Aula 2
Processamento de Imagens - Convolução

Eduardo L. L. Cabral

Objetivos

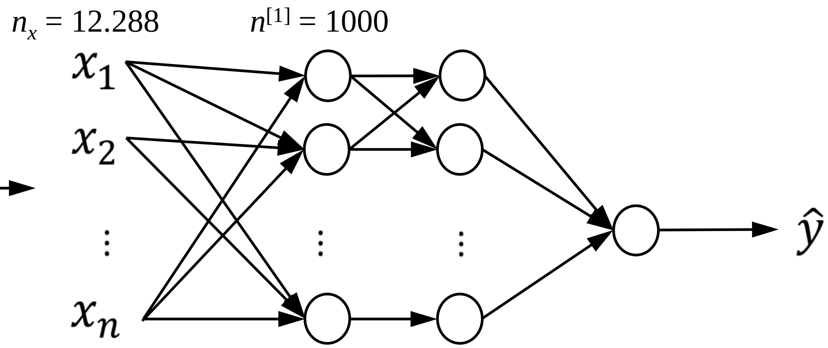
- Apresentar princípios básicos de processamento de imagens.
- Apresentar a operação de convolução.
- Exemplo de operação de convolução para detectar bordas em imagens.
- Apresentar aspectos operacionais da convolução (“padding” e “stride”).

Desafios da visão computacional

- Como visto na Aula 13, a visão computacional apresenta inúmeros desafios.
- Um dos maiores desafios da visão computacional é a dimensão das imagens, que pode ser da ordem de vários mega-pixels.
- Se as imagens forem pequenas, por exemplo, 64x64x3 pixels totalizando 12.288 números $\Rightarrow$ o uso de uma RNA com camadas densas não é um problema.
- Se as imagens forem, por exemplo, padrão HD de 1080x720x3 pixels tem-se 2.332.800 números $\Rightarrow$ nesse caso o uso de uma RNA com camadas densas se torna um grande problema computacional.
- A solução para diminuir o número de parâmetros da RNA é usar as RNA convolucionais.

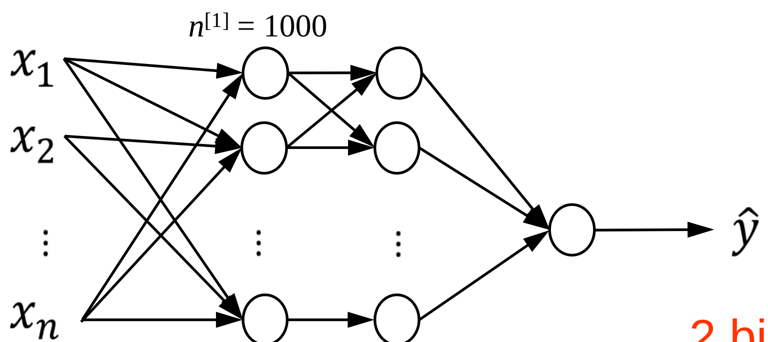
Desafios da visão computacional

64 x 64 x 3 = 12.288



$\mathbf{W}^{[1]} \rightarrow (n^{[1]}, n_x)$
(1000 x 12.288 = 1.228.800 parâmetros)

1080 x 720 x 3 = 2.332.800



$\mathbf{W}^{[1]} \rightarrow (n^{[1]}, n_x)$
(1000 x 2.332.800 = 2.332.800.000 parâmetros)

2 bilhões de parâmetros !!!

Operação de convolução

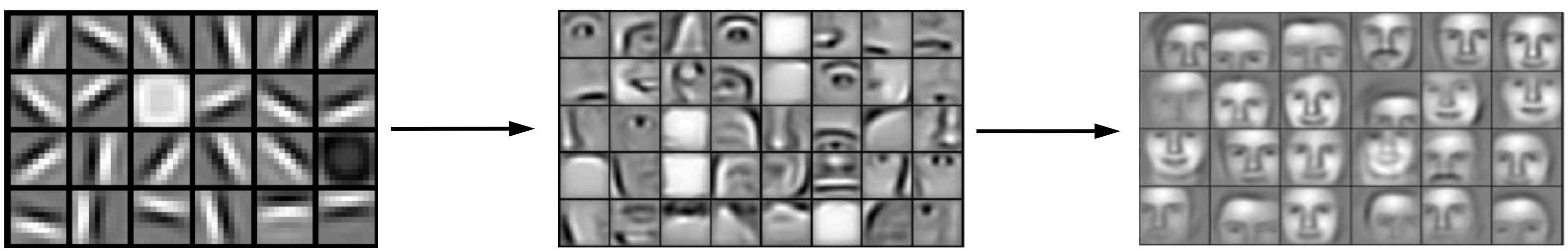
- Convolução é a operação mais utilizada em visão computacional para processamento de imagens.
- Convolução pode ser utilizada para realizar inúmeras operações:
  - detecção de bordas;
  - detecção de cantos;
  - detecção de cores;
  - suavização da imagem (filtro passa baixa);
  - ressaltamento (filtro passa alta);
  - etc.

Operação de convolução

- Dado o seu potencial de identificar características nas imagens a convolução se tornou a operação básica das RNA convolucionais.
- RNA convolucionais $\Rightarrow$ especializadas em trabalhar com imagens.
- RNA convolucional de várias camadas, cada camada acrescenta e agrega informações das camadas anteriores para realizar a tarefa desejada.

Operação de convolução

- Exemplo das saídas das camadas de uma RNA convolucional treinada para identificar faces.
- Cada camada agrega informação das camadas anteriores.



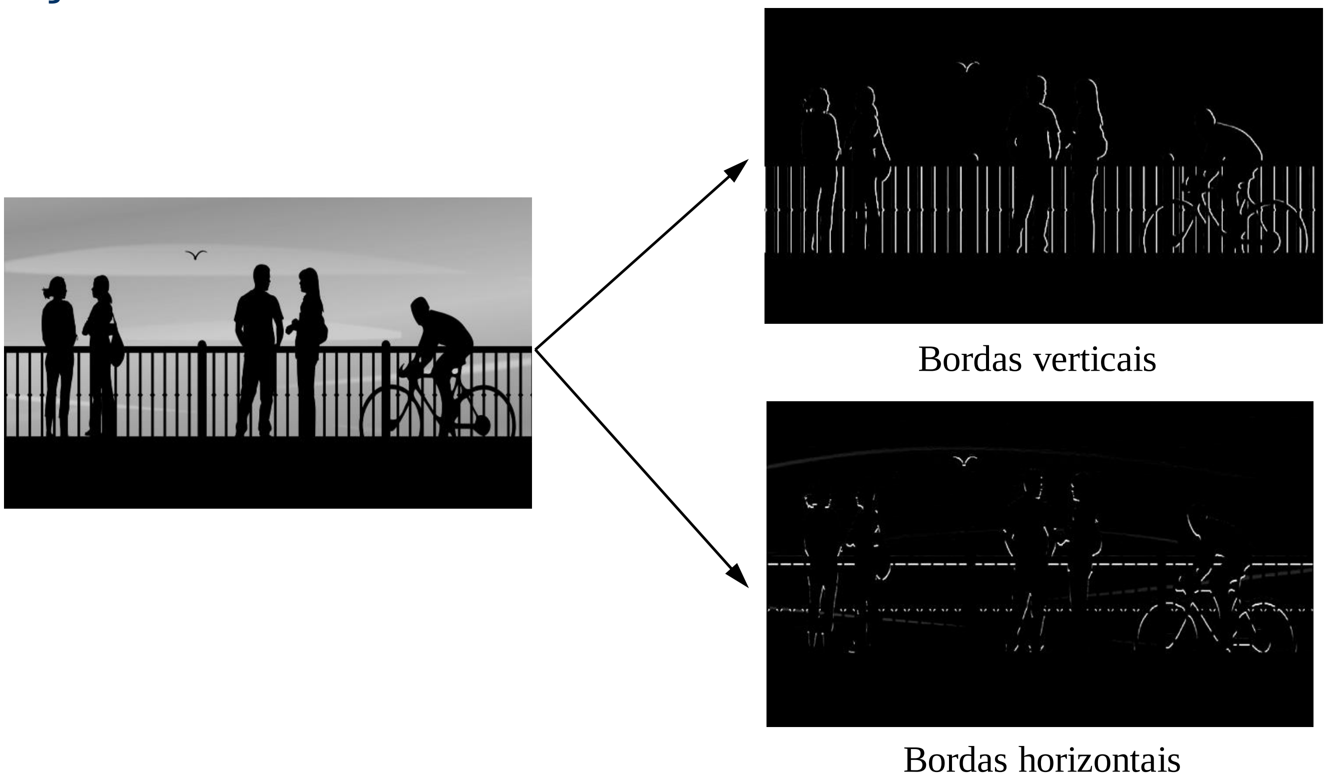
1ª camada → detecção de bordas

2ª camada → detecção de olhos e narizes

3ª camada → detecção de faces

Detecção de bordas

- Processamento típico de imagens usando convolução ⇒ detecção de bordas.



Exemplo de detecção de bordas verticais e horizontais em uma imagem.

Detecção de bordas verticais

3*1+1*1+2*1+0*0+5*0+7*0+1*(-1)+8*(-1)+2*(-1) = -5

3^{1}	0^{0}	1^{-1}	2	7	4
1^{1}	5^{0}	8^{-1}	9	3	1
2^{1}	7^{0}	2^{-1}	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

Imagem 6x6

*

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

Filtro 3x3

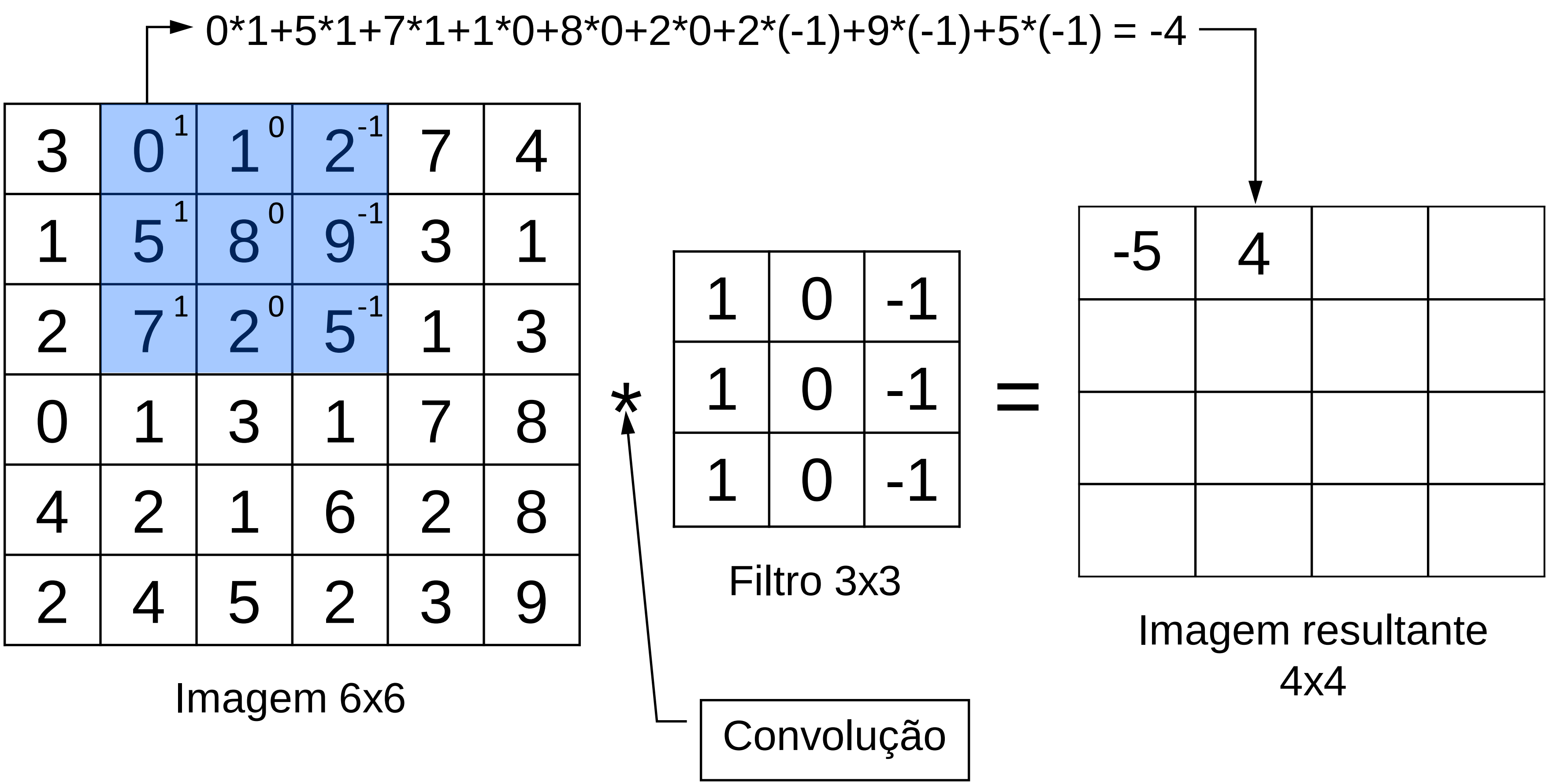
=

-5			

Imagem resultante 4x4

Convolução

Detecção de bordas verticais



Detecção de bordas verticais

1*1+8*1+2*1+2*0+9*0+5*0+7*(-1)+3*(-1)+1*(-1) = 0

3	0	1^{1}	2^{0}	7^{-1}	4
1	5	8^{1}	9^{0}	3^{-1}	1
2	7	2^{1}	5^{0}	1^{-1}	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

Imagem 6x6

*

Convolução

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

Filtro 3x3

=

-5	4	0	

Imagem resultante 4x4

Detecção de bordas verticais

2*1+9*1+5*1+7*0+3*0+1*0+4*(-1)+1*(-1)+3*(-1) = 8

3	0	1	2^{1}	7^{0}	4^{-1}
1	5	8	9^{1}	3^{0}	1^{-1}
2	7	2	5^{1}	1^{0}	3^{-1}
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

Imagem 6x6

*

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

Filtro 3x3

=

-5	4	0	8

Imagem resultante 4x4

Convolução

Detecção de bordas verticais

3	0	1	2	7	4
1^{1}	5^{0}	8^{-1}	9	3	1
2^{1}	7^{0}	2^{-1}	5	1	3
0^{1}	1^{0}	3^{-1}	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

Imagem 6x6

*

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

Filtro 3x3

=

-5	4	0	8
-10			

Imagem resultante
4x4

Convolução

Detecção de bordas verticais

3	0	1	2	7	4
1	5^{1}	8^{0}	9^{-1}	3	1
2	7^{1}	2^{0}	5^{-1}	1	3
0	1^{1}	3^{0}	1^{-1}	7	8
4	2	1	6	2	8
2	4	5	2	3	9

Imagem 6x6

*

Convolução

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

Filtro 3x3

=

-5	4	0	8
-10	-2		

Imagem resultante 4x4

Detecção de bordas verticais

3	0	1	2	7	4
1	5	8^{1}	9^{0}	3^{-1}	1
2	7	2^{1}	5^{0}	1^{-1}	3
0	1	3^{1}	1^{0}	7^{-1}	8
4	2	1	6	2	8
2	4	5	2	3	9

Imagem 6x6

*

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

Filtro 3x3

=

-5	4	0	8
-10	-2	-2	

Imagem resultante 4x4

Convolução

Detecção de bordas verticais

3	0	1	2	7	4
1	5	8	9^{1}	3^{0}	1^{-1}
2	7	2	5^{1}	1^{0}	3^{-1}
0	1	3	1^{1}	7^{0}	8^{-1}
4	2	1	6	2	8
2	4	5	2	3	9

Imagem 6x6

*

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

Filtro 3x3

=

-5	4	0	8
-10	-2	2	3

Imagem resultante 4x4

Convolução

Detecção de bordas verticais

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

Imagem 6x6

*

Convolução

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

Filtro 3x3

=

-5	4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-10

Imagem resultante 4x4

Detecção de bordas verticais

- Outro exemplo de imagem onde existe de fato uma borda vertical.

