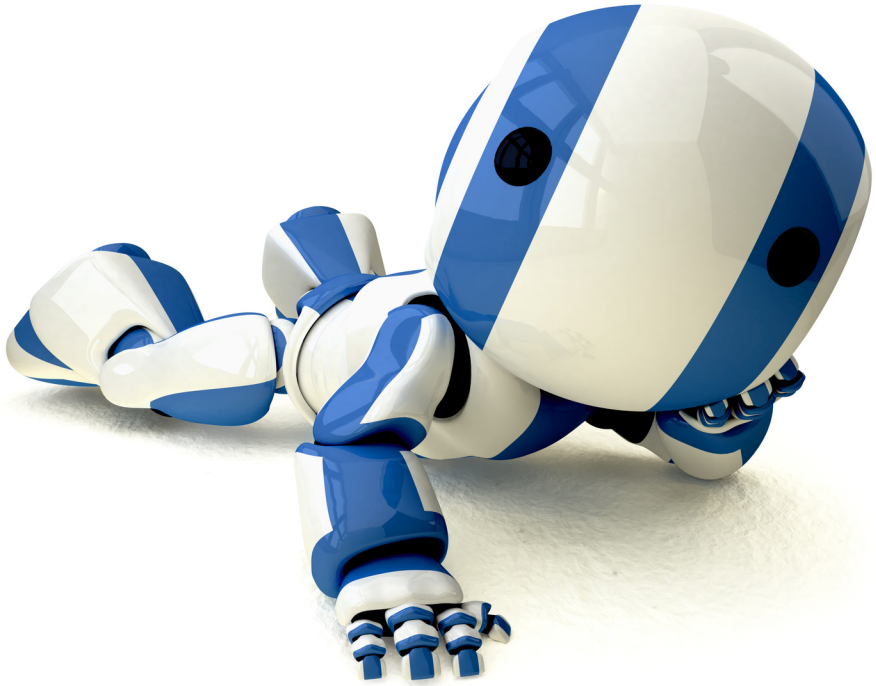
PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
ESCOLA POLITÉCNICA
CURSO DE EXTENSÃO EM DATA SCIENCE

Aprendizado de Máquina Supervisionado

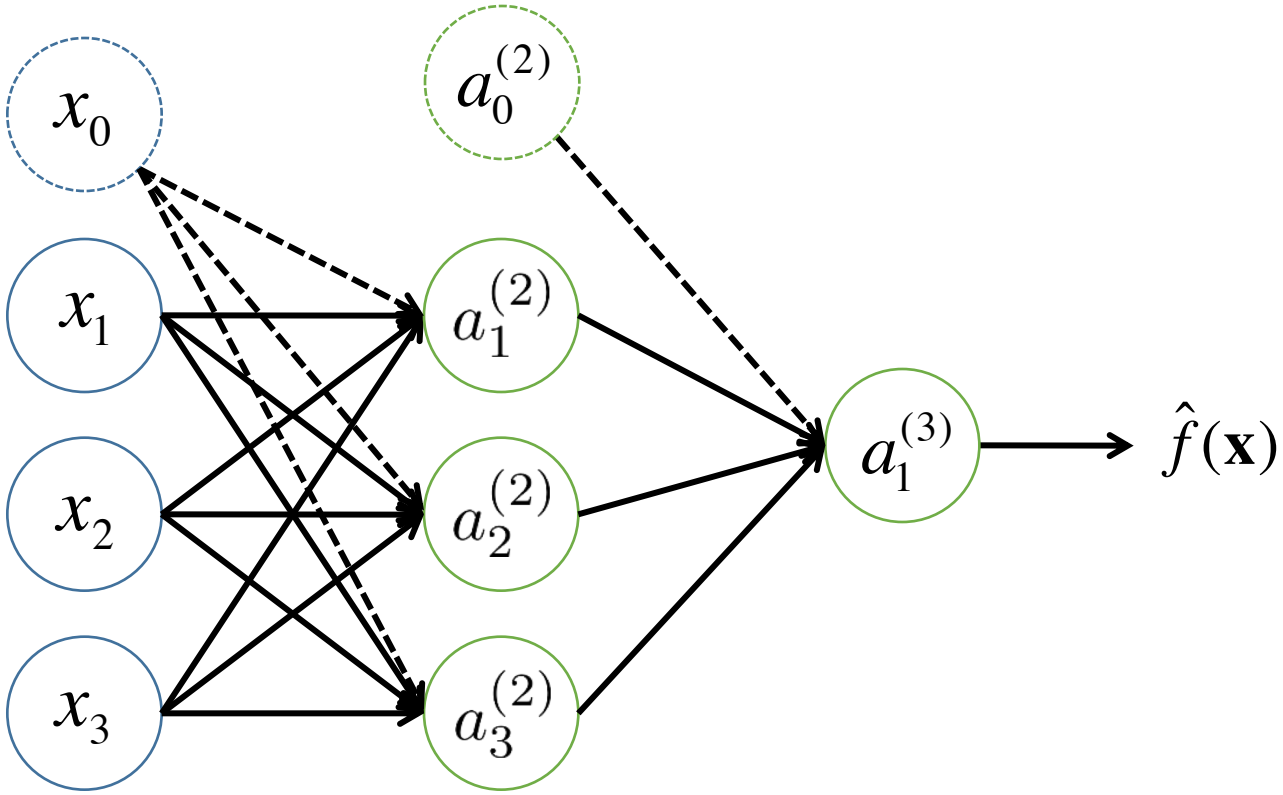


Paradigma baseado em Otimização
Parte III: Redes Neurais Convolucionais

Prof. Dr. Rodrigo C. Barros

BUSINESS INTELLIGENCE AND
MACHINE LEARNING RESEARCH GROUP

Aula Passada



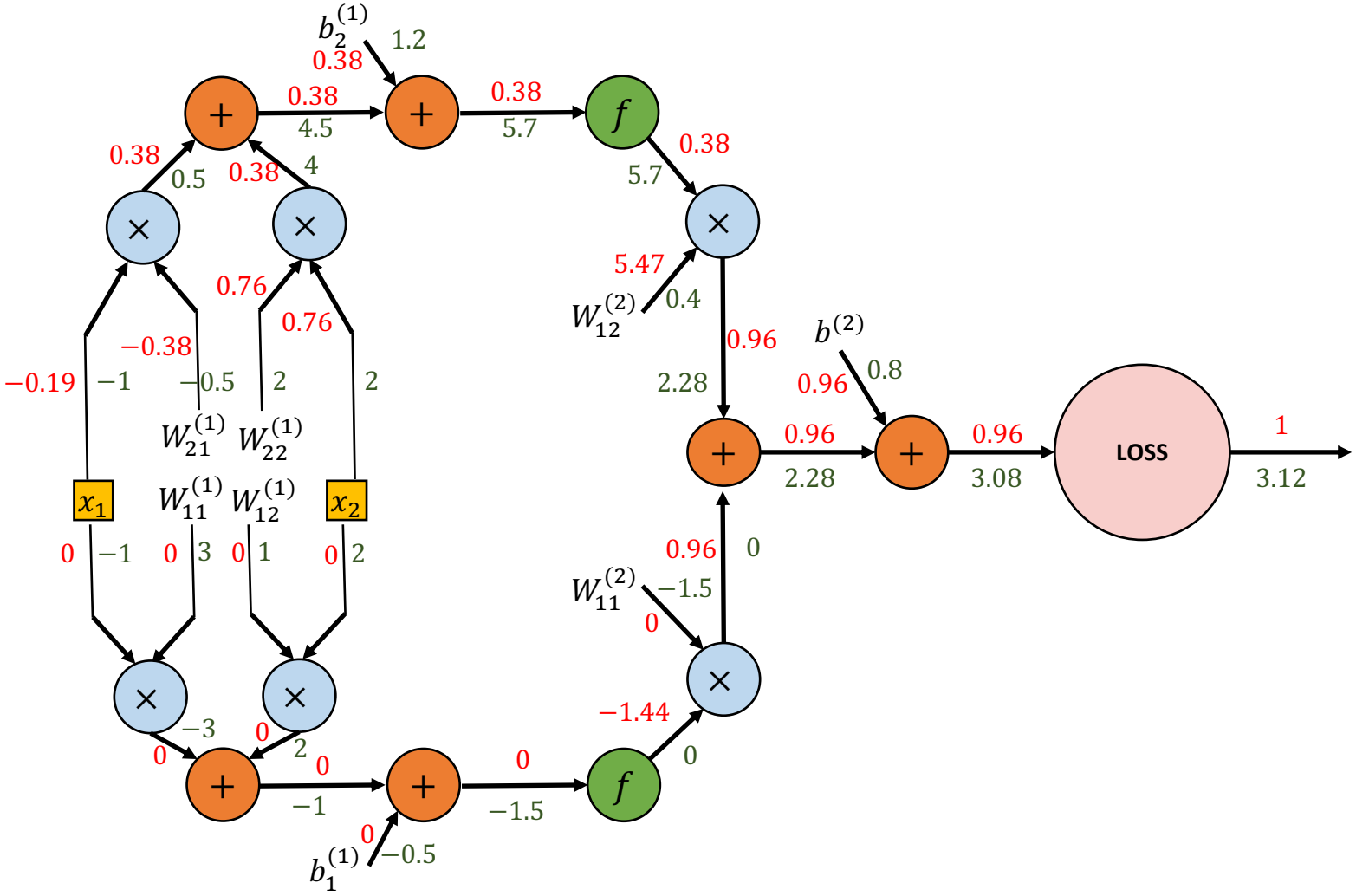
Camada 1 Camada 2 Camada 3

camada de entrada

camada escondida

camada de saída

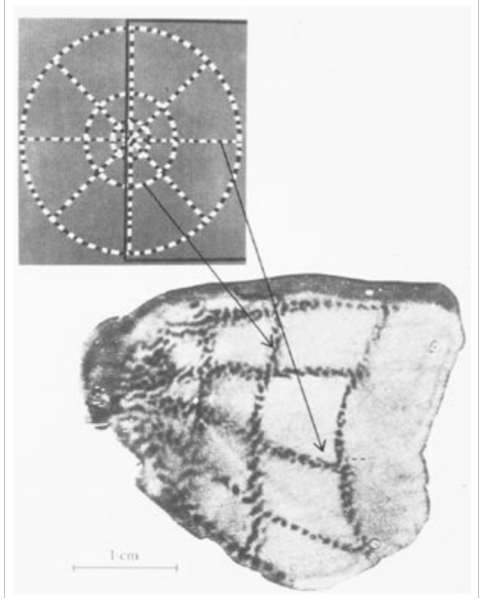
Aula Passada

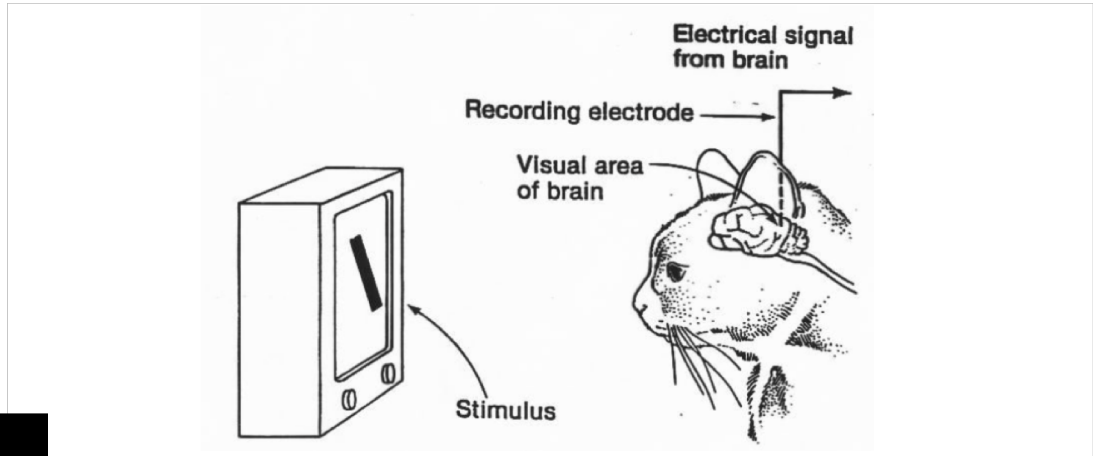


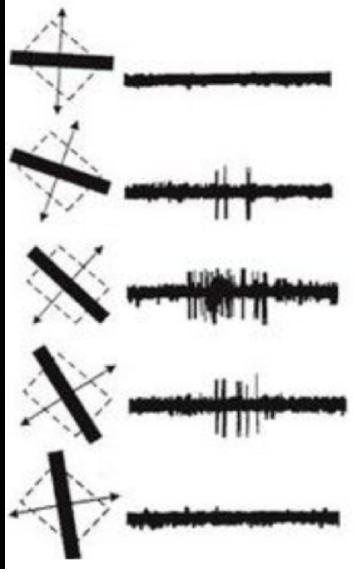
Agenda

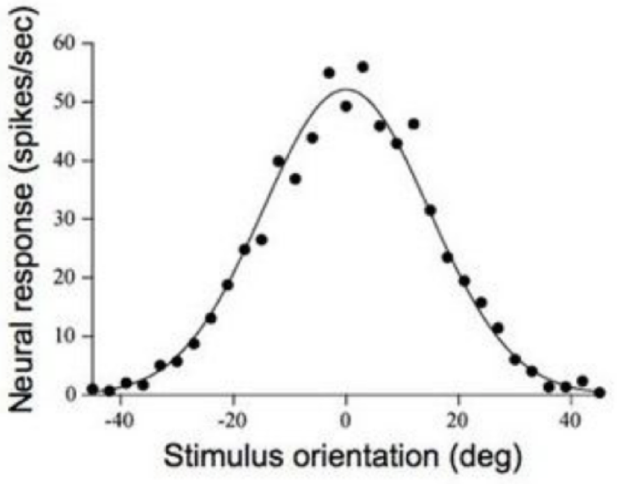
- Introdução a Redes Neurais Convolucionais
  - *Background* e Motivação
  - Camada de Convolução
  - Cálculo de Variáveis Importantes
  - Camada de *Pooling*
  - Camada *Fully-Connected*
  - Intuições e Analogia com Córtex Visual
- Exemplos de Arquiteturas
- Aplicações

Um pouco de história...



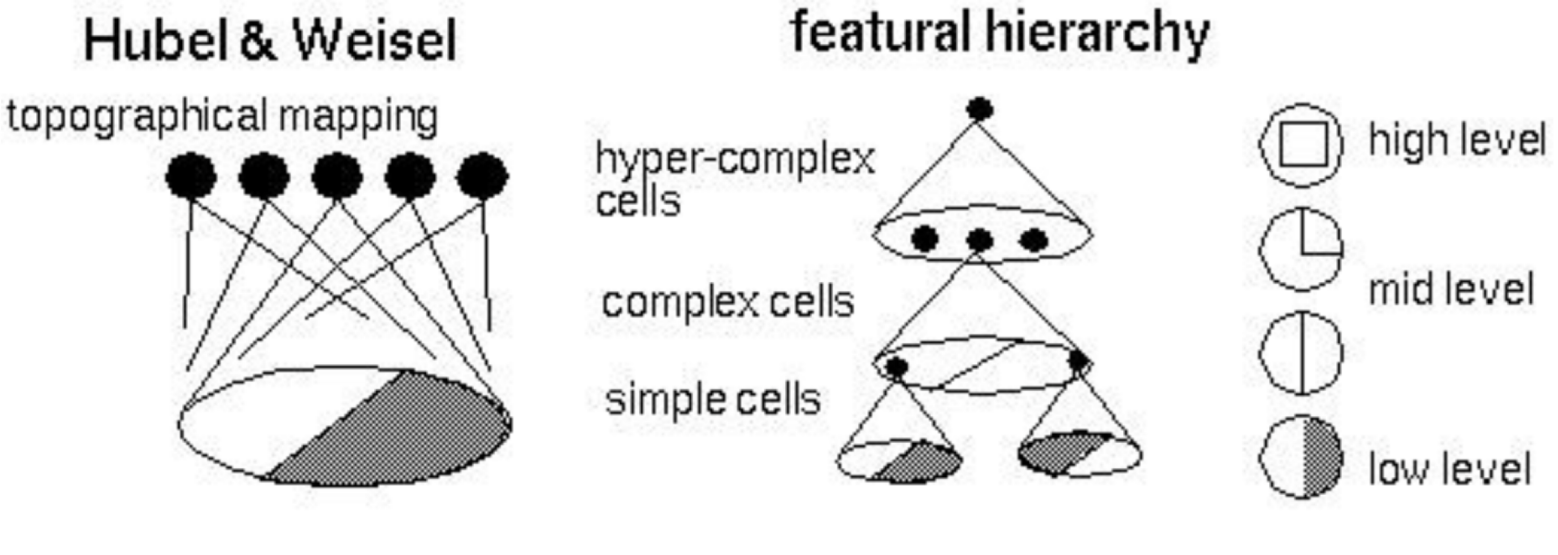






Hubel e Wiesel (1959)

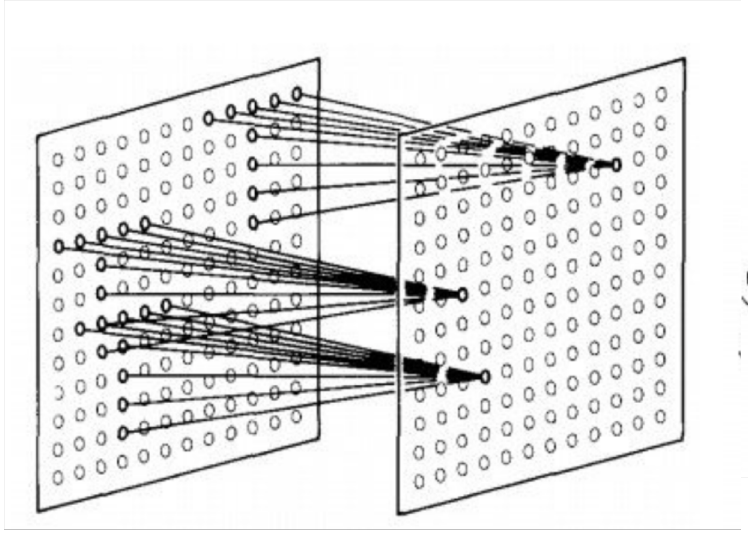
Um pouco de história…

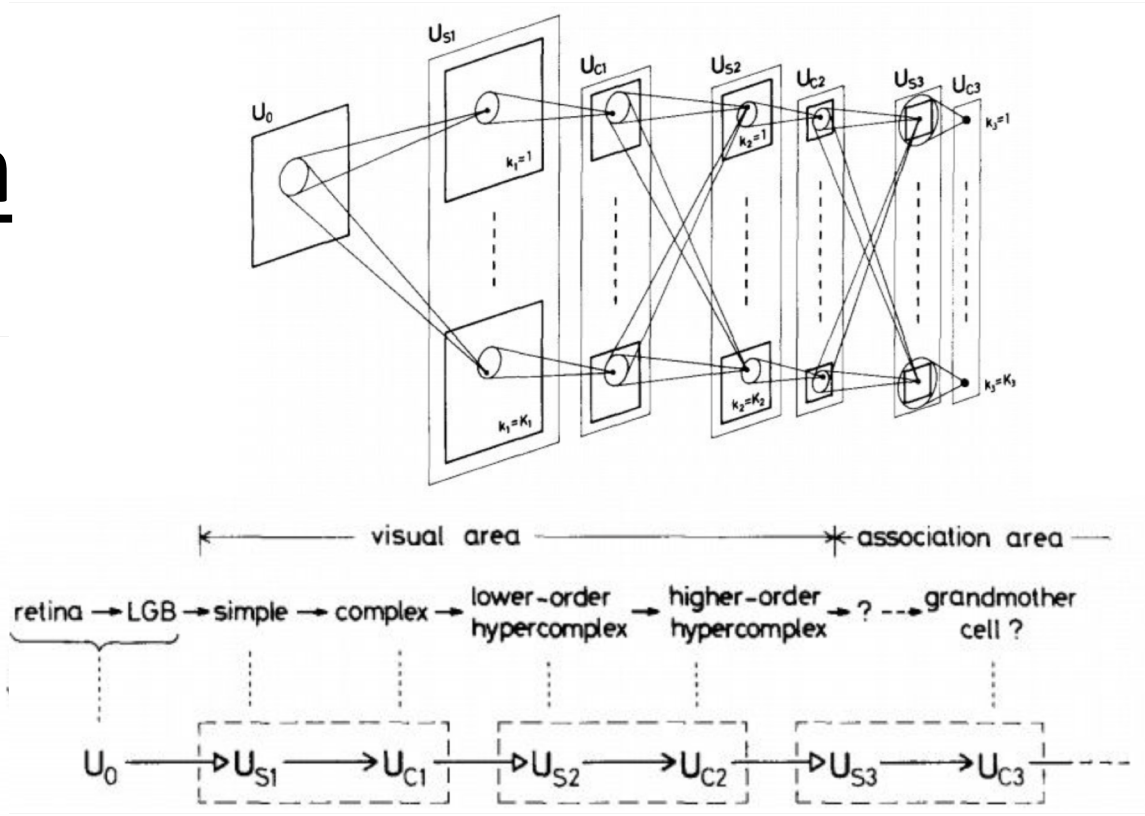


Um pouco de história...

