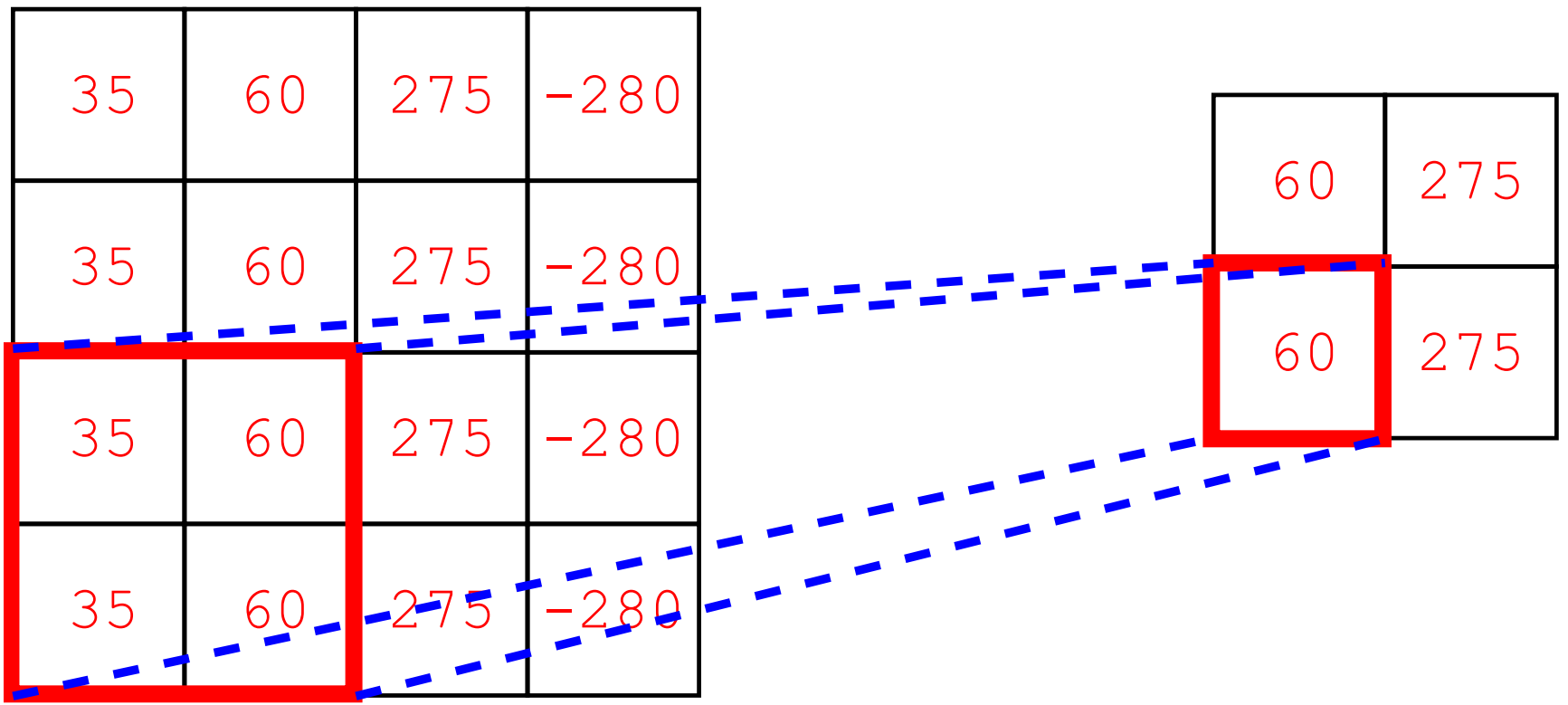


Figura: Exemplo de Max-Pooling.

Pooling

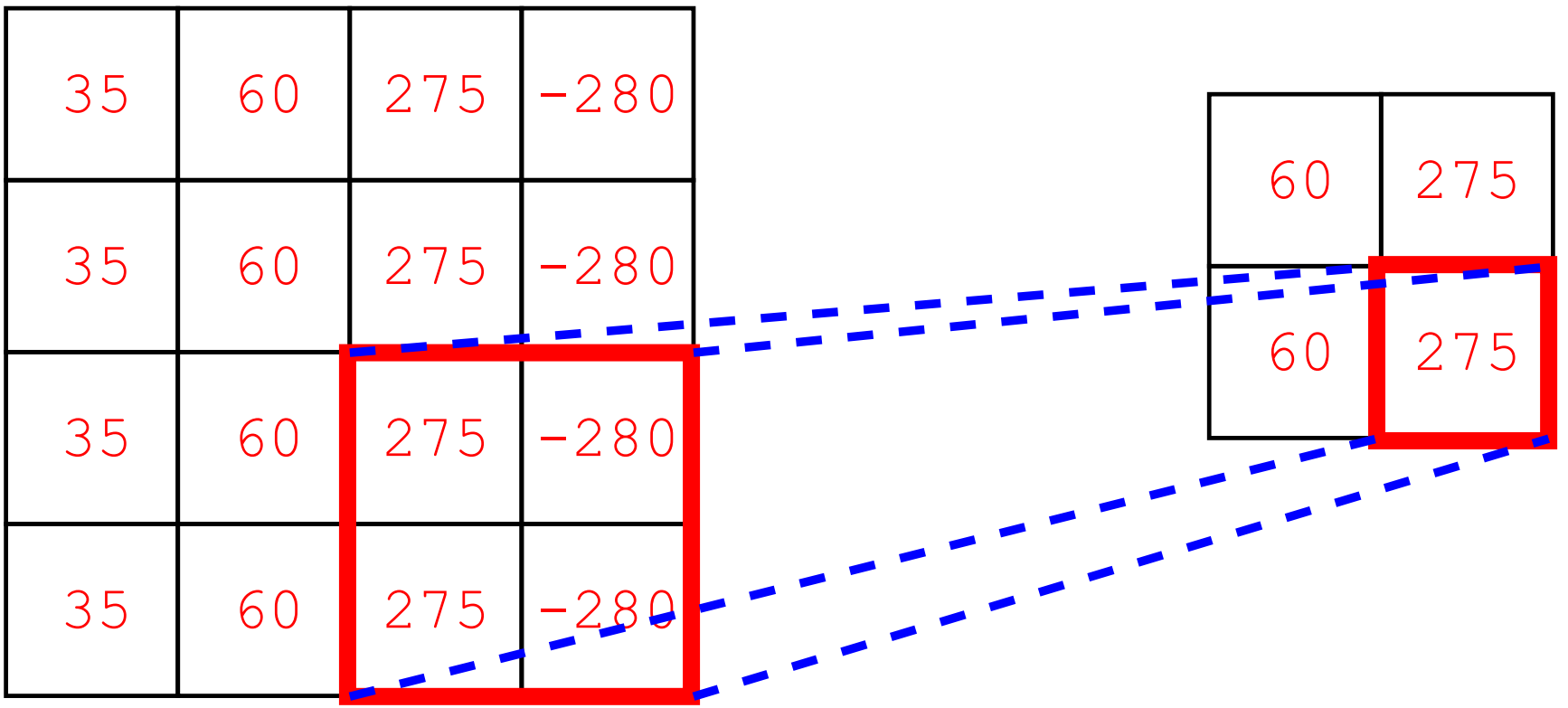


Figura: Exemplo de Max-Pooling.

Field of View do Pooling

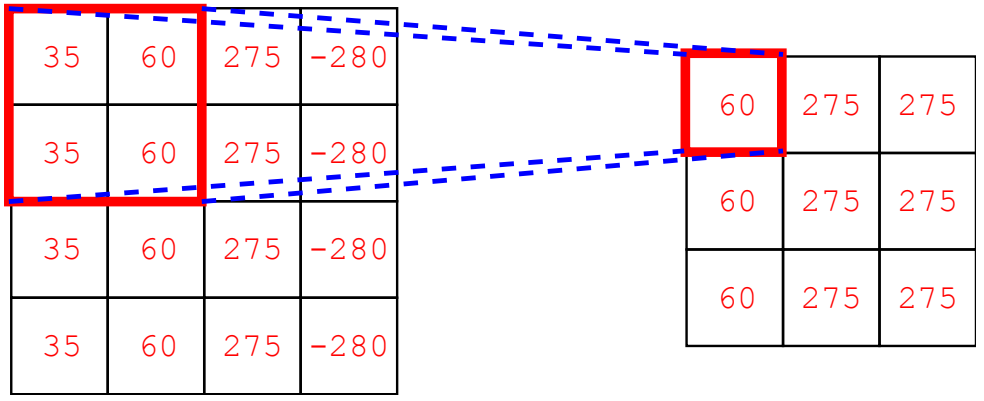


Figura: FoV de 2x2 de uma camada de Max-Pooling.

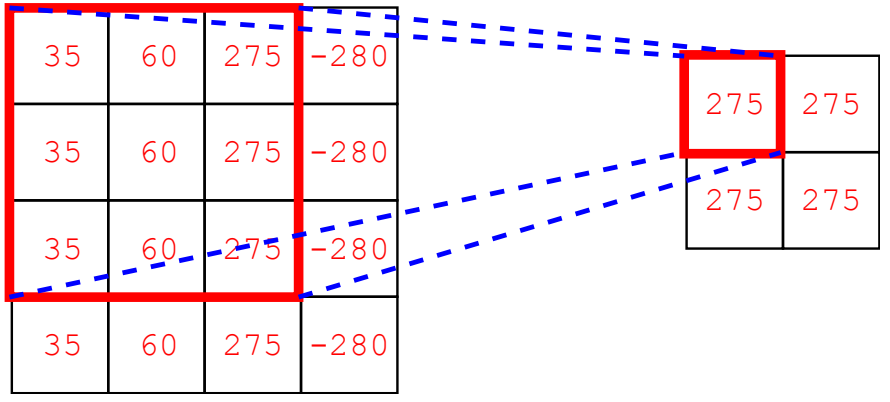
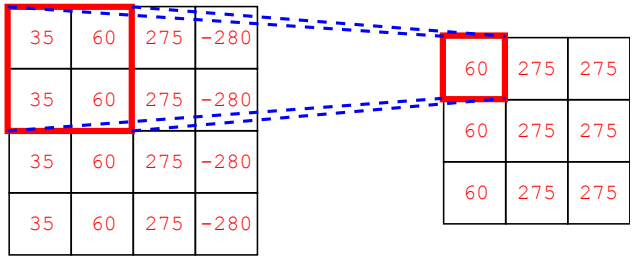
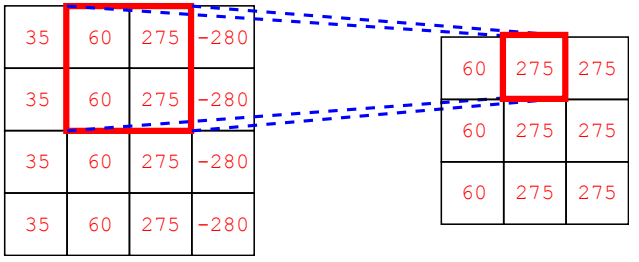


Figura: FoV de 3x3 de uma camada de Max-Pooling.

Stride do Pooling

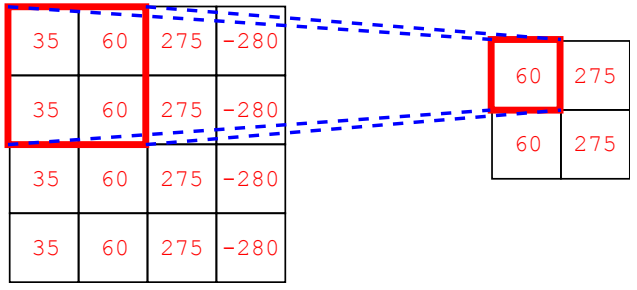


(a)

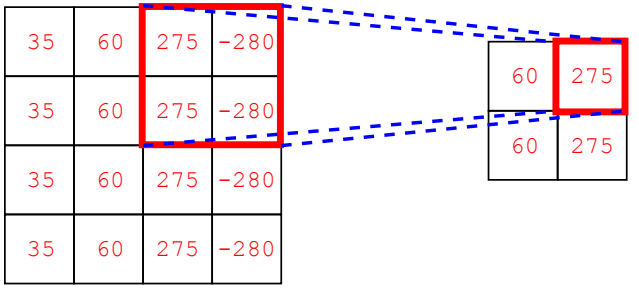


(b)

Figura: Stride de tamanho 1.



(a)



(b)

Figura: Stride de tamanho 2.

Resolução de Saída do Pooling

Calculando a Resolução Espacial de Saída

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

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

Camada de Pooling - PyTorch

Demo Pooling

Camadas-CNN.ipynb

Batch Normalization

- Vamos primeiro entender a intuição que motivou a criação desta camada.
  - O pré-processamento dos dados de entrada é uma etapa crucial.
  - Operações como padronização e normalização da entrada aceleram a convergência do treinamento. **Por que?**

Batch Normalization

- Vamos primeiro entender a intuição que motivou a criação desta camada.
  - O pré-processamento dos dados de entrada é uma etapa crucial.
  - Operações como padronização e normalização da entrada aceleram a convergência do treinamento.
  - **Com os valores na mesma escala a convergência é mais estável.**

    Ex: Imagine duas características, idade e salário. Enquanto uma varia de 0 a 100, a outra pode variar de 0 a 10.000 ou até mais!

    Escalas tão distintas dificultam a busca por relações entre elas.

Batch Normalization

- Operações como padronização e normalização da entrada aceleram a convergência do treinamento.
- Por que não fazer isso também com as ativações das camadas intermediárias?

Batch Normalization

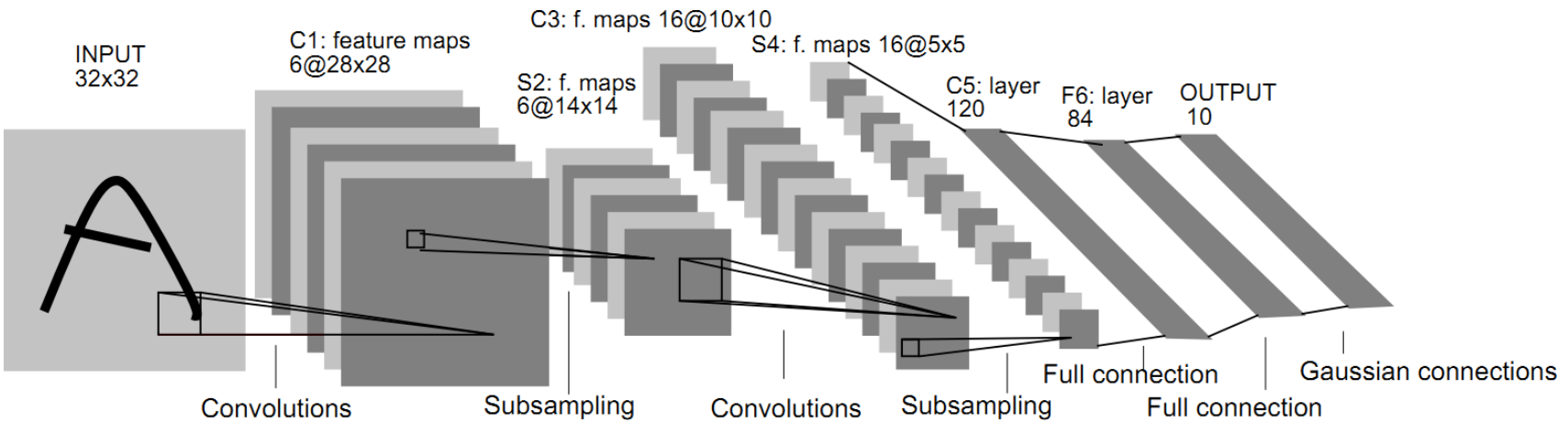


Figura: Arquitetura da LeNet [1].

Batch Normalization

- Operações como padronização e normalização da entrada aceleram a convergência do treinamento.
- Por que não fazer isso também com as ativações das camadas intermediárias?
- Os autores do batch normalization postularam que esta diferença na distribuição das ativações poderia dificultar a convergência da rede. [6]

[6] https://arxiv.org/abs/1502.03167

Batch Normalization

- Em cada iteração de treinamento, a camada de batch normalization normaliza as ativações de cada neurônio subtraindo sua média $\hat{\mu}$ e dividindo pelo seu desvio padrão $\hat{\sigma}$.

$$\mathrm{BN}(\mathbf{x}) = \gamma \odot \frac{\mathbf{x} - \hat{\mu}}{\hat{\sigma}} + \beta$$

- $\hat{\mu}$ e $\hat{\sigma}$ **são estimados com base no batch atual**.

Batch Normalization

- Em cada iteração de treinamento, a camada de batch normalization normaliza as ativações de cada neurônio subtraindo sua média $\hat{\mu}$ e dividindo pelo seu desvio padrão $\hat{\sigma}$.

$$\mathrm{BN}(\mathbf{x}) = \gamma \odot \frac{\mathbf{x} - \hat{\mu}}{\hat{\sigma}} + \beta$$

- Para dar mais liberdade à normalização, definimos um coeficiente de escala de coordenadas γ e um offset β.
- γ e β **são aprendidos durante o treinamento**.