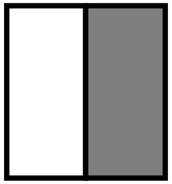
10^{1}	10^{0}	10^{-1}	0	0	0
10^{1}	10^{0}	10^{-1}	0	0	0
10^{1}	10^{0}	10^{-1}	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

*

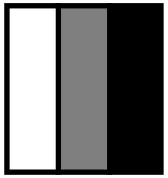
1	0	-1
1	0	-1
1	0	-1

=

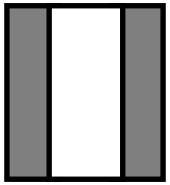
0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0



*



=



Detecção de bordas verticais

- Outro exemplo de imagem onde existe de fato uma borda vertical.

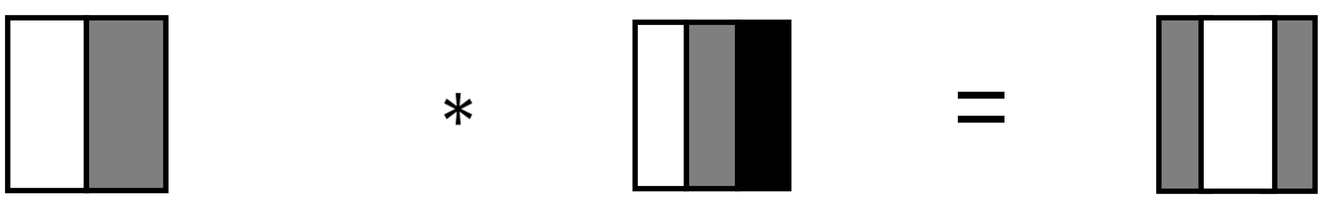
10	10^{1}	10^{0}	0^{-1}	0	0
10	10^{1}	10^{0}	0^{-1}	0	0
10	10^{1}	10^{0}	0^{-1}	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

*

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

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0



Detecção de bordas verticais

- Outro exemplo de imagem onde existe de fato uma borda vertical.

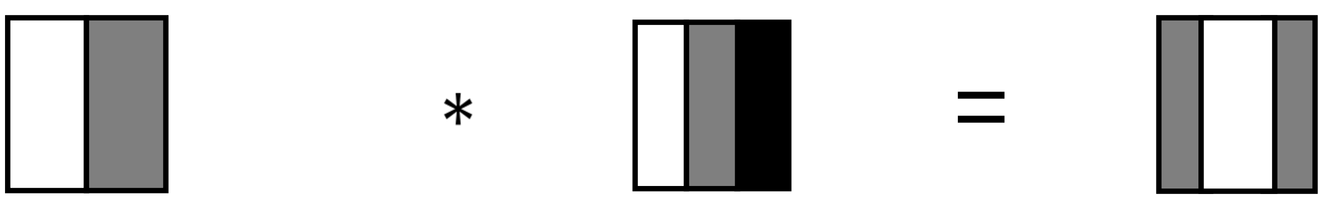
10	10	10^{1}	0^{0}	0^{-1}	0
10	10	10^{1}	0^{0}	0^{-1}	0
10	10	10^{1}	0^{0}	0^{-1}	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

*

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

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0



Detecção de bordas verticais

- Outro exemplo de imagem onde existe de fato uma borda vertical.

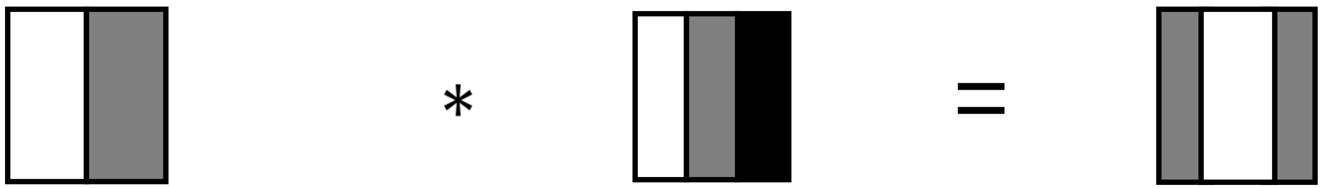
10	10	10	0^{1}	0^{0}	0^{-1}
10	10	10	0^{1}	0^{0}	0^{-1}
10	10	10	0^{1}	0^{0}	0^{-1}
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

*

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

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0



Detecção de bordas verticais

- Na figura anterior ⇒ quanto maior o valor do pixel mais claro é o pixel.
- Borda detectada na imagem é espessa ⇒ isso de fato ocorre, mas no caso da imagem do exemplo que tem dimensão reduzida isso parece exagero ⇒ para imagens de dimensões normais essa borda não vai aparentar ser espessa.

Detecção de bordas verticais

- Outro exemplo de detecção de borda vertical.

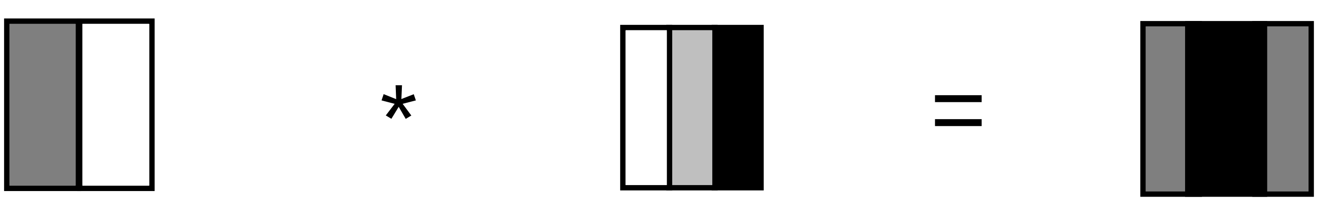
0^{1}	0^{0}	0^{-1}	10	10	10
0^{1}	0^{0}	0^{-1}	10	10	10
0^{1}	0^{0}	0^{-1}	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10

*

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

=

0	-30	-30	0
0	-30	-30	0
0	-30	-30	0
0	-30	-30	0



Detecção de bordas verticais

- Outro exemplo de detecção de borda vertical.

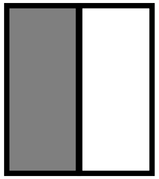
0	0 [1]	0 [0]	10 [-1]	10	10
0	0 [1]	0 [0]	10 [-1]	10	10
0	0 [1]	0 [0]	10 [-1]	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10

*

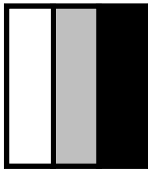
1	0	-1
1	0	-1
1	0	-1

=

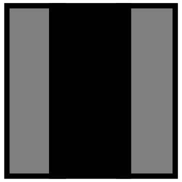
0	-30	-30	0
0	-30	-30	0
0	-30	-30	0
0	-30	-30	0



*



=



Detecção de bordas verticais

- Outro exemplo de detecção de borda vertical.

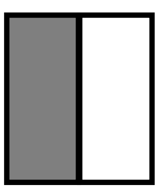
0	0	0 ^{1}	10 ^{0}	10 $^{-1}$	10
0	0	0 ^{1}	10 ^{0}	10 $^{-1}$	10
0	0	0 ^{1}	10 ^{0}	10 $^{-1}$	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10

*

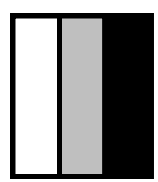
1	0	-1
1	0	-1
1	0	-1

=

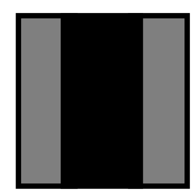
0	-30	-30	0
0	-30	-30	0
0	-30	-30	0
0	-30	-30	0



*



=



Detecção de bordas horizontais

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

Filtro para borda vertical

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

Filtro para borda horizontal

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10

*

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

=

0	0	0	0
30	10	-10	-30
30	10	-10	-30
0	0	0	0

Detecção de bordas horizontais

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

Filtro para borda vertical

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

Filtro para borda horizontal

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10

*

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

=

0	0	0	0
30	10	-10	-30
30	10	-10	-30
0	0	0	0

Detecção de bordas horizontais

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

Filtro para borda vertical

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

Filtro para borda horizontal

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10

*

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

=

0	0	0	0
30	10	-10	-30
30	10	-10	-30
0	0	0	0

Detecção de bordas horizontais

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

Filtro para borda vertical

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

Filtro para borda horizontal

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10

*

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

=

0	0	0	0
30	10	-10	-30
30	10	-10	-30
0	0	0	0

Detecção de bordas horizontais

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

Filtro para borda vertical

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

Filtro para borda horizontal

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10

*

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

=

0	0	0	0
30	10	-10	-30
30	10	-10	-30
0	0	0	0

Detecção de bordas horizontais

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

Filtro para borda vertical

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

Filtro para borda horizontal

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10

*

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

=

0	0	0	0
30	10	-10	-30
30	10	-10	-30
0	0	0	0

Detecção de bordas

- Existem outros tipos de filtros para detectar bordas.

1	0	-1
2	0	-2
1	0	-1

Filtro Sobel

3	0	-3
10	0	-10
3	0	-3

Filtro Scharr

0	1	0
1	0	-1
0	-1	0

Borda a 45º

- Podemos também detectar bordas a -45º, 75º etc $\Rightarrow$ o que altera são os números dentro do filtro.

Suavização de imagens

- Outro exemplo de convolução $\Rightarrow$ suavização **(filtragem passa baixa).**

- Substitui o valor do pixel pela média aritmética simples dos valores dos pixels vizinhos.

$$\Rightarrow \frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline\end{array}$$

- Substitui o valor do pixel pela média ponderada dos valores dos pixels vizinhos.

$$\Rightarrow \frac{1}{16}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline\end{array}$$

Suavização de imagens

- Exemplo de imagem suavizada.

Imagem original

Imagem suavizada com filtro 7x7

Convolução nas RNAs

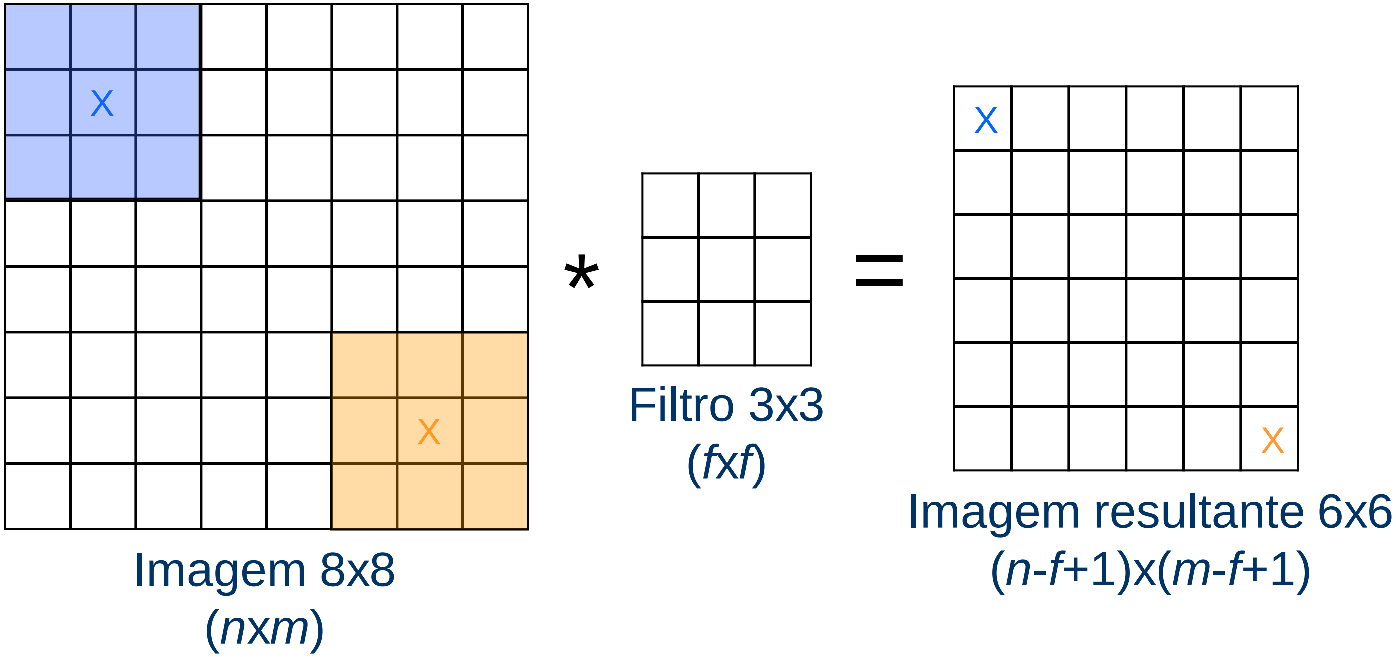
- Em uma RNA convolucional os valores dos filtros utilizados são aprendidos durante o treinamento.
- Valores numéricos dos filtros são parâmetros da RNA aprendidos para detectar características, tais como:
  - Bordas em qualquer ângulo;
  - Cores;
  - Cantos;
  - Etc.
- A RNA aprende os filtros que forem necessários para realizar a tarefa desejada.
- Nos filtros com dimensão 3x3, tem-se 9 parâmetros da RNA a serem aprendidos.