(Fukushima, 1980)

Neurocognitron

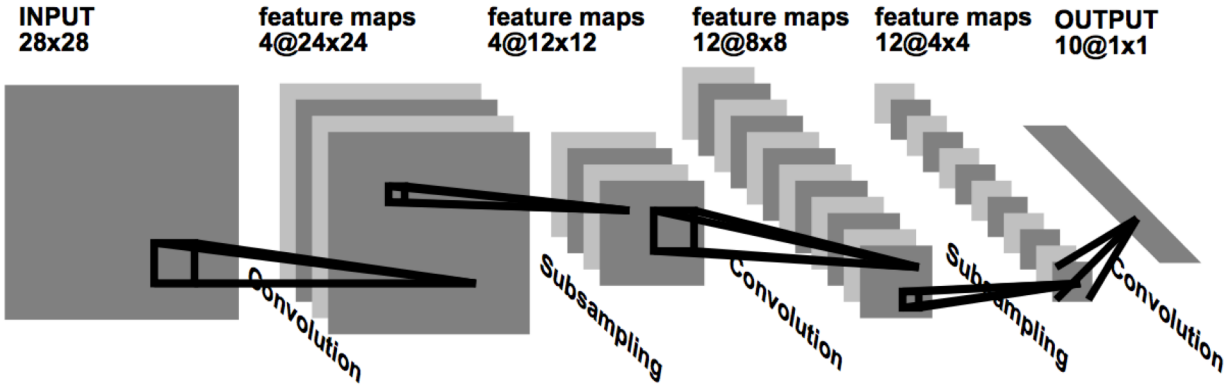


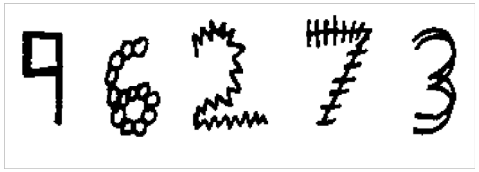


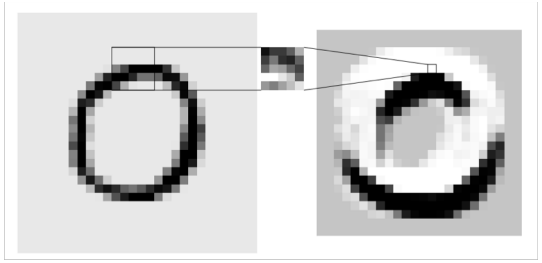
Um pouco de história…

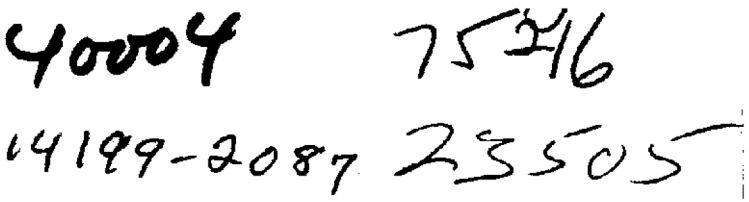
1990 – A primeira convolucional! (LeNet)

98.3% de acurácia no MNIST









Y. LeCun, B. Boser, J. S. Denker, Richard E. Howard, W. Habbard, L. D. Jackel, and D. Henderson. Handwritten digit recognition with a back-propagation network. In Advances in neural information processing systems, 1990.

Um pouco de história...

1995 – LeNet 5

99.1% de acurácia no MNIST

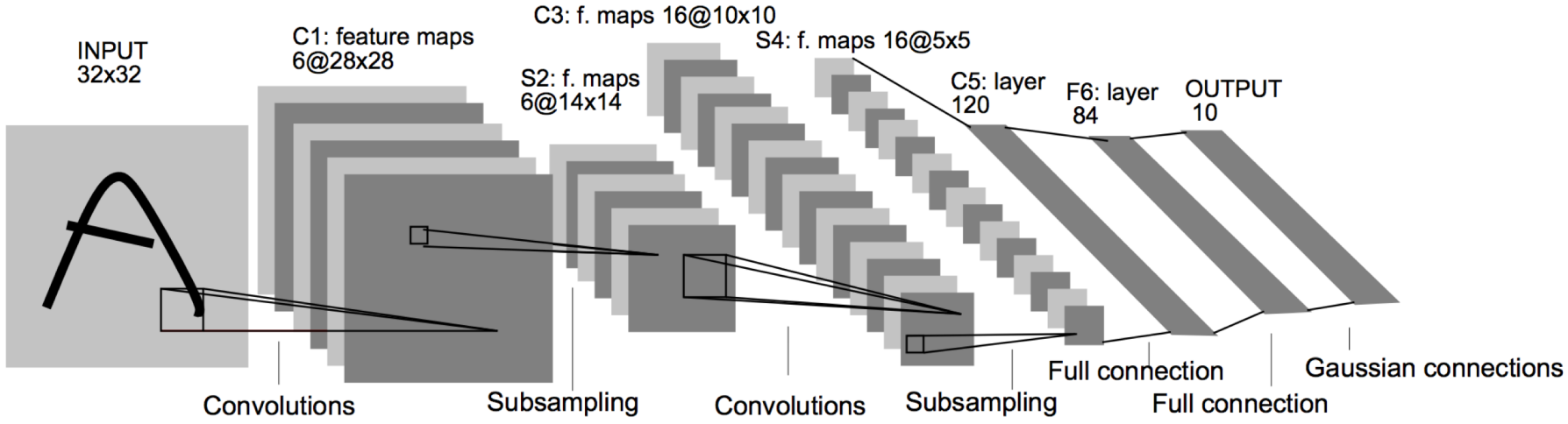


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

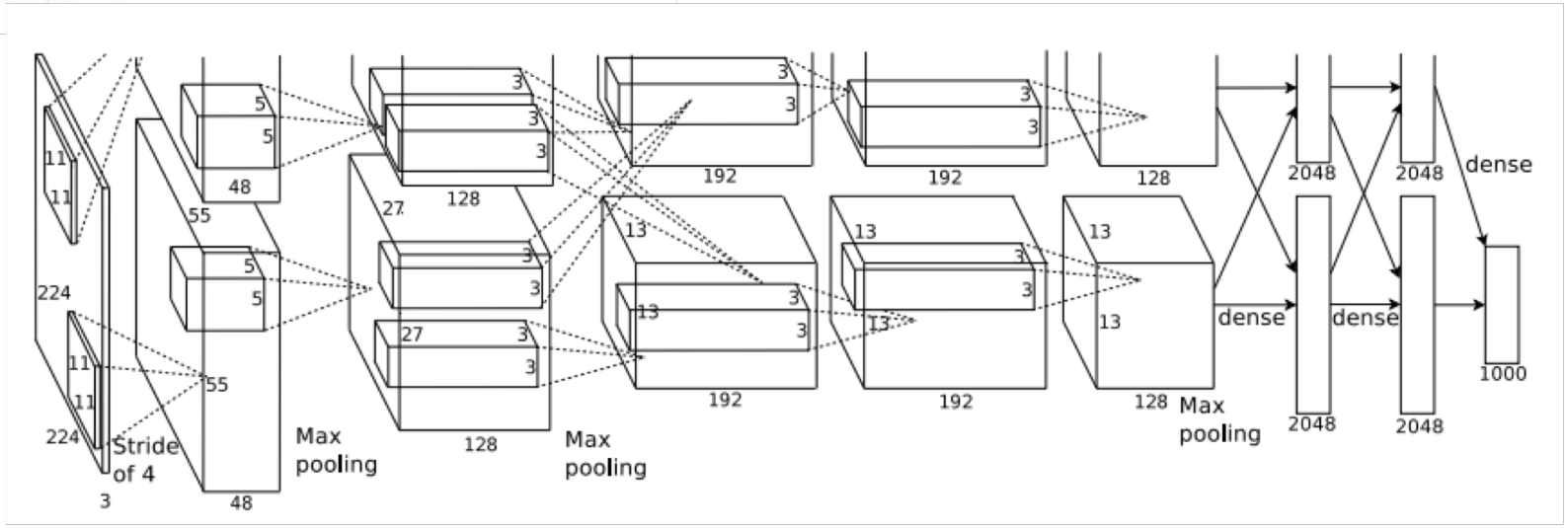
Y. LeCun, L. D. Jackel, L. Bottou, A. Brunot, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, U. A. Muller, E. Sackinger, P. Simard, and V. Vapnik. Comparison of learning algorithms for handwritten digit recognition. In F. Fogelman and P. Gallinari, editors, *International Conference on Artificial Neural Networks*, pages 53-60, Paris, 1995.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, november 1998.

Um pouco de história…

84.7% de acurácia no ImageNet

2012 – AlexNet

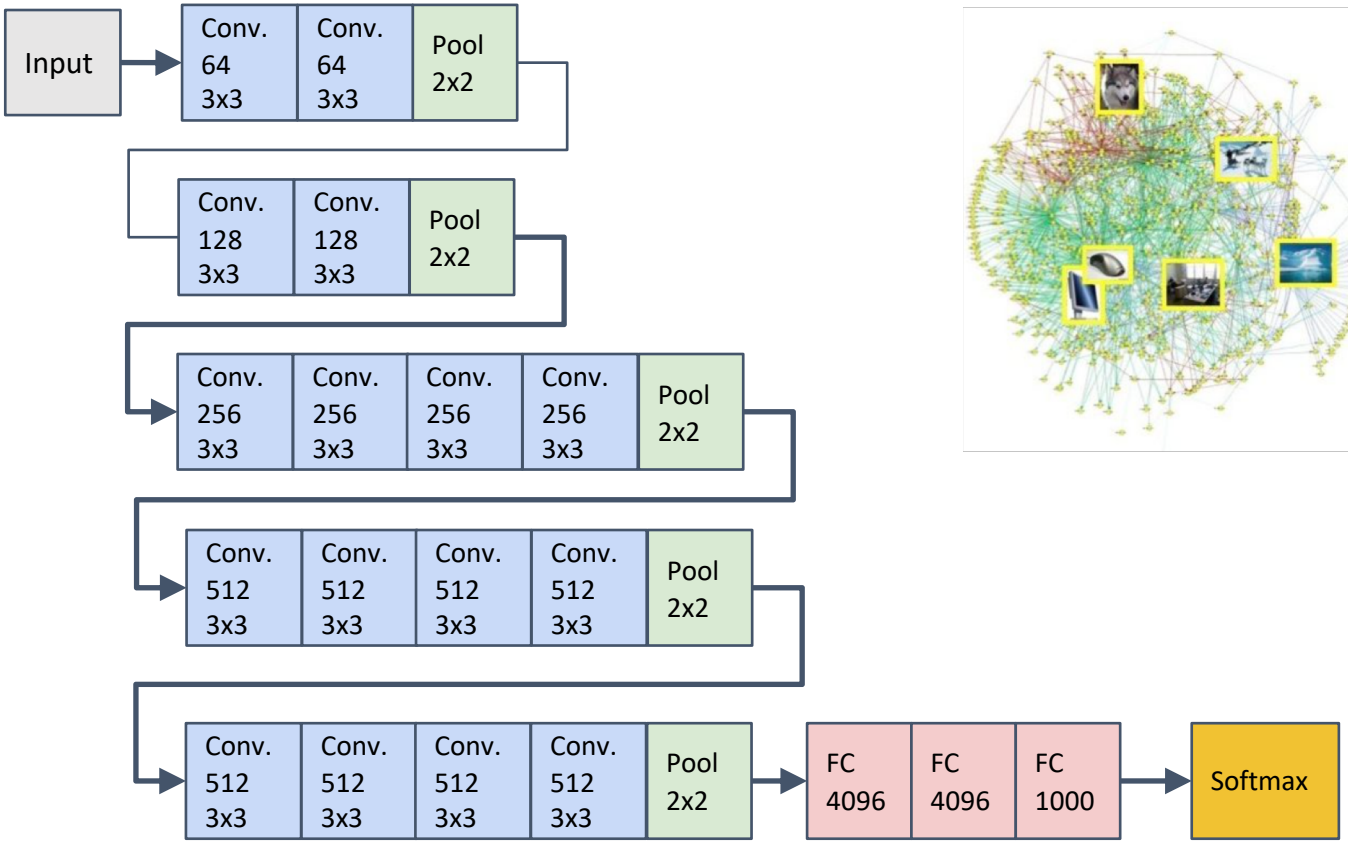


Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems , pages 1097–1105, 2012.

Um pouco de história…

2013 – VGG

92.5% de acurácia no ImageNet

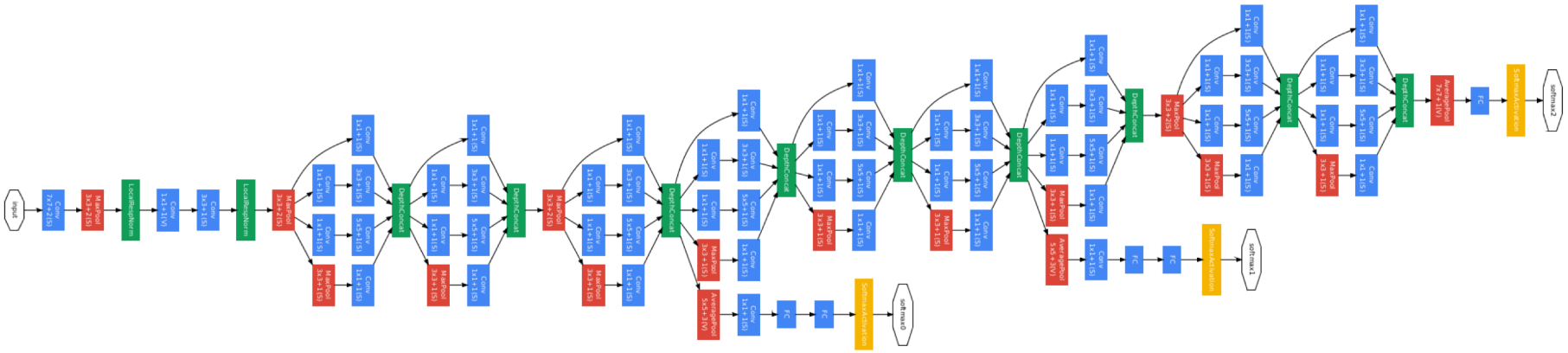


Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. 2014.

Um pouco de história...

93.33% de acurácia
no ImageNet

2014 - GoogLeNet



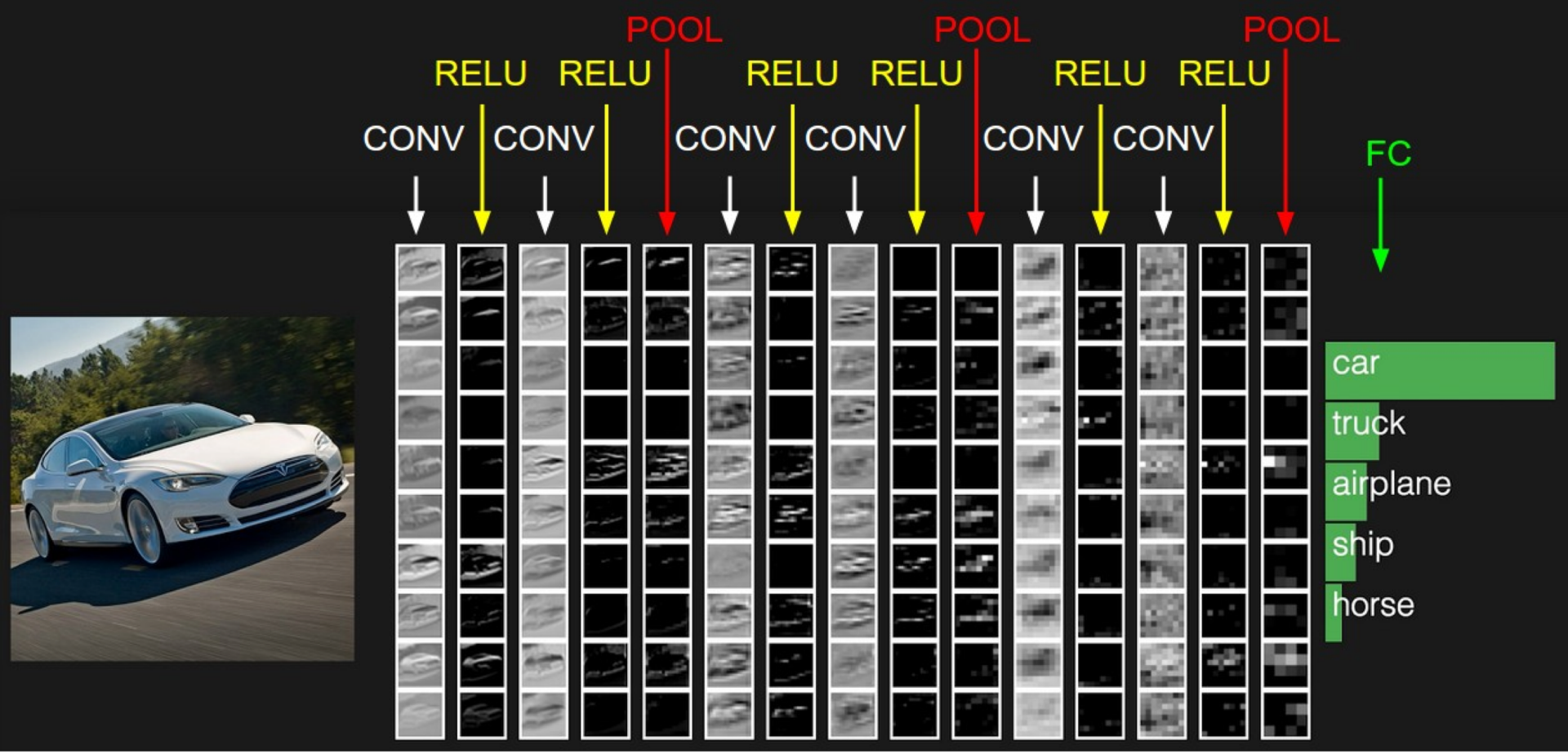
Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. 2014.

O que **É** uma Rede Neural Convolucional:

- ”Rede Neural *feed-forward* cujos neurônios estão dispostos de tal forma a responder por regiões sobrepostas (**campos receptores**) que preenchem o campo visual”

- ”Múltiplas camadas de **pequenas coleções de neurônios** que processam porções da imagem de entrada (campos receptores)”

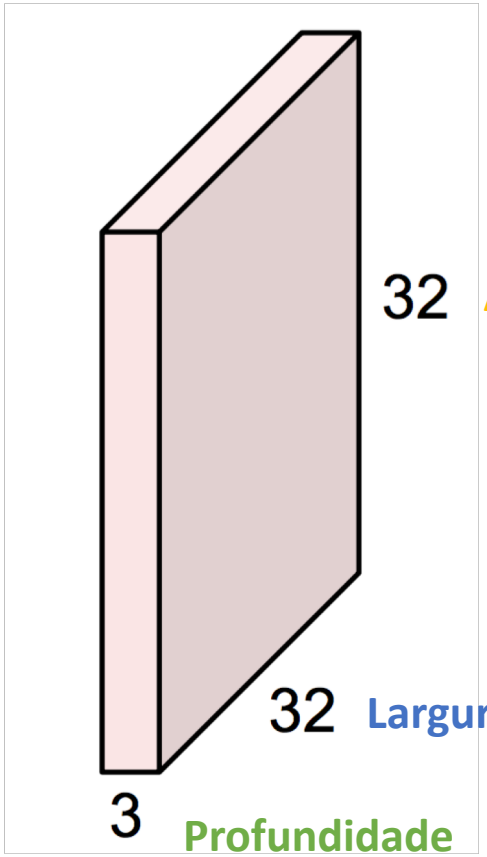
- ”Sequências de **camadas de convolução** intercaladas por funções de ativação”



Começaremos com a camada de convolução...

Camada de Convolução

Imagem 32×32×3

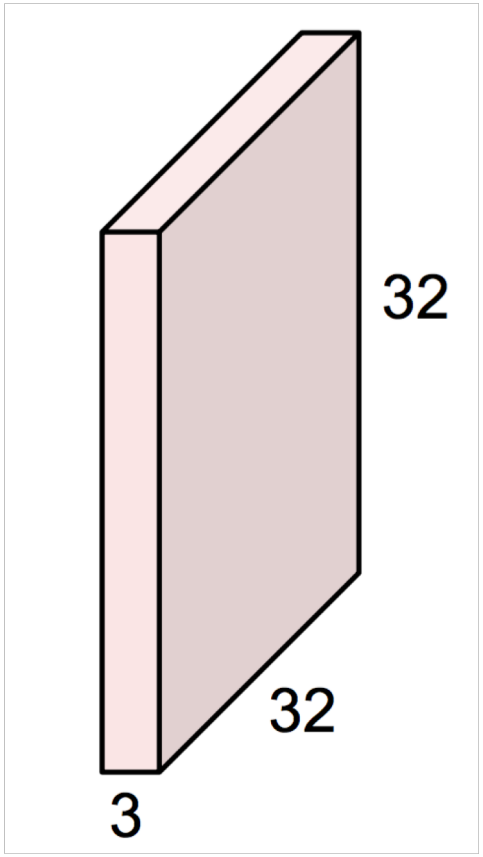


(canais de cor, ex: RGB)