Batch Normalization

- Em geral, o batch normalization é aplicado após a operação de convolução e antes da ativação.

```
1 ConvolutionBlock = nn.Sequential(
2     nn.Conv2d(3, 16, kernel_size=3, padding=1),
3     nn.BatchNorm2d(16),
4     nn.ReLU(),
5     nn.MaxPool2d(kernel_size=3),
6 )
```

Batch Normalization - PyTorch

Demo Batch Normalization

Camadas-CNN.ipynb

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

Arquitetura Famosas

- LeNet [1] (1998)
- AlexNet [2] (2012)
- VGG [3] (2014)
- GoogLeNet [4] (2015)
- ResNets [5] (2016)
- DenseNets [6] (2017)

Arquitetura Famosas

- AlexNet [2] – primeiro uso de CNN em datasets desafiadores
- *ImageNet Large Scale Visual Recognition Competition* (ILSVRC)
- Inovações:
  - Novas funções de ativação (ReLU)
  - Implemetação em GPU
  - Regularização via Dropout

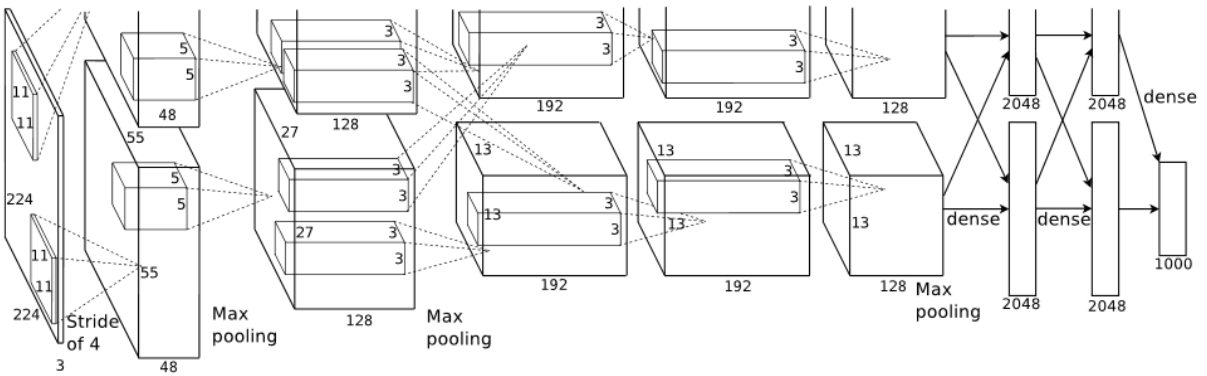


Figura: Arquitetura da AlexNet. Fonte: [2].

Arquitetura Famosas

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

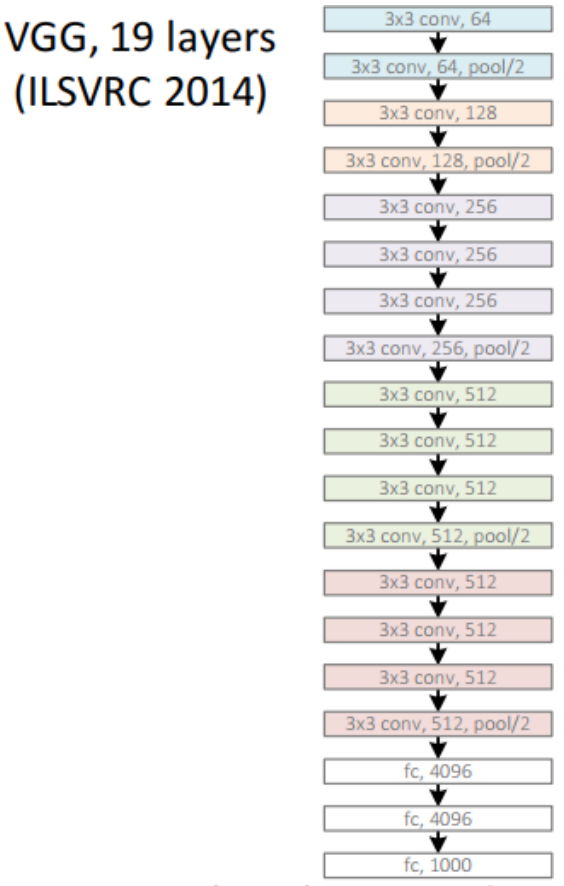


Figura: Arquitetura da VGG. Fonte: [5].

Arquitetura Famosas

- GoogLeNet [4] – módulos de convoluções paralelas (Inception)
- Combinam convoluções com diferentes tamanhos de kernel
- Redes mais profundas possuem representação semântica de mais alto nível

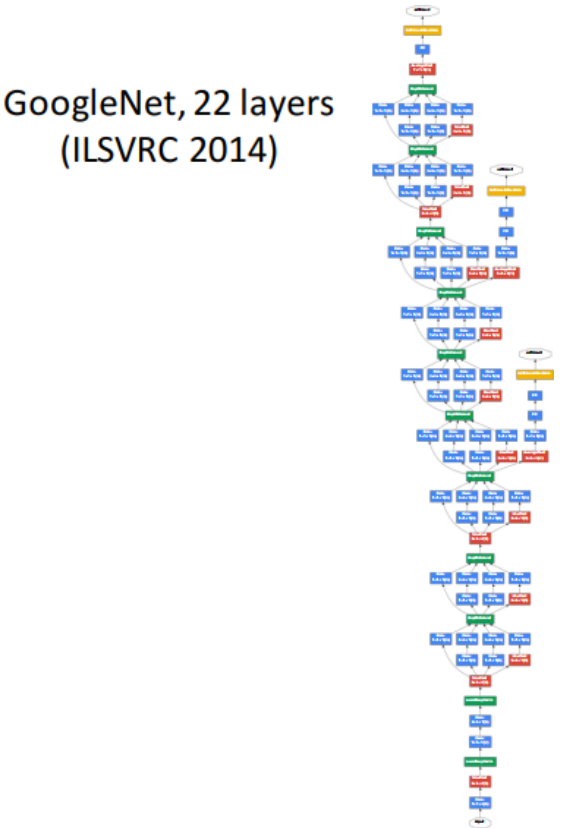


Figura: Arquitetura da GoogLeNet. Fonte: [4].

Arquitetura Famosas

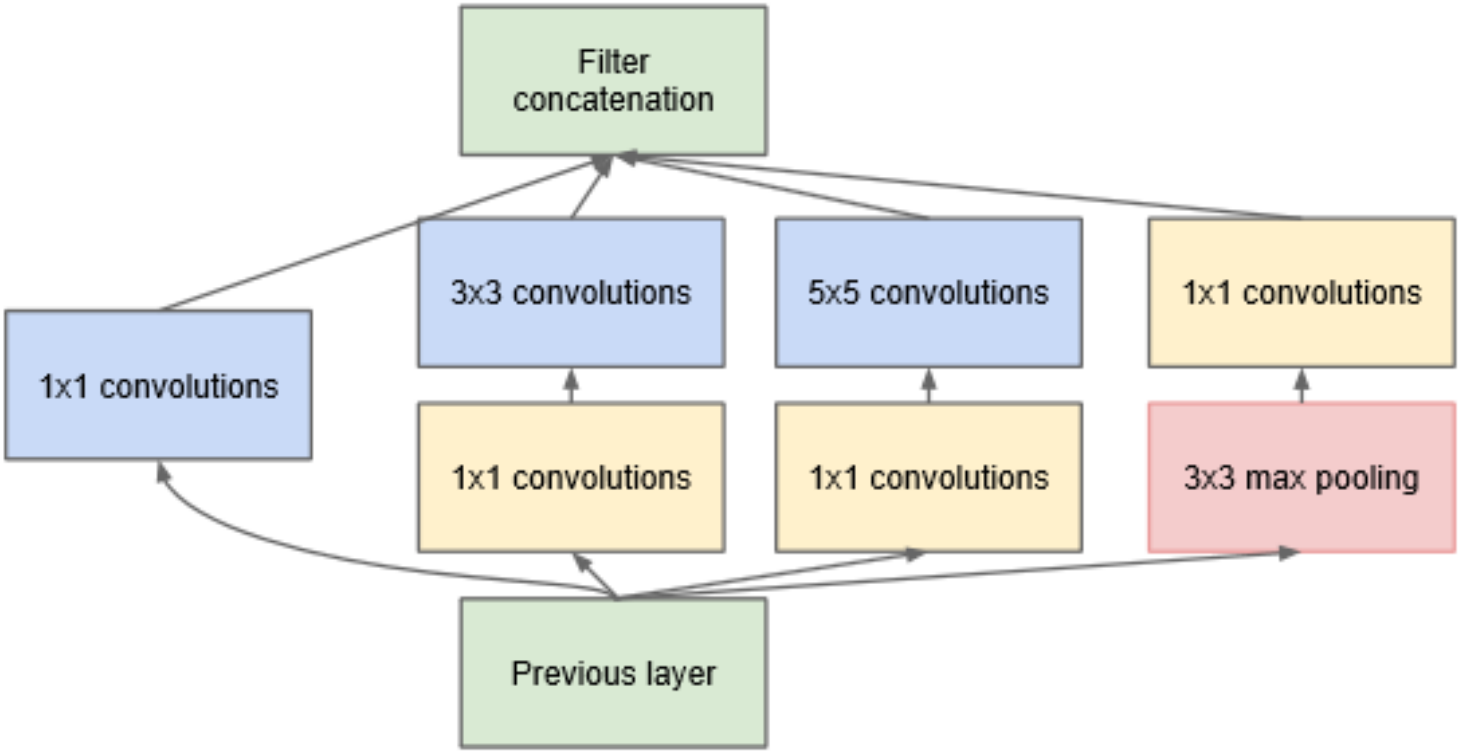


Figura: Módulo Inception da GoogLeNet [4]

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?

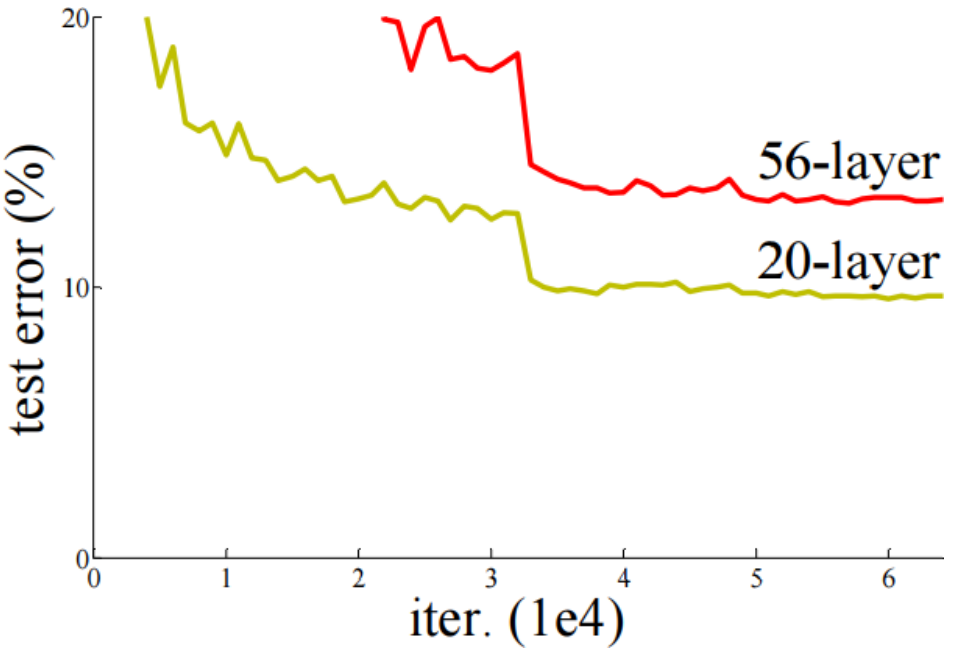


Figura: Número de Camadas vs. Erro no conjunto de teste. Fonte: [5].

Vanishing Gradient

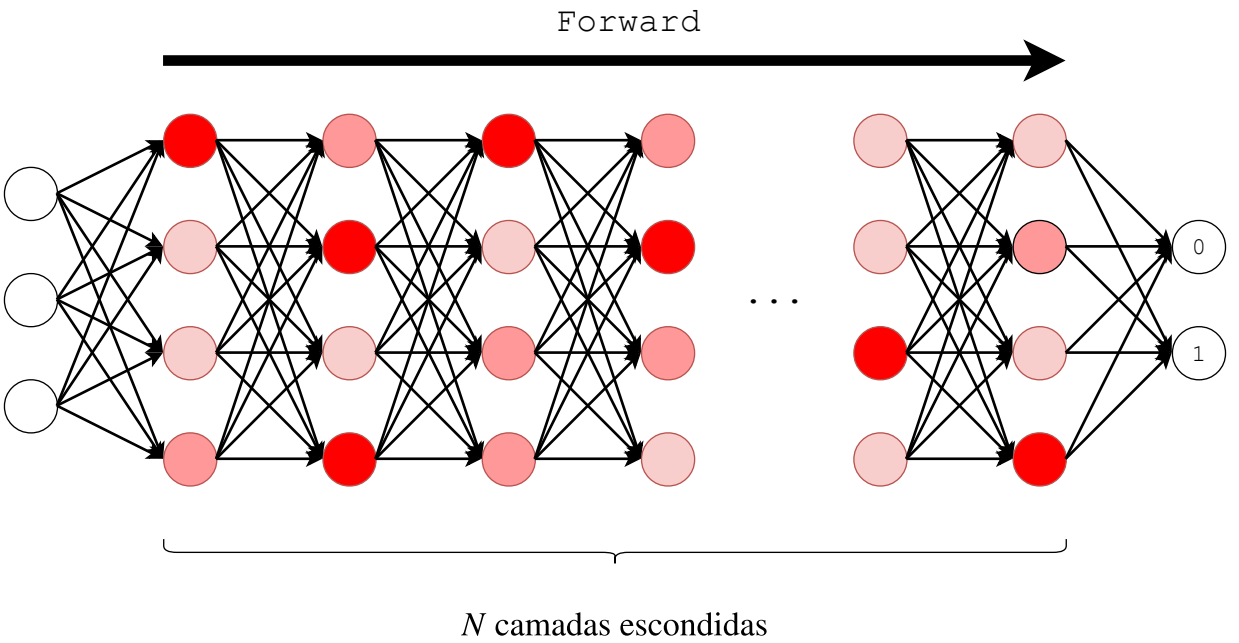


Figura: Exemplo do problema do Vanishing Gradient.

Vanishing Gradient

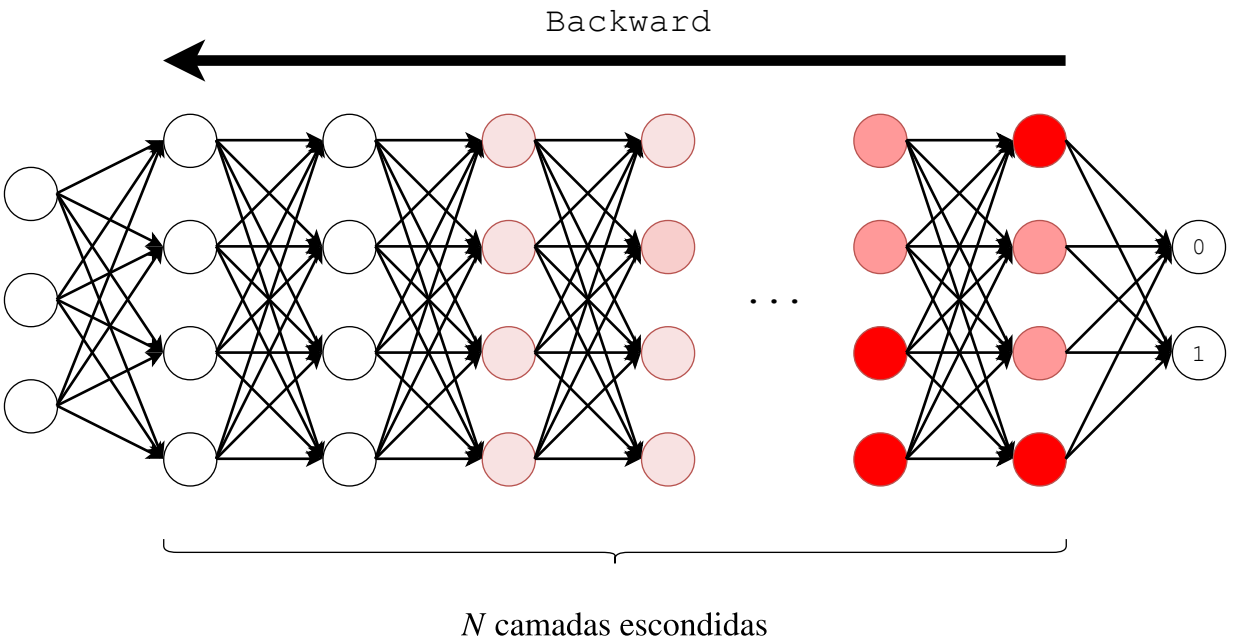


Figura: Exemplo do problema do Vanishing Gradient.