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Filtro a ser aprendido

Efeito de borda

- Dimensão da imagem resultante após convolução:
  - Convolução de imagem 8x8 com filtro 3x3 $\Rightarrow$ resulta imagem 6x6;
  - Filtro tem que caber completamente na imagem;
  - Imagem encolhe a cada etapa de filtragem.



Efeito das bordas

- Dimensão da imagem resultante $\Rightarrow$ $\boxed{(n - f + 1)\text{x}(m - f + 1)}$

  n = altura da imagem em pixels;

  m = largura da imagem em pixels;

  f = tamanho do filtro em pixels.

- Exemplo: (8x8) * (3x3) $\Rightarrow$ (8-3+1)x(8-3+1) = 6x6.
- Imagem encolhe a cada operação de filtragem $\Rightarrow$ isso pode ser indesejável em uma RNA de muitas camadas.

Efeito das bordas

- Informação contida nos pixels de cantos e bordas não está sendo considerada da mesma forma que para os pixels centrais $\Rightarrow$ informação importante pode estar sendo jogada fora.
- Exemplo de convolução com filtro 3x3:
  - Pixel de canto da imagem é usado 1 única vez;
  - Pixel de borda da imagem é usado 3 vezes;
  - Pixel de centro da imagem é usado 9 vezes.

Efeito das bordas

- Como lidar com as bordas da imagem?
  - Se for ignorada ⇒ imagem resultante é menor do que a imagem original;
  - Uma solução ⇒ colocar valor constante nas bordas (“padding”).
- **“Padding”** ⇒ acrescentar pixels nas bordas da imagem na quantidade necessária para que a imagem resultante tenha a mesma dimensão da imagem original:
  - No caso da imagem 8x8 com filtro 3x3 acrescenta-se 1 pixel em todas as bordas da imagem resultando em uma imagem 10x10, que convolucionada com um filtro 3x3 resulta em uma imagem 8x8.
- Pixels são incluídos nas bordas com valor zero.

Efeito das bordas

- Exemplo de “padding” de imagem original 8x8 e filtro 3x3 (p=1):

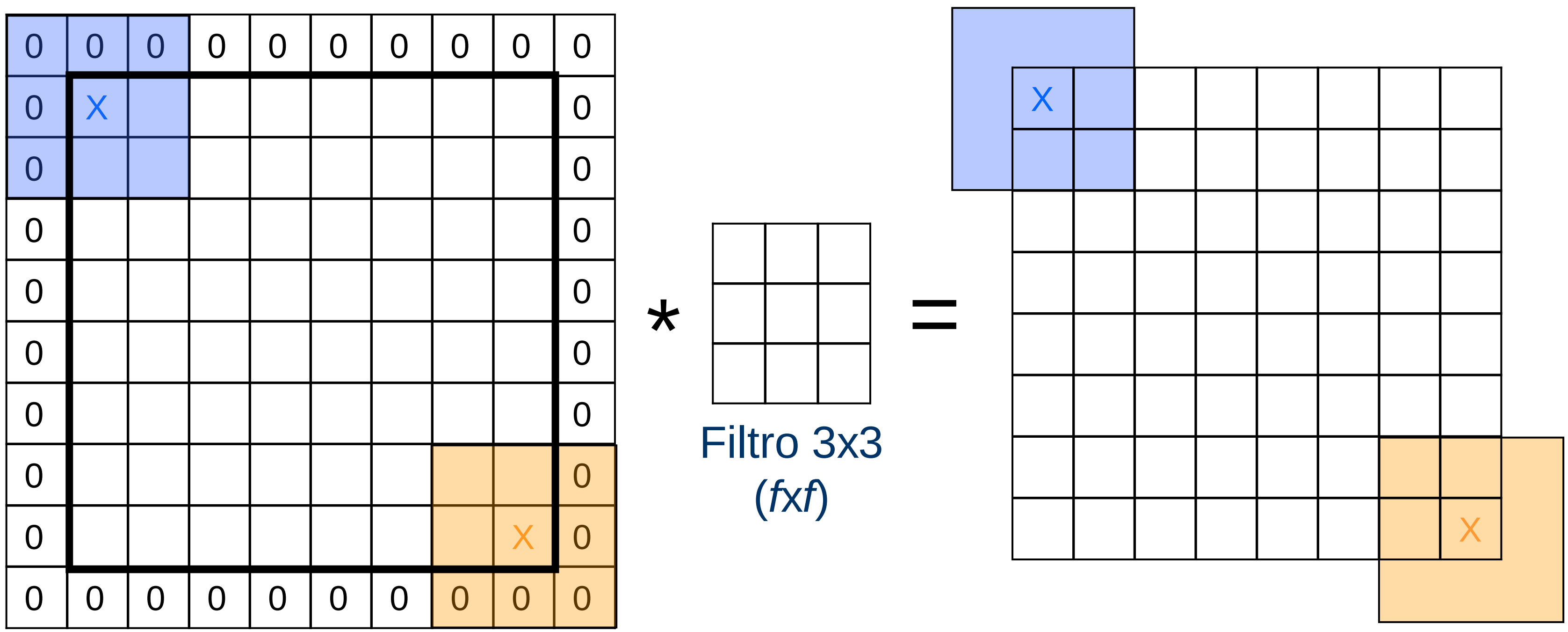


Imagem original 8x8 (nxm)
Imagem com “padding” (p=1) 10x10
(n+2p)x(m+2p)

Filtro 3x3
(fxf)

Imagem resultante 8x8
(n+2p-f+1)x(m+2p-f+1)

Efeito das bordas

- Número de pixels adicionados em cada borda ⇒ p
- Dimensão da imagem resultante ⇒ $\boxed{(n+2p-f+1)\text{x}(m+2p\text{-}f+1)}$

  n = altura da imagem original em pixels;

  m = largura da imagem original em pixels;

  f = dimensão do filtro em pixles (*f*x*f*).
- Usando "padding" ⇒ pixels de cantos e bordas são considerados com maior peso.
- Pode fazer "padding" com um número maior de pixels, por exemplo, *p* = 2 ⇒ *p* depende da dimensão do filtro.
- As imagens não precisam ser quadradas ⇒ altura e largura podem ser diferentes.

Efeito das bordas

- Nomenclatura usada:
  - Convolução válida (“valid convolution”) $\Rightarrow$ convolução realizada sem “padding”:

    Dimensões: $(n \mathrm{x} m) * (f \mathrm{x} f) = (n-f+1)\mathrm{x}(m-f+1)$
  - Convolução mesma (“same convolution”) $\Rightarrow$ convolução realizada adicionando pixels nas bordas em número suficiente para que a imagem resultante tenha mesma dimensão da imagem original:

    Dimensões: $(n+2p)\mathrm{x}(m+2p) * (f \mathrm{x} f) = (n+2p-f+1)\mathrm{x}(m+2p-f+1)$
- Para obter uma imagem resultante de mesma dimensão da imagem original a quantidade de “padding” deve ser:

$$\left.\begin{array}{l}(n+2p-f+1) = n \\ (m+2p-f+1) = m\end{array}\right\} \Rightarrow \text{resolvendo para } p \Rightarrow \boxed{p = (f - 1)/2}$$

Efeito das bordas

- Deve-se sempre usar filtros de dimensão ímpar (f = ímpar).
- Quando f é ímpar quantidade de "padding" necessária para manter imagem resultante com mesma dimensão é inteiro:

  $f = 3 \rightarrow p = (3 - 1)/2 = 1$

  $f = 5 \rightarrow p = (5 - 1)/2 = 2$

- Se f for par o que acontece?
  - A quantidade de "padding" para manter a imagem resultante com mesma dimensão será fracionária:

    $f = 4 \rightarrow p = (4 - 1)/2 = 1{,}5$!!

- Outra vantagem de f ímpar $\Rightarrow$ filtro tem pixel central, o que facilita os cálculos, pois define uma referência simples.
- Dimensões comuns de filtros $\Rightarrow$ 1x1, 3x3, 5x5, 7x7.

Deslocamento (“stride”)

- Outra característica da convolução é o quanto o filtro é deslocado na horizontal e vertical ao ser “passado” pela imagem (“stride”).
  - “Stride” = 1 o filtro se desloca 1 pixel de cada vez na horizontal e depois na vertical;
  - “Stride” = 2 o filtro se desloca 2 pixels de cada vez na horizontal e depois na vertical.
- Dimensão da imagem resultante:

$$(n x m) * (f x f) \text{ com } (s, p) = [(n + 2p - f)/s + 1] x [(m + 2p - f)/s + 1]$$

Deslocamento ("stride")

- Exemplo de convolução com filtro 3x3, "padding", p = 0, e "stride", s = 2:

2^{3}	3^{4}	7^{4}	4	6	2	9
6^{1}	6^{0}	9^{2}	8	7	4	3
3^{-1}	4^{0}	8^{3}	3	8	9	7
7	8	3	6	6	3	4
4	2	1	8	3	4	6
3	2	4	1	9	8	3
0	1	3	9	2	1	4

Imagem original 7x7 (nxm)
p = 0, s = 2

*

3	4	4
1	0	2
-1	0	3

Filtro 3x3
(fxf)

=

91		

Imagem resultante 3x3
$\lfloor (n+2p-f)/s+1 \rfloor \times \lfloor (m+2p-f)/s+1 \rfloor$

Deslocamento (“stride”)

- Exemplo de convolução com filtro 3x3, “padding”, p = 0, e “stride”, s = 2:

2	3	7^{3}	4^{4}	6^{4}	2	9
6	6	9^{1}	8^{0}	7^{2}	4	3
3	4	8^{-1}	3^{0}	8^{3}	9	7
7	8	3	6	6	3	4
4	2	1	8	3	4	6
3	2	4	1	9	8	3
0	1	3	9	2	1	4

Imagem original 7x7 (nxm)
p = 0, s = 2

*

3	4	4
1	0	2
-1	0	3

Filtro 3x3
(fxf)

=

91	100	

Imagem resultante 3x3
$\lfloor (n+2p-f)/s+1 \rfloor \times \lfloor (m+2p-f)/s+1 \rfloor$

Deslocamento ("stride")

- Exemplo de convolução com filtro 3x3, "padding", p = 0, e "stride", s = 2:

2	3	7	4	6^{3}	2^{4}	9^{4}
6	6	9	8	7^{1}	4^{0}	3^{2}
3	4	8	3	8^{-1}	9^{0}	7^{3}
7	8	3	6	6	3	4
4	2	1	8	3	4	6
3	2	4	1	9	8	3
0	1	3	9	2	1	4

Imagem original 7x7 (nxm)
p = 0, s = 2

*

3	4	4
1	0	2
-1	0	3

Filtro 3x3
(fxf)

=

91	100	83

Imagem resultante 3x3
$\lfloor (n+2p-f)/s+1 \rfloor$ x $\lfloor (m+2p-f)/s+1 \rfloor$

Deslocamento ("stride")

- Exemplo de convolução com filtro 3x3, "padding", p = 0, e "stride", s = 2:

2	3	7	4	6	2	9
6	6	9	8	7	4	3
3^{3}	4^{4}	8^{4}	3	8	9	7
7^{1}	8^{0}	3^{2}	6	6	3	4
4^{-1}	2^{0}	1^{3}	8	3	4	6
3	2	4	1	9	8	3
0	1	3	9	2	1	4

Imagem original 7x7 (nxm)
p = 0, s = 2

*

3	4	4
1	0	2
-1	0	3

Filtro 3x3
(fxf)

=

91	100	83
69		

Imagem resultante 3x3
$\lfloor (n+2p-f)/s+1 \rfloor \text{x} \lfloor (m+2p-f)/s+1 \rfloor$

Deslocamento (“stride”)

- Exemplo de convolução com filtro 3x3, “padding”, $p = 0$, e “stride”, $s = 2$:

2	3	7	4	6	2	9
6	6	9	8	7	4	3
3	4	8^{3}	3^{4}	8^{4}	9	7
7	8	3^{1}	6^{0}	6^{2}	3	4
4	2	1^{-1}	8^{0}	3^{3}	4	6
3	2	4	1	9	8	3
0	1	3	9	2	1	4

Imagem original 7x7 (nxm)
$p = 0$, $s = 2$

*

3	4	4
1	0	2
-1	0	3

Filtro 3x3
(fxf)

=

91	100	83
69	91	

Imagem resultante 3x3
$\lfloor (n+2p-f)/s+1 \rfloor$ x $\lfloor (m+2p-f)/s+1 \rfloor$

Deslocamento (“stride”)

- Exemplo de convolução com filtro 3x3, “padding”, $p = 0$, e “stride”, $s = 2$:

2	3	7	4	6	2	9
6	6	9	8	7	4	3
3	4	8	3	8^{3}	9^{4}	7^{4}
7	8	3	6	6^{1}	3^{0}	4^{2}
4	2	1	8	3^{-1}	4^{0}	6^{3}
3	2	4	1	9	8	3
0	1	3	9	2	1	4

Imagem original 7x7 (*nxm*)
$p = 0$, $s = 2$

*

3	4	4
1	0	2
-1	0	3

Filtro 3x3
(*fxf*)

=

91	100	83
69	91	127

Imagem resultante 3x3
$\lfloor (n+2p-f)/s+1 \rfloor \times \lfloor (m+2p-f)/s+1 \rfloor$

Deslocamento ("stride")

- Exemplo de convolução com filtro 3x3, "padding", p = 0, e "stride", s = 2:

2	3	7	4	6	2	9
6	6	9	8	7	4	3
3	4	8	3	8	9	7
7	8	3	6	6	3	4
4^{3}	2^{4}	1^{4}	8	3	4	6
3^{1}	2^{0}	4^{2}	1	9	8	3
0^{-1}	1^{0}	3^{3}	9	2	1	4

Imagem original 7x7 (nxm)
p = 0, s = 2

*

3	4	4
1	0	2
-1	0	3

Filtro 3x3
(fxf)

=

91	100	83
69	91	127
44		

Imagem resultante 3x3
$\lfloor (n+2p-f)/s+1 \rfloor \times \lfloor (m+2p-f)/s+1 \rfloor$

Deslocamento ("stride")

- Exemplo de convolução com filtro 3x3, "padding", $p = 0$, e "stride", $s = 2$:

2	3	7	4	6	2	9
6	6	9	8	7	4	3
3	4	8	3	8	9	7
7	8	3	6	6	3	4
4	2	1^{3}	8^{4}	3^{4}	4	6
3	2	4^{1}	1^{0}	9^{2}	8	3
0	1	3^{-1}	9^{0}	2^{3}	1	4

Imagem original 7x7 (nxm)
$p = 0$, $s = 2$

*

3	4	4
1	0	2
-1	0	3

Filtro 3x3
(fxf)

=

91	100	83
69	91	127
44	72	

Imagem resultante 3x3
$\lfloor (n+2p-f)/s+1 \rfloor$ x $\lfloor (m+2p-f)/s+1 \rfloor$

Deslocamento (“stride”)

- Exemplo de convolução com filtro 3x3, “padding”, p = 0, e “stride”, s = 2:

2	3	7	4	6	2	9
6	6	9	8	7	4	3
3	4	8	3	8	9	7
7	8	3	6	6	3	4
4	2	1	8	3^{3}	4^{4}	6^{4}
3	2	4	1	9^{1}	8^{0}	3^{2}
0	1	3	9	2^{-1}	1^{0}	4^{3}

Imagem original 7x7 (nxm)
p = 0, s = 2

*

3	4	4
1	0	2
-1	0	3

Filtro 3x3
(fxf)

=

91	100	83
69	91	127
44	72	74

Imagem resultante 3x3
$\lfloor (n+2p-f)/s+1 \rfloor$ x $\lfloor (m+2p-f)/s+1 \rfloor$

Deslocamento ("stride")

Arredondamento

- Dimensão da imagem resultante do exemplo:

  $(nxm) * (fxf) = [(n + 2p - f)/s + 1]x\ [(m + 2p - f)/s + 1]$

  $(7x7) * (3x3) \text{ com } (p{=}0 \text{ e } s{=}2) = \underbrace{[(7+0-3)/2+1]x[(7+0-3)/2+1]}_{(3x3)}$

- Pode ocorrer que a divisão por 2 resulta em um número fracionário:
  - Nesse caso o resultado é arredondado para baixo;
  - A parte da máscara que não se encaixa na imagem é desconsiderada no cálculo da imagem resultante, ou seja, não são usados.

Deslocamento ("stride")

- Fórmula geral para calcular a dimensão da imagem resultante:

$$(nxm) * (fxf) = \lfloor (n + 2p - f)/s + 1 \rfloor x \lfloor (m + 2p - f)/s + 1 \rfloor$$

Arredondamento para baixo

- Exemplo de convolução de uma imagem 8x8 com filtro 3x3, "padding", $p = 0$, e "stride", $s = 3 \Rightarrow$ nesse caso parte da imagem é desconsiderada.

Deslocamento ("stride")

2^{3}	3^{4}	7^{4}	4	6	2	9	8
6^{1}	6^{0}	9^{2}	8	7	4	3	1
3^{-1}	4^{0}	8^{3}	3	8	9	7	2
7	8	3	6	6	3	4	5
4	2	1	8	3	4	6	7
3	2	4	1	9	8	3	0
0	1	3	9	2	1	4	3
5	7	2	0	5	6	7	4

Imagem original 8x8 (nxm)

$p = 0$, $s = 2$

*

3	4	4
1	0	2
-1	0	3

Filtro 3x3

(fxf)

=

91		

Imagem resultante

$\lfloor (n+2p-f)/s+1 \rfloor \times \lfloor (m+2p-f)/s+1 \rfloor$

$\lfloor 3{,}5 \rfloor \times \lfloor 3{,}5 \rfloor = 3 \times 3$

Deslocamento ("stride")

2	3	7^{3}	4^{4}	6^{4}	2	9	8
6	6	9^{1}	8^{0}	7^{2}	4	3	1
3	4	8^{-1}	3^{0}	8^{3}	9	7	2
7	8	3	6	6	3	4	5
4	2	1	8	3	4	6	7
3	2	4	1	9	8	3	0
0	1	3	9	2	1	4	3
5	7	2	0	5	6	7	4

Imagem original 8x8 (nxm)
$p = 0$, $s = 2$

*

3	4	4
1	0	2
-1	0	3

Filtro 3x3
(fxf)

=

91	100	

Imagem resultante
$\lfloor (n+2p-f)/s+1 \rfloor \times \lfloor (m+2p-f)/s+1 \rfloor$
$\lfloor 3{,}5 \rfloor \times \lfloor 3{,}5 \rfloor = 3\text{x}3$

Deslocamento (“stride”)

2	3	7	4	6^{3}	2^{4}	9^{4}	8
6	6	9	8	7^{1}	4^{0}	3^{2}	1
3	4	8	3	8^{-1}	9^{0}	7^{3}	2
7	8	3	6	6	3	4	5
4	2	1	8	3	4	6	7
3	2	4	1	9	8	3	0
0	1	3	9	2	1	4	3
5	7	2	0	5	6	7	4

Imagem original 8x8 (*nxm*)

$p = 0$, $s = 2$

*

3	4	4
1	0	2
-1	0	3

Filtro 3x3 (*fxf*)

=

91	100	83

Imagem resultante

$\lfloor (n+2p-f)/s+1 \rfloor \times \lfloor (m+2p-f)/s+1 \rfloor$

$\lfloor 3{,}5 \rfloor \times \lfloor 3{,}5 \rfloor = 3\text{x}3$

Deslocamento ("stride")

Máscara não chega na última coluna

2	3	7	4	6	2	9	8
6	6	9	8	7	4	3	1
3	4	8	3	8	9	7	2
7	8	3	6	6	3	4	5
4	2	1	8	3	4	6	7
3	2	4	1	9	8	3	0
0	1	3	9	2	1	4	3
5	7	2	0	5	6	7	4

Imagem original 8x8 (*nxm*)

$p = 0$, $s = 2$

*

3	4	4
1	0	2
-1	0	3

Filtro 3x3

(*fxf*)

=

91	100	83

Imagem resultante

$\lfloor (n+2p-f)/s+1 \rfloor \times \lfloor (m+2p-f)/s+1 \rfloor$

$\lfloor 3{,}5 \rfloor \times \lfloor 3{,}5 \rfloor = 3\times3$

Deslocamento ("stride")

2	3	7	4	6	2	9	8
6	6	9	8	7	4	3	1
3	4	8	3	8	9	7	2
7	8	3	6	6	3	4	5
4^{3}	2^{4}	1^{4}	8	3	4	6	7
3^{1}	2^{0}	4^{2}	1	9	8	3	0
0^{-1}	1^{0}	3^{3}	9	2	1	4	3
5	7	2	0	5	6	7	4

Imagem original 8x8 (*nxm*)
$p = 0$, $s = 2$

*

3	4	4
1	0	2
-1	0	3

Filtro 3x3
(*fxf*)

=

91	100	83
69	91	127
44		

Imagem resultante
$\lfloor (n+2p-f)/s+1 \rfloor \times \lfloor (m+2p-f)/s+1 \rfloor$
$\lfloor 3,5 \rfloor \times \lfloor 3,5 \rfloor = 3\text{x}3$

Deslocamento ("stride")

2	3	7	4	6	2	9	8
6	6	9	8	7	4	3	1
3	4	8	3	8	9	7	2
7	8	3	6	6	3	4	5
4	2	1	8	3	4	6	7
3	2	4	1	9	8	3	0
0	1	3	9	2	1	4	3
5	7	2	0	5	6	7	4

Imagem original 8x8 (*nxm*)
$p = 0$, $s = 2$

Máscara não chega na última coluna

*

3	4	4
1	0	2
-1	0	3

Filtro 3x3
(*fxf*)

=

91	100	83
69	91	127
44	72	74

Imagem resultante
$\lfloor (n+2p-f)/s+1 \rfloor \times \lfloor (m+2p-f)/s+1 \rfloor$
$\lfloor 3{,}5 \rfloor \times \lfloor 3{,}5 \rfloor = 3\text{x}3$

Deslocamento ("stride")

2	3	7	4	6	2	9	8
6	6	9	8	7	4	3	1
3	4	8	3	8	9	7	2
7	8	3	6	6	3	4	5
4	2	1	8	3	4	6	7
3	2	4	1	9	8	3	0
0	1	3	9	2	1	4	3
5	7	2	0	5	6	7	4

Imagem original 8x8 (*nxm*)

$p = 0$, $s = 2$

*

3	4	4
1	0	2
-1	0	3

Filtro 3x3

(*fxf*)

=

91	100	83
69	91	127
44	72	74

Imagem resultante

$\lfloor (n+2p-f)/s+1 \rfloor \times \lfloor (m+2p-f)/s+1 \rfloor$

$\lfloor 3{,}5 \rfloor \times \lfloor 3{,}5 \rfloor = 3\text{x}3$

Máscara não chega na última linha

Deslocamento ("stride")

Pixels descartados

2	3	7	4	6	2	9	8
6	6	9	8	7	4	3	1
3	4	8	3	8	9	7	2
7	8	3	6	6	3	4	5
4	2	1	8	3	4	6	7
3	2	4	1	9	8	3	0
0	1	3	9	2	1	4	3
5	7	2	0	5	6	7	4

Imagem original 8x8 (*nxm*)

$p = 0$, $s = 2$

*

3	4	4
1	0	2
-1	0	3

Filtro 3x3

(*fxf*)

=

91	100	83
69	91	127
44	72	74

Imagem resultante

$\lfloor (n+2p-f)/s+1 \rfloor \times \lfloor (m+2p-f)/s+1 \rfloor$

$\lfloor 3{,}5 \rfloor \times \lfloor 3{,}5 \rfloor = 3\text{x}3$

Convolução em volume

- Imagens coloridas tem 3 canais (RGB) $\Rightarrow$ como realizar convolução em imagens de vários canais?
- Convolução em imagem RGB é realizada com um filtro 3D.
- Filtro 3D:
  - Imagem de dimensão (nxmx3);
  - Filtro de dimensão (fxfx3) $\Rightarrow$ filtro deve ter o mesmo número de canais que a imagem;
  - **Número de canais da imagem = número de canais do filtro.**
  - Dimensão da imagem resultante da convolução usando $p = 0$, $s = 1$:

(nxmx3) * (fxfx3) = (n-f+1)x(m-f+1) $\Rightarrow$ imagem resultante tem somente um canal.

Convolução em volume

- Exemplo: imagem (8x8x3), com filtro (3x3x3), p = 0, s = 1:

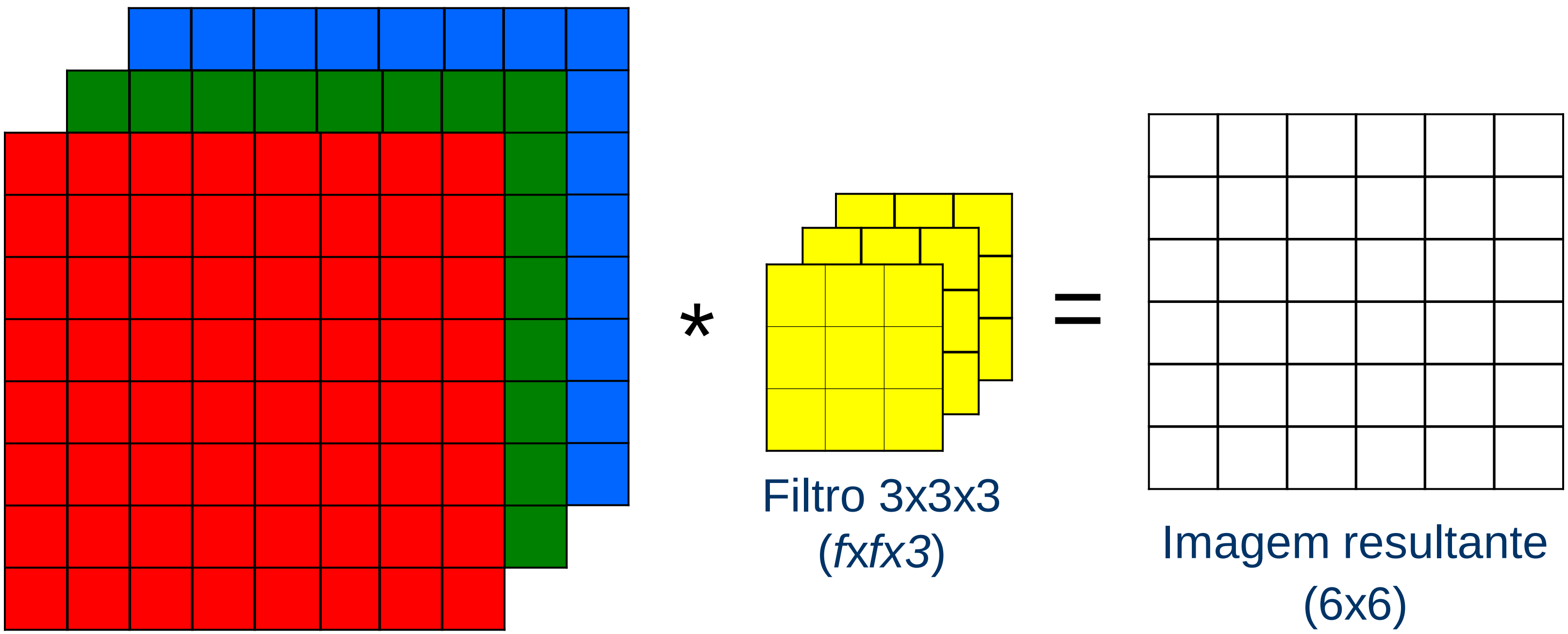


Imagem original 8x8x3 (nxmx3)