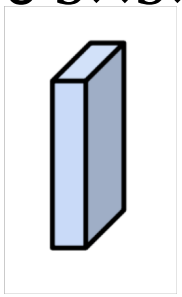
Camada de Convolução

Filtros sempre compreendem a profundidade inteira do volume de entrada

Imagem 32×32×3

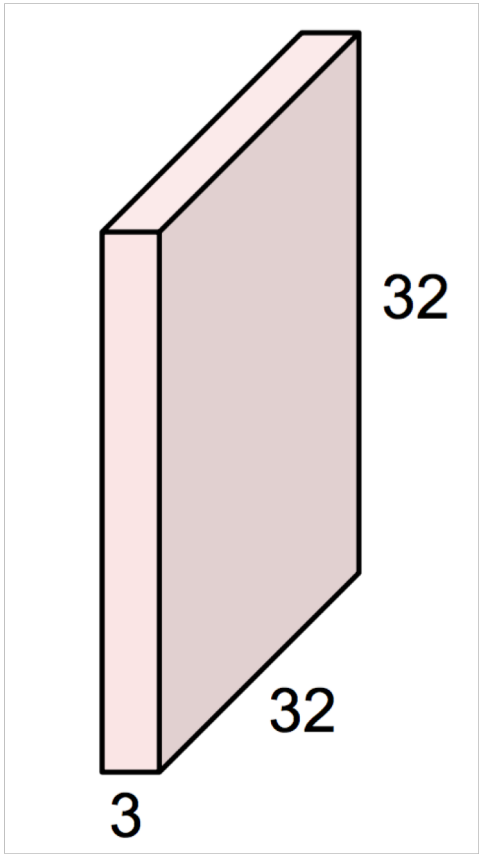


Filtro 5×5×3

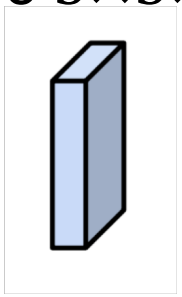


Camada de Convolução

Imagem $32 \times 32 \times 3$



Filtro $5 \times 5 \times 3$



Convoluir o filtro com a imagem

Deslizar espacialmente o filtro pela imagem, computando produtos internos

Camada de Convolução

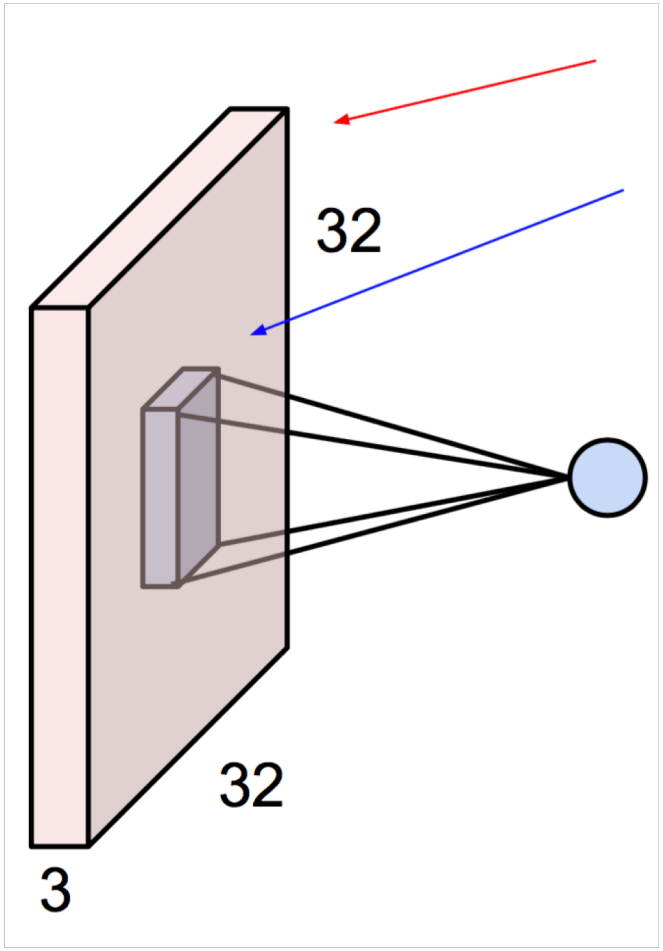
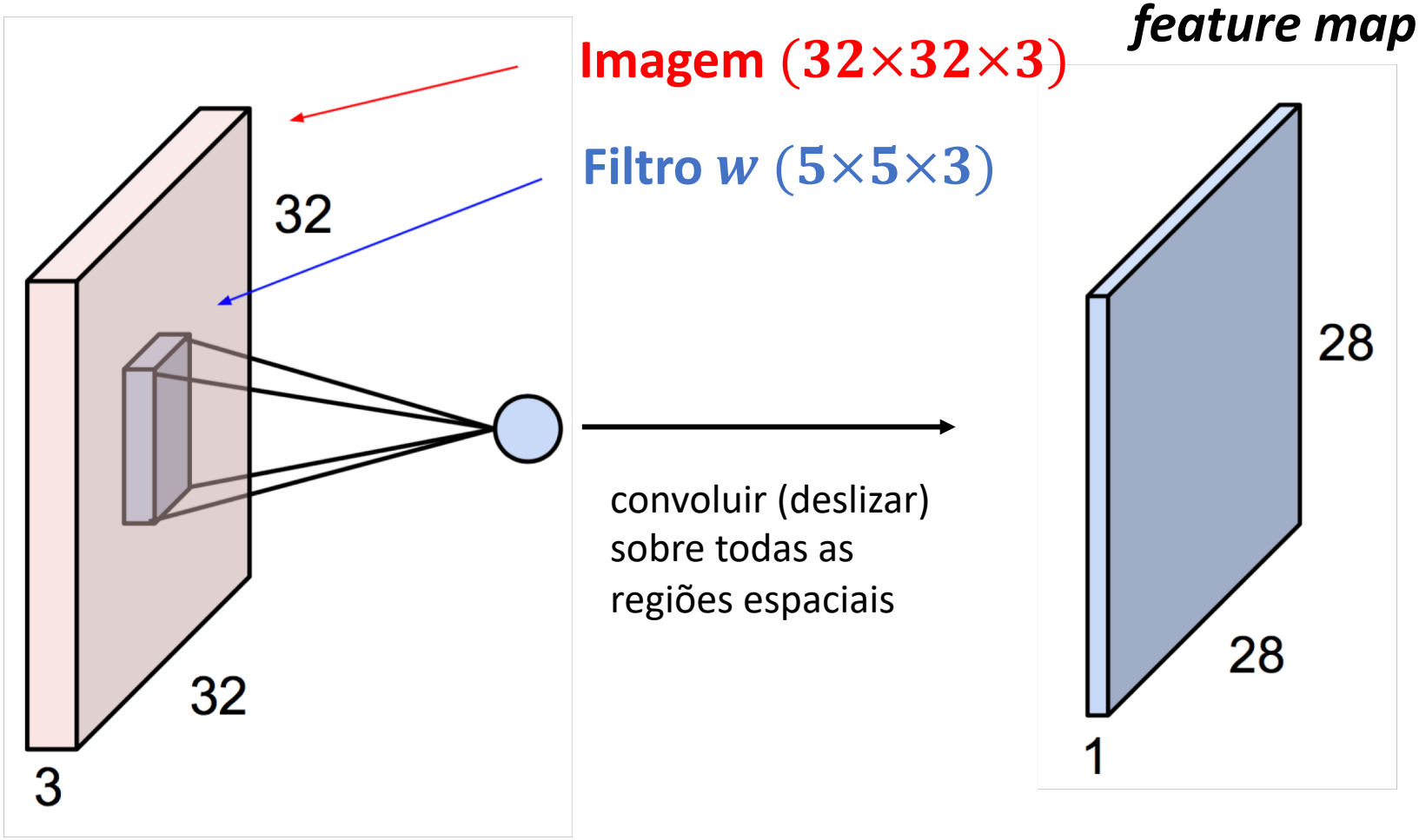


Imagem $(32\times32\times3)$

Filtro w $(5\times5\times3)$

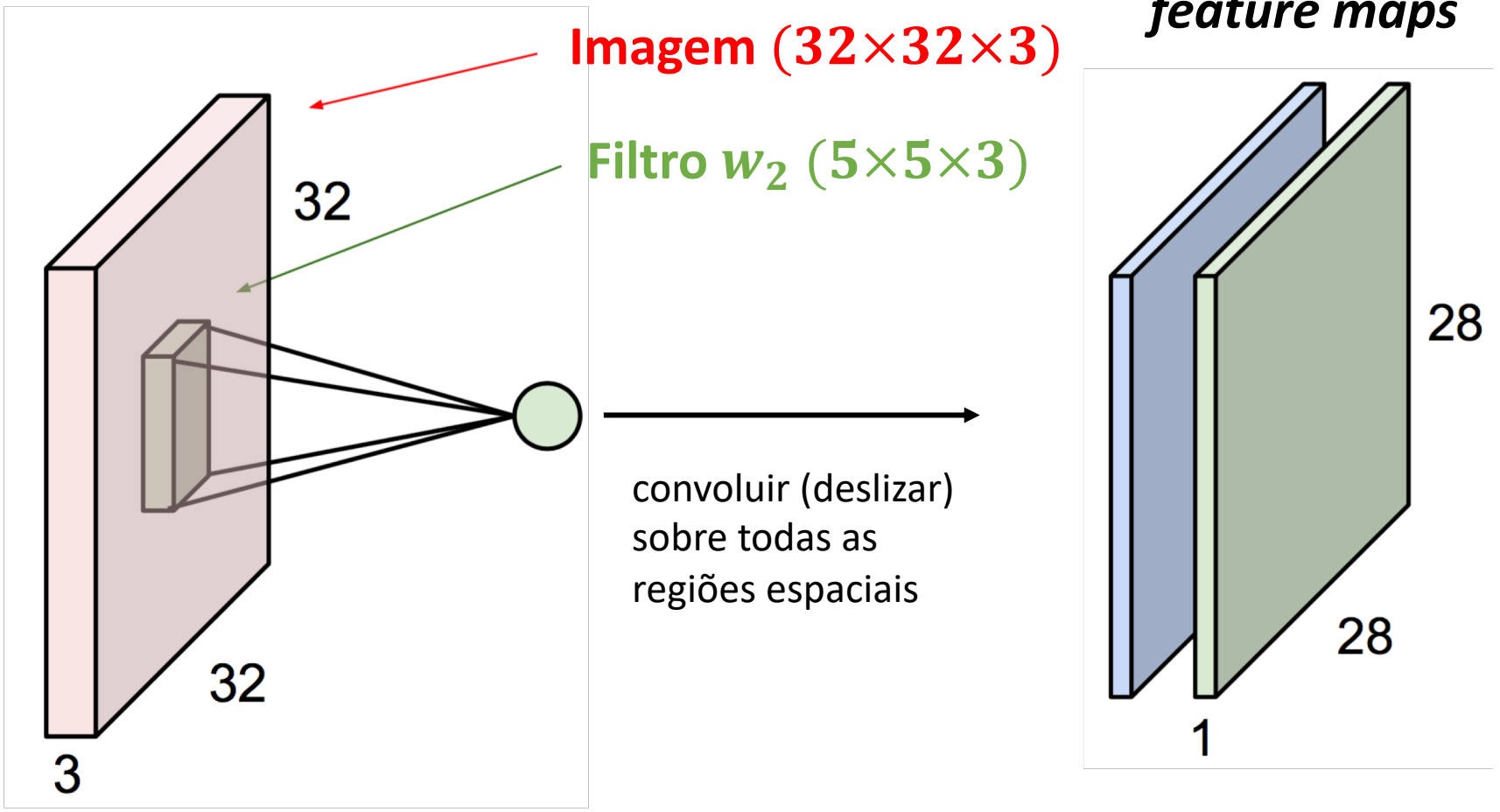
1 número:
resultado do produto interno entre o filtro e uma porção $5\times5\times3$ da imagem
($5\times5\times3 = 75$ parâmetros + bias)

Camada de Convolução



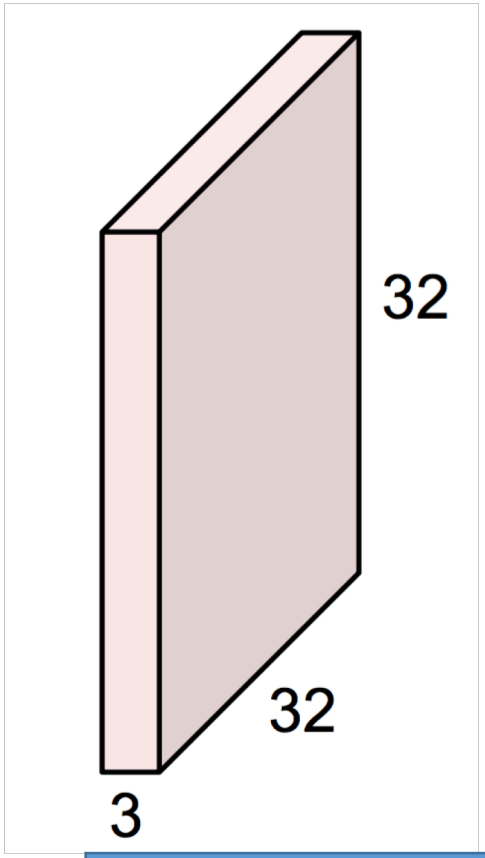
Camada de Convolução

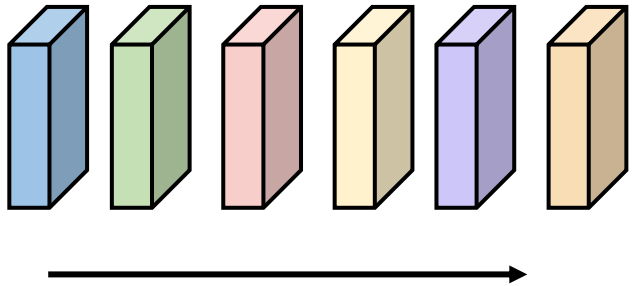
Considere a existência de um segundo filtro (verde)



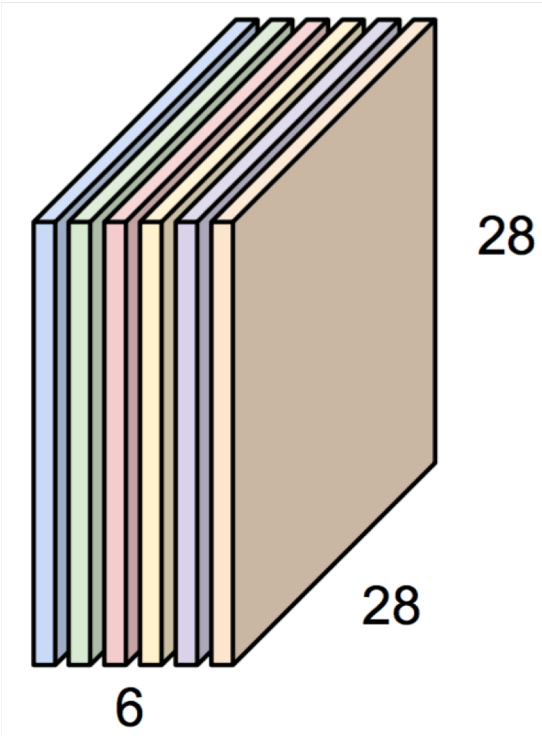
Camada de Convolução

Se tivermos 6 filtros $5 \times 5 \times 3$, teremos 6 feature maps distintos:





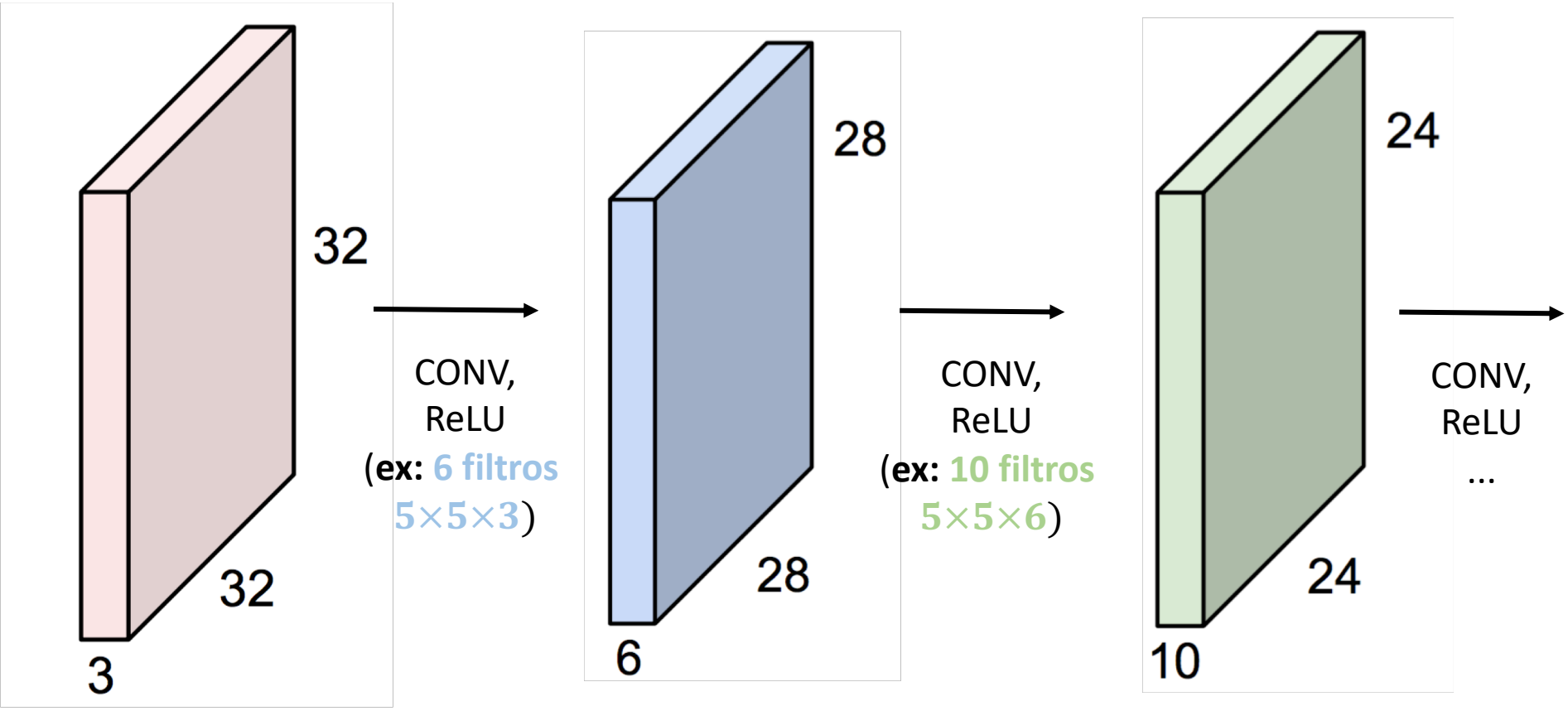
feature maps



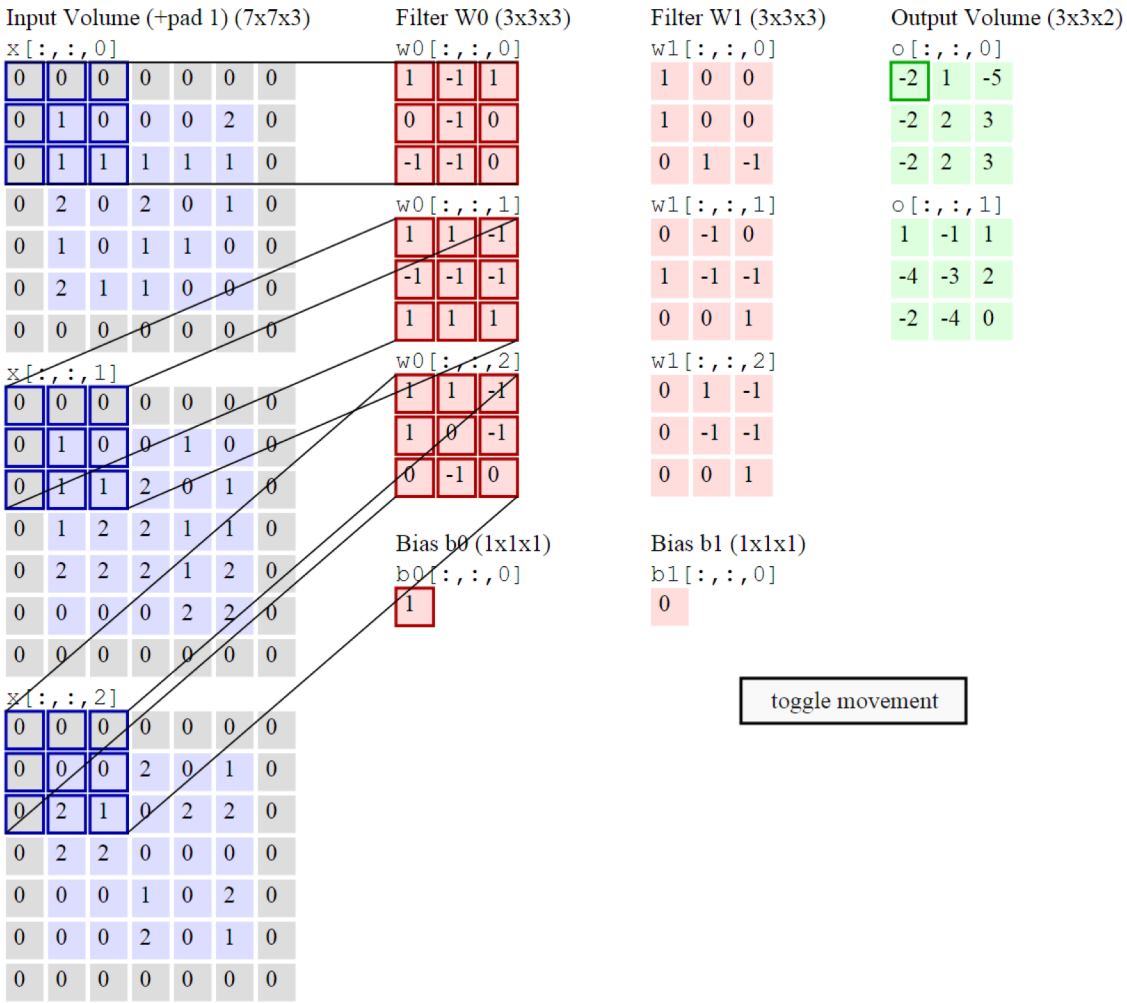
Empilhamos os *feature maps* para gerar uma ”nova imagem” de tamanho $28 \times 28 \times 6$

Rede Convolucional

”Sequência de camadas de convolução intercaladas por funções de ativação”