Arquitetura Famosas

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

Arquitetura Famosas

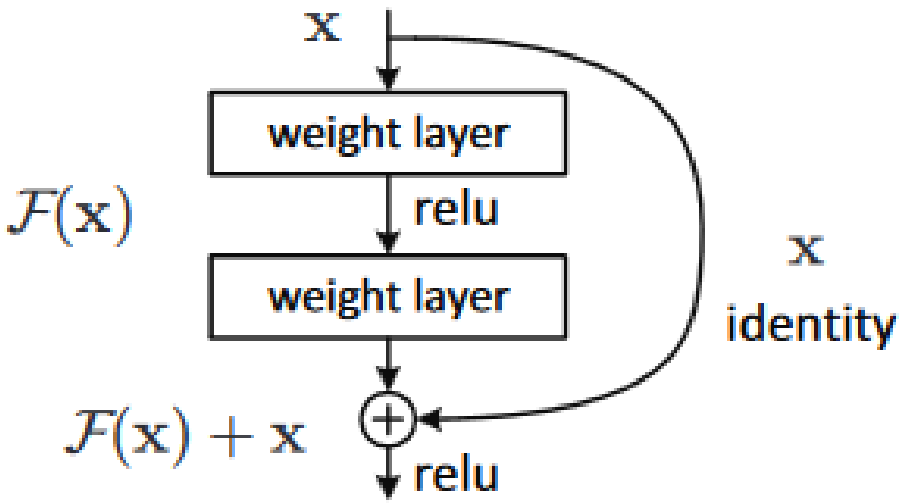
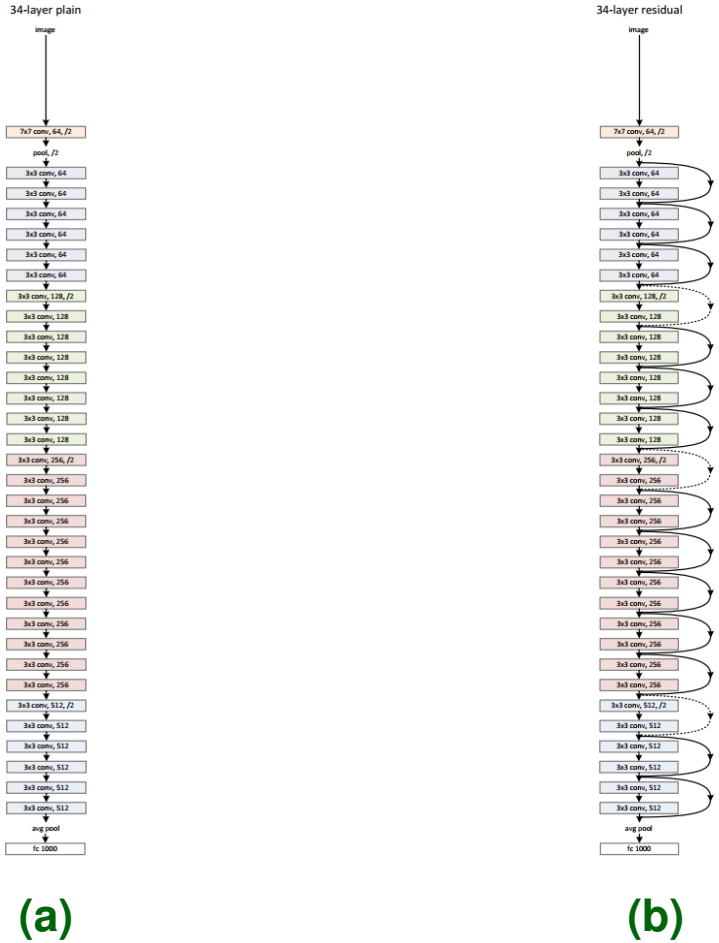


Figura: Conexões residuais das ResNets. Fonte: [5].

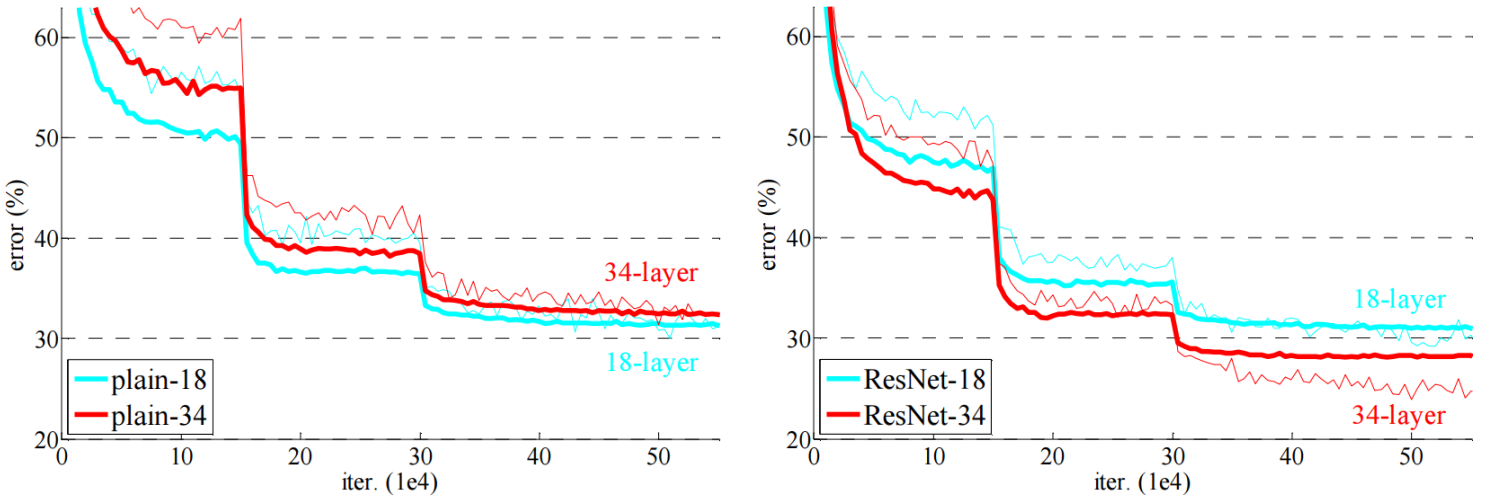
Arquitetura Famosas



(a) **(b)**

Figura: Arquitetura de (a) uma CNN comum e (b) uma ResNet com 34 camadas. Fonte: [5].

Arquitetura Famosas



(a) **(b)**

Figura: CNNs comuns (a) vs. ResNets (b). Fonte: [5].

Arquitetura Famosas

Eficiência de Parâmetros

Foi observado [5] 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 [7, 8, 9] começaram a ser propostas.

Arquitetura Famosas

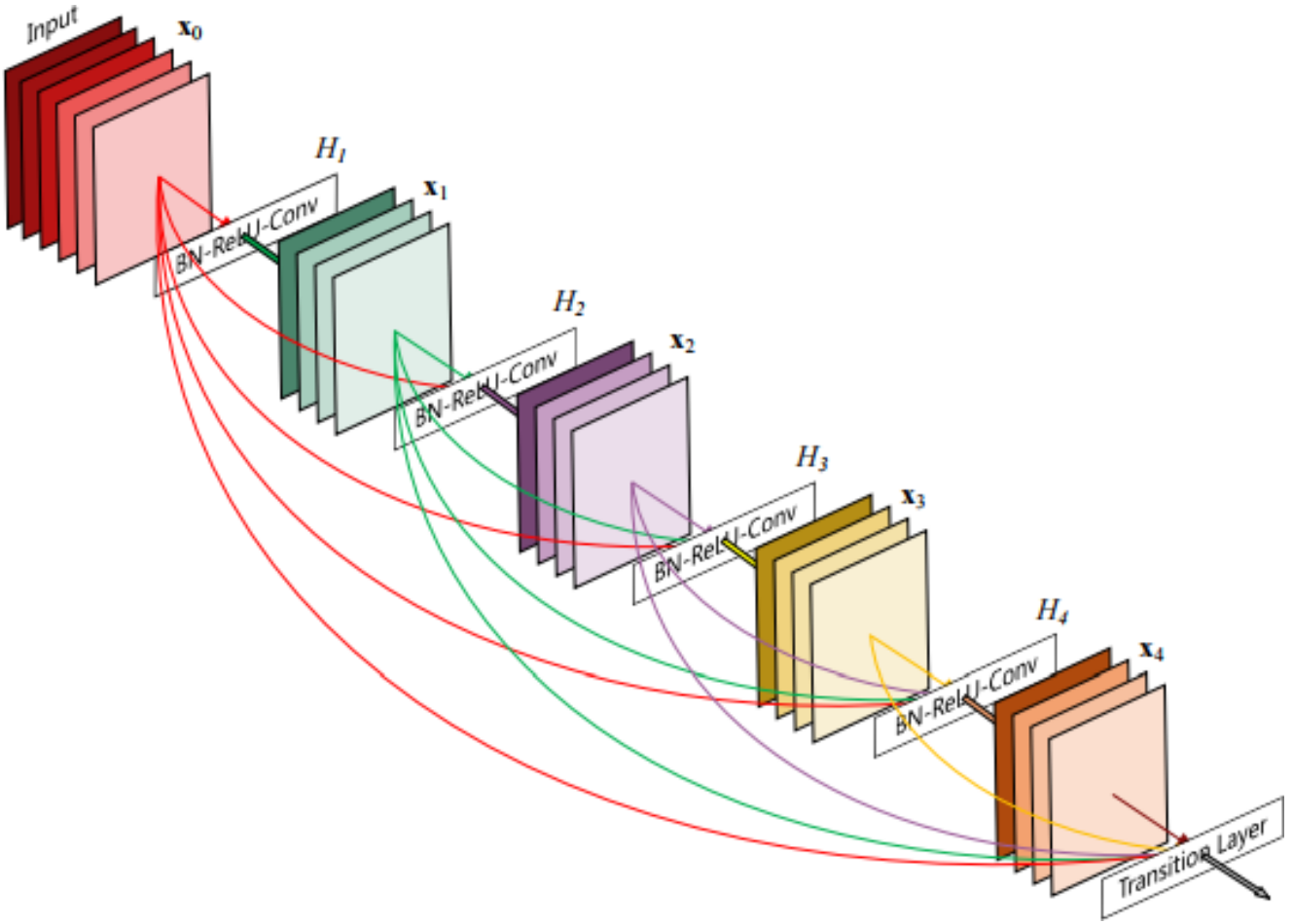


Figura: Bloco denso de uma DenseNet. Fonte: [6].

Arquitetura Famosas

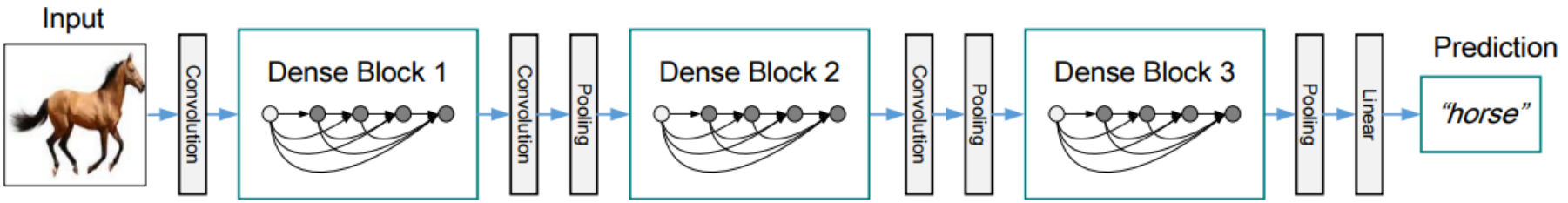


Figura: Arquitetura de uma DenseNet. Fonte: [6].

Arquitetura Famosas

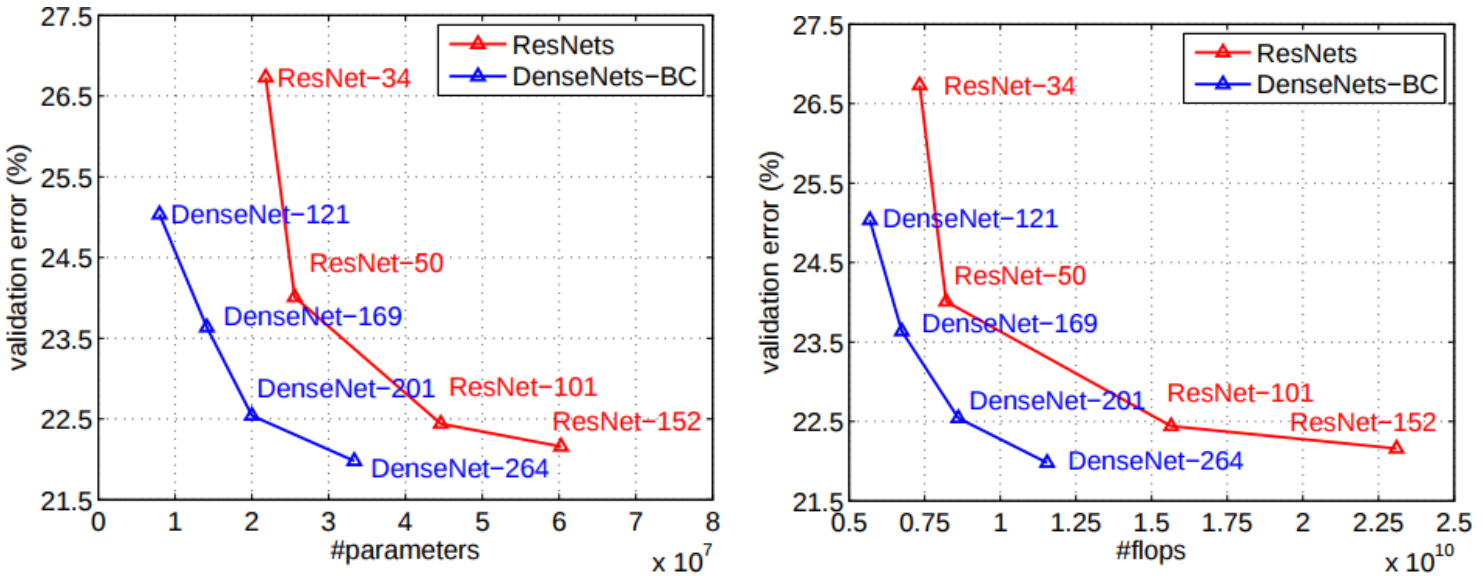


Figura: ResNets vs. DenseNets. Fonte: [6].

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

Estratégias de Treino

- Treinamento completo
- Extração de features
- Fine-Tuning

Treinamento Completo

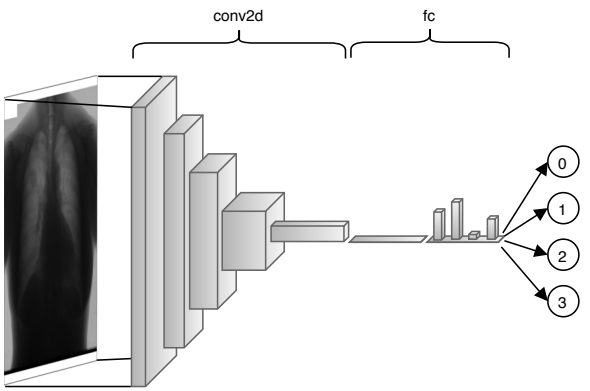


Figura: Treinamento completo de uma CNN.

Extração de Features

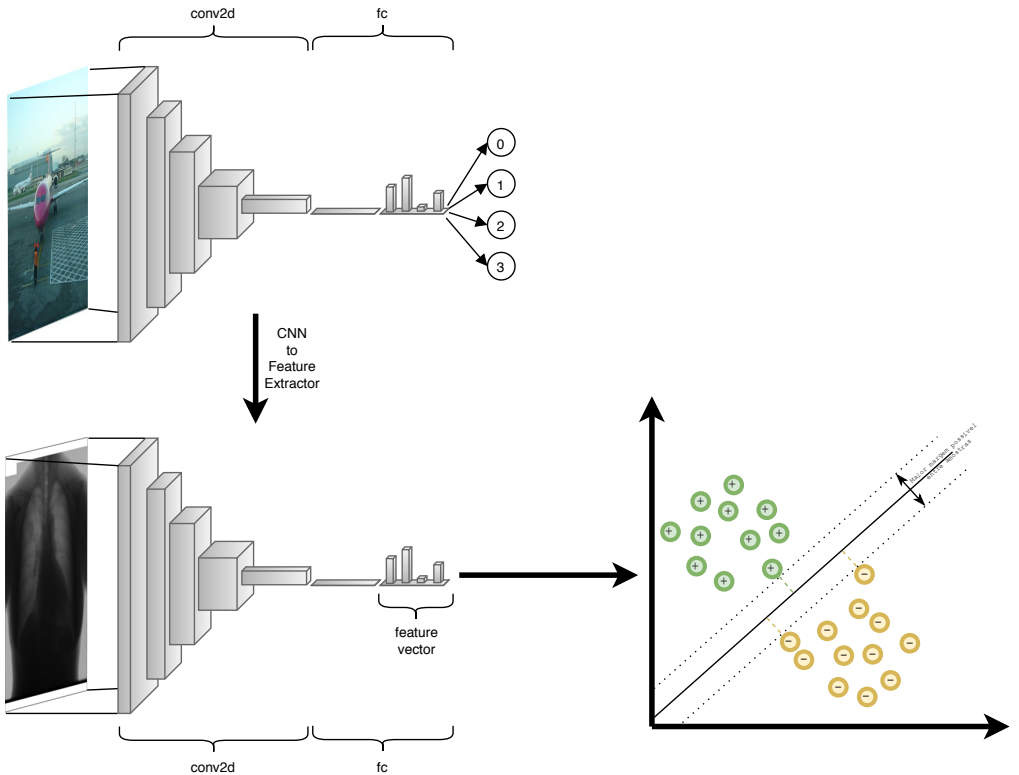


Figura: Extração de features usando uma CNN.

Fine-Tuning

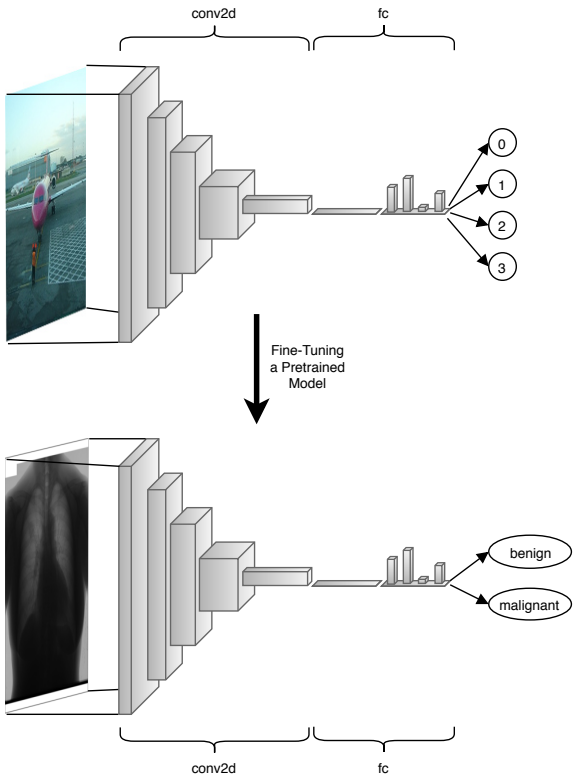


Figura: Fine-tuning em uma CNN.