Filtro 3x3x3 (fxfx3)

Imagem resultante (6x6) (n-f+1)x(m-f+1)

Convolução em volume

- Exemplo: imagem (8x8x3), com filtro (3x3x3), $p = 0$, $s = 1$:
  - Filtro 3x3x3 possui 27 parâmetros;
  - Multiplica-se os 27 parâmetros do filtro pelos pixels correspondentes na imagem nos 3 canais e adiciona todos os valores, resultando no valor do pixel da nova imagem;
  - Passando o filtro em toda a imagem, como feito para uma imagem de 1 canal, tem-se a imagem resultante.
- Convolução em volume $\Rightarrow$ convolução de um volume (várias imagens) por um filtro que é também um volume (vários filtros)

Convolução em volume

- Exemplo: imagem (8x8x3), com filtro (3x3x3), *p* = 0,*s* = 1:

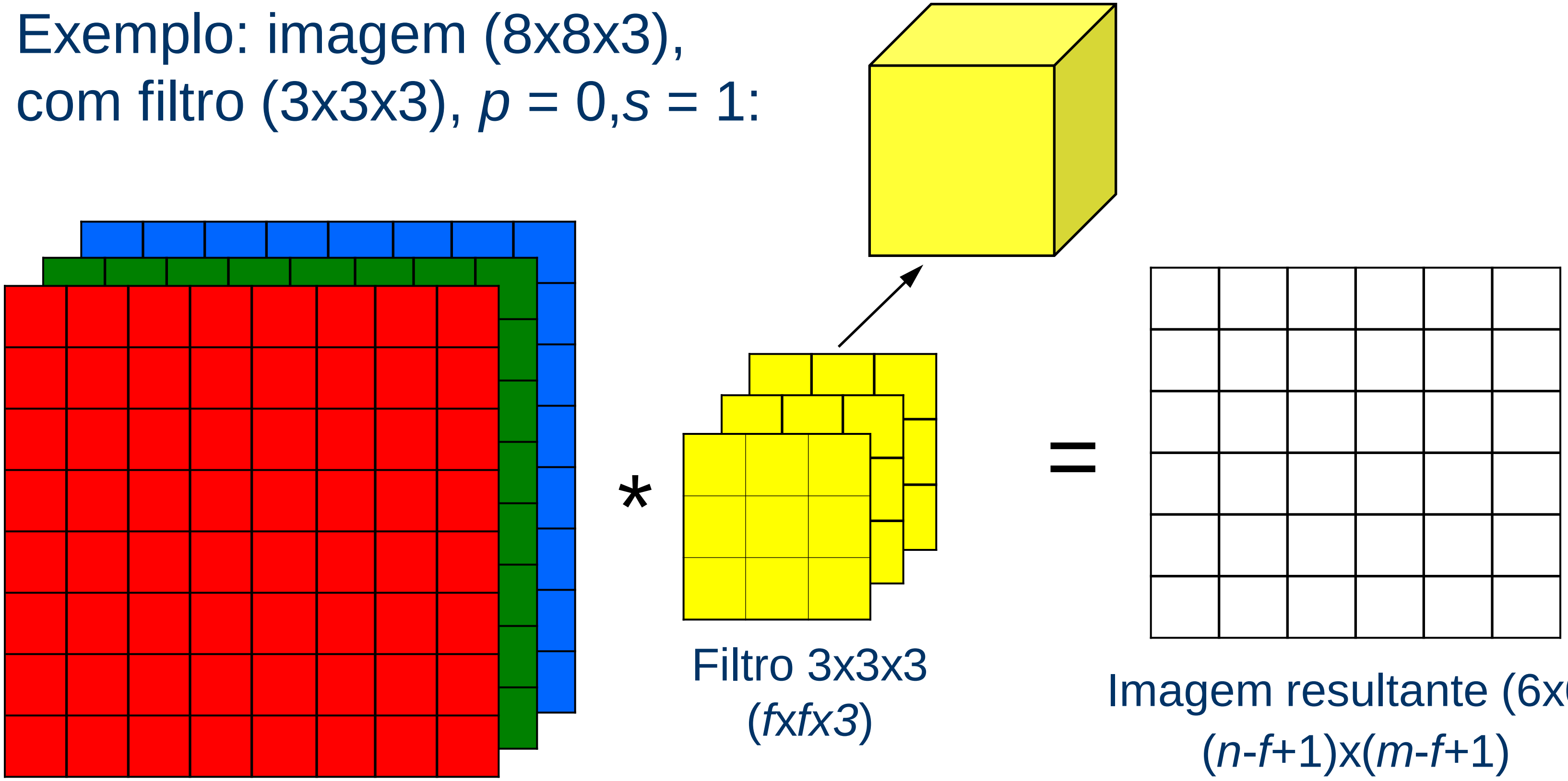


Imagem original 8x8x3 (*nxmx3*)

Filtro 3x3x3 (*fxfx3*)

Imagem resultante (6x6) (*n*-*f*+1)x(*m*-*f*+1)

Convolução em volume

- Exemplo: imagem (8x8x3), com filtro (3x3x3), *p* = 0, *s* = 1:

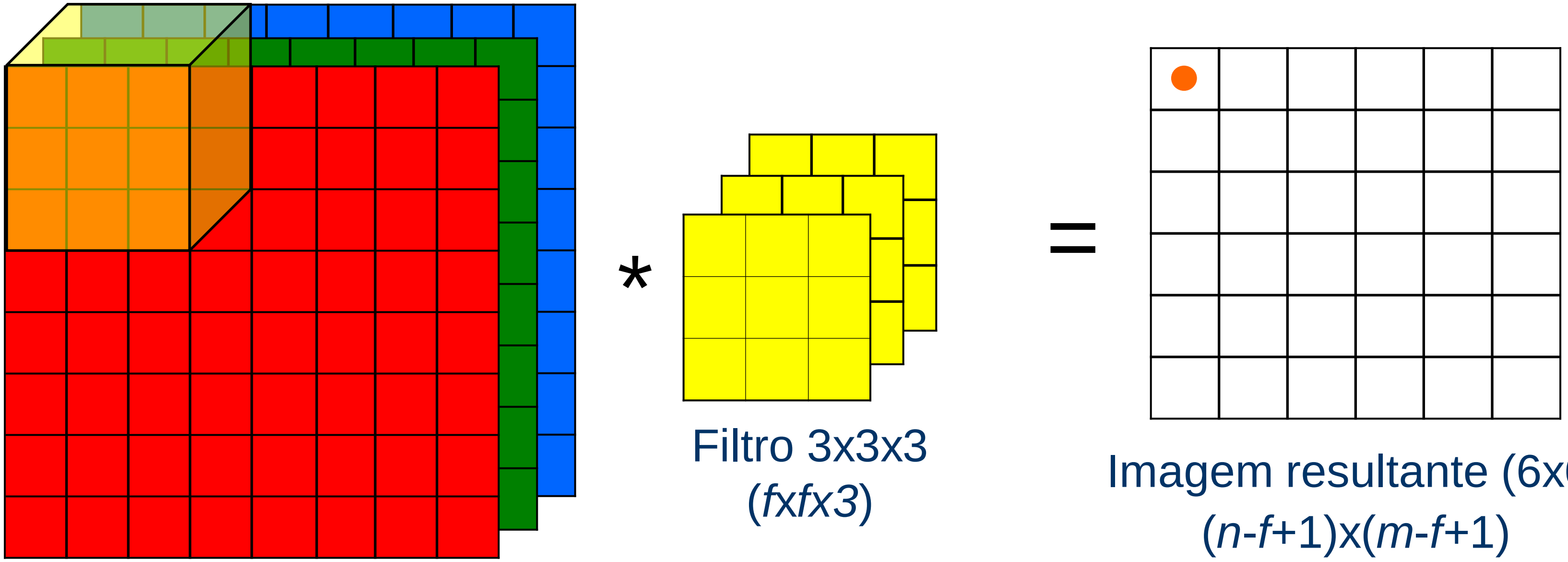


Imagem original 8x8x3
(*nxmx3*)

Filtro 3x3x3
(*fxfx3*)

Imagem resultante (6x6)
(*n*-*f*+1)x(*m*-*f*+1)

Convolução em volume

- Exemplo: imagem (8x8x3), com filtro (3x3x3), *p* = 0, *s* = 1:

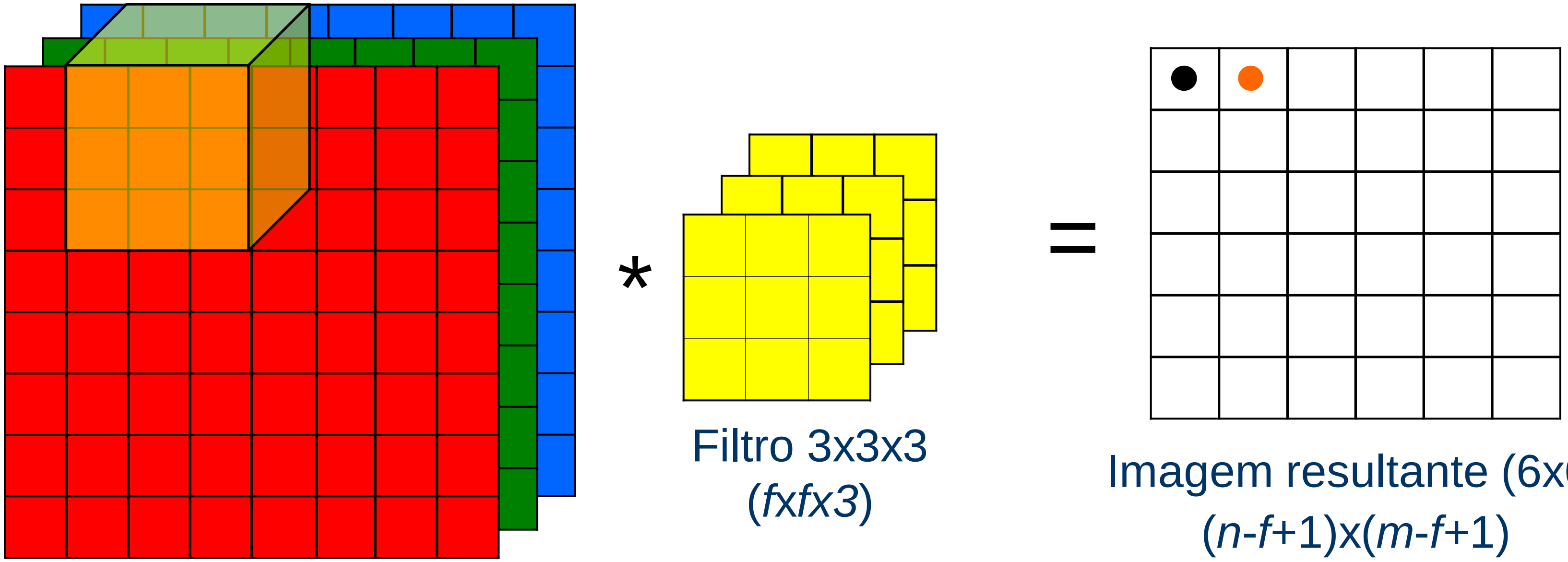


Imagem original 8x8x3
(*nxmx3*)

Filtro 3x3x3
(*fxfx3*)

Imagem resultante (6x6)
(*n*-*f*+1)x(*m*-*f*+1)

Convolução em volume

- Exemplo: imagem (8x8x3), com filtro (3x3x3), *p* = 0, *s* = 1:

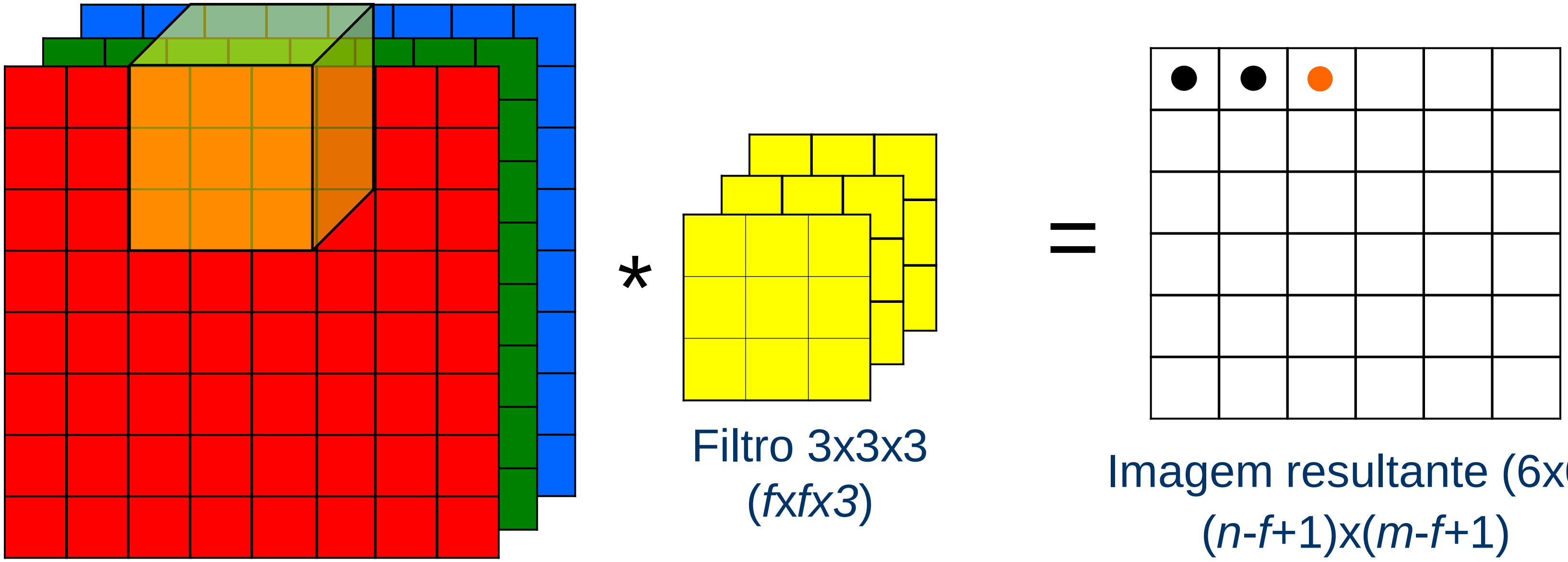


Imagem original 8x8x3
(*nxmx3*)