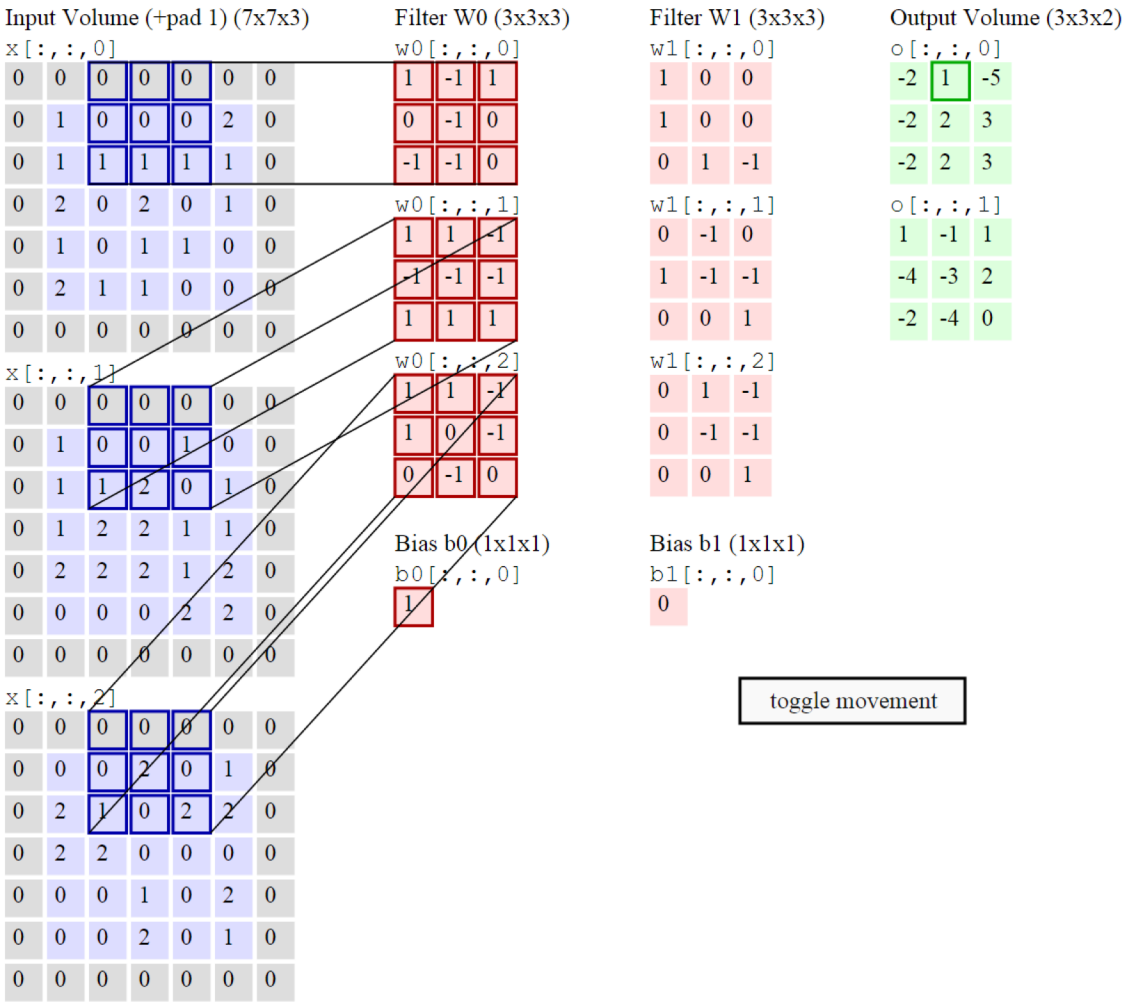


Redes Neurais Convolucionais



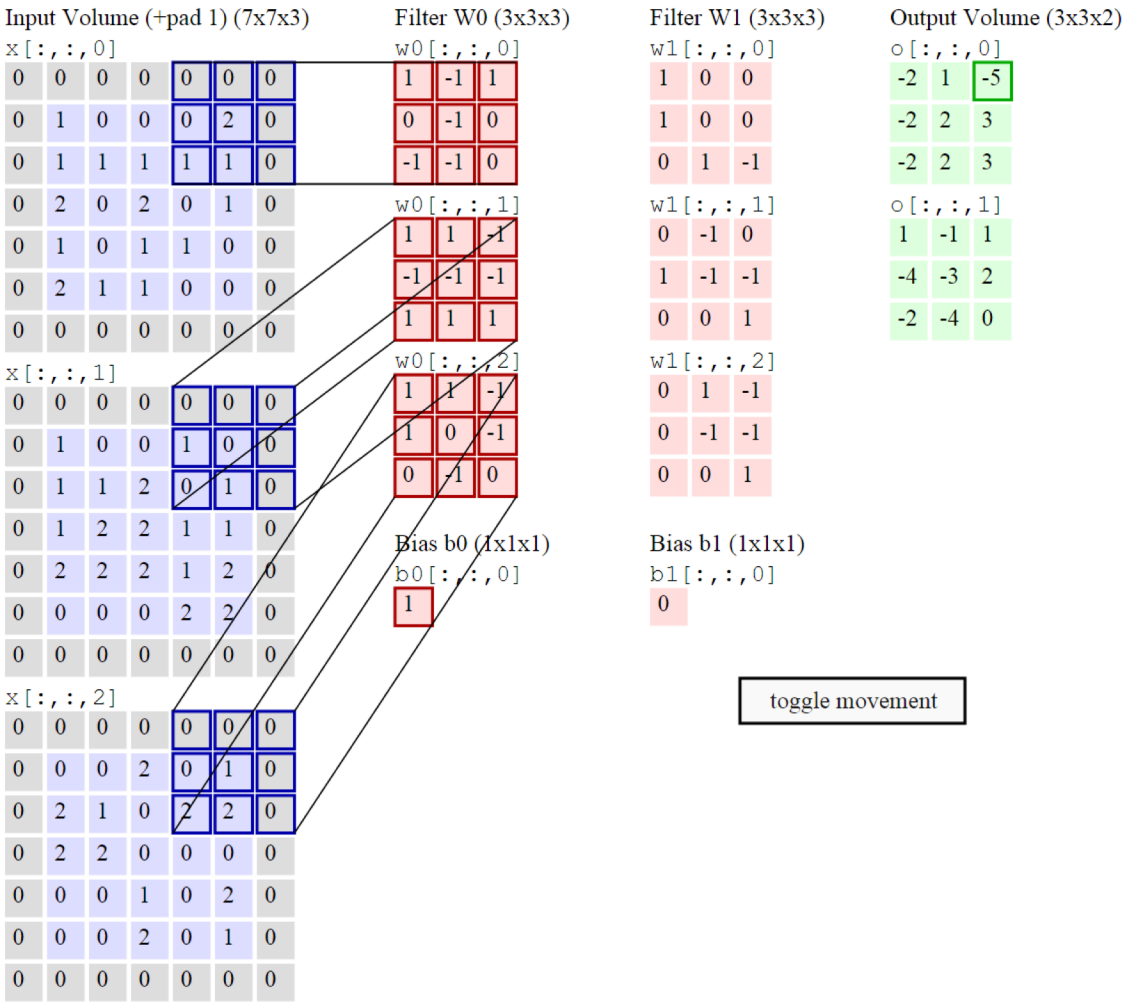
Fonte: http://cs231n.github.io/convolutional-networks/

Redes Neurais Convolucionais



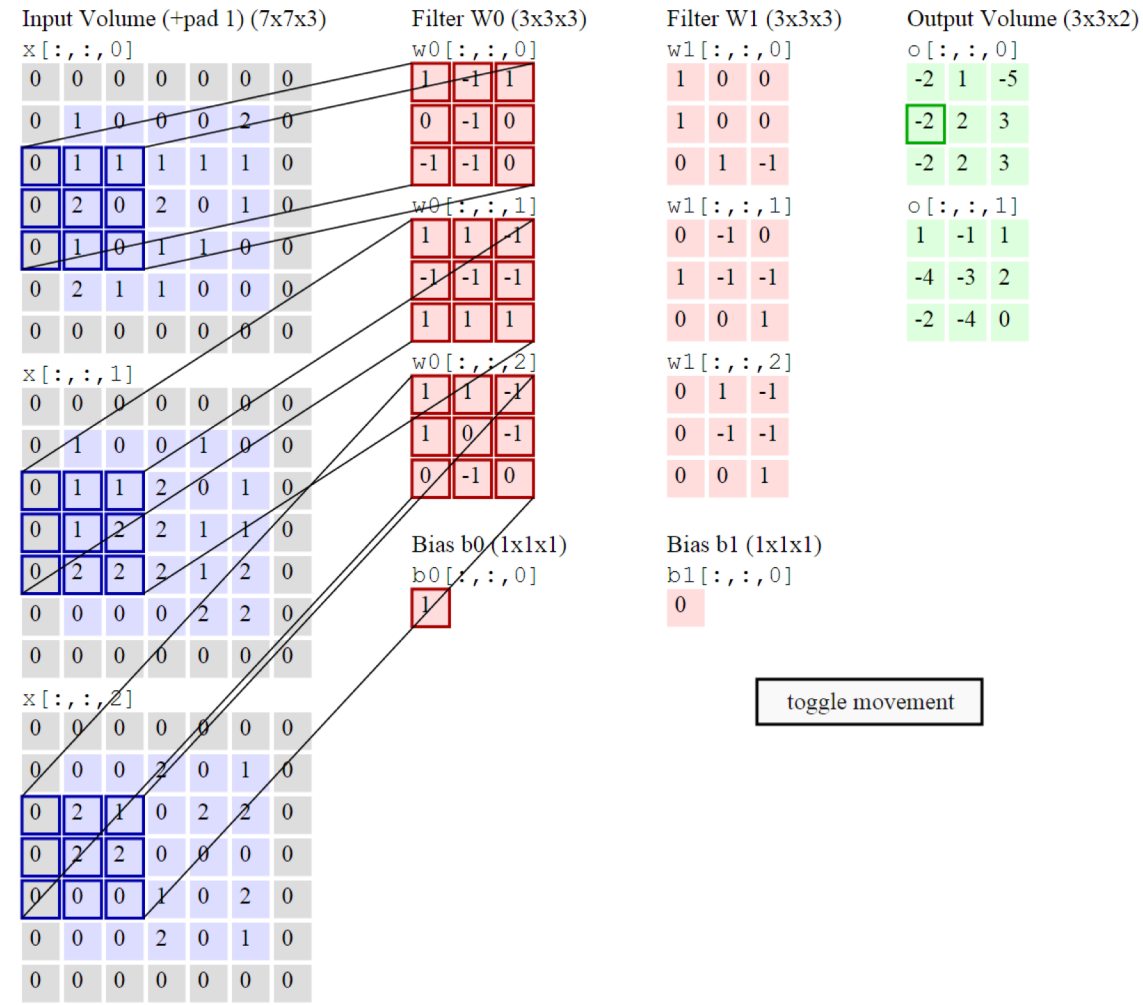
Fonte: http://cs231n.github.io/convolutional-networks/

Redes Neurais Convolucionais



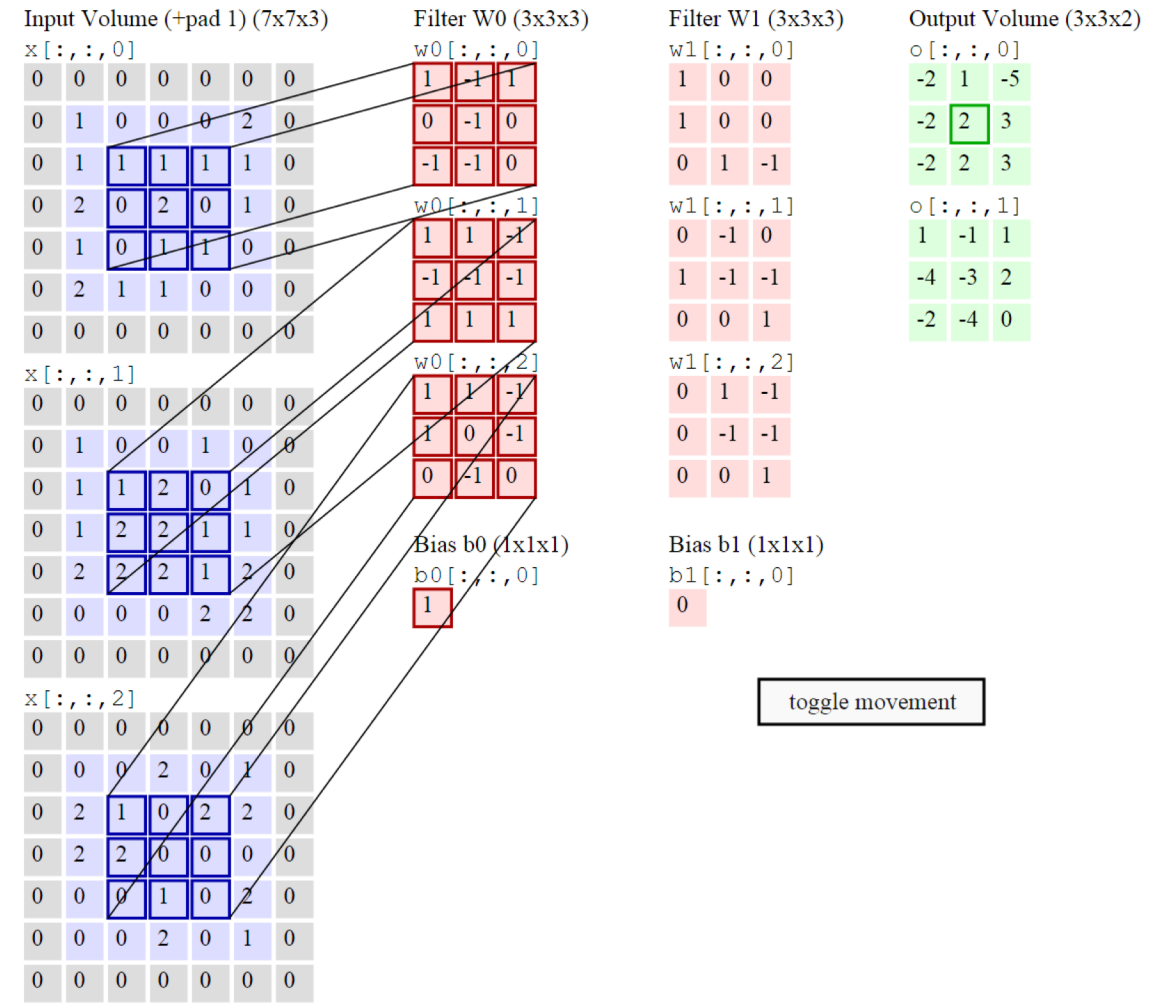
Fonte: http://cs231n.github.io/convolutional-networks/

Redes Neurais Convolucionais



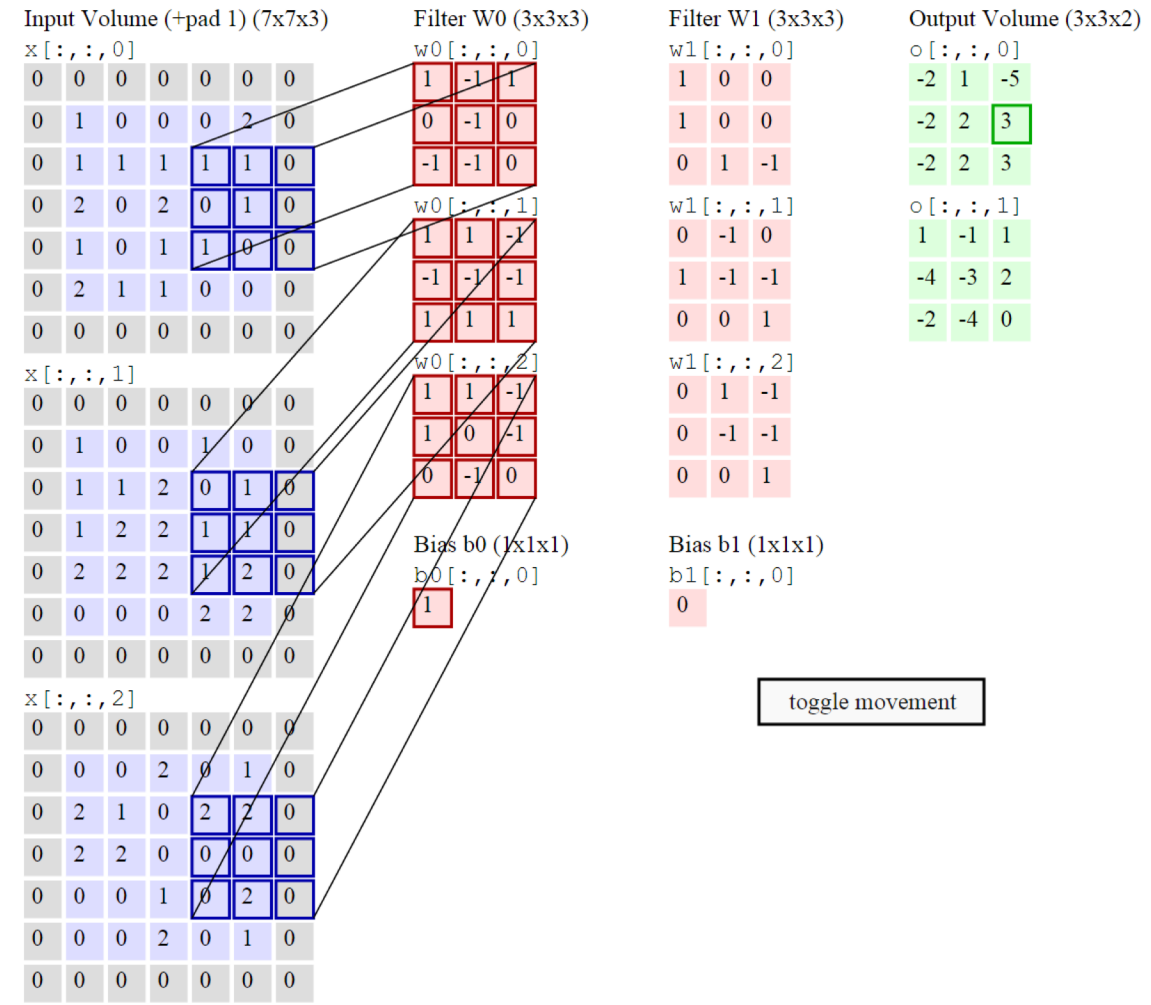
Fonte: http://cs231n.github.io/convolutional-networks/

Redes Neurais Convolucionais



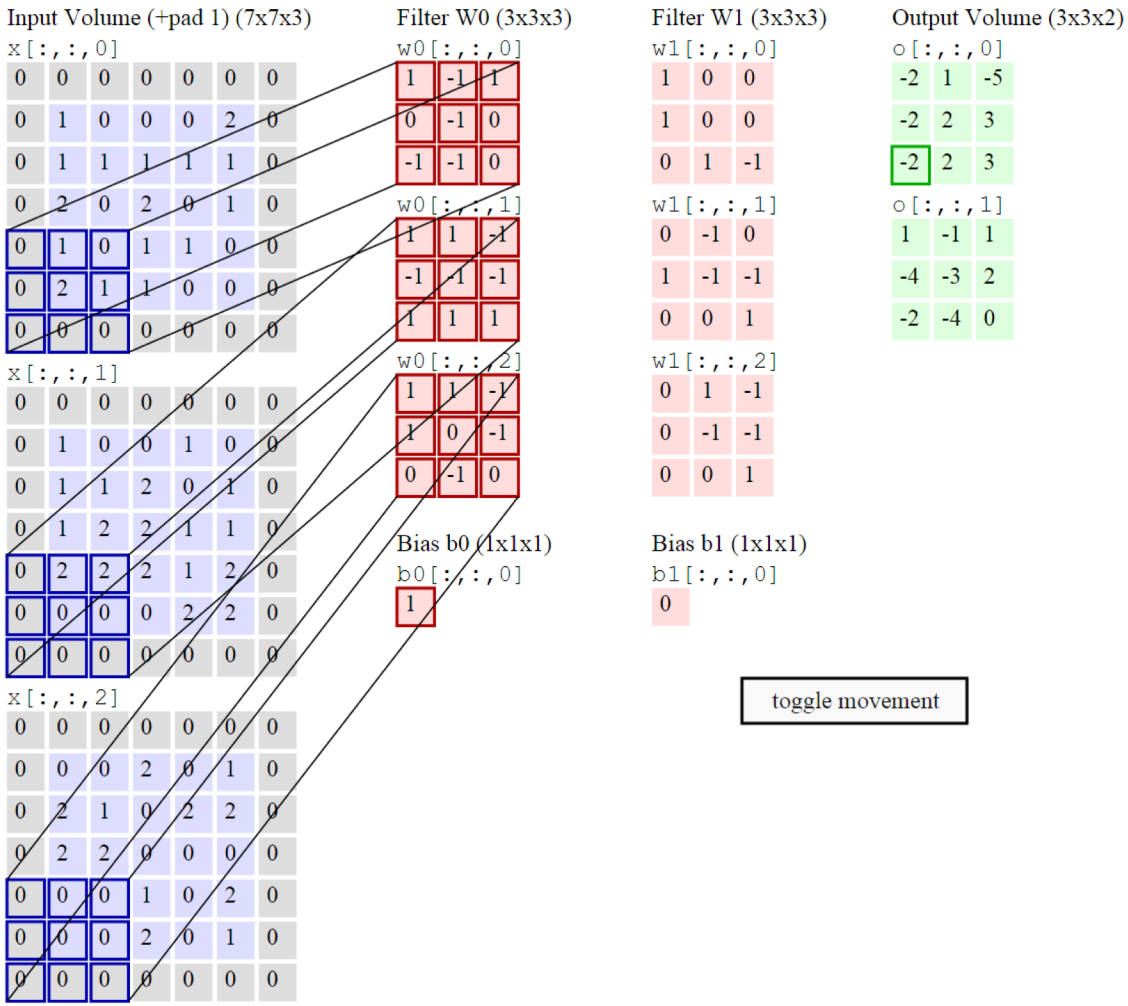
Fonte: http://cs231n.github.io/convolutional-networks/

Redes Neurais Convolucionais



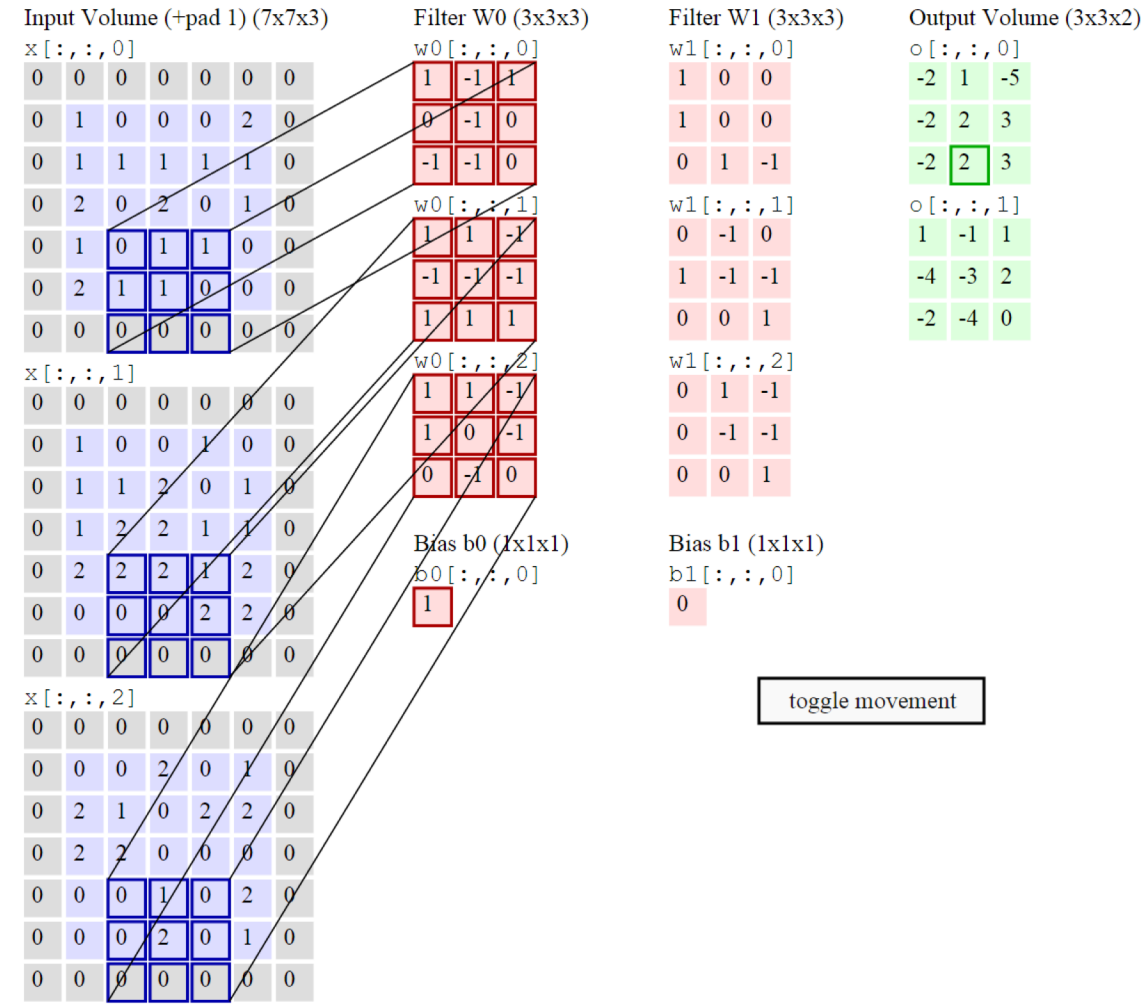
Fonte: http://cs231n.github.io/convolutional-networks/