Fine-Tuning

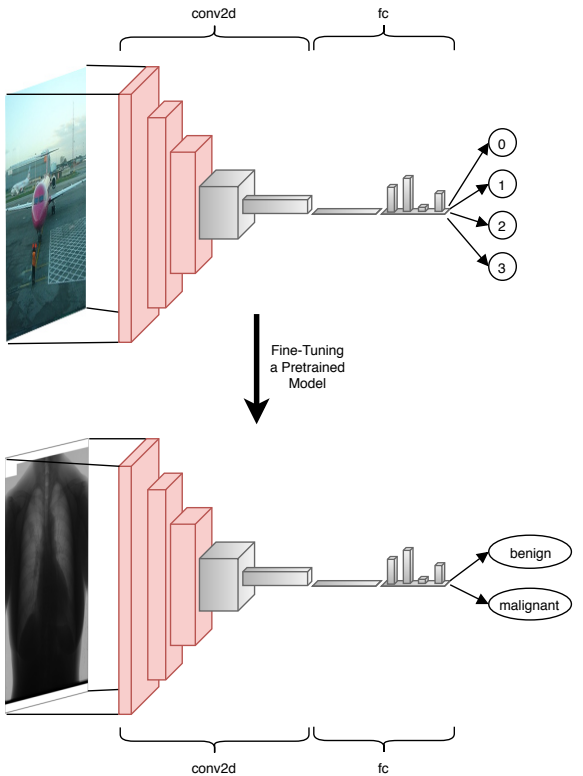


Figura: Fine-tuning em uma CNN.

Fine-Tuning

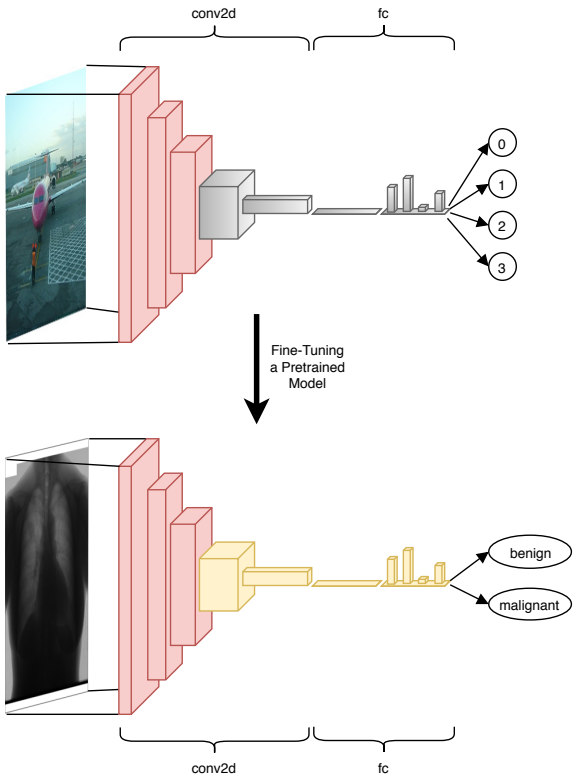


Figura: Fine-tuning em uma CNN.

Transfer Learning

Demo - Transfer Learning

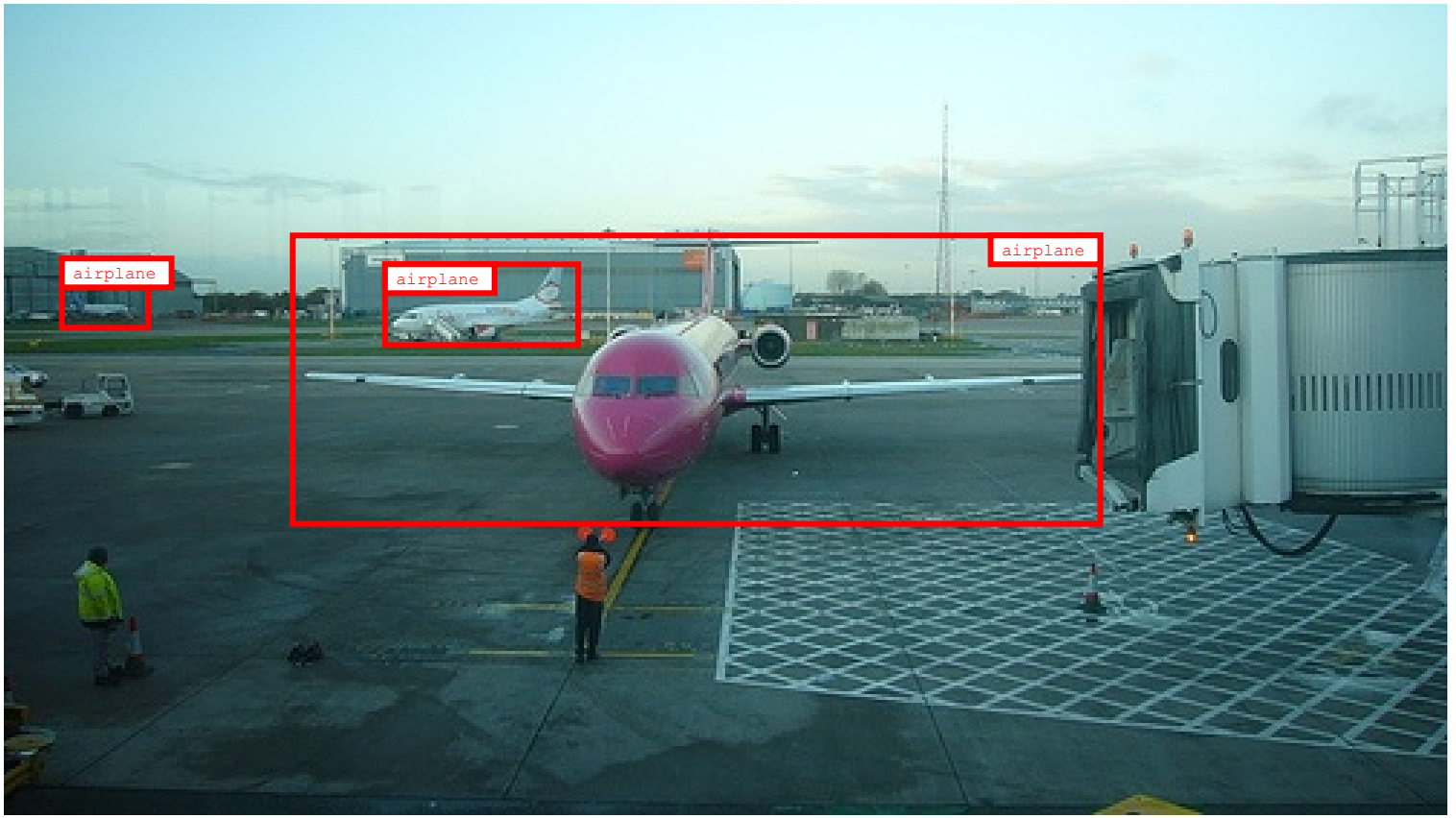
Transfer_Learning.ipynb

Agenda

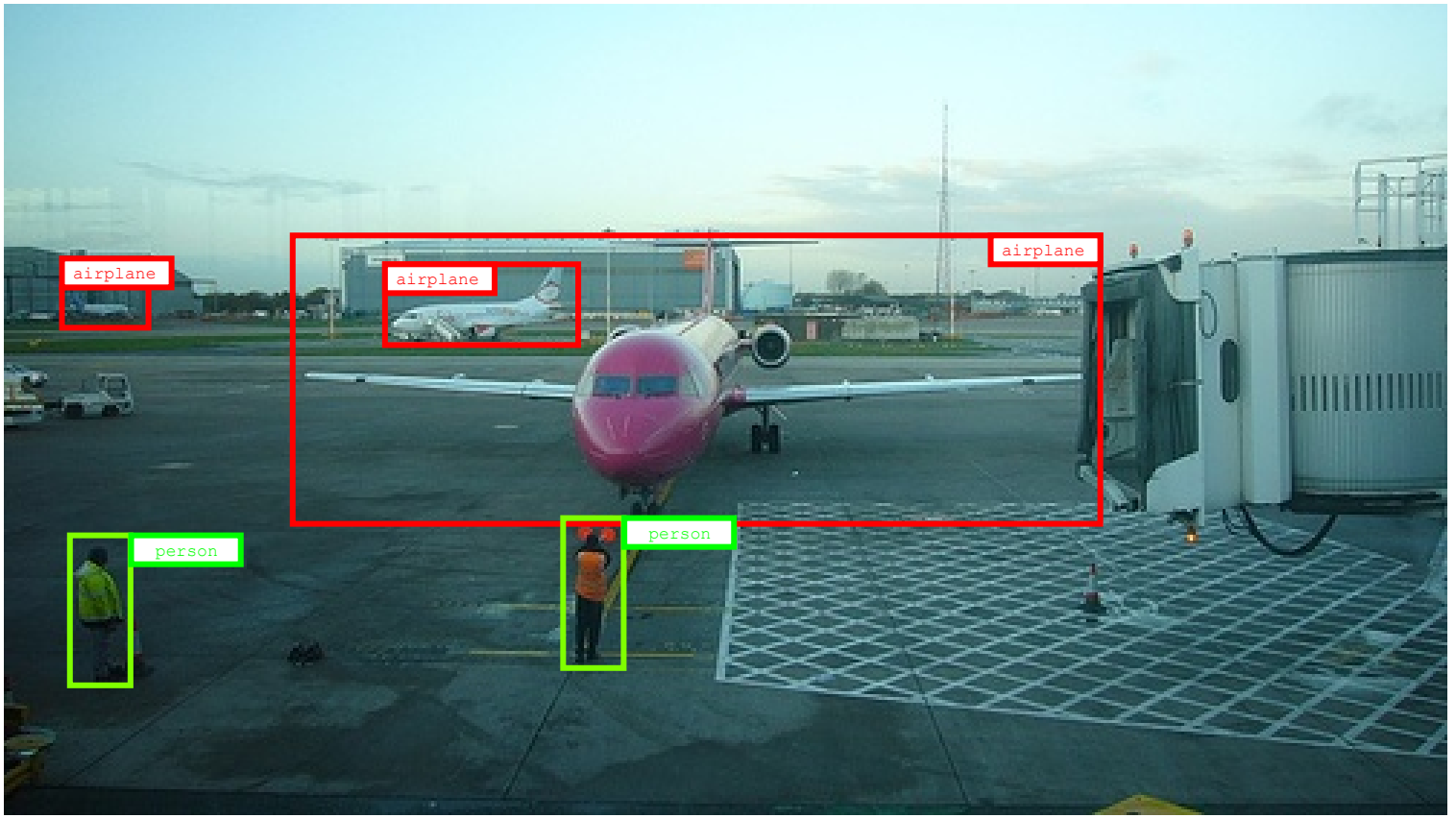
Detecção em imagens

Figura: Exemplo de Detecção em imagens.

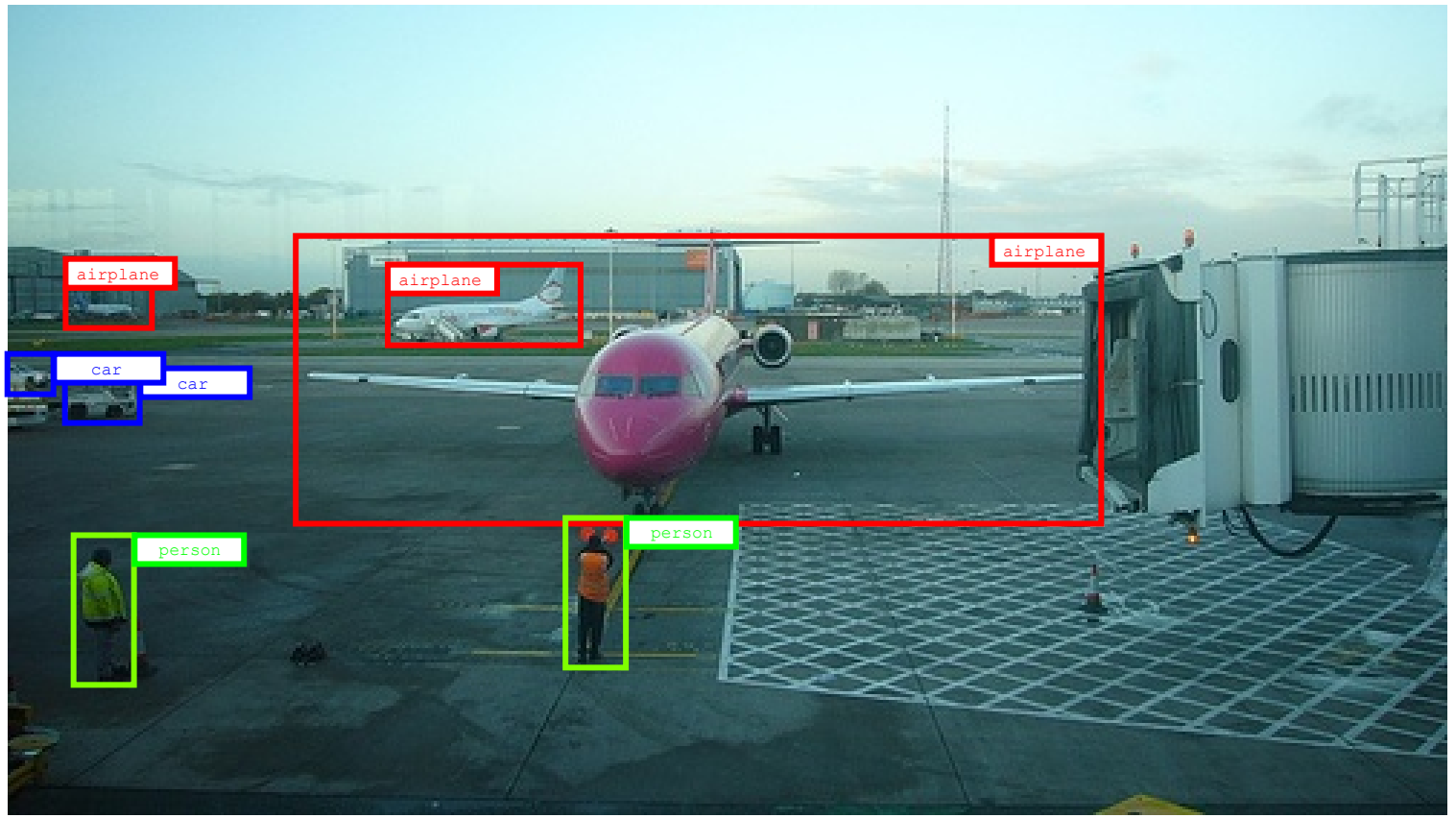
Detecção em imagens



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

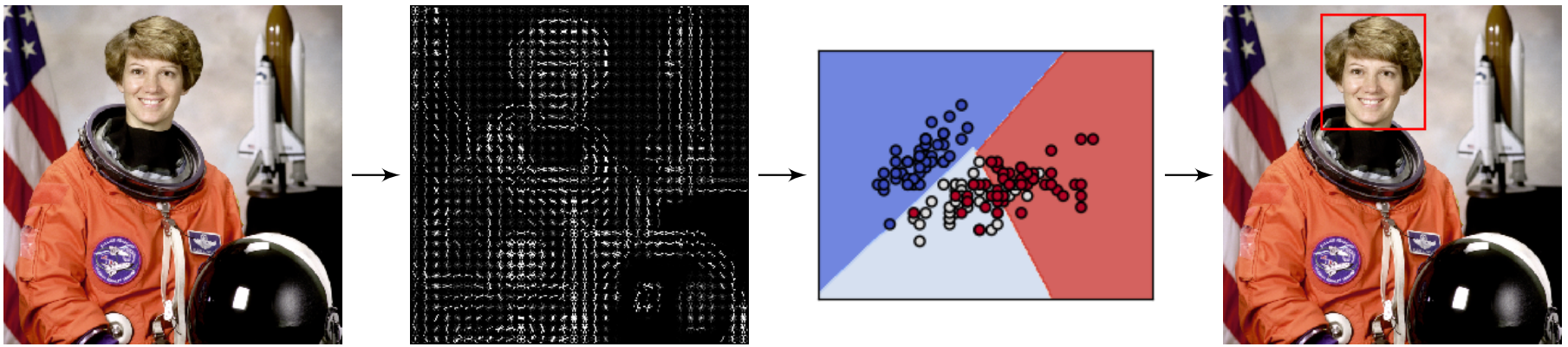


Figura: Pipeline padrão de um algoritmo de Machine Learning clássico.