Filtro 3x3x3
(*fxfx3*)

Imagem resultante (6x6)
(*n*-*f*+1)x(*m*-*f*+1)

Convolução em volume

- Exemplo: imagem (8x8x3), com filtro (3x3x3), *p* = 0, *s* = 1:

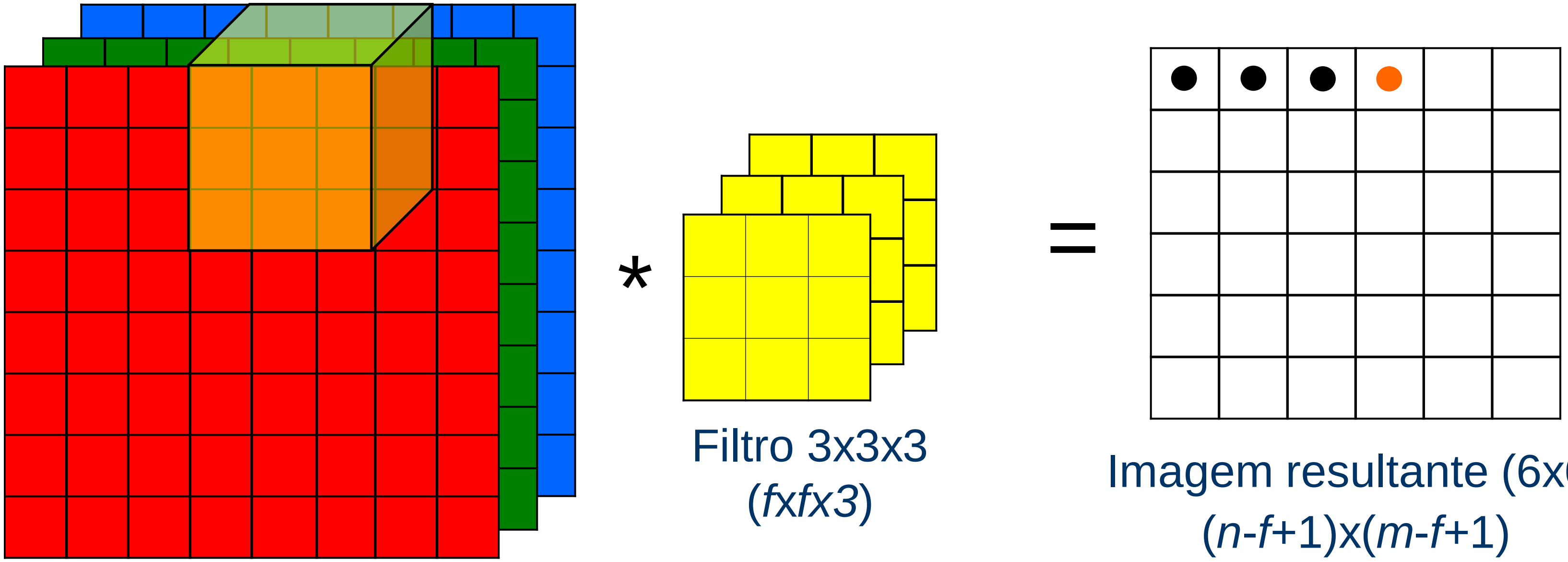


Imagem original 8x8x3 (*nxmx3*)

Filtro 3x3x3 (*fxfx3*)

Imagem resultante (6x6) (*n*-*f*+1)x(*m*-*f*+1)

Convolução em volume

- Exemplo: imagem (8x8x3), com filtro (3x3x3), *p* = 0, *s* = 1:

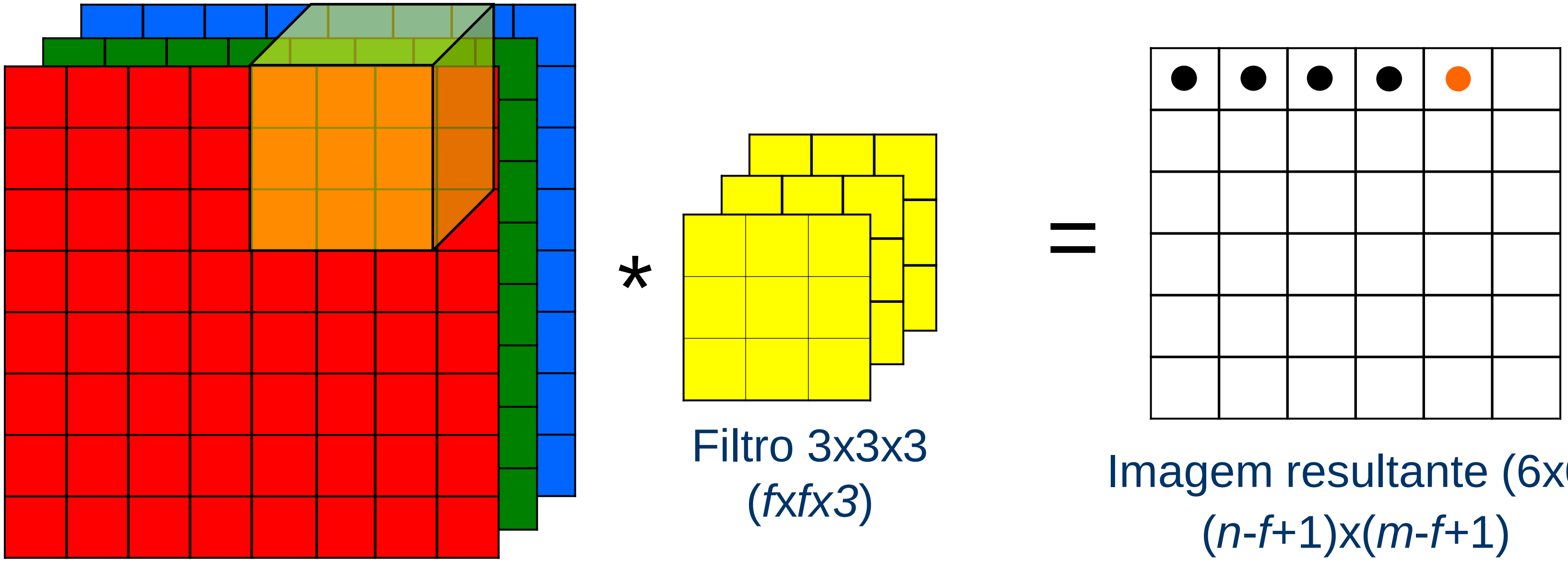


Imagem original 8x8x3
(*nxmx3*)

Filtro 3x3x3
(*fxfx3*)

Imagem resultante (6x6)
(*n*-*f*+1)x(*m*-*f*+1)

Convolução em volume

- Exemplo: imagem (8x8x3), com filtro (3x3x3), *p* = 0, *s* = 1:

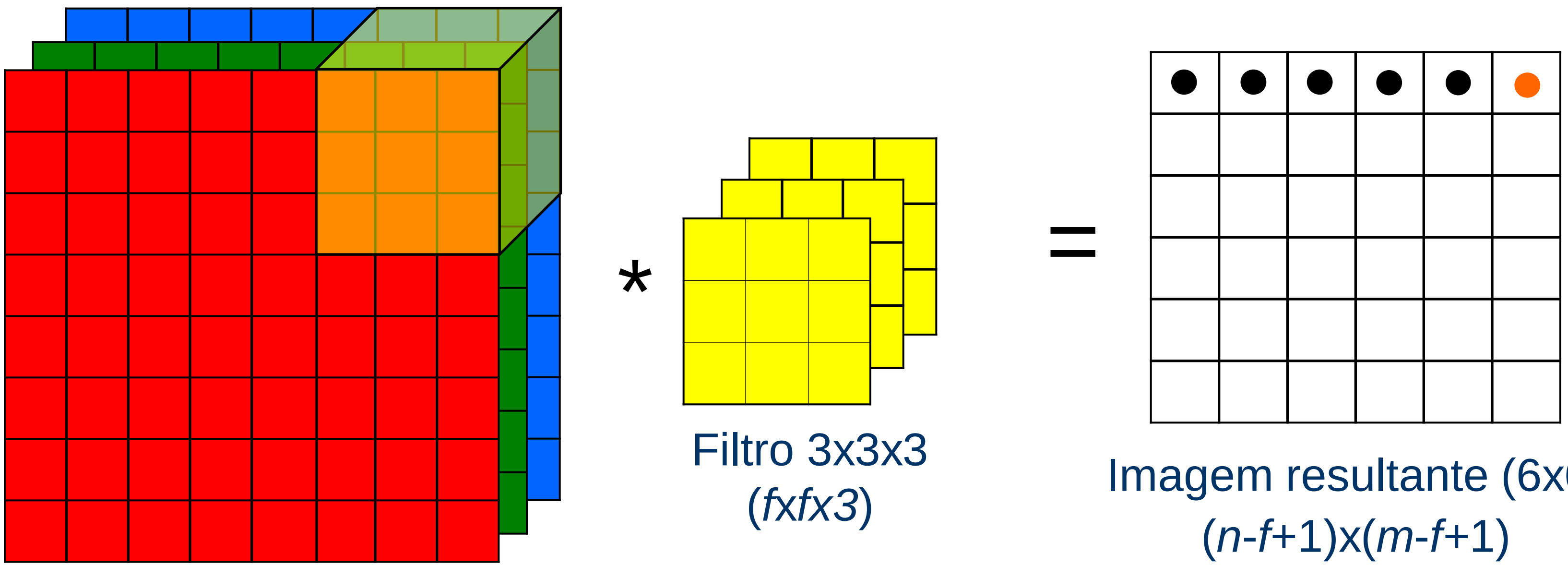


Imagem original 8x8x3
(*nxmx3*)

Filtro 3x3x3
(*fxfx3*)

Imagem resultante (6x6)
(*n*-*f*+1)x(*m*-*f*+1)

Convolução em volume

- Exemplo: imagem (8x8x3), com filtro (3x3x3), p = 0, s = 1:

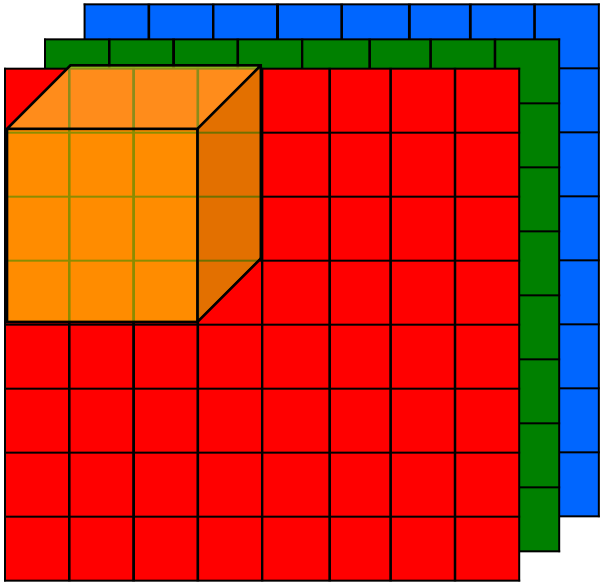
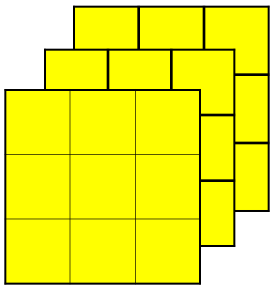


Imagem original 8x8x3
(*nxmx3*)

*



Filtro 3x3x3
(*fxfx3*)

=

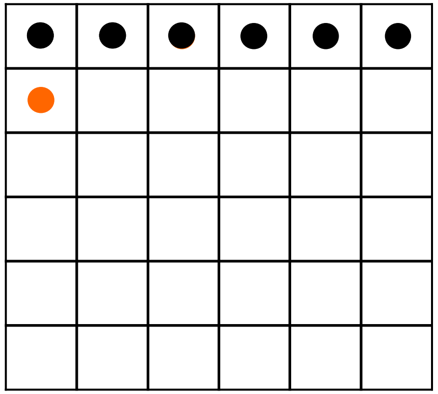


Imagem resultante (6x6)
(n-f+1)x(m-f+1)

Convolução em volume

- Exemplo: imagem (8x8x3), com filtro (3x3x3), *p* = 0, *s* = 1:

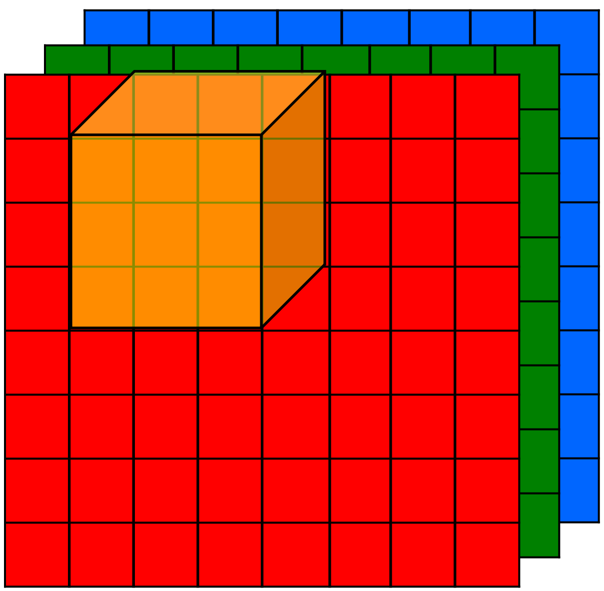
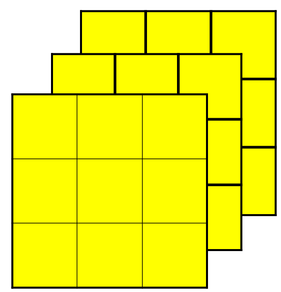


Imagem original 8x8x3
(*nxmx3*)

*



Filtro 3x3x3
(*fxfx3*)