Redes Neurais Convolucionais



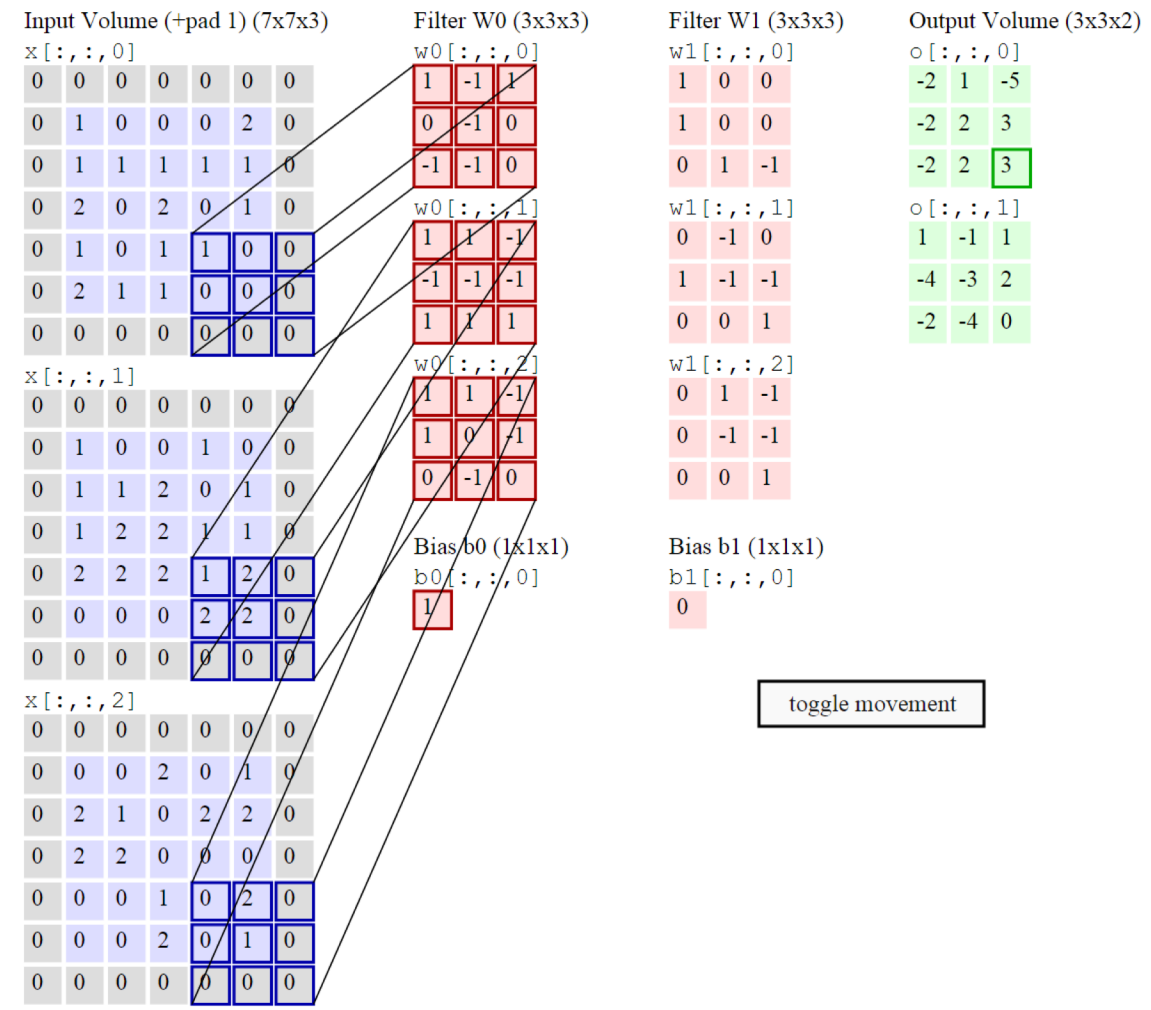
Fonte: http://cs231n.github.io/convolutional-networks/

Redes Neurais Convolucionais



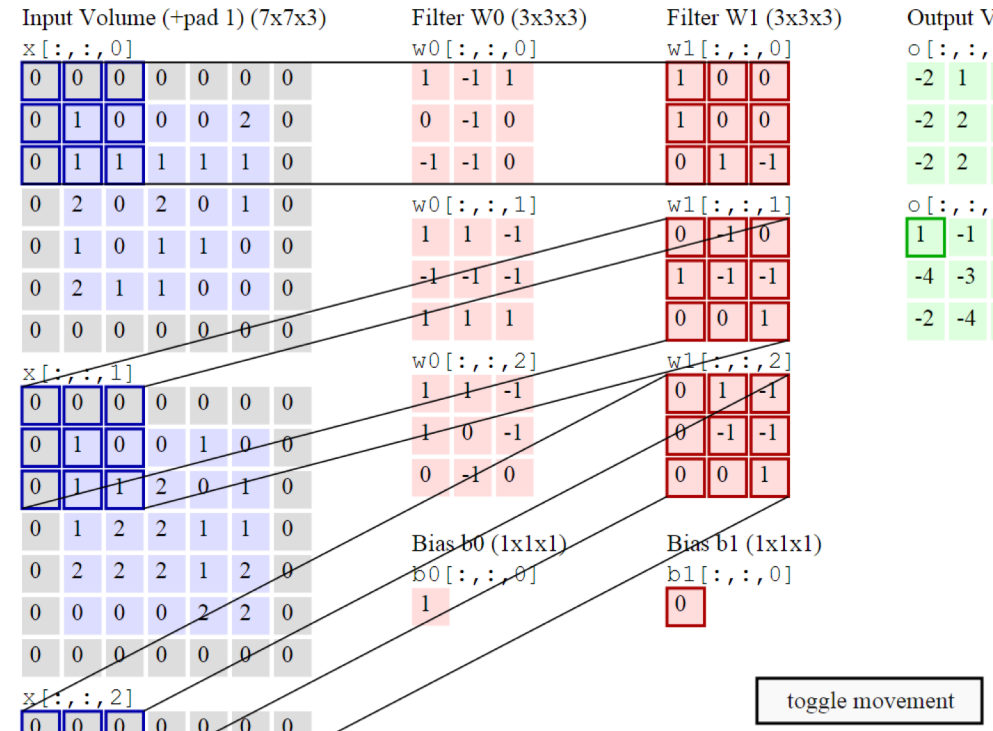
Fonte: http://cs231n.github.io/convolutional-networks/

Redes Neurais Convolucionais



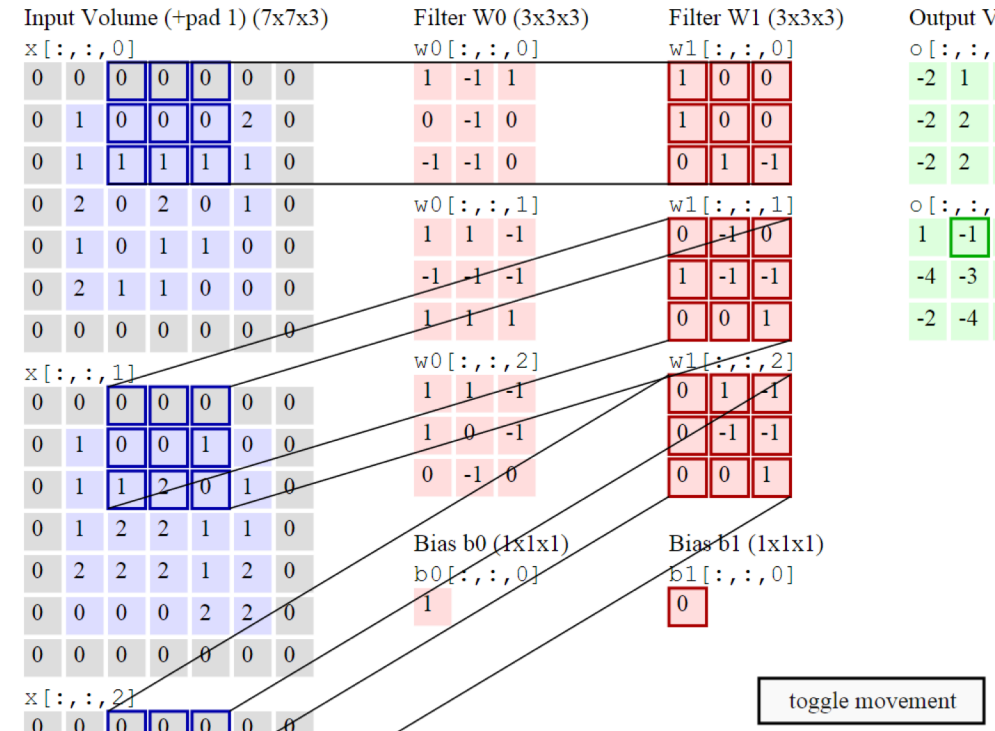
Fonte: http://cs231n.github.io/convolutional-networks/

Redes Neurais Convolucionais



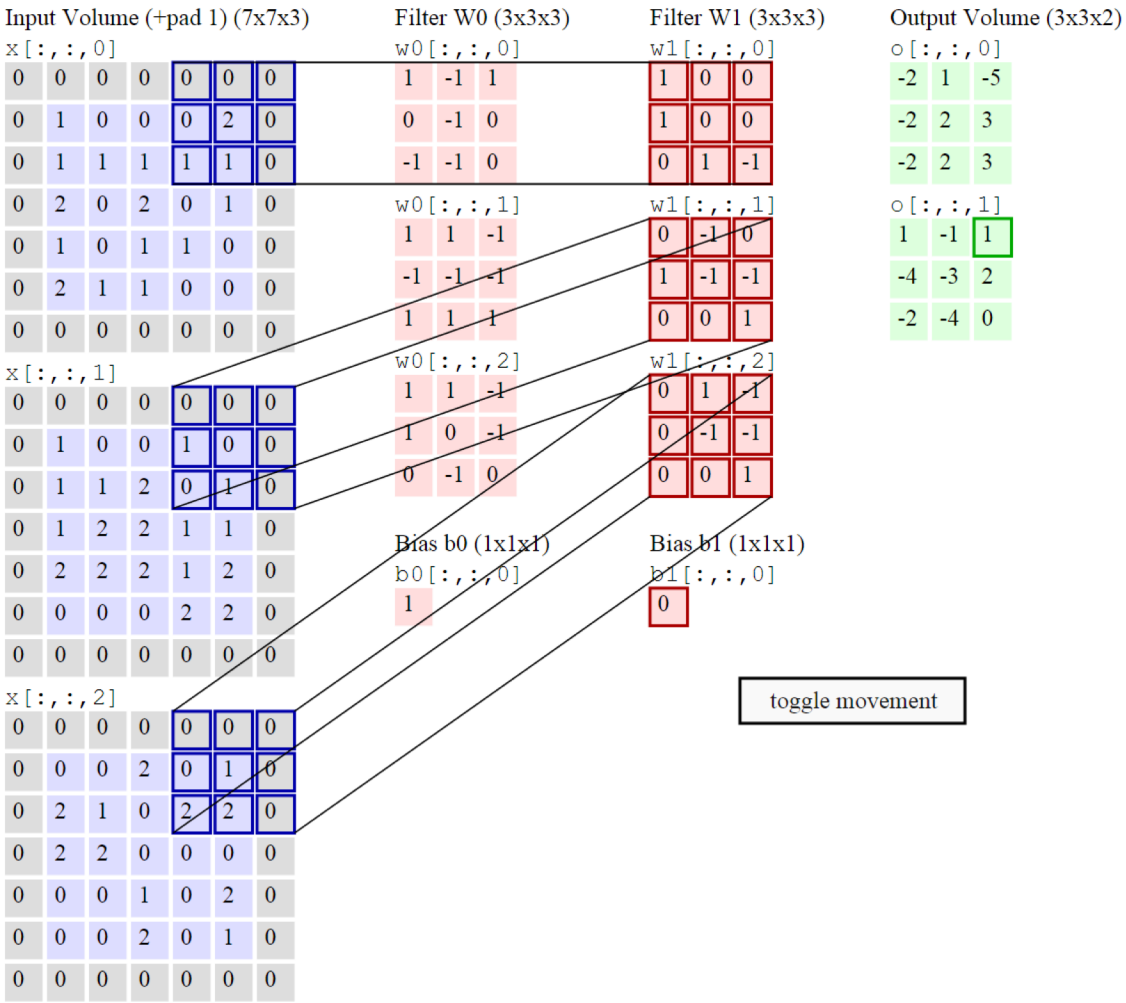
Fonte: http://cs231n.github.io/convolutional-networks/

Redes Neurais Convolucionais



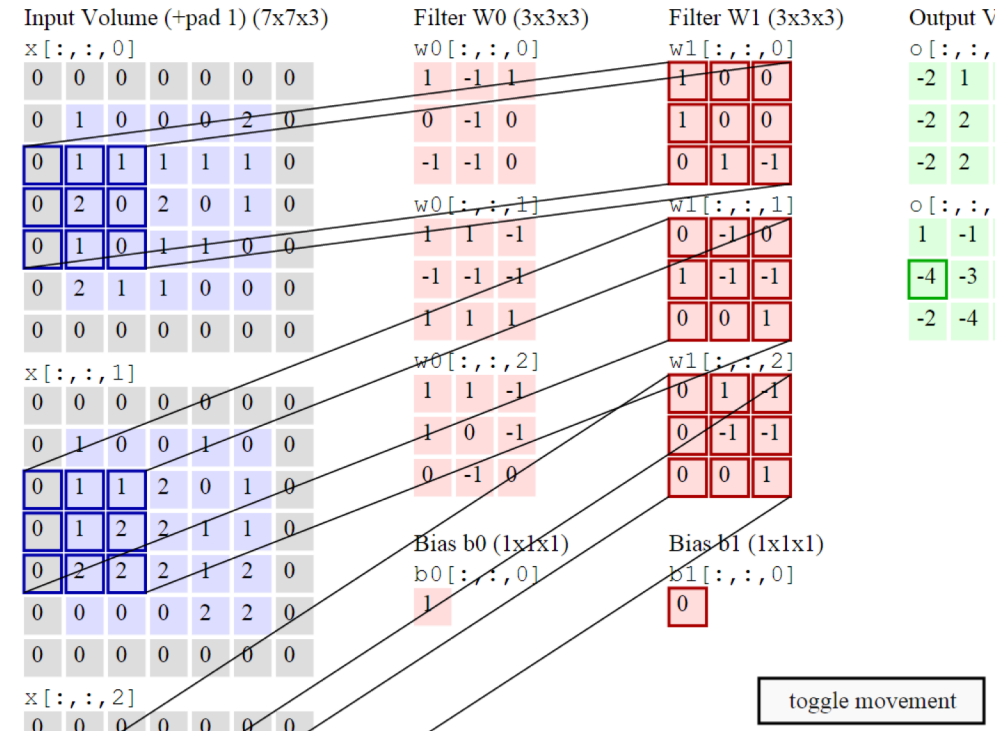
Fonte: http://cs231n.github.io/convolutional-networks/

Redes Neurais Convolucionais



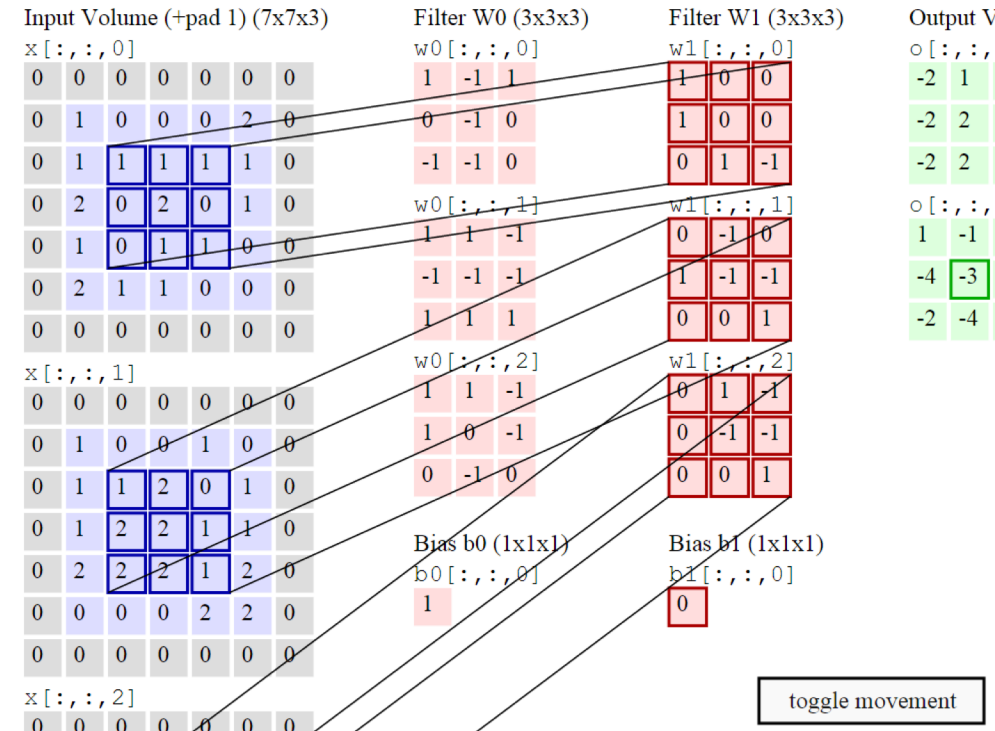
Fonte: http://cs231n.github.io/convolutional-networks/

Redes Neurais Convolucionais



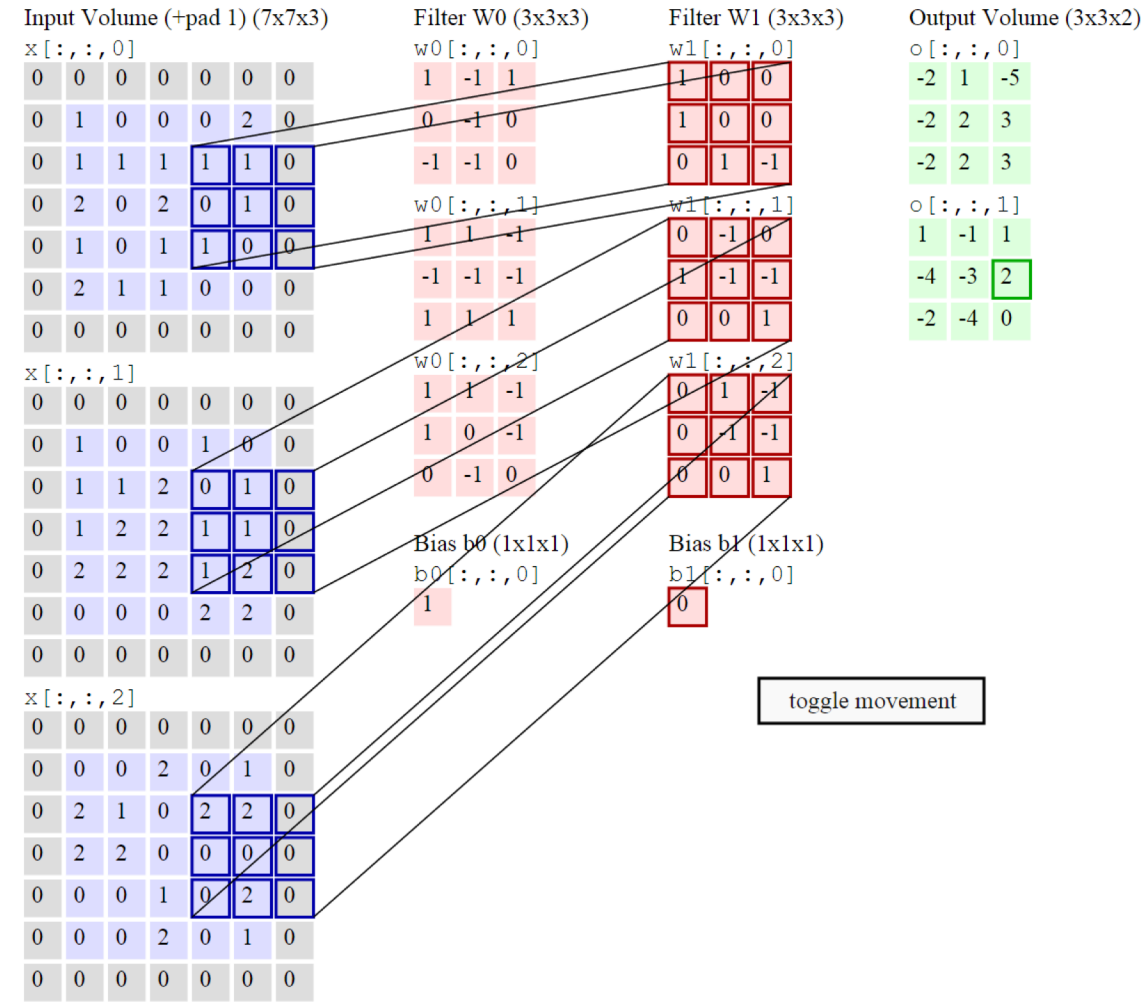
Fonte: http://cs231n.github.io/convolutional-networks/