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

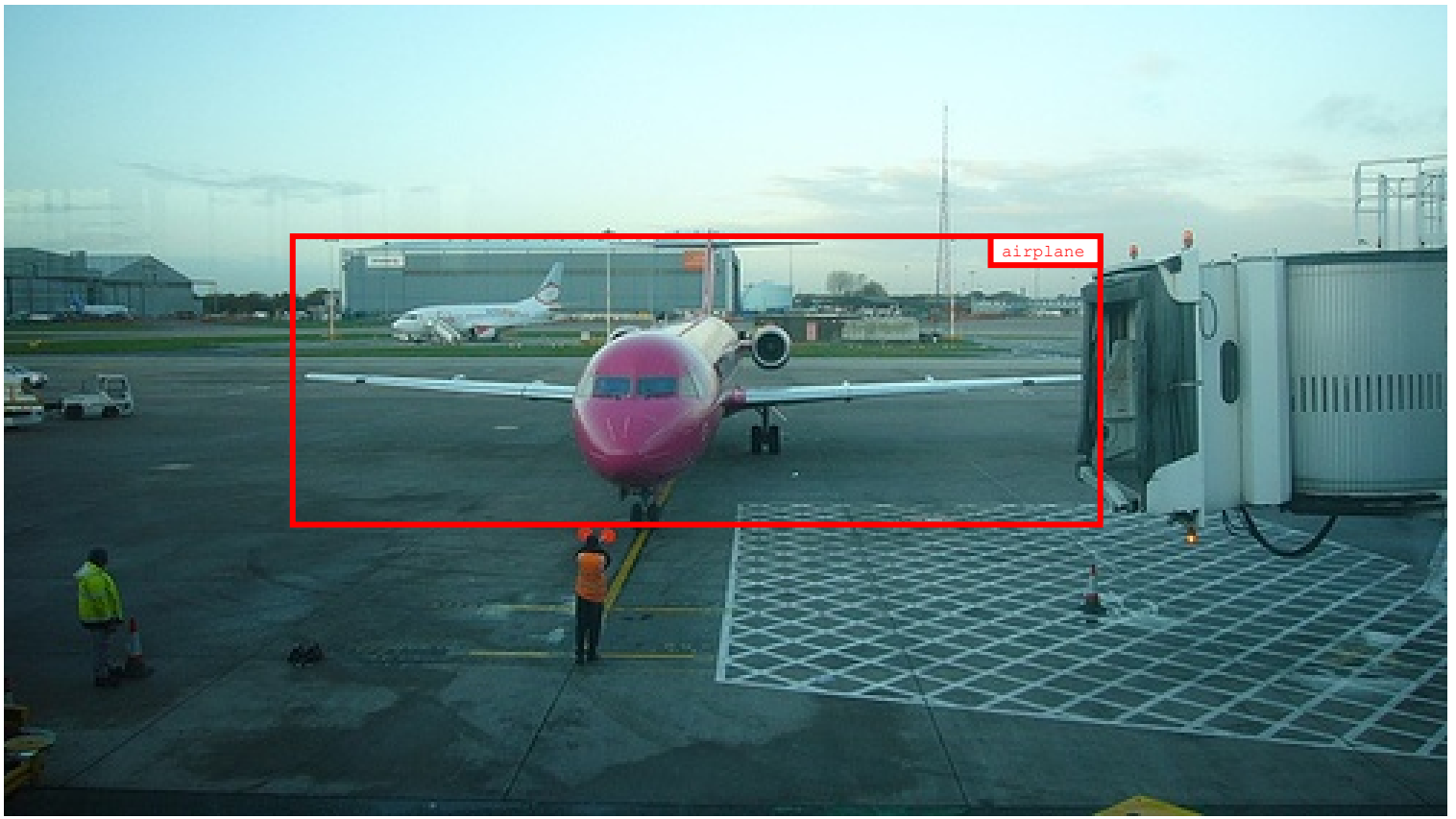


Figura: Parâmetros dos Bounding Boxes.

Detecção Deep

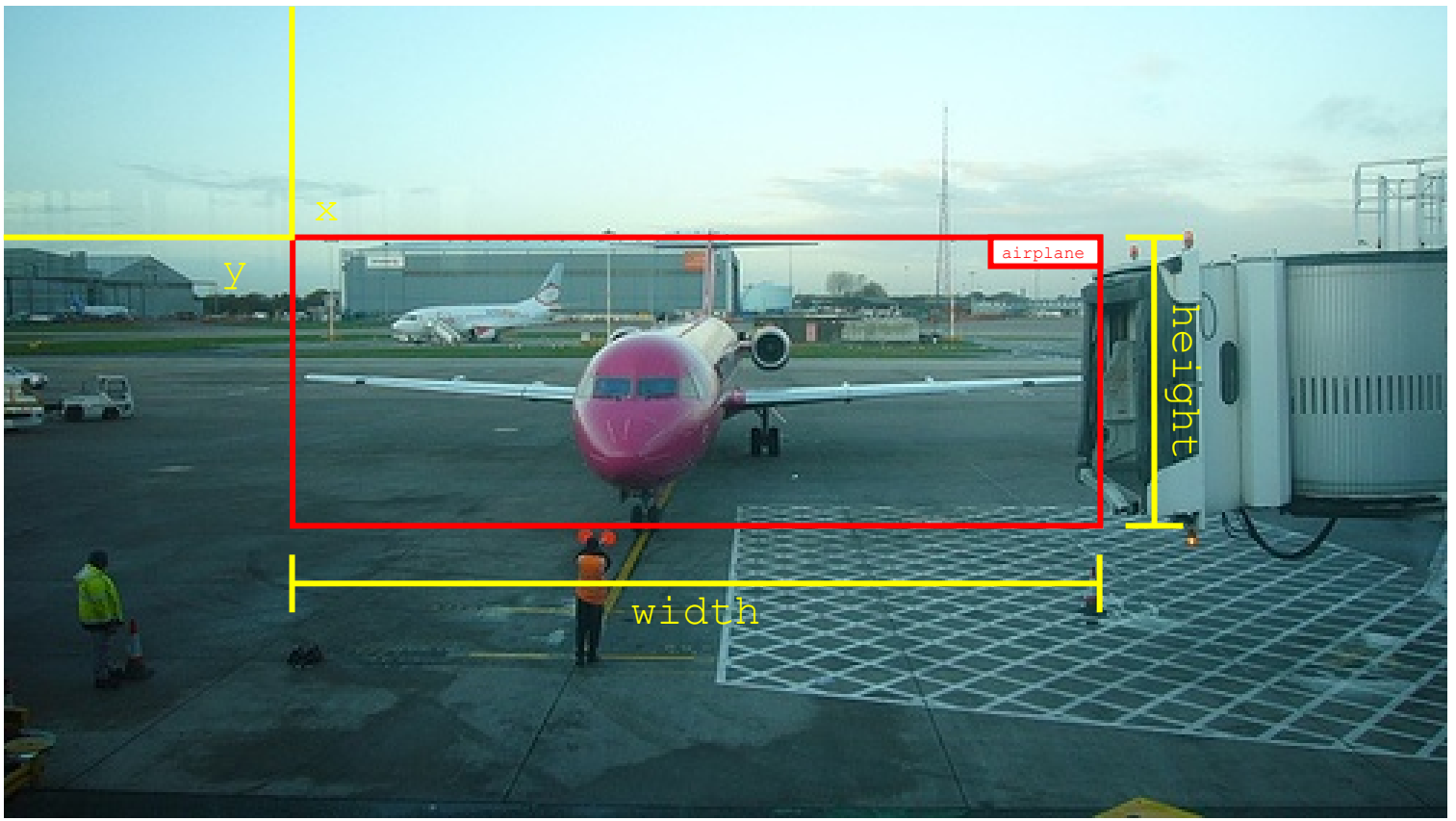


Figura: Parâmetros dos Bounding Boxes.

Detecção Deep

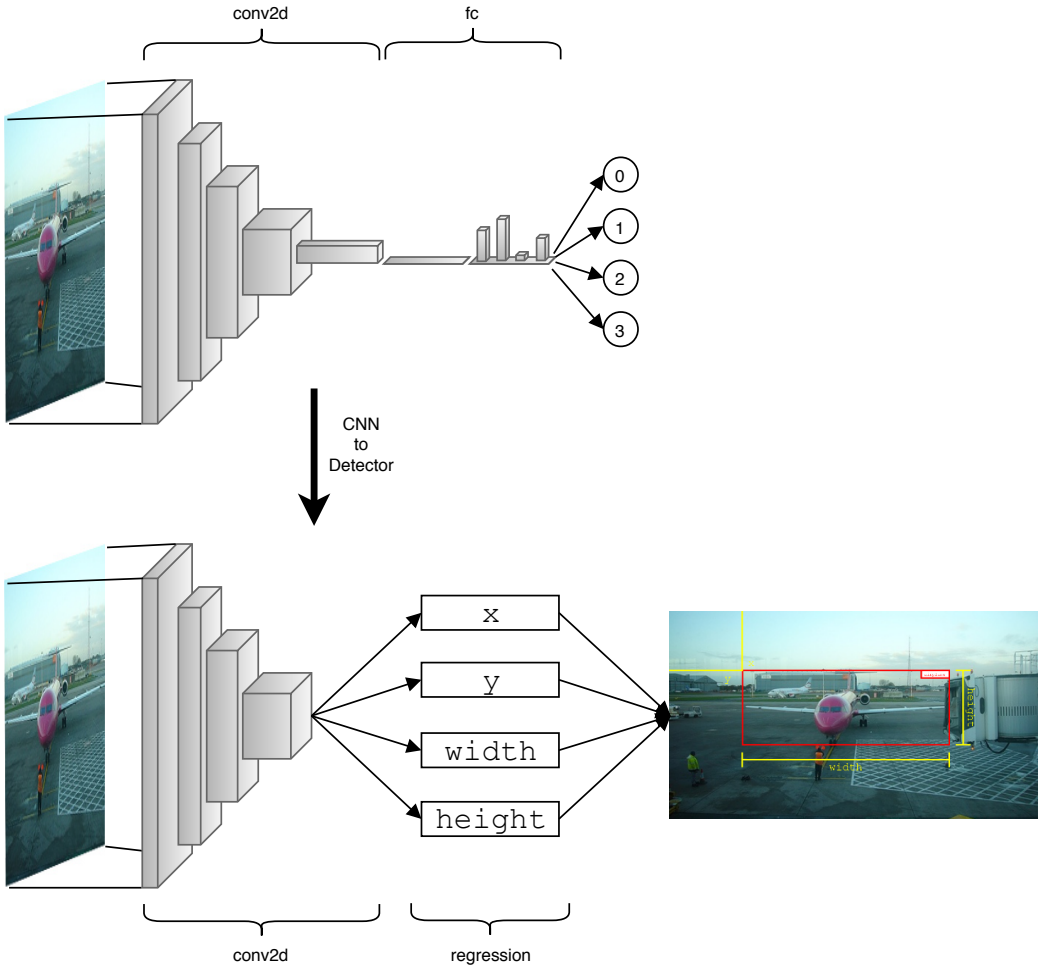


Figura: Pipeline básico de uma estratégia Deep para detecção em imagens.

Detecção Deep em Imagens

- Regional CNNs (R-CNNs) [10]
- YOLO [11]

R-CNNs

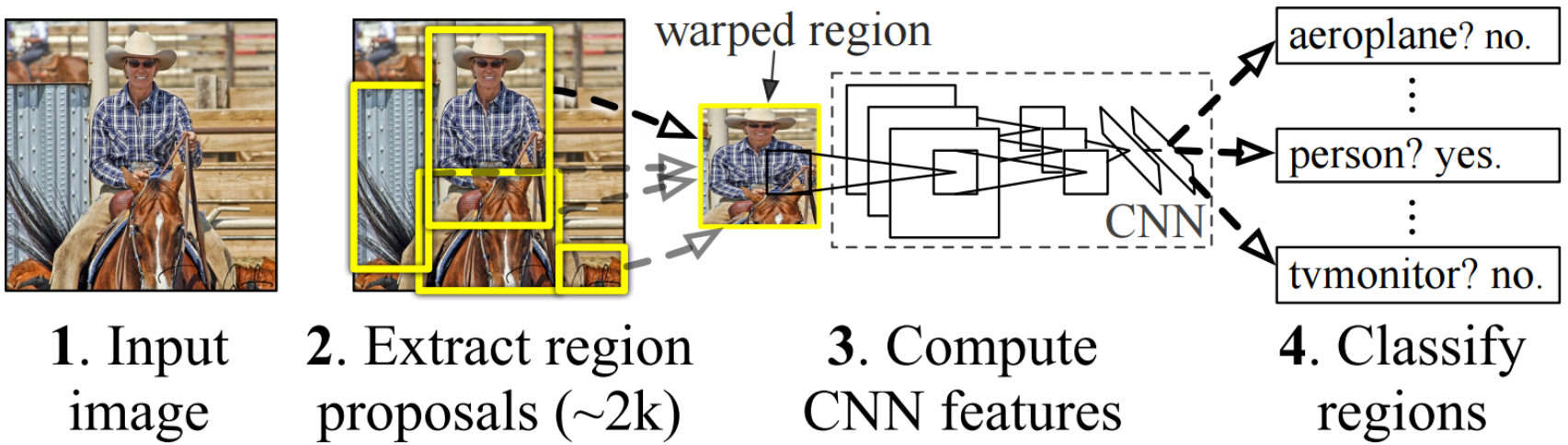


Figura: Pipeline de uma R-CNN. Fonte: [10].

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

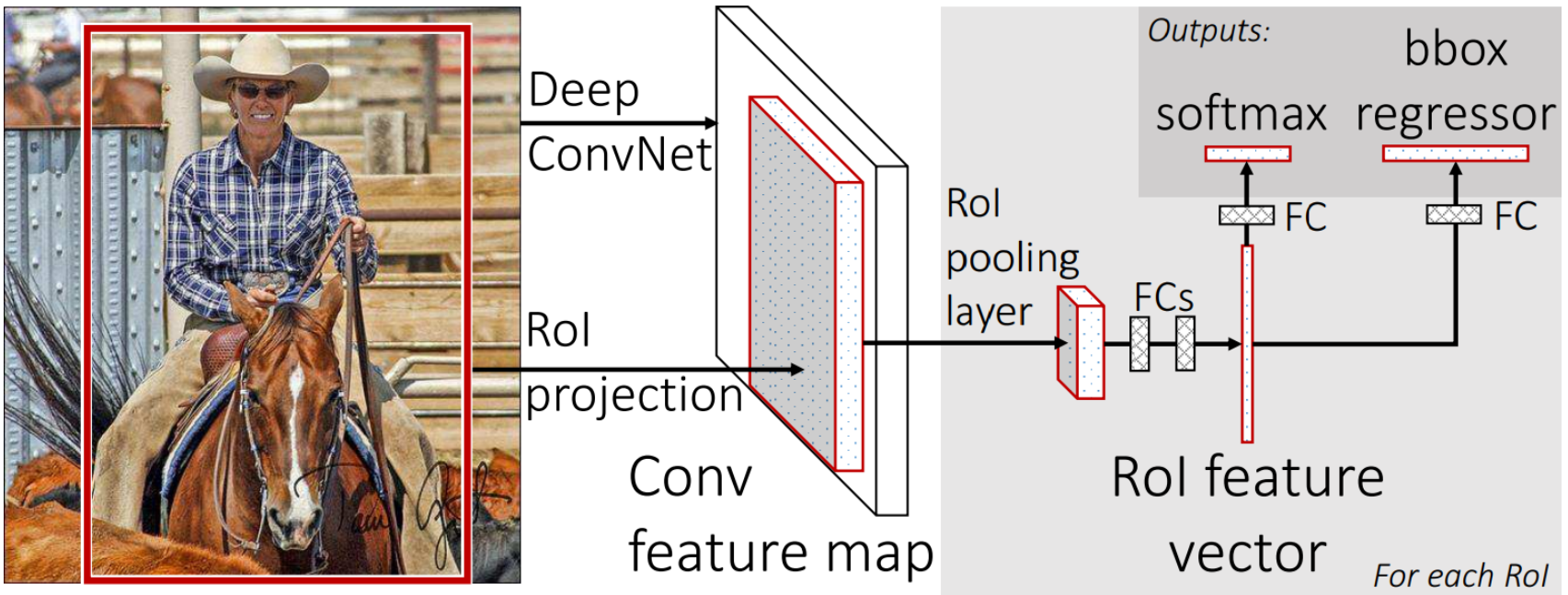
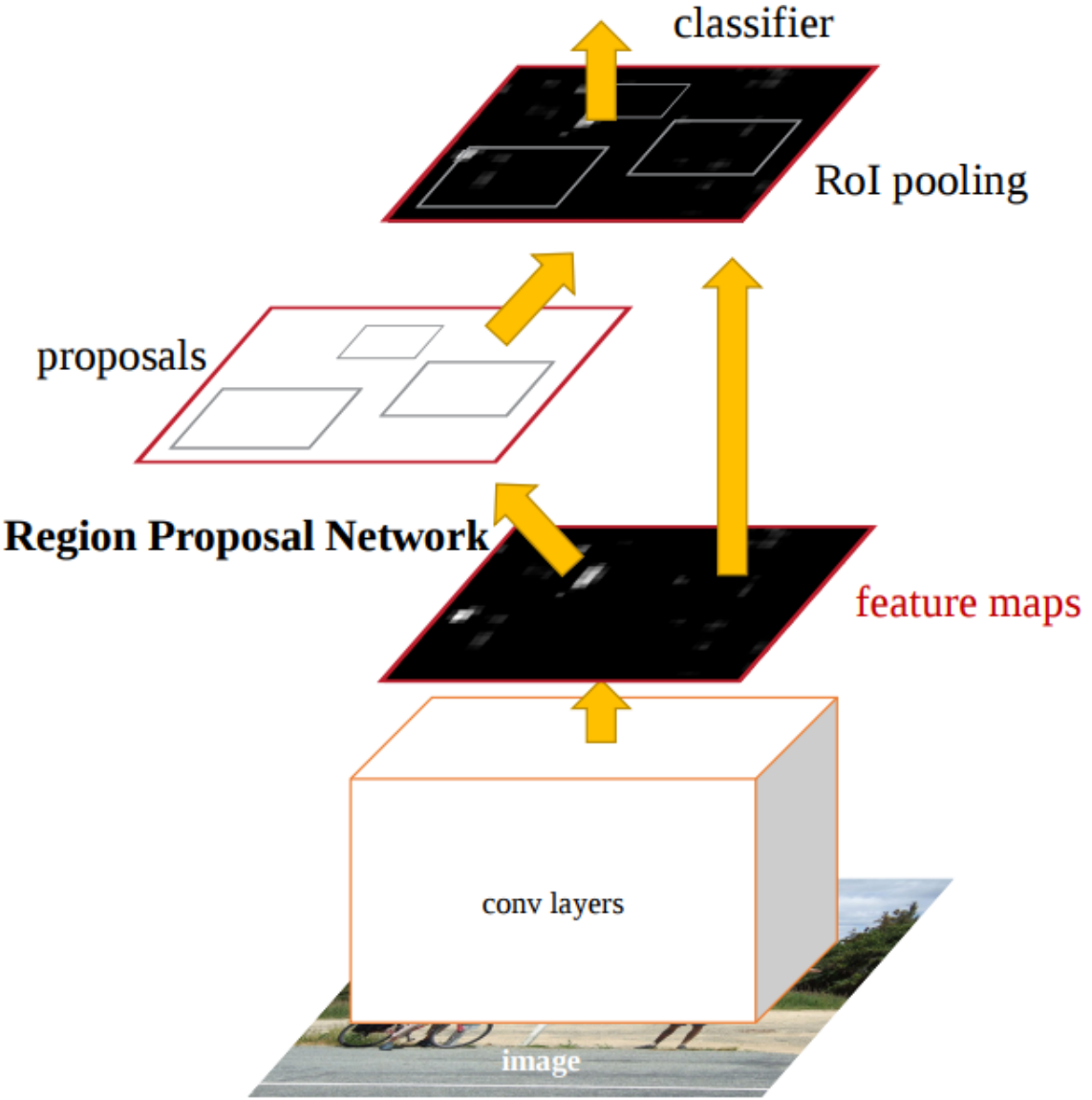


Figura: Pipeline de uma Fast R-CNN. Fonte: [12].