=

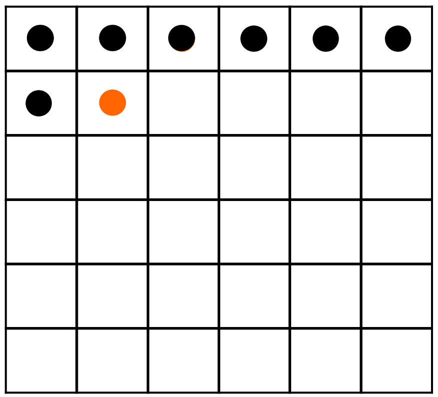


Imagem resultante (6x6)
(*n*-*f*+1)x(*m*-*f*+1)

Convolução em volume

- Exemplo: imagem (8x8x3), com filtro (3x3x3), p = 0, s = 1:

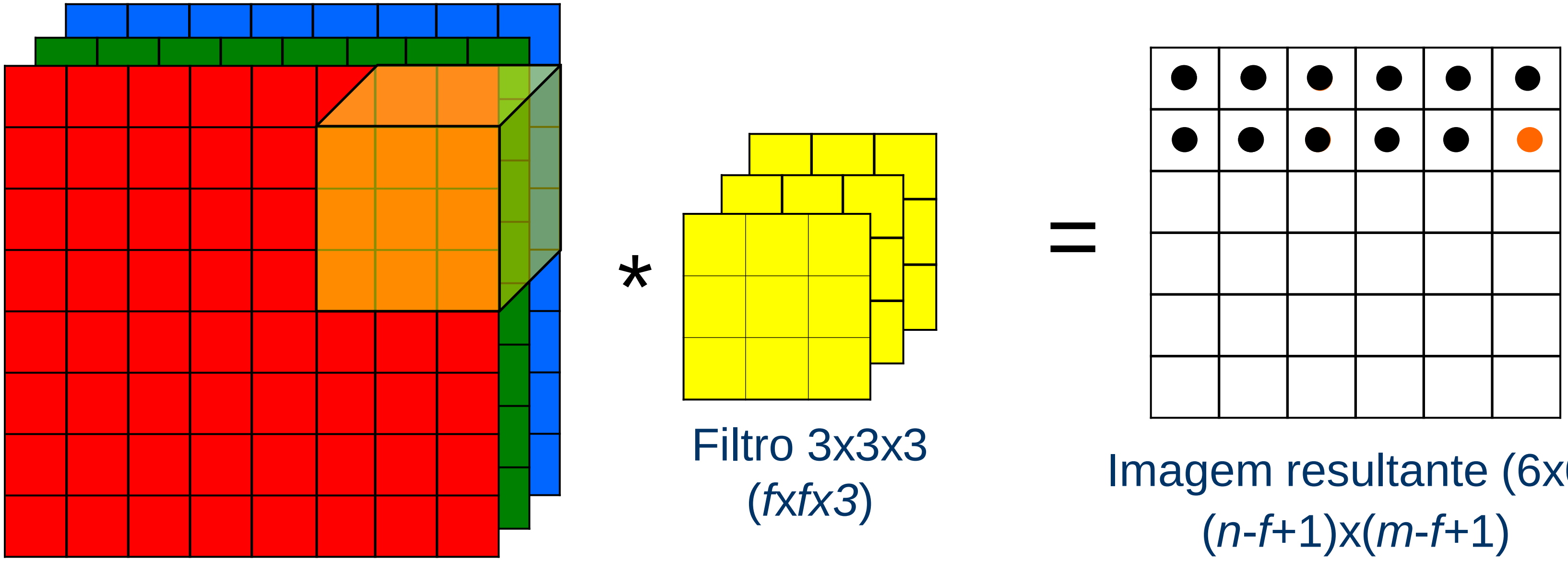


Imagem original 8x8x3 (*nxmx3*)

Filtro 3x3x3 (*fxfx3*)

Imagem resultante (6x6) (n-f+1)x(m-f+1)

Convolução em volume

- Exemplo: imagem (8x8x3), com filtro (3x3x3), *p* = 0, *s* = 1:

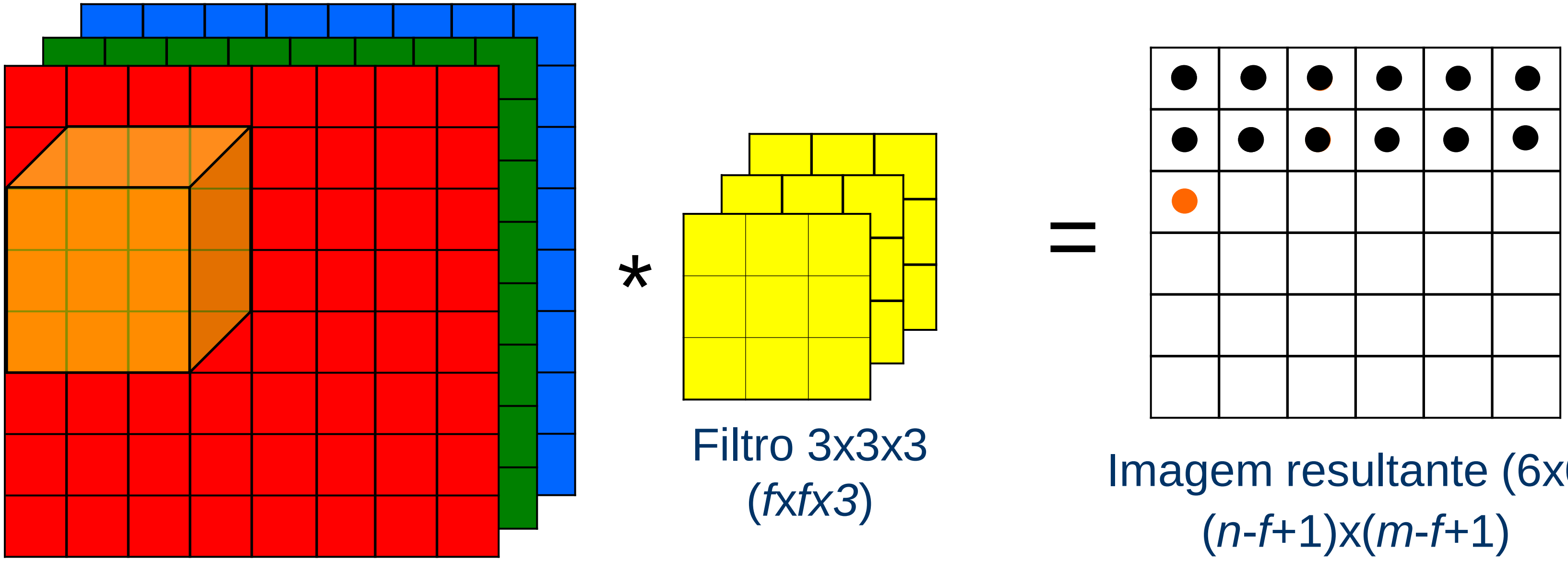


Imagem original 8x8x3 (*nxmx3*)

Filtro 3x3x3 (*fxfx3*)

Imagem resultante (6x6) (*n*-*f*+1)x(*m*-*f*+1)

Convolução em volume

- Exemplo: imagem (8x8x3), com filtro (3x3x3), *p* = 0, *s* = 1:

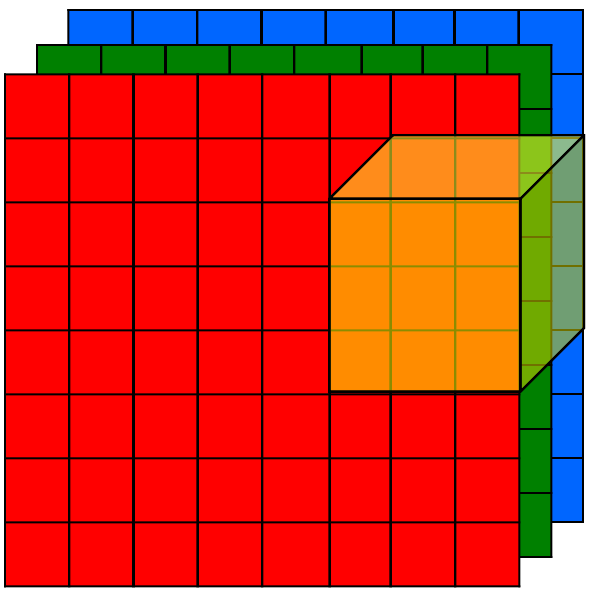
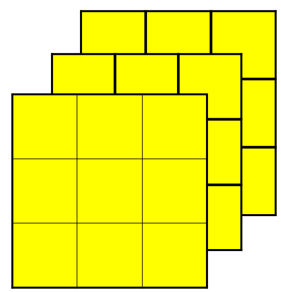


Imagem original 8x8x3
(*nxmx3*)

*



Filtro 3x3x3
(*fxfx3*)

=

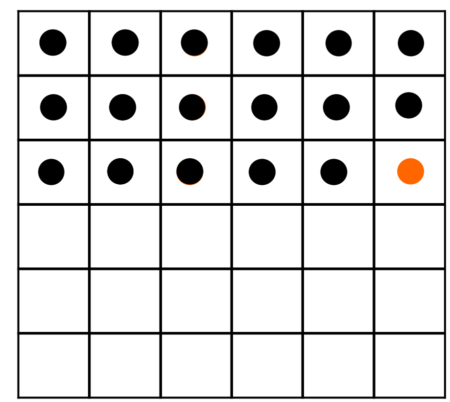
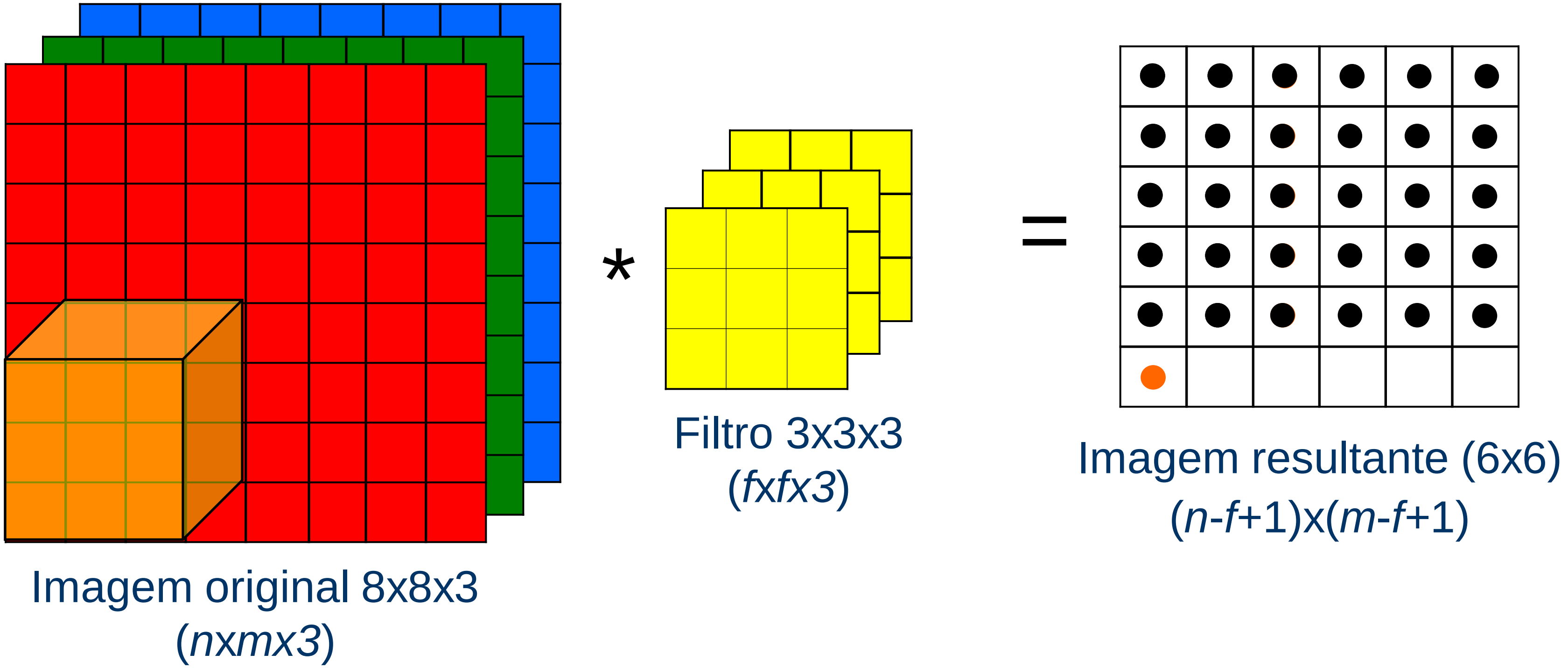


Imagem resultante (6x6)
(*n*-*f*+1)x(*m*-*f*+1)

Convolução em volume

- Exemplo: imagem (8x8x3), com filtro (3x3x3), *p* = 0, *s* = 1:



Convolução em volume

- Exemplo: imagem (8x8x3), com filtro (3x3x3), *p* = 0, *s* = 1:

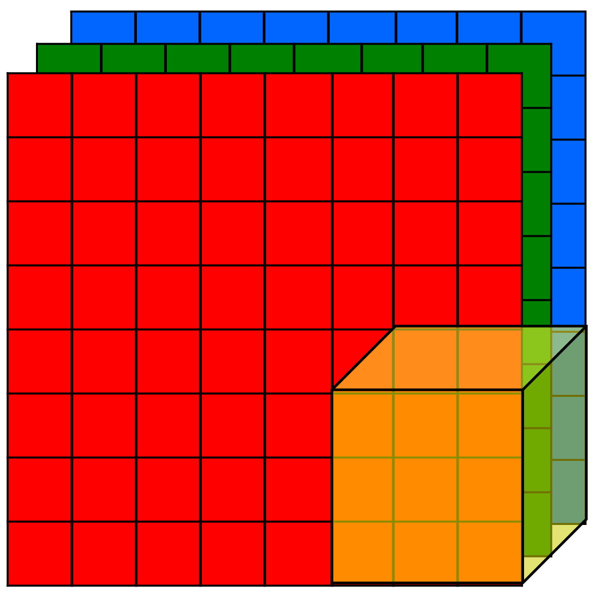
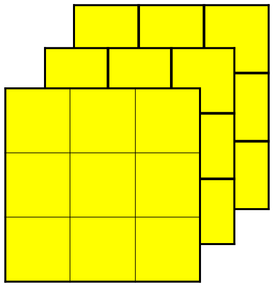