Redes Neurais Convolucionais



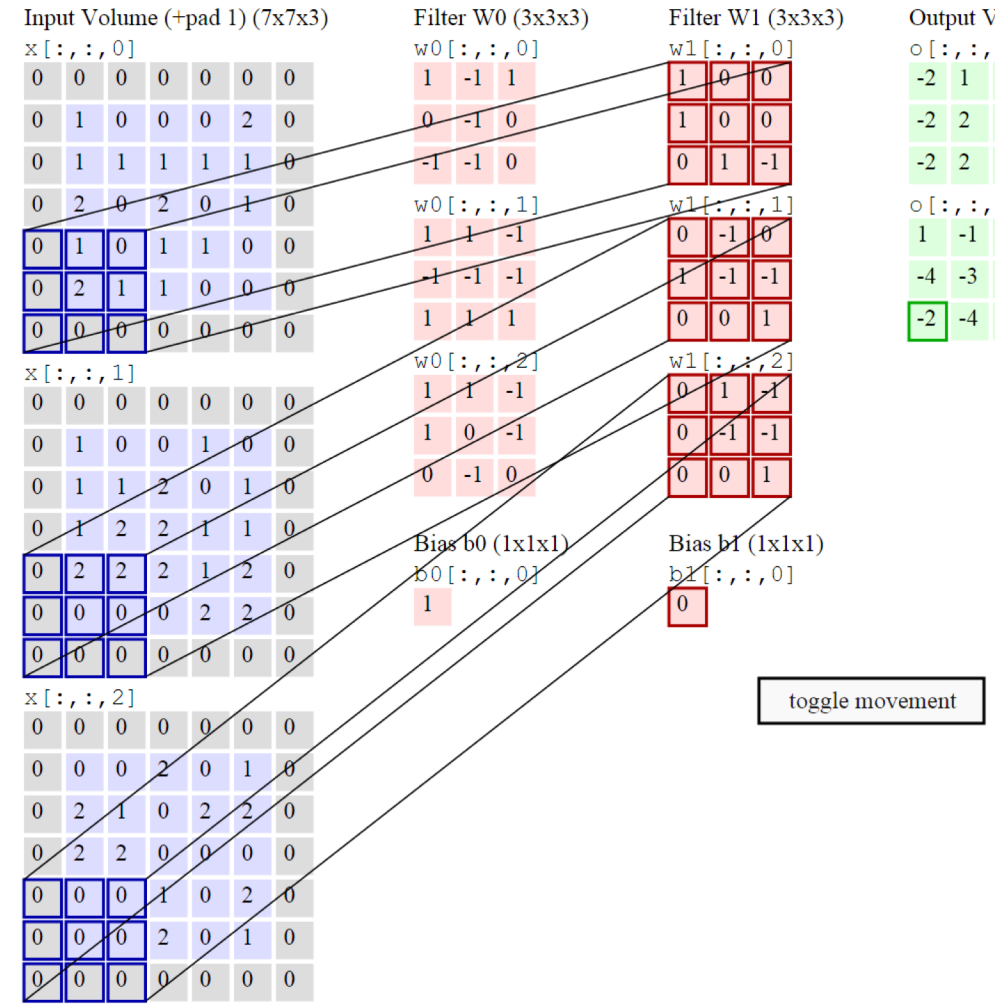
Fonte: http://cs231n.github.io/convolutional-networks/

Redes Neurais Convolucionais



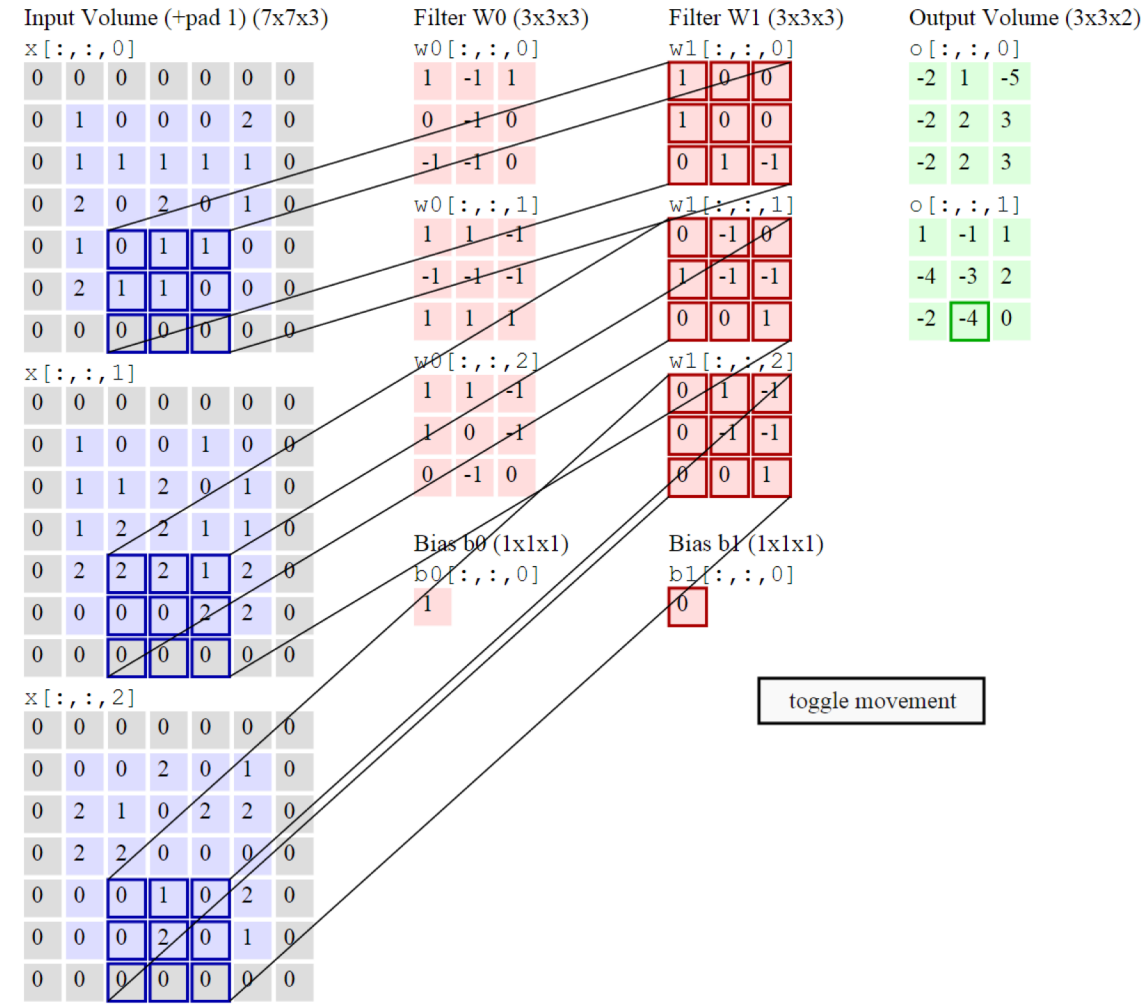
Fonte: http://cs231n.github.io/convolutional-networks/

Redes Neurais Convolucionais



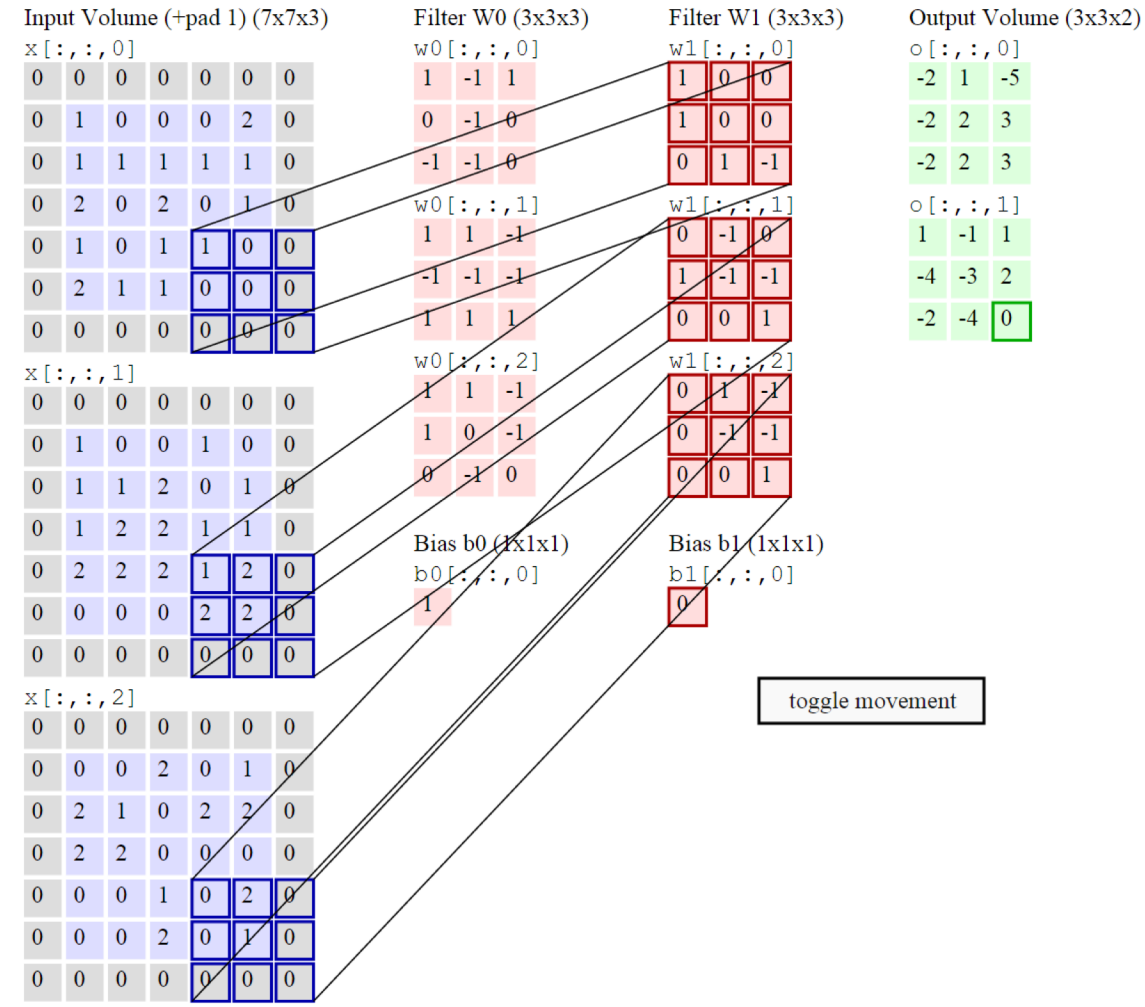
Fonte: http://cs231n.github.io/convolutional-networks/

Redes Neurais Convolucionais

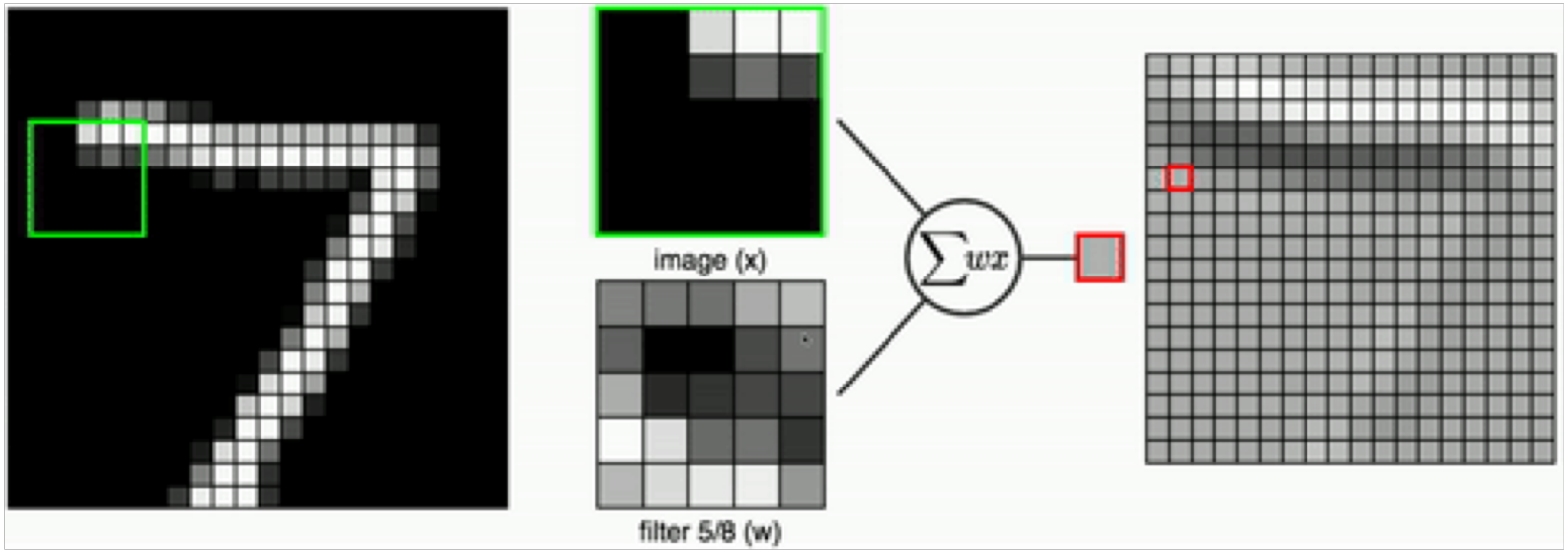


Fonte: http://cs231n.github.io/convolutional-networks/

Redes Neurais Convolucionais

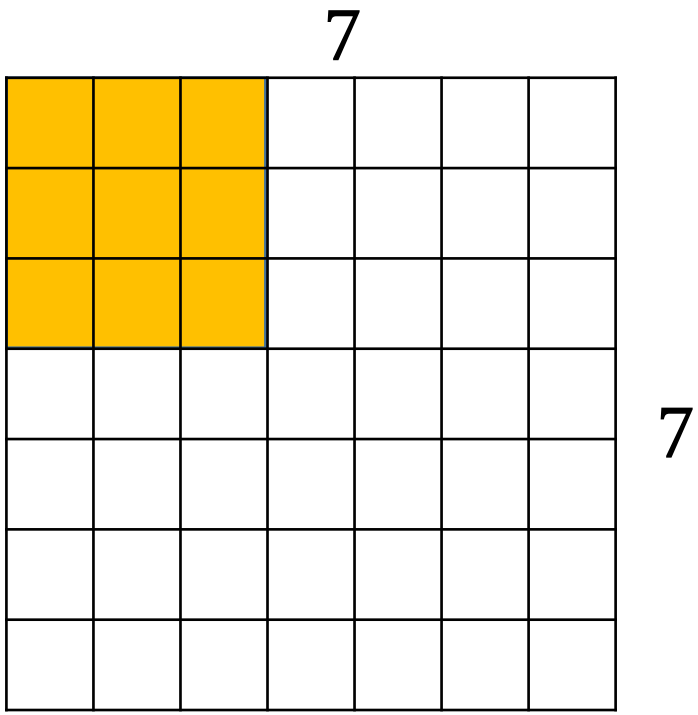


Fonte: http://cs231n.github.io/convolutional-networks/



http://ml4a.github.io/dev/demos/demo_convolution.html

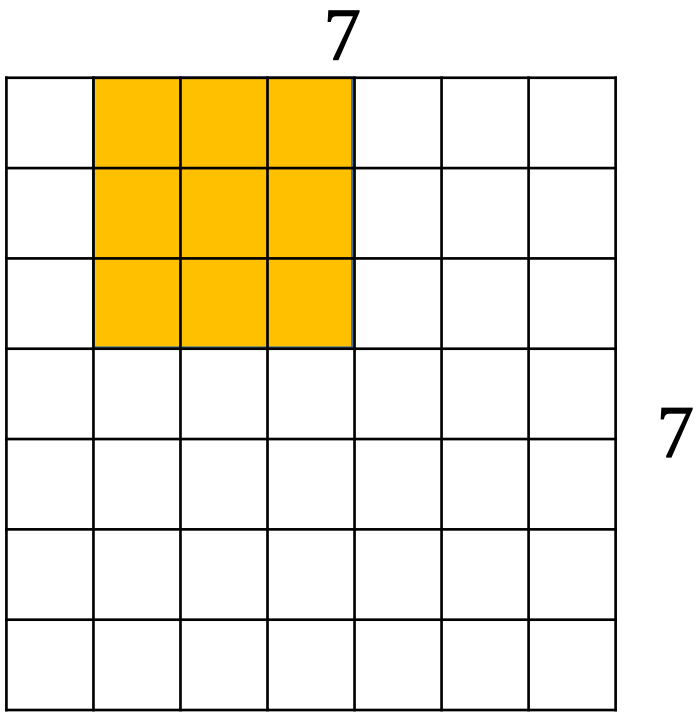
Acertando os Cálculos



Entrada 7×7

Filtro 3×3

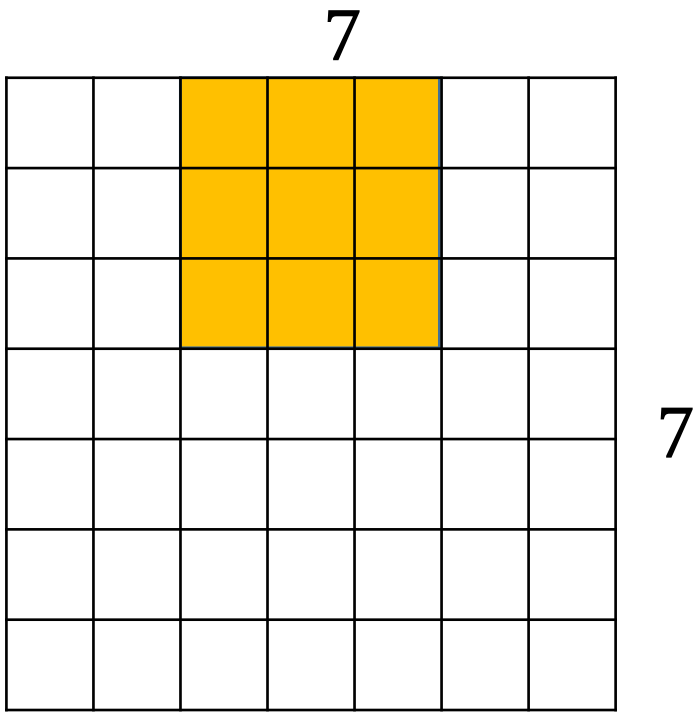
Acertando os Cálculos



Entrada 7×7

Filtro 3×3

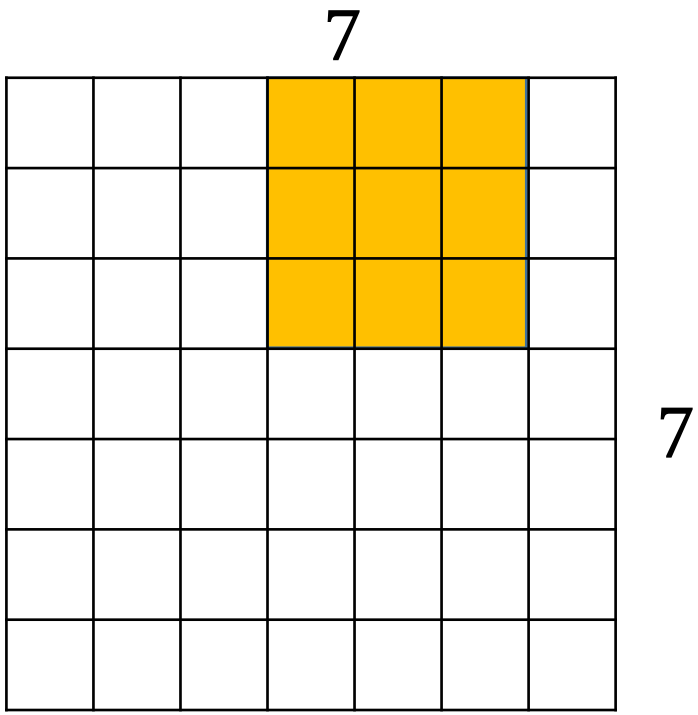
Acertando os Cálculos



Entrada 7×7

Filtro 3×3

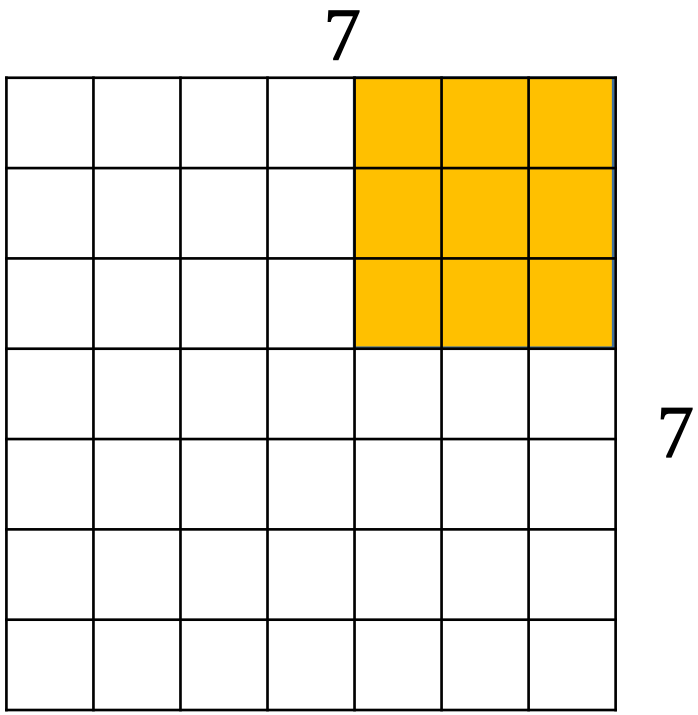
Acertando os Cálculos



Entrada 7×7

Filtro 3×3

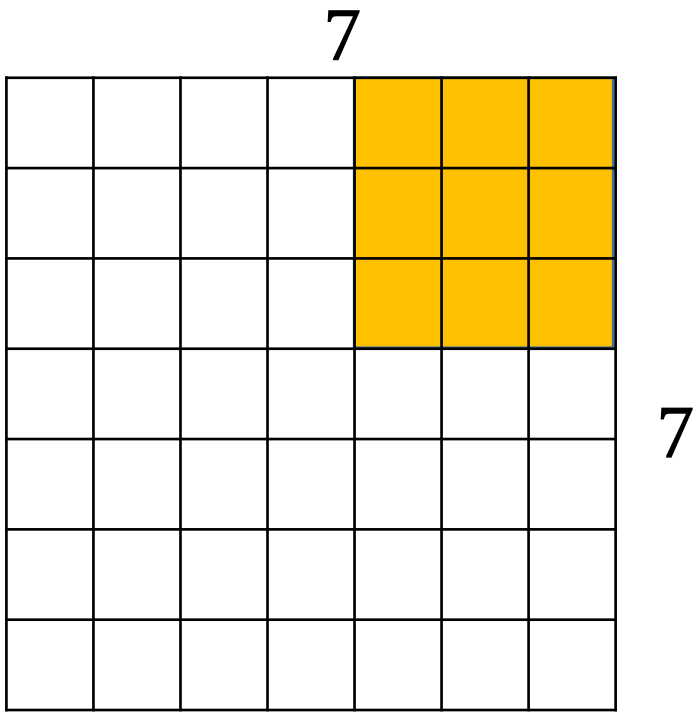
Acertando os Cálculos



Entrada 7×7

Filtro 3×3

Acertando os Cálculos

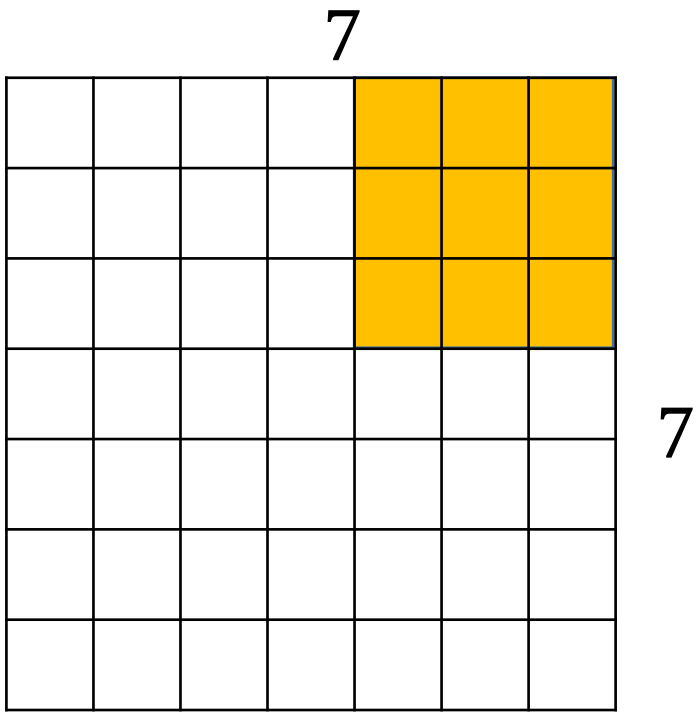


Entrada 7×7

Filtro 3×3

Tamanho do mapa de ativação?