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)

Agenda

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

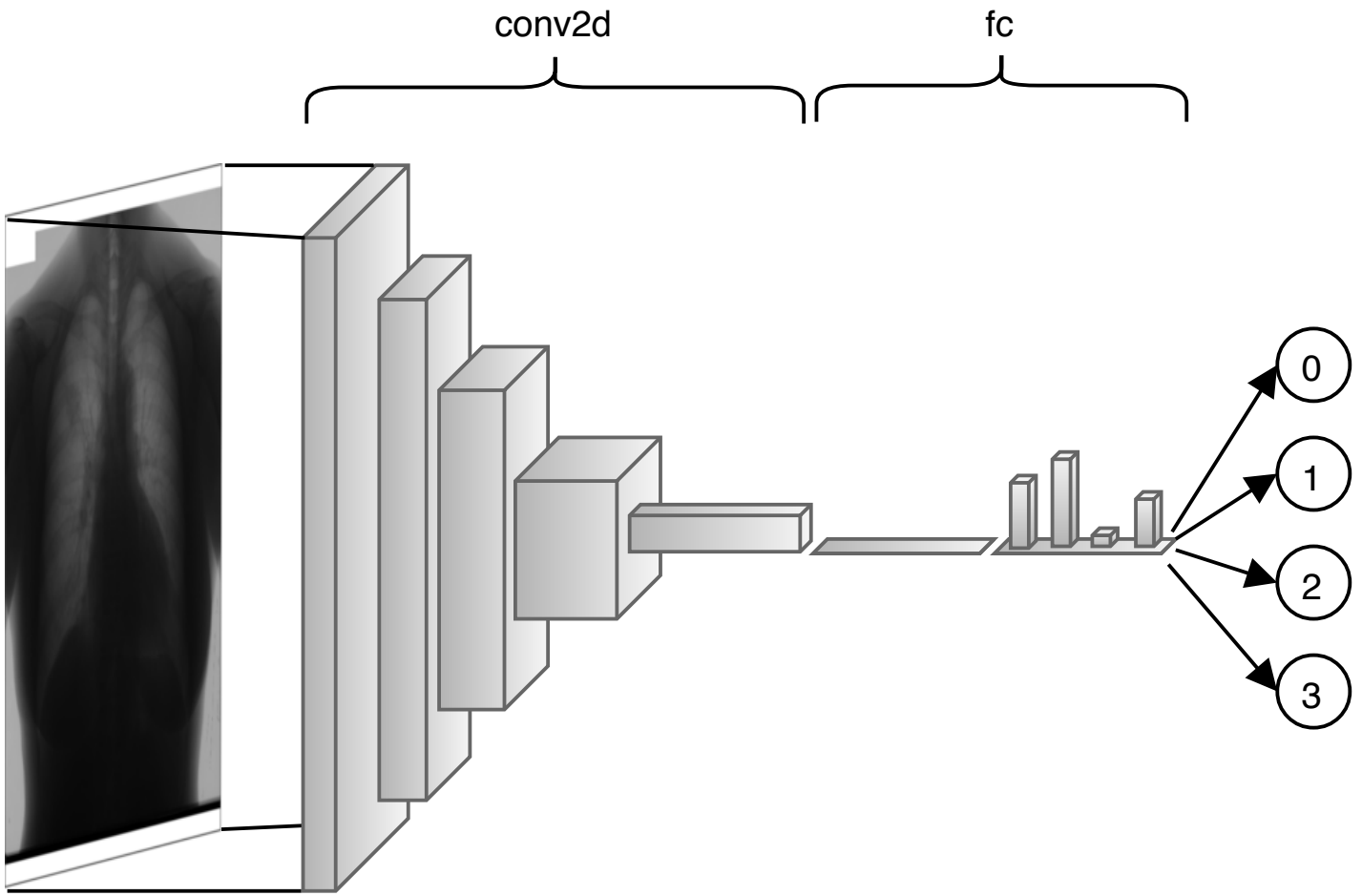


Figura: Arquitetura de uma CNN.

Rotulação Esparsa $\rightarrow$ 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

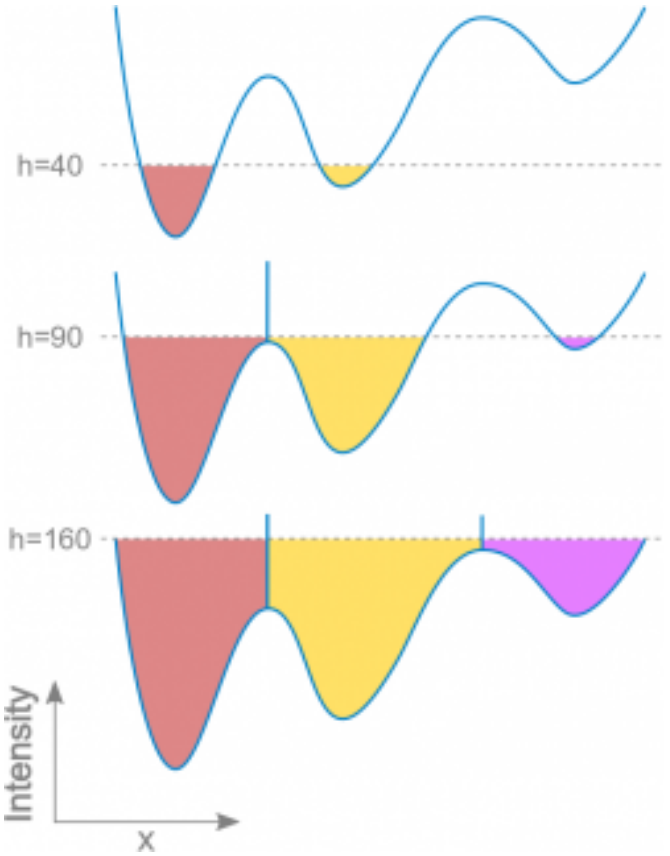
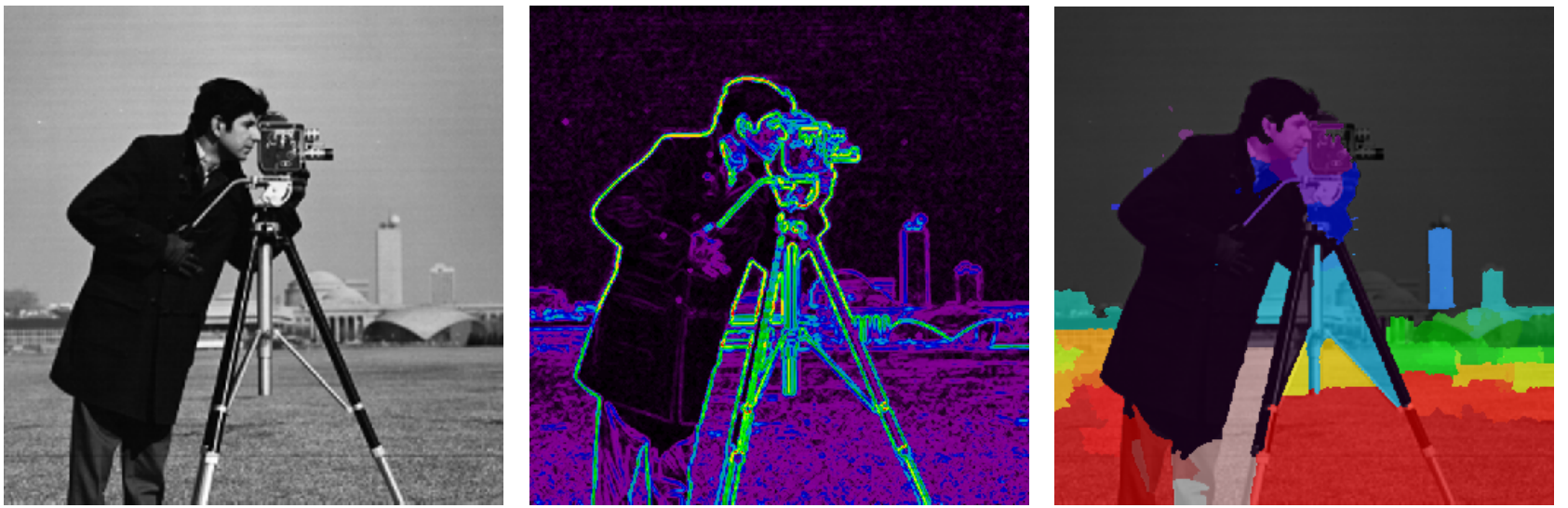


Figura: Flooding do algoritmo do Watershed em 1D. A ideia básica do algoritmo é considerar a imagem como um mapa topográfico e colocar uma fonte de água em cada mínimo local. Em seguida, barragens são adicionadas nas regiões em que as fontes se encontram[7].

Watershed



(a) (b) (c)

Figura: Segmentação de imagem via Watershed[7].

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

Métodos Clássicos de Segmentação

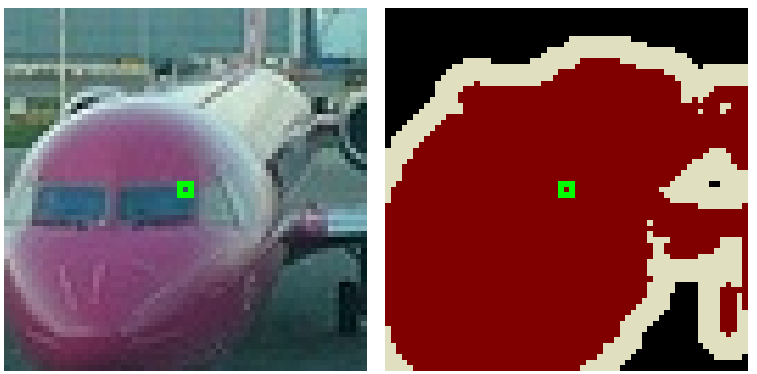
- Problemas com os métodos clássicos
  - Processamentos locais, geralmente envolvendo features muito rasos
  - Generalidade
  - Falta de informação semântica

Métodos Deep de Segmentação

- Abordagens Deep para Rotulação Densa
  - Classificação de Pixels
  - Fully Convolutional Networks (FCNs) [14]
  - Redes de Deconvolução
    - U-Nets [15]
    - SegNets [16]

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 $\times$ 256 possui 65536 pixels
  - Alta redundância nos cálculos das convoluções para janelas de contexto sobrepostas

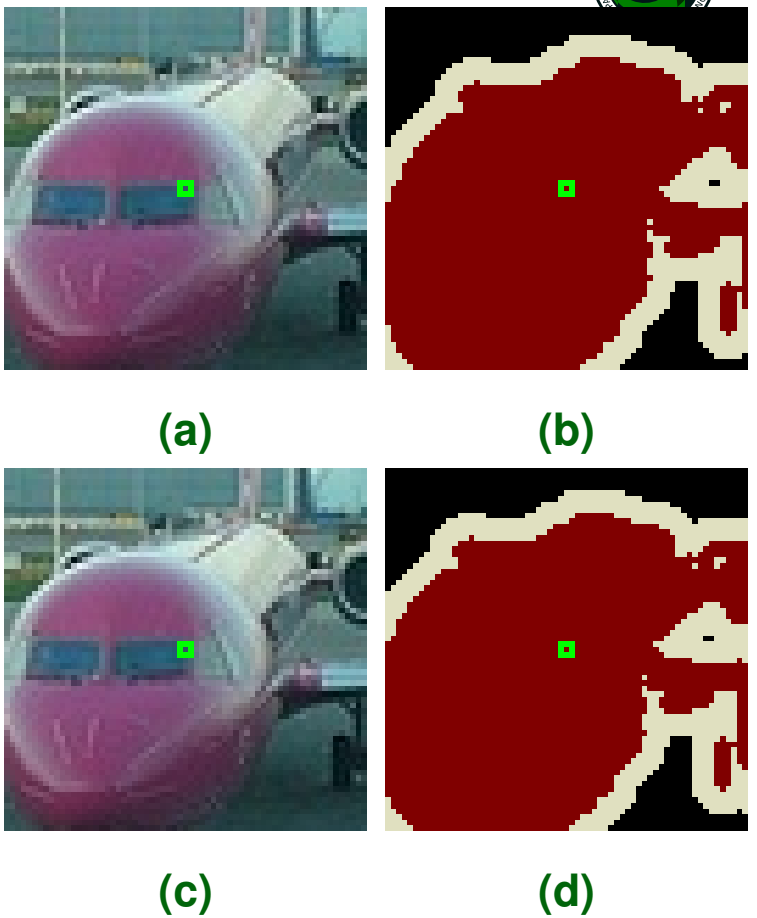


Figura: Pixel central da imagem original e da máscara de labels circundados por janela de contexto.

CNN → FCN

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

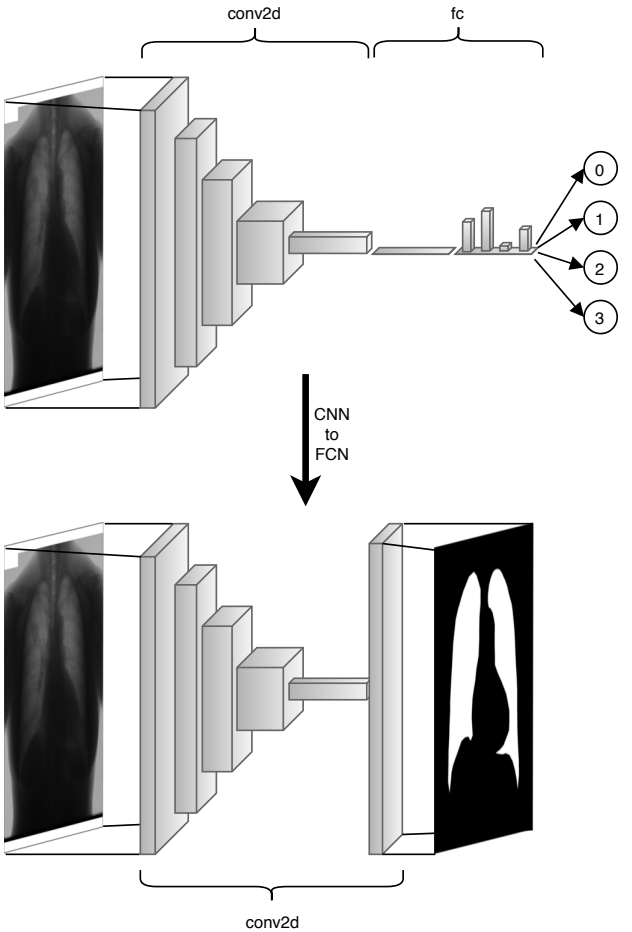
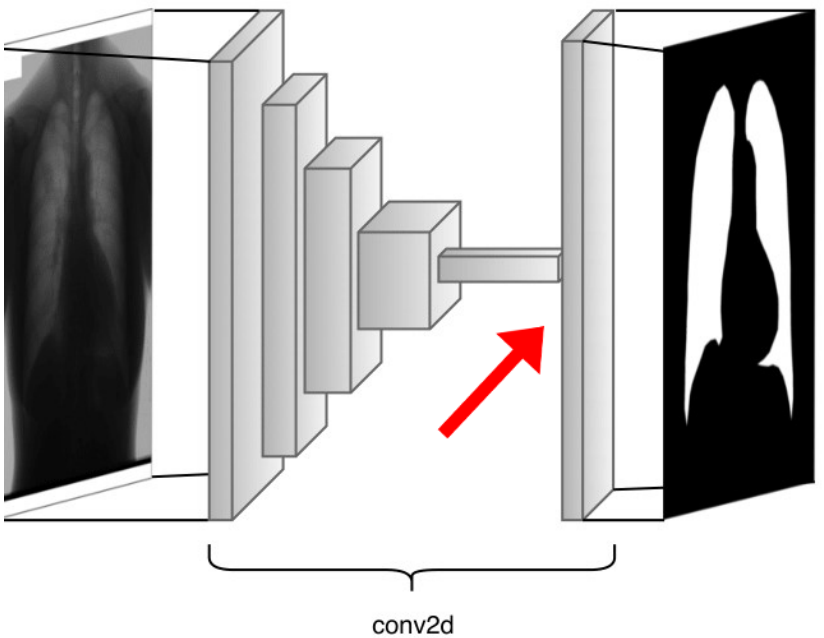


Figura: Transformando uma CNN [2] em uma FCN [14].