Imagem original 8x8x3
(*nxmx3*)

*



Filtro 3x3x3
(*fxfx3*)

=

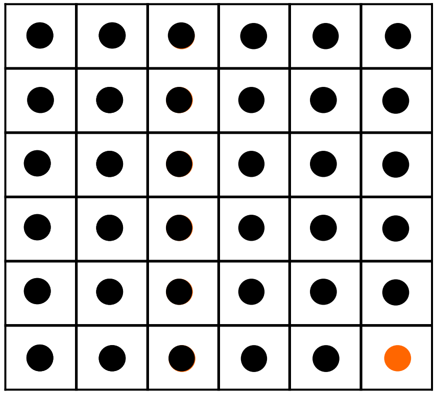
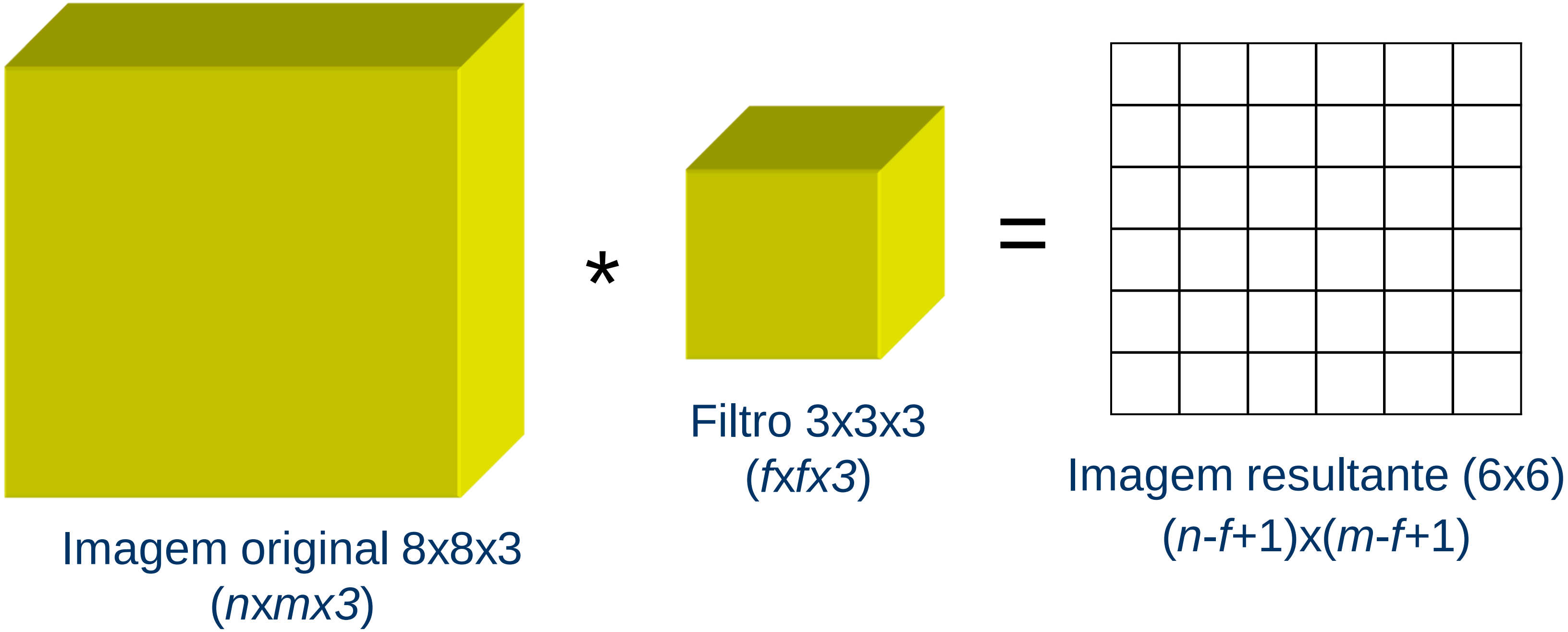


Imagem resultante (6x6)
(*n*-*f*+1)x(*m*-*f*+1)

Convolução em volume

- Convolução de um volume (8x8x3) por um filtro 3D (volume, 3x3x3) $\Rightarrow$ imagem (6x6)



Convolução em volume

- Com será um filtro 3D para detectar bordas verticais na cor vermelha?

Filtro 3D (3x3)

R

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

G

0	0	0
0	0	0
0	0	0

B

0	0	0
0	0	0
0	0	0

- Filtro 3D (3x3) para detectar bordas horizontais na cor azul:

R

0	0	0
0	0	0
0	0	0

G

0	0	0
0	0	0
0	0	0

B

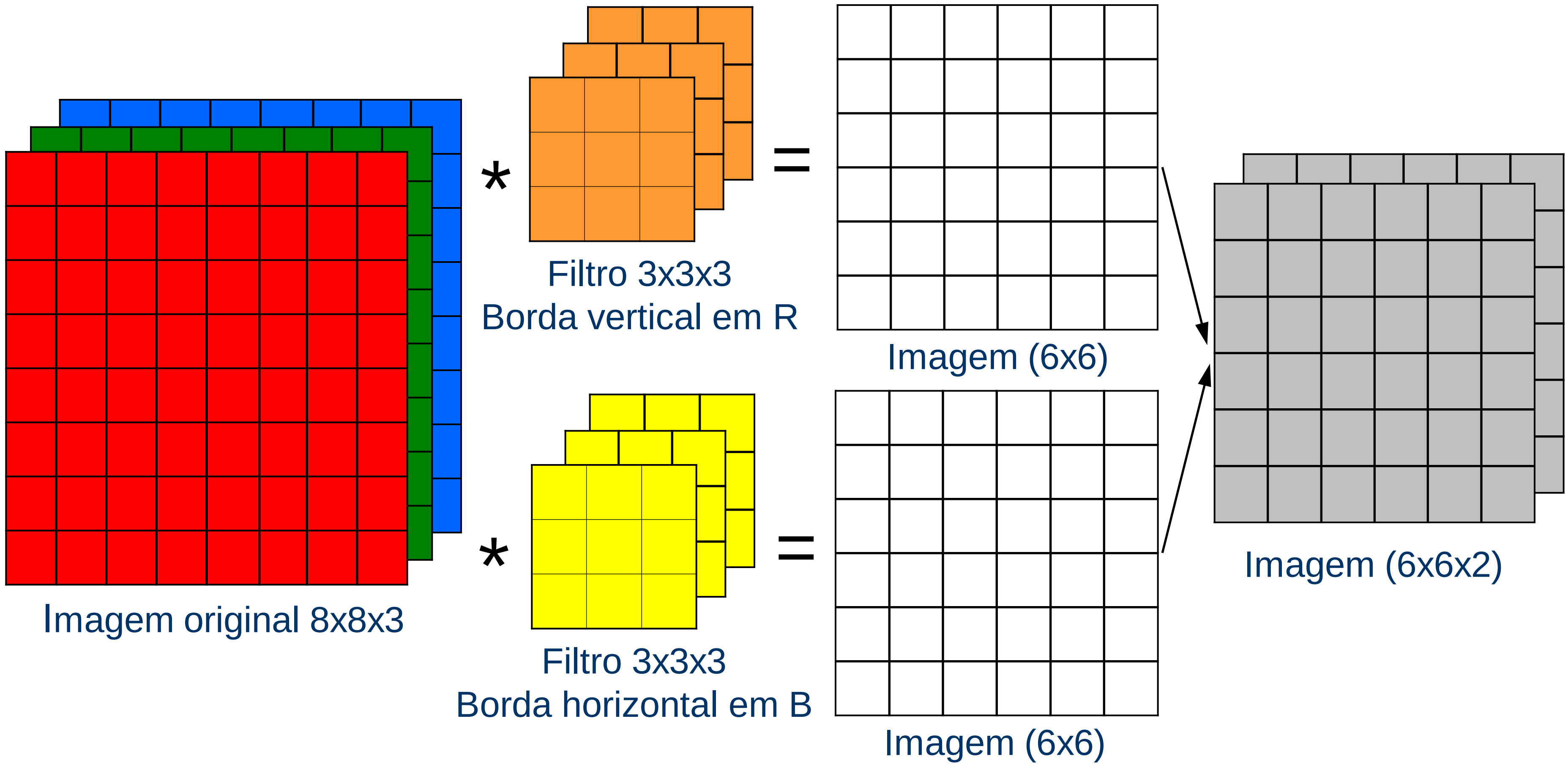
1	1	1
0	0	0
-1	-1	-1

Convolução em volume

- Pode-se usar múltiplos filtros 3D, por exemplo:
  - Um filtro para detectar borda vertical na imagem vermelha $\Rightarrow$ $(n \text{x} m \text{x} 3) * (f \text{x} f \text{x} 3) \Rightarrow (n-f+1) \text{x} (m-f+1)$;
  - Um filtro para detectar borda horizontal na imagem azul $\Rightarrow$ $(n \text{x} m \text{x} 3) * (f \text{x} f \text{x} 3) \Rightarrow (n-f+1) \text{x} (m-f+1)$;
  - Unindo as duas imagens resultantes $\Rightarrow$ tem-se um volume final de dimensão $(n-f+1) \text{x} (m-f+1) \text{x} 2$.

Convolução em volume

- Exemplo: imagem (8x8x3), com filtro (3x3x3x2), *p* = 0, *s* = 1:



Convolução em volume

- Convolução de um volume (8x8x3) por dois filtros 3D (3x3x3) $\Rightarrow$ imagem (6x6x2)

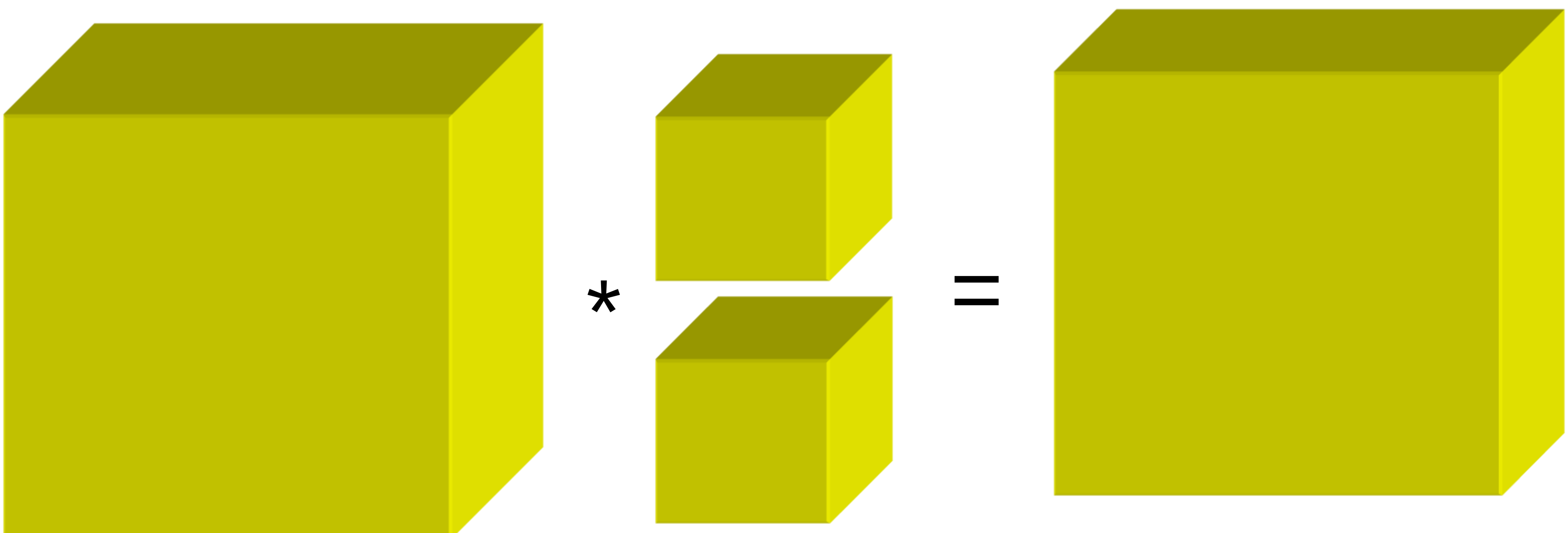


Imagem original 8x8x3
(*nxmx3*)

2 filtros 3x3x3
(*fxfx3*)x2

Imagem resultante (6x6x2)
$(n\text{-}f+1)\text{x}(m\text{-}f+1)\text{x}n_c$

Convolução em volume

- Usando vários filtros é possível detectar muitas características na imagem.
- Considerando “padding” e “stride”, tem a fórmula geral para a dimensão do volume resultante:

$$\boxed{(n x m x n_c) * (f x f x n_c) x n_f = \lfloor (n + 2p - f)/s + 1 \rfloor x \lfloor (m + 2p - f)/s + 1 \rfloor x n_f}$$

- Por exemplo:

$$\left.\begin{array}{l} - \ n = 32 \\ - \ m = 64 \\ - \ n_c = 3 \\ - \ f = 5; \\ - \ n_f = 10 \\ - \ s = 2 \\ - \ p = 1 \end{array}\right\} \Rightarrow \text{volume resultante (15x31x10)}$$

Convolução nas RNAs

- Na operação de convolução real os cálculos são realizados com a máscara (filtro) invertida de cima para baixo e da esquerda para a direita.

3	4	5
1	0	2
-1	9	7

$\Rightarrow$

7	2	5
0	0	4
-1	1	3

- Nas RNAs não é realizada a operação de inversão da máscara.
- De fato o que se realiza nas RNAs é uma operação de correlação cruzada da máscara com a imagem, mas mesmo assim é chamada de convolução.

Convolução nas RNAs

- A não inversão da máscara é feito para simplificação e não afeta em nada os resultados.
- A convolução apresenta a propriedade distributiva que a correlação cruzada não apresenta.
- Propriedade distributiva $\Rightarrow$ $(\mathbf{A} * \mathbf{B}) * \mathbf{C} = \mathbf{A} * (\mathbf{B} * \mathbf{C})$