Acertando os Cálculos



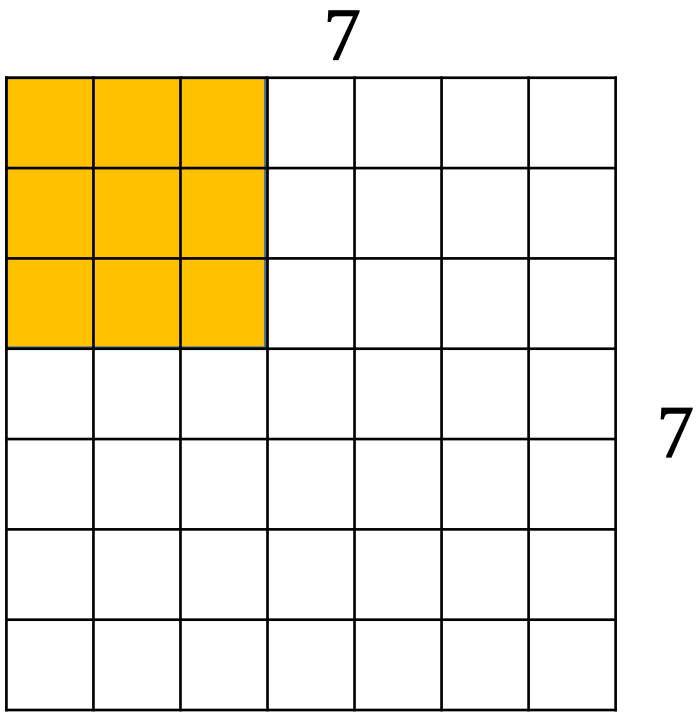
Entrada 7×7

Filtro 3×3

Tamanho do mapa de ativação?

5×5

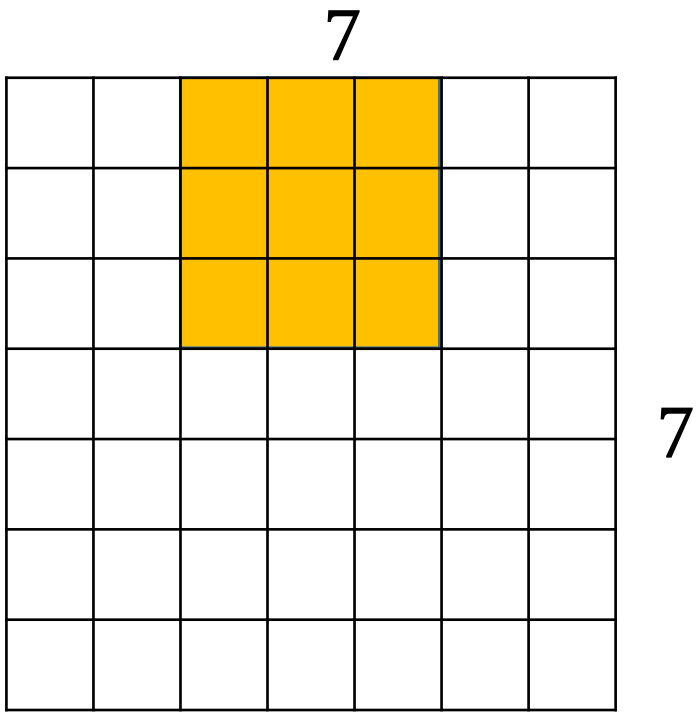
Acertando os Cálculos



Entrada 7×7

Filtro 3×3, *stride* (passo) 2

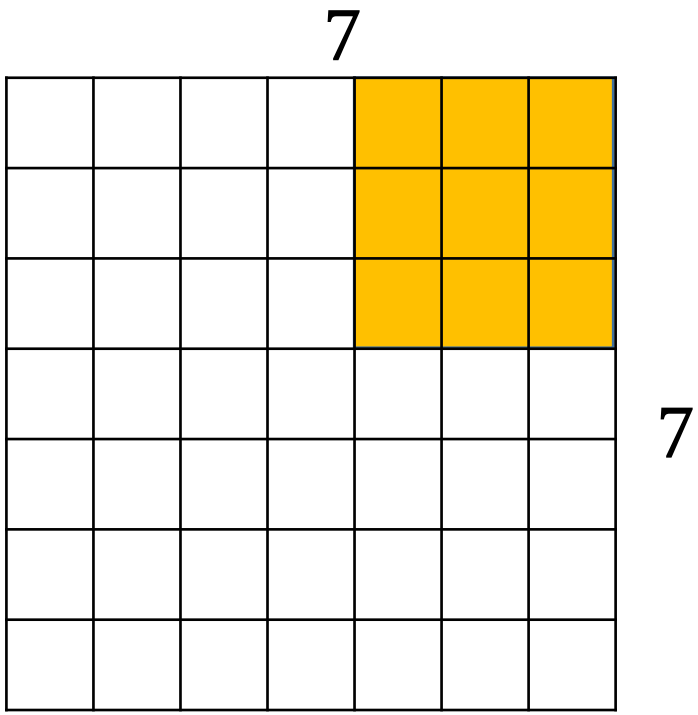
Acertando os Cálculos



Entrada 7×7

Filtro 3×3, *stride* (passo) 2

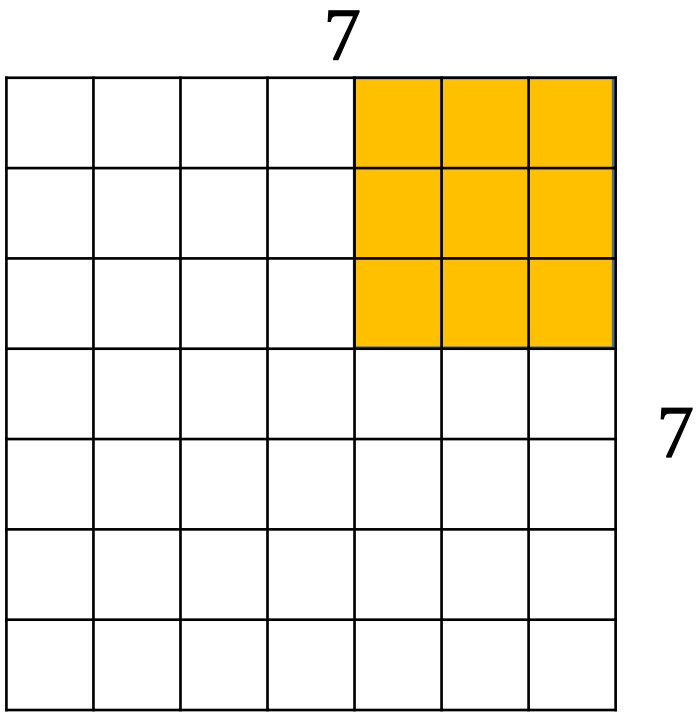
Acertando os Cálculos



Entrada 7×7

Filtro 3×3, *stride* (passo) 2

Acertando os Cálculos

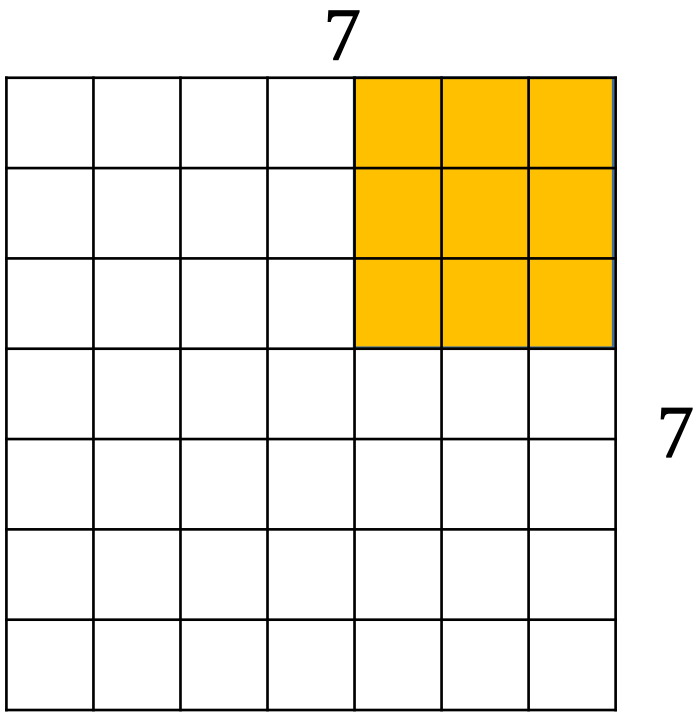


Entrada 7×7

Filtro 3×3, *stride* (passo) 2

Tamanho do mapa de ativação?

Acertando os Cálculos



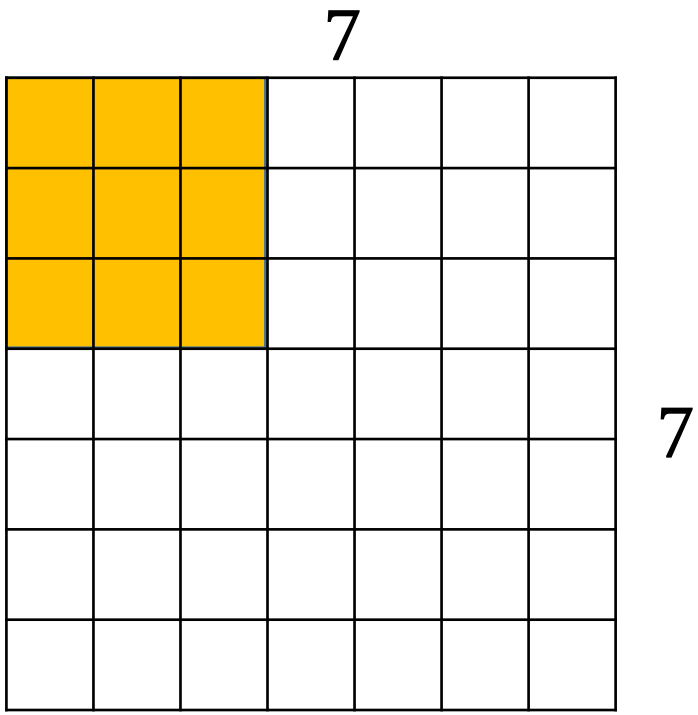
Entrada 7×7

Filtro 3×3, *stride* (passo) 2

Tamanho do mapa de ativação?

3×3

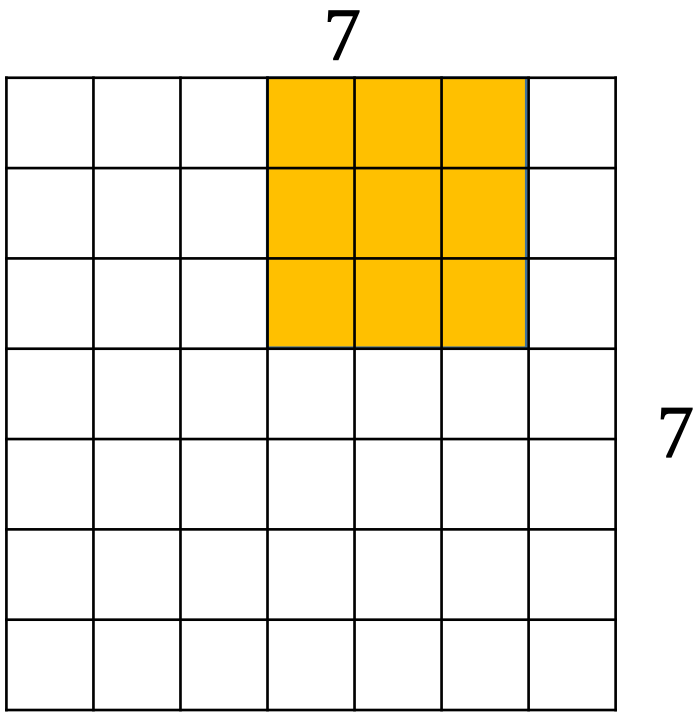
Acertando os Cálculos



Entrada 7×7

Filtro 3×3, *stride* (passo) 3

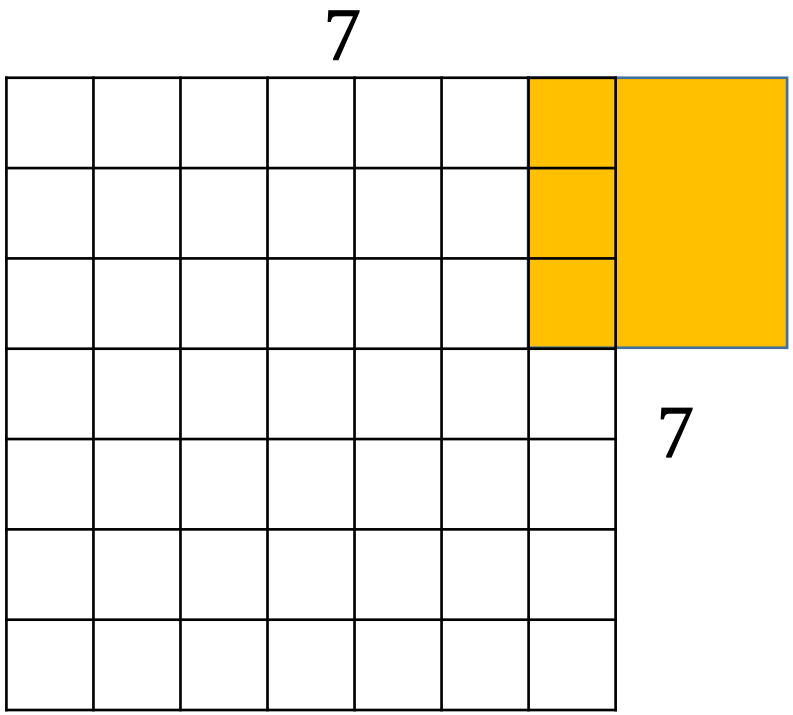
Acertando os Cálculos



Entrada 7×7

Filtro 3×3, *stride* (passo) 3

Acertando os Cálculos



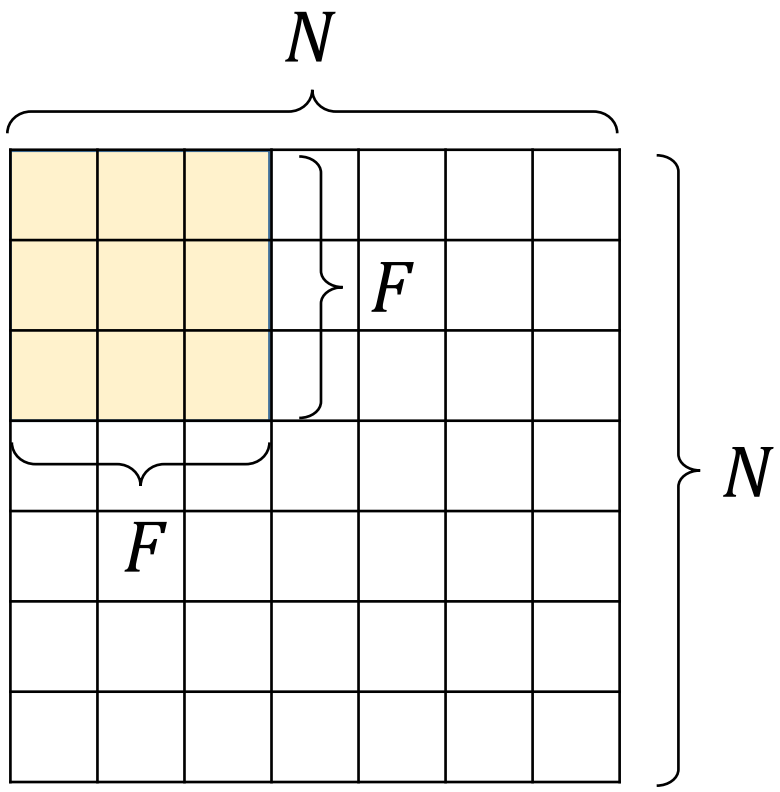
Entrada 7×7

Filtro 3×3, *stride* (passo) 3

Não cabe!!

Não podemos aplicar filtro 3×3 em entrada 7×7 com stride 3

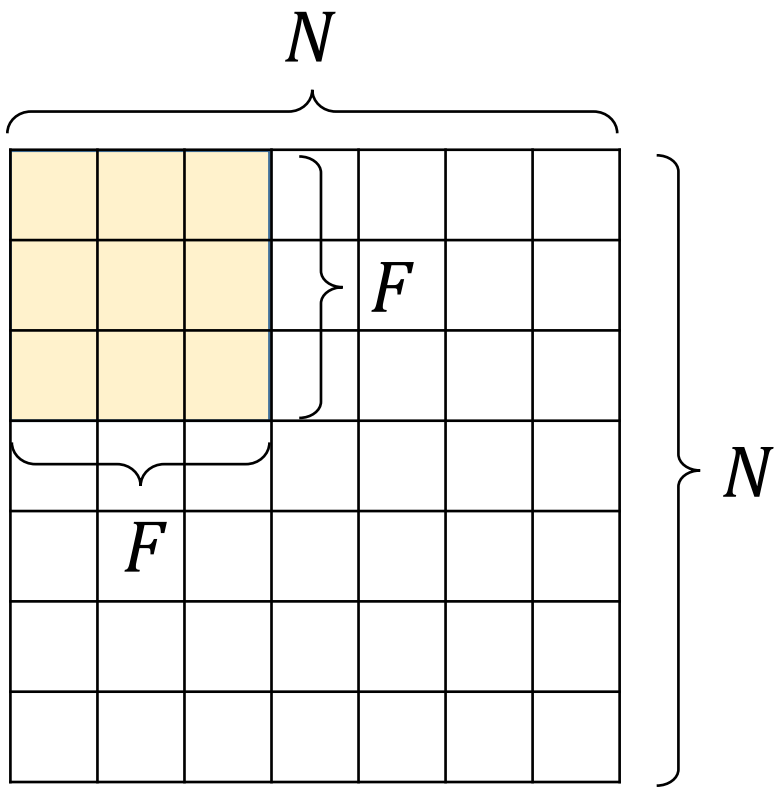
Acertando os Cálculos



Fórmula para Tamanho da Saída:

$$\frac{(N-F)}{stride}+1$$

Acertando os Cálculos



Fórmula para Tamanho da Saída:

$$\frac{(N-F)}{stride}+1$$

ex: $N=7, F=3$

$stride\ 1$: $\frac{(7-3)}{1}+1=5$

$stride\ 2$: $\frac{(7-3)}{2}+1=3$

$stride\ 3$: $\frac{(7-3)}{3}+1=2.33$

Acertando os Cálculos

Dica: preencher bordas com zeros (zero-padding)

0	0	0	0	0	0	0	0	0
0								0
0								0
0								0
0								0
0								0
0								0
0								0
0	0	0	0	0	0	0	0	0

Entrada 7×7

Filtro 3×3, ***stride*** **1**

pad 1 (pixel nas bordas)

Acertando os Cálculos

Dica: preencher bordas com zeros (zero-padding)

0	0	0	0	0	0	0	0	0
0								0
0								0
0								0
0								0
0								0
0								0
0								0
0								0
0	0	0	0	0	0	0	0	0

Entrada 7×7

Filtro 3×3, ***stride*** **1**

pad 1 (pixel nas bordas)

Tamanho do mapa de ativação?

Lembre-se: $\frac{(N-F)}{stride} + 1$

Acertando os Cálculos

Dica: preencher bordas com zeros (zero-padding)

0	0	0	0	0	0	0	0	0
0								0
0								0
0								0
0								0
0								0
0								0
0								0
0	0	0	0	0	0	0	0	0

Entrada 7×7

Filtro 3×3, ***stride*** **1**

pad 1 (pixel nas bordas)

Tamanho do mapa de ativação?

7×7

Acertando os Cálculos

Dica: preencher bordas com zeros (zero-padding)

0	0	0	0	0	0	0	0	0
0								0
0								0
0								0
0								0
0								0
0								0
0								0
0								0
0	0	0	0	0	0	0	0	0

Entrada 7×7

Filtro 3×3, ***stride* 1**

pad 1 (pixel nas bordas)

Tamanho do mapa de ativação?

7×7

É comum utilizar camadas CONV com stride 1, filtros $F\times F$, zero-padding de $(F-1)/2$ (preserva tamanho espacial da entrada)

Acertando os Cálculos

Dica: preencher bordas com zeros (zero-padding)

0	0	0	0	0	0	0	0	0
0								0
0								0
0								0
0								0
0								0
0								0
0								0
0	0	0	0	0	0	0	0	0

Entrada 7×7

Filtro 3×3, ***stride* 1**

pad 1 (pixel nas bordas)

Tamanho do mapa de ativação?