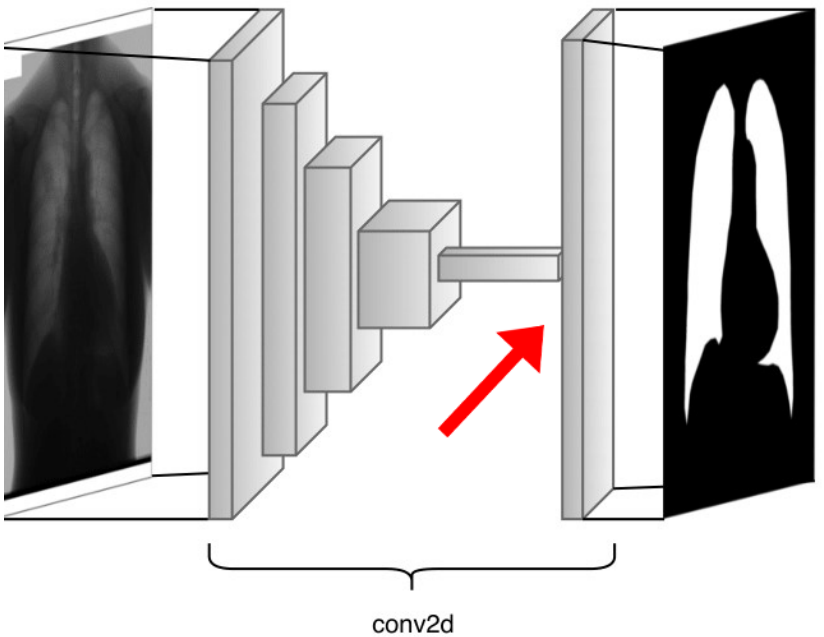
FCN

- Ao longo das convoluções perdemos resolução espacial nos feature maps. É preciso **recuperar esta resolução** para obter o resultado da segmentação.



FCN

- FCNs tradicionais realizam a operação de **upsample** antes de prosseguir com as últimas convoluções.



Redes de Deconvolução

- Arquitetura Encoder-Decoder convolucionais.
- Mais camadas dedicadas à recuperação da informação espacial
- Implementam **Convoluções Transpostas** (Deconvoluções)

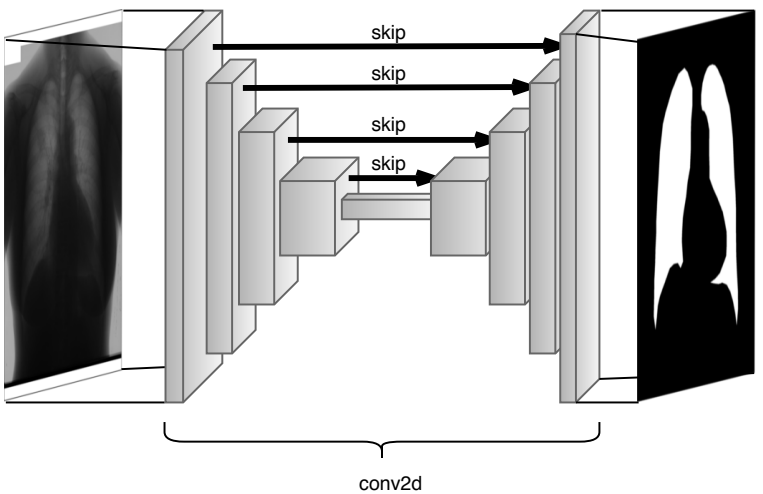


Figura: Arquitetura da U-Net.

Convoluções Normais

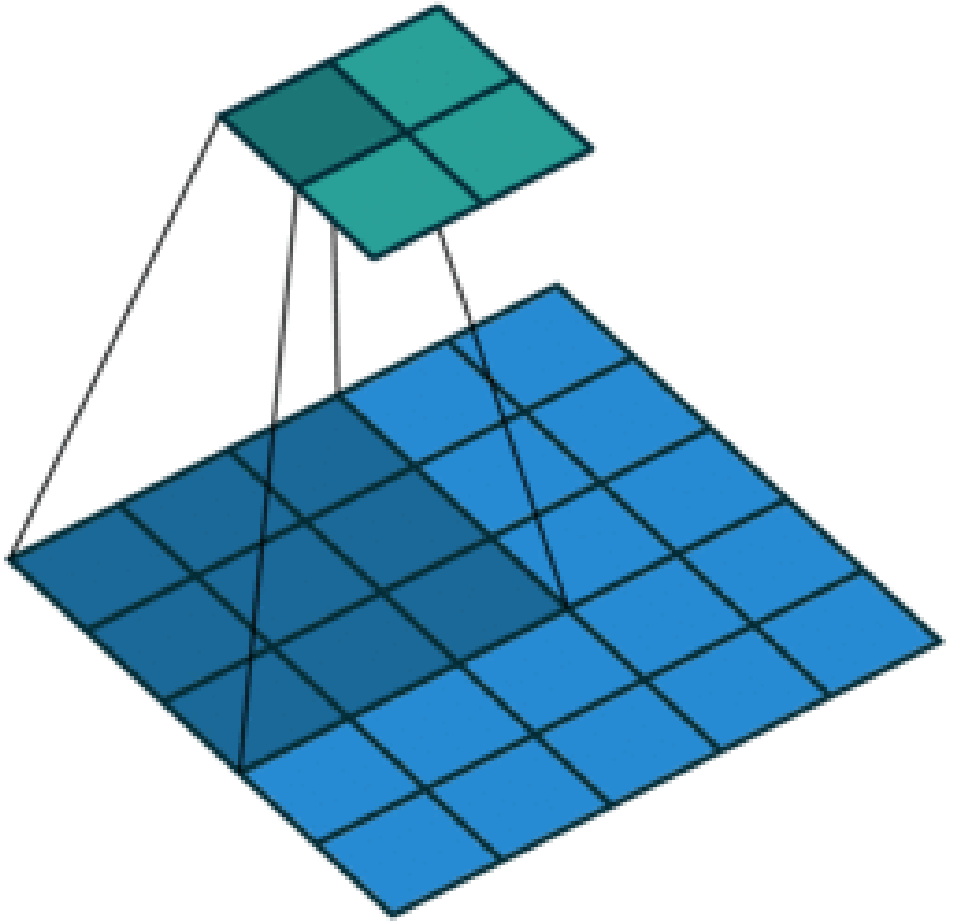


Figura: Convolução Normais[7].

[7] https://towardsdatascience.com/types-of-convolutions-in-deep-learning-717013397f4d

Convoluções Normais

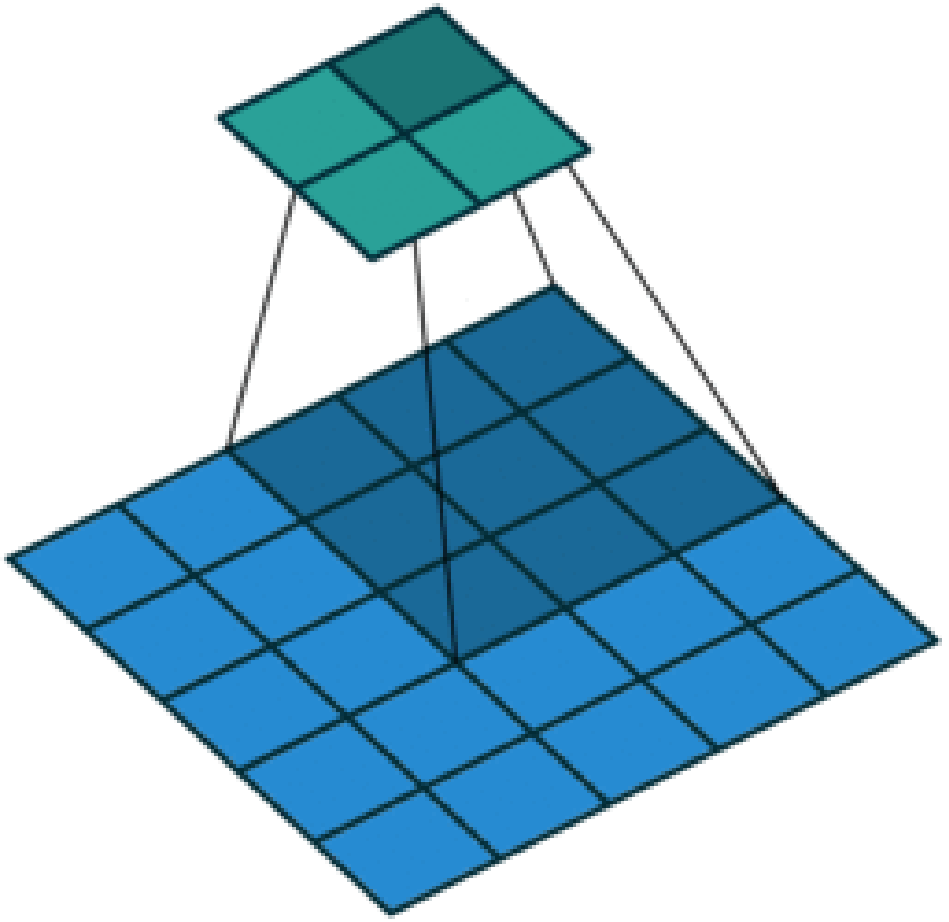


Figura: Convolução Normais[7].

[7] https://towardsdatascience.com/types-of-convolutions-in-deep-learning-717013397f4d

Convoluções Normais

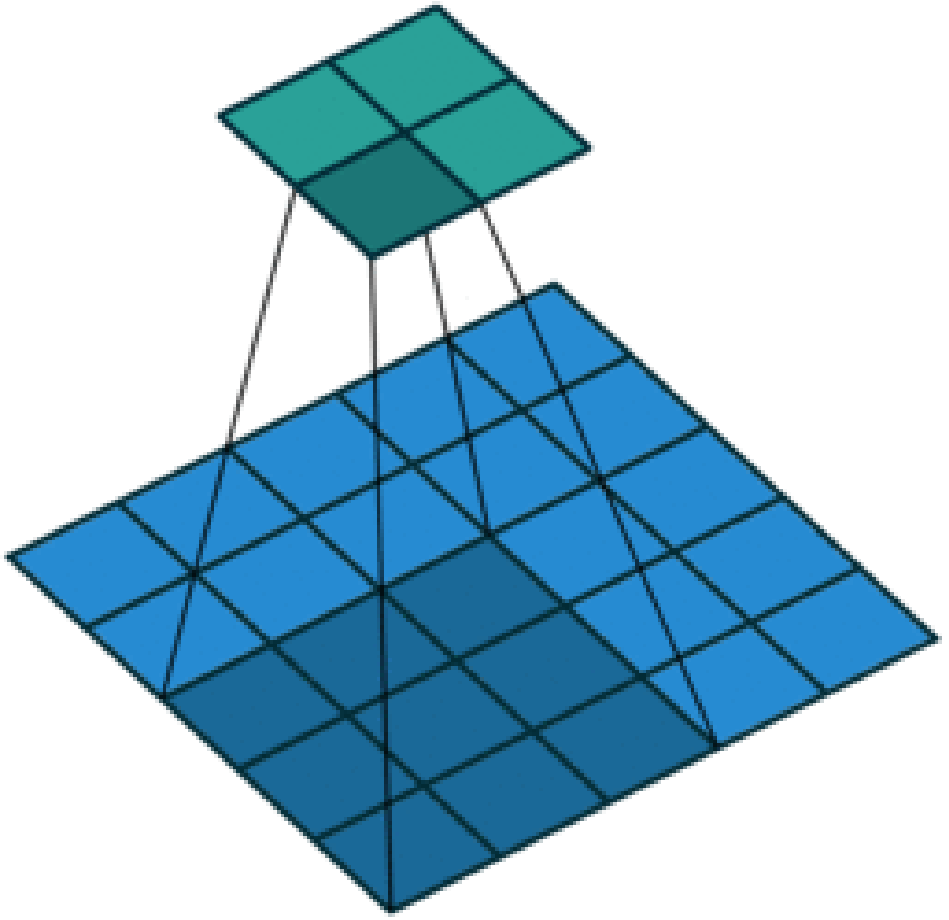


Figura: Convolução Normais[7].

[7] https://towardsdatascience.com/types-of-convolutions-in-deep-learning-717013397f4d

Convoluções Normais

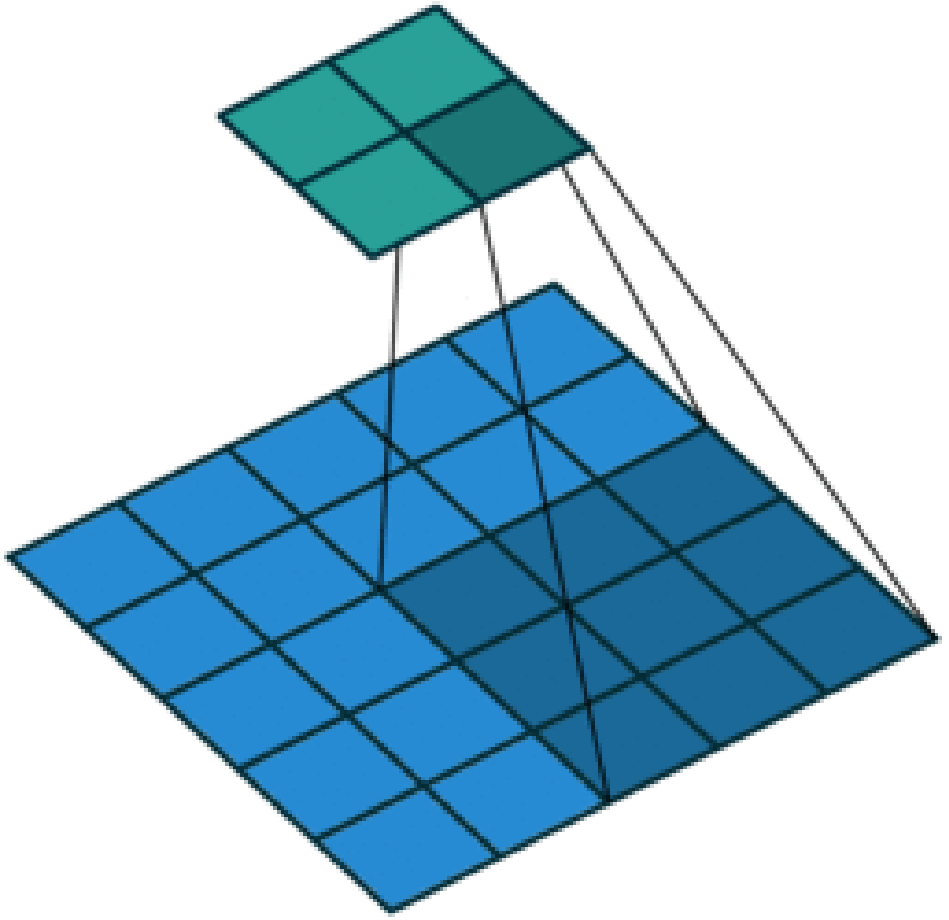


Figura: Convolução Normais[7].

[7] https://towardsdatascience.com/types-of-convolutions-in-deep-learning-717013397f4d

Convoluções Transpostas

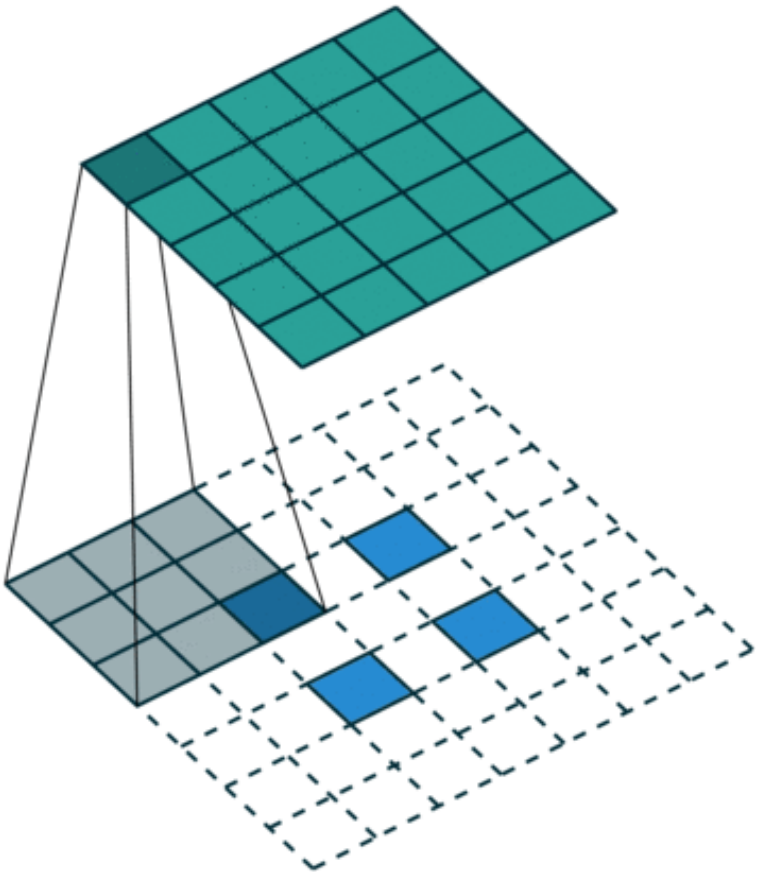


Figura: Convolução Transposta[7].

[7] https://towardsdatascience.com/types-of-convolutions-in-deep-learning-717013397f4d

Convoluções Transpostas

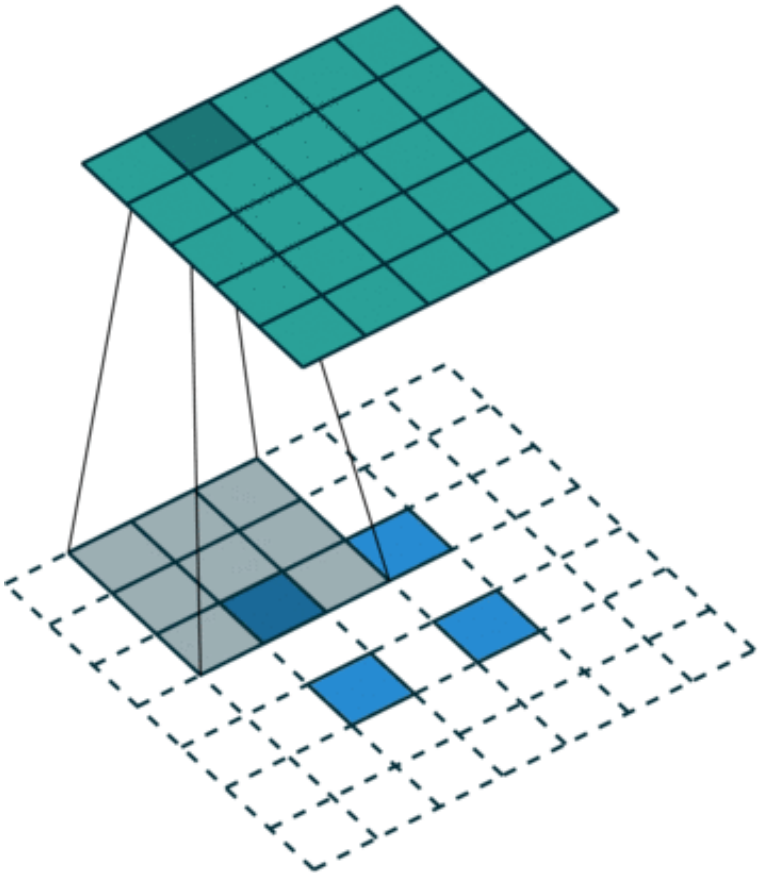


Figura: Convolução Transposta[7].

[7] https://towardsdatascience.com/types-of-convolutions-in-deep-learning-717013397f4d

Convoluções Transpostas

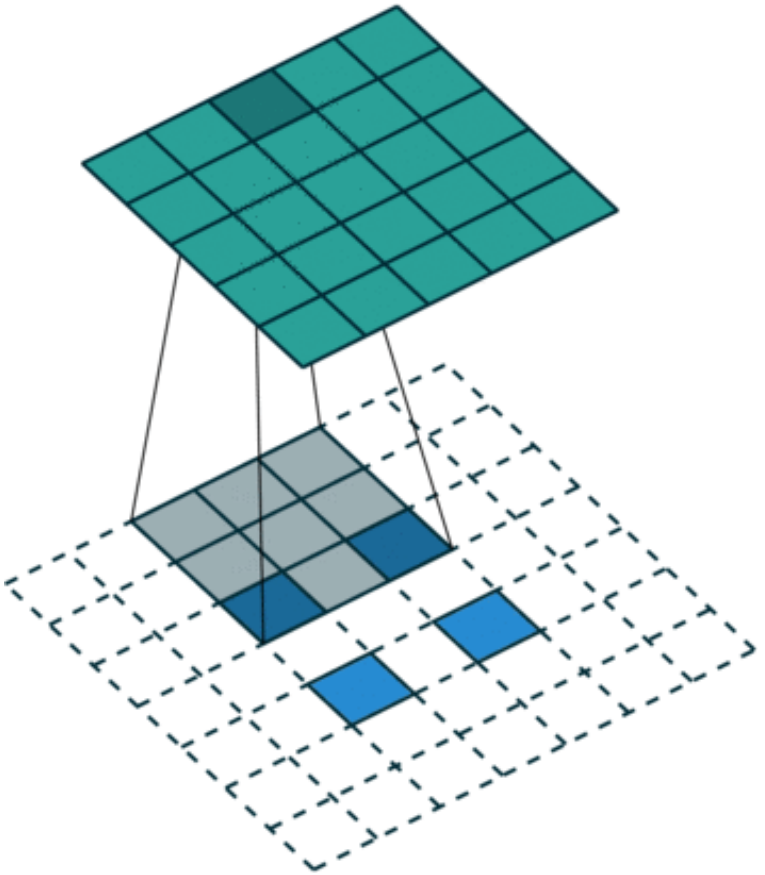


Figura: Convolução Transposta[7].

[7] https://towardsdatascience.com/types-of-convolutions-in-deep-learning-717013397f4d

Convoluções Transpostas

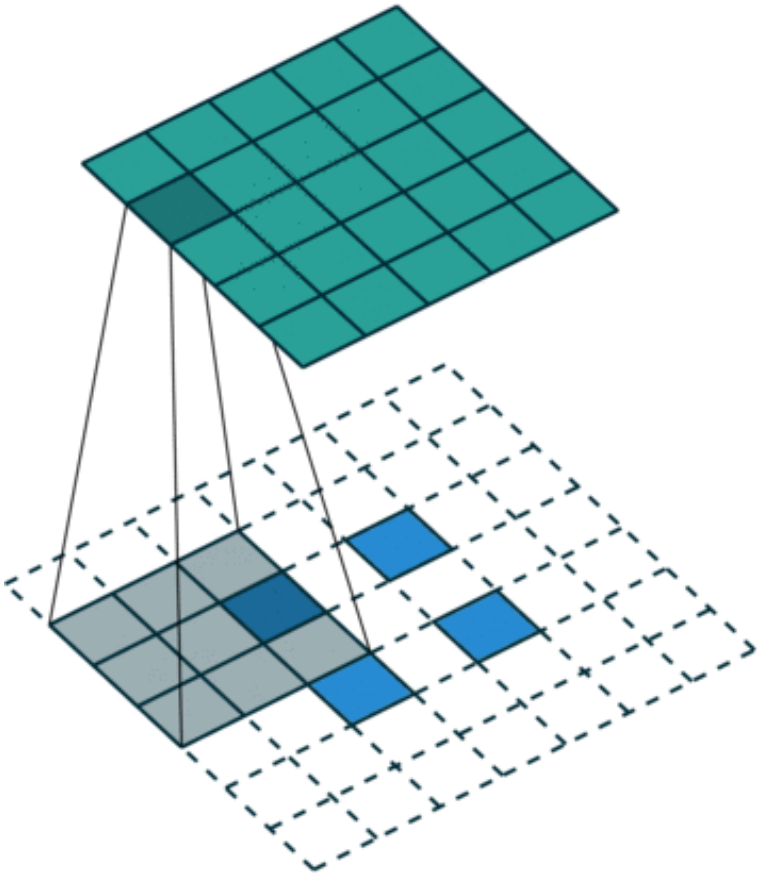


Figura: Convolução Transposta[7].

[7] https://towardsdatascience.com/types-of-convolutions-in-deep-learning-717013397f4d

Convoluções Transpostas

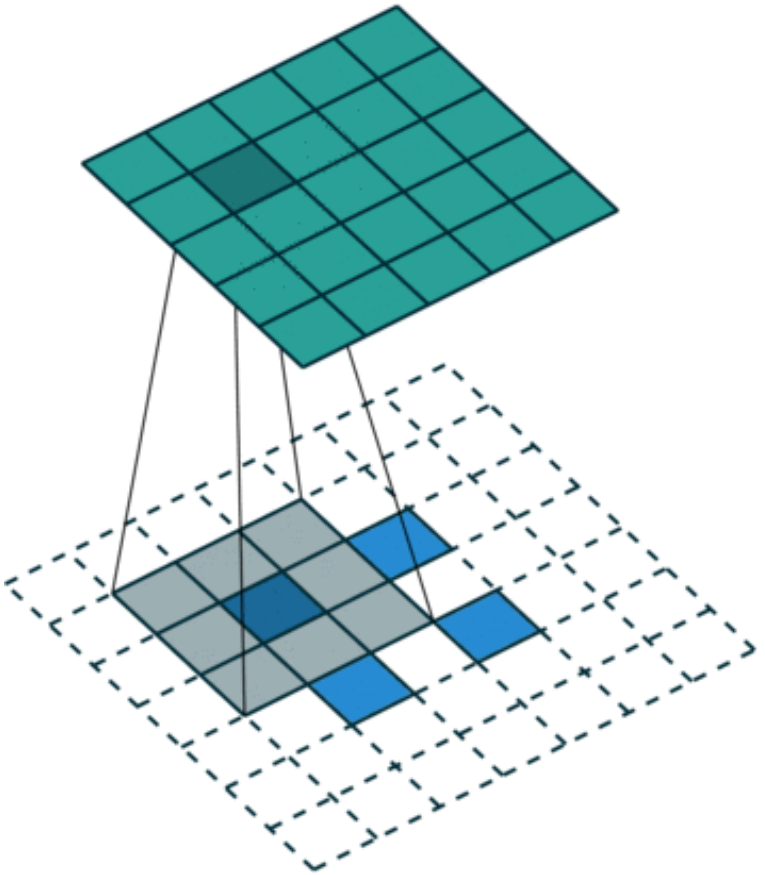


Figura: Convolução Transposta[7].

[7] https://towardsdatascience.com/types-of-convolutions-in-deep-learning-717013397f4d

Convoluções Transpostas

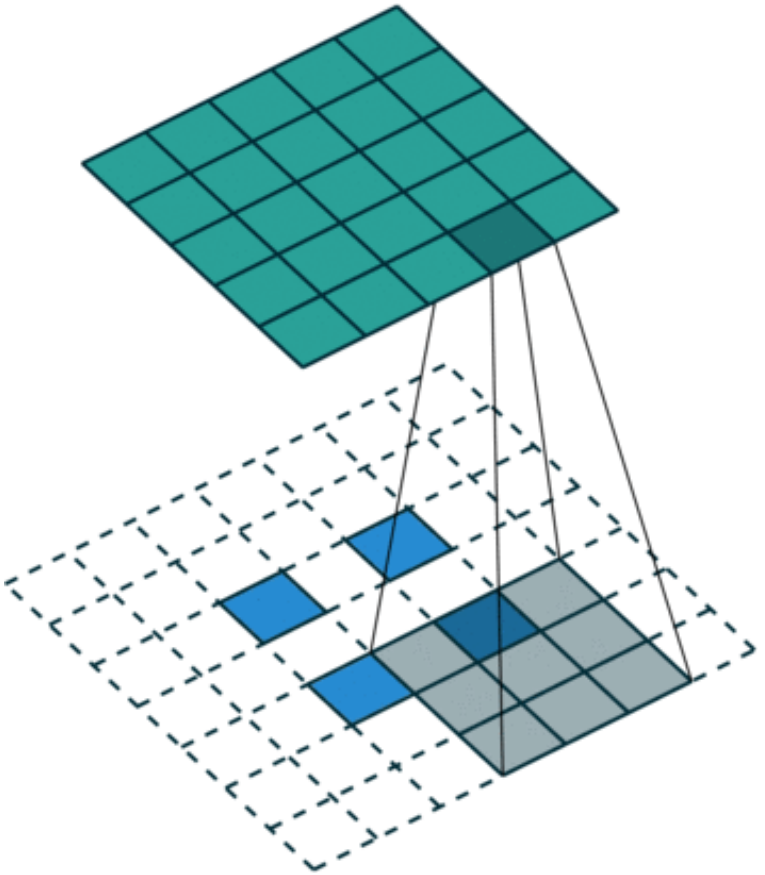


Figura: Convolução Transposta[7].

[7] https://towardsdatascience.com/types-of-convolutions-in-deep-learning-717013397f4d

Convoluções Transpostas

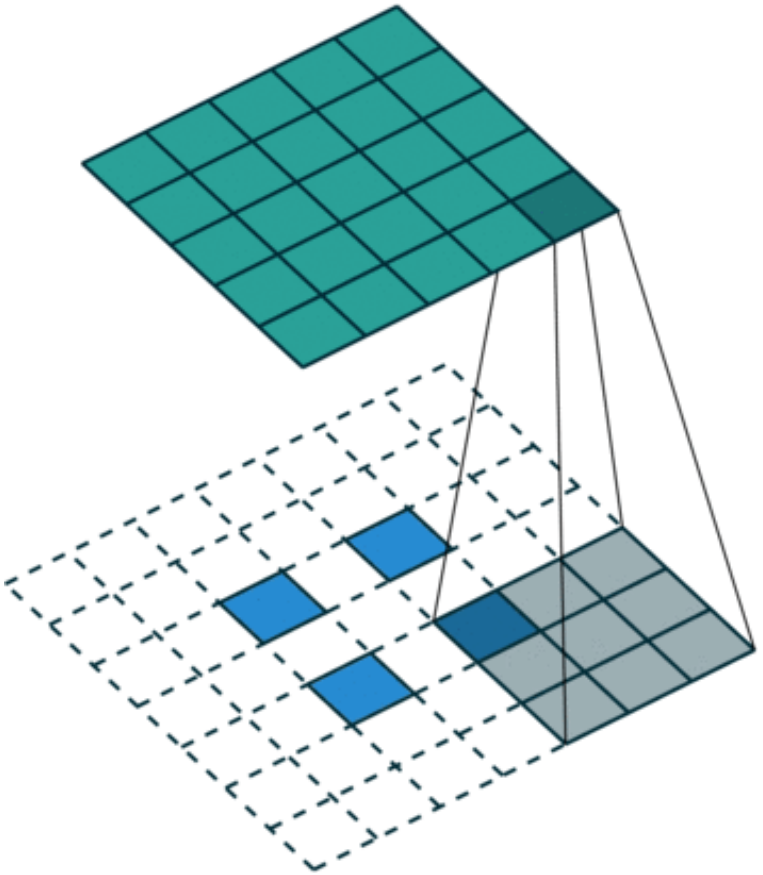
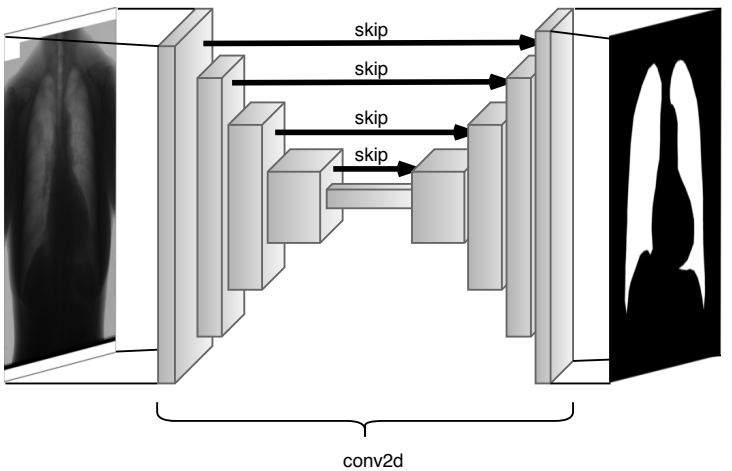


Figura: Convolução Transposta[7].

[7] https://towardsdatascience.com/types-of-convolutions-in-deep-learning-717013397f4d

Redes de Deconvolução



- U-nets [15]
  - **Skip Connections** entre camadas simétricas.
  - 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. Adaptado de [15].

Skip Connections

- Propostos por [14]
- 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

Redes de Deconvolução

- SegNets [16]
  - Uso dos índices de pooling do Encoder para a reconstrução do Decoder

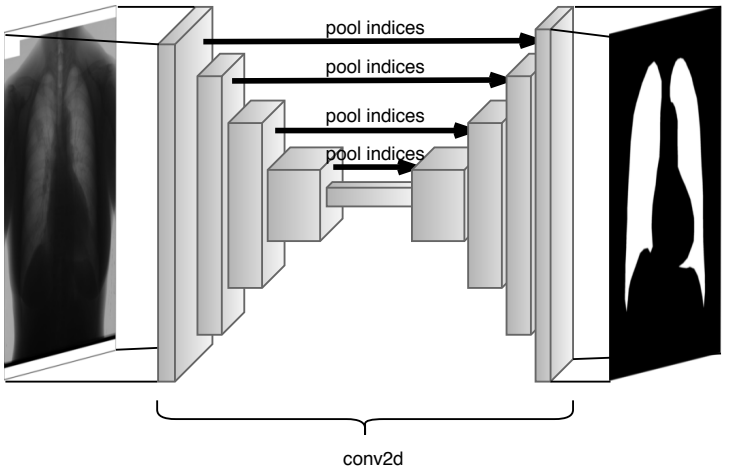


Figura: Ilustração de uma SegNet. Setas correspondem à passagem dos índices de pooling para camadas posteriores. Adaptado de [16].

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

Segmentação de Imagens

Demo - FCN

FCN_Segmentation.ipynb

[1] 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.

[2] 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.

[3] Karen Simonyan and Andrew Zisserman.
Very deep convolutional networks for large-scale image recognition.
arXiv preprint arXiv:1409.1556, 2014.

[4] 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.

[5] 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.

[6] 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.

[7] Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber.
Training very deep networks.
In Advances in neural information processing systems, pages 2377–2385, 2015.

[8] 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.

[9] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich.
Fractalnet: Ultra-deep neural networks without residuals.
arXiv preprint arXiv:1605.07648, 2016.

[10] 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.

[11] 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.

[12] Ross Girshick.
Fast r-cnn.
In Proceedings of the IEEE international conference on computer vision, pages 1440–1448, 2015.

[13] 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.

[14] J. Long, E. Shelhamer, and T. Darrell.
Fully convolutional networks for semantic segmentation.
pages 3431–3440, June 2015.

[15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox.
U-net: Convolutional networks for biomedical image segmentation.
In MICCAI, pages 234–241. Springer, 2015.

[16] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla.
Segnet: A deep convolutional encoder-decoder architecture for image segmentation.
39(12):2481–2495, 2017.