7×7

É comum utilizar camadas CONV com stride 1, filtros $F\times F$, zero-padding de $(F-1)/2$ (preserva tamanho espacial da entrada)

Exemplos: $F = 3 \rightarrow pad = 1$

$F = 5 \rightarrow pad = 2$

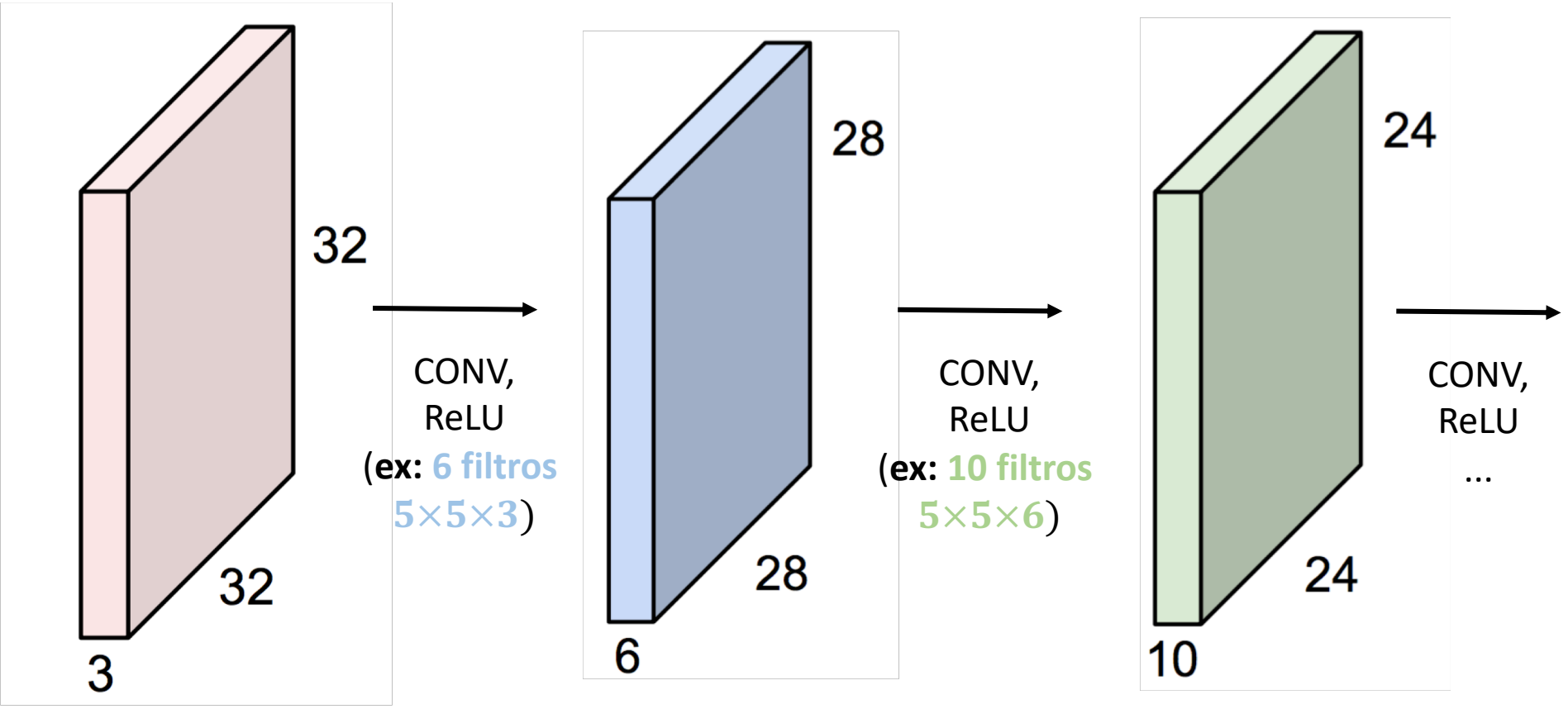
$F = 7 \rightarrow pad = 3$

$$\frac{(N + (2\times pad) - F)}{stride} + 1$$

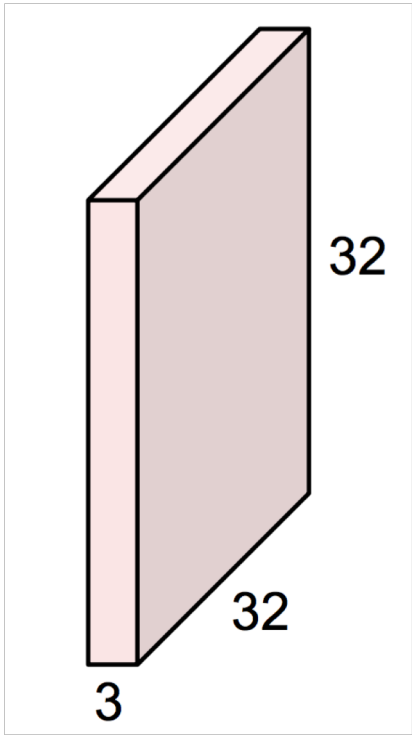
Voltando ao exemplo anterior…

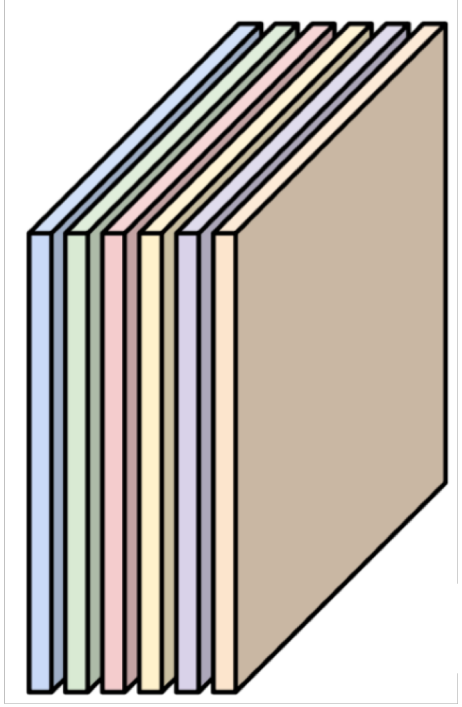
Note que a entrada 32×32 convoluída repetidamente por filtros 5×5 encolhe os volumes espacialmente! (32 → 28 → 24 …)

Encolher os volumes muito rapidamente NÃO funciona bem!



Vamos praticar?



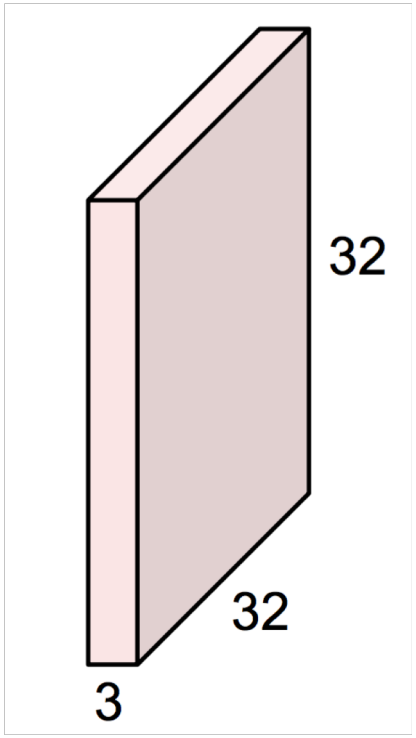


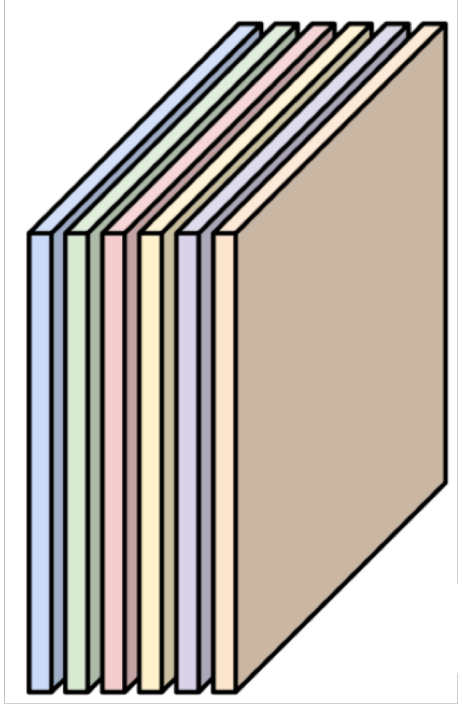
Volume de entrada:
$32 \times 32 \times 3$

10 filtros 5×5, stride 1, pad 2

Tamanho do volume de saída?

Vamos praticar?





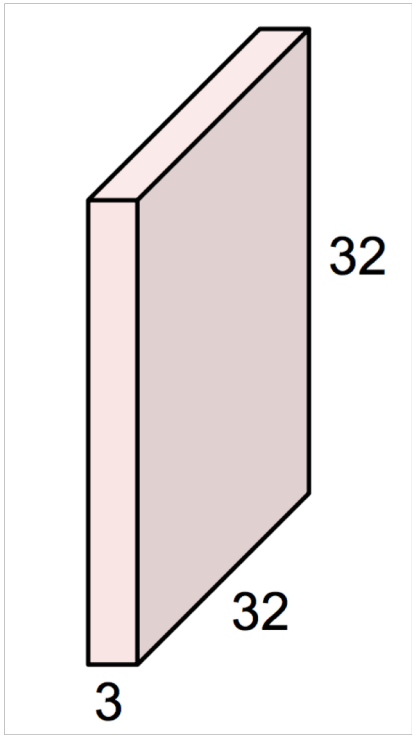
Volume de entrada:
$32 \times 32 \times 3$

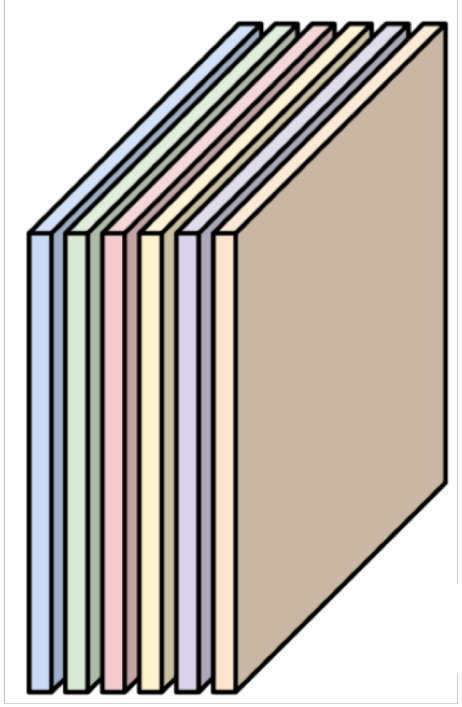
10 filtros 5×5, stride 1, pad 2

Tamanho do volume de saída?

$$\frac{(N + (2 \times pad) - F)}{stride} + 1$$

Vamos praticar?





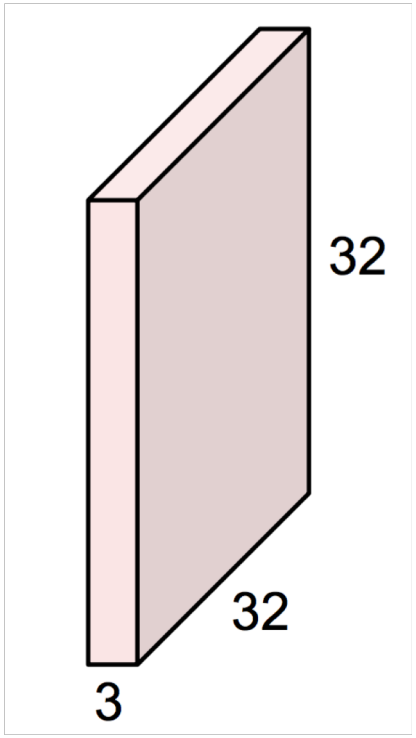
Volume de entrada:
$32 \times 32 \times 3$

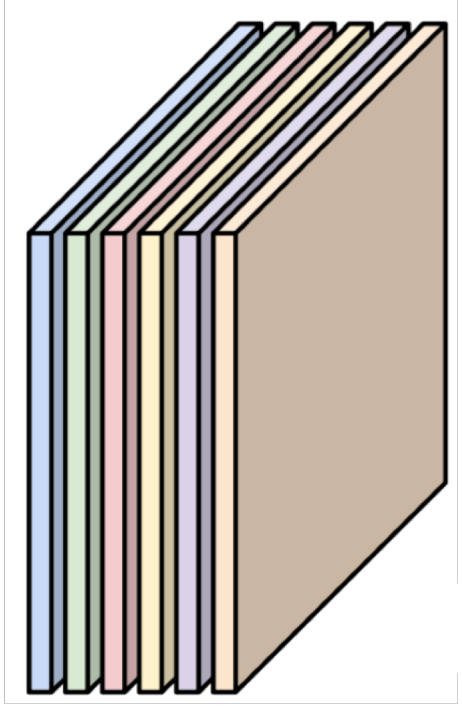
10 filtros **5×5**, **stride 1**, **pad 2**

Tamanho do volume de saída?

$$\frac{(N + (2 \times pad) - F)}{stride} + 1$$

Vamos praticar?





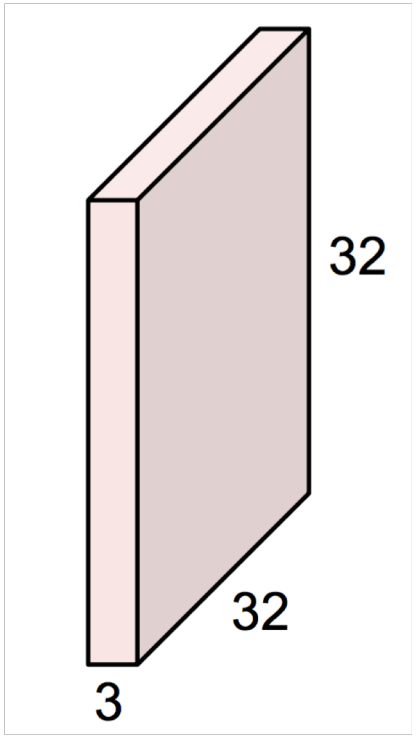
Volume de entrada:
$\mathbf{32}\times\mathbf{32}\times 3$

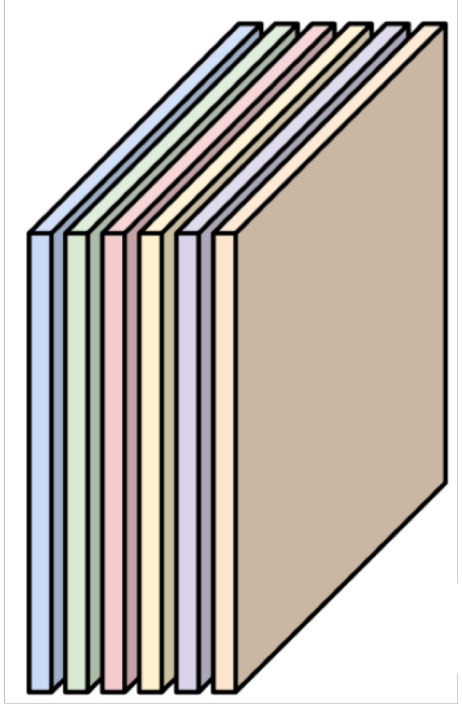
10 filtros $\mathbf{5}\times\mathbf{5}$, **stride 1**, **pad 2**

Tamanho do volume de saída?

$$\frac{(N + (2\times pad) - F)}{stride} + 1 \quad = \frac{(\mathbf{32} + (2\times\mathbf{2}) - \mathbf{5})}{\mathbf{1}} + 1$$

Vamos praticar?





Volume de entrada:
$\mathbf{32}\times\mathbf{32}\times 3$

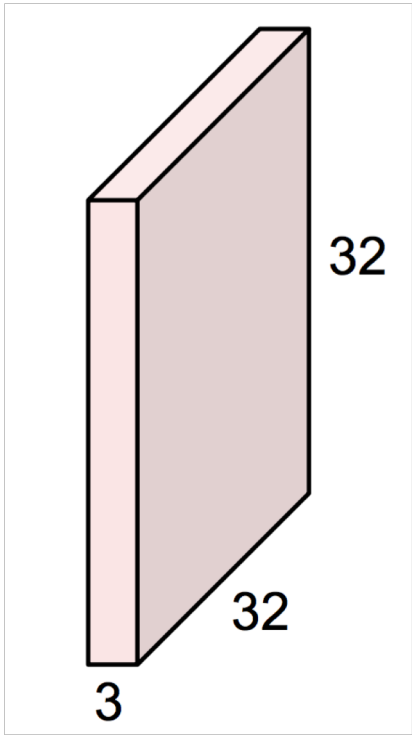
10 filtros $\mathbf{5}\times\mathbf{5}$, **stride 1**, **pad 2**

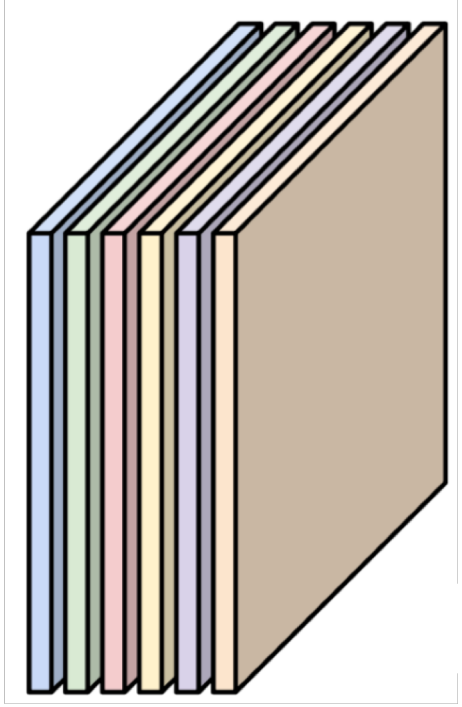
Tamanho do volume de saída?

$$\frac{(N + (2\times pad) - F)}{stride} + 1 = \frac{(\mathbf{32} + (2\times\mathbf{2}) - \mathbf{5})}{\mathbf{1}} + 1$$

$$= 32 \text{ (espacialmente)}$$

Vamos praticar?





Volume de entrada:
$32 \times 32 \times 3$

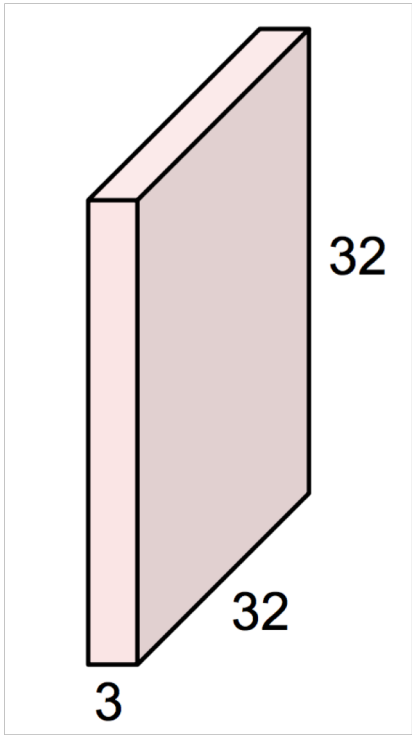
10 filtros **5×5**, **stride 1**, **pad 2**

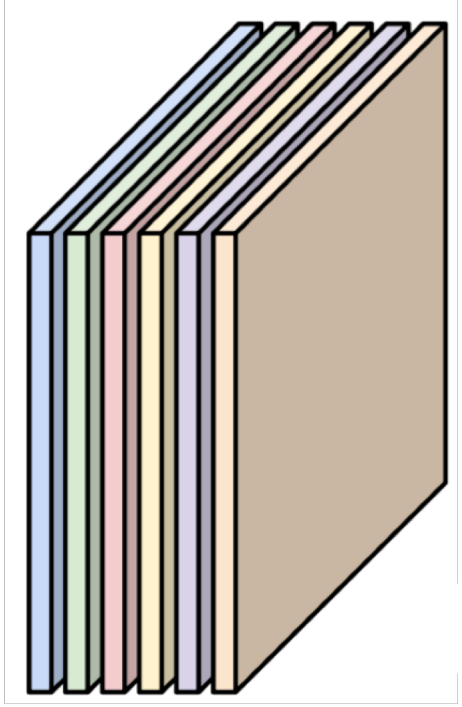
Tamanho do volume de saída?

$$\frac{(N + (2 \times pad) - F)}{stride} + 1 = \frac{(32 + (2 \times 2) - 5)}{1} + 1$$

$$= 32 \text{ (espacialmente)} \qquad 32 \times 32 \times 10$$

Vamos praticar?



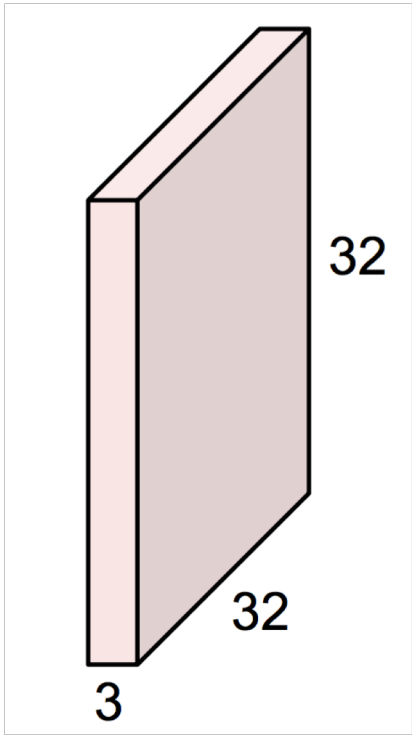


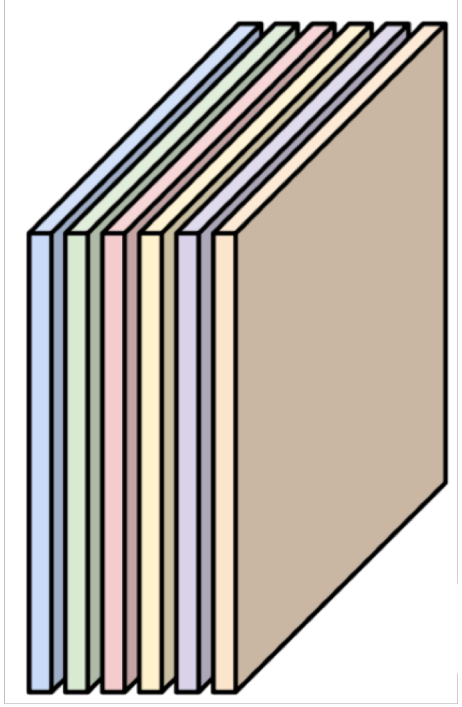
Volume de entrada:
$32 \times 32 \times 3$

10 filtros 5×5, stride 1, pad 2

Número de parâmetros desta camada?

Vamos praticar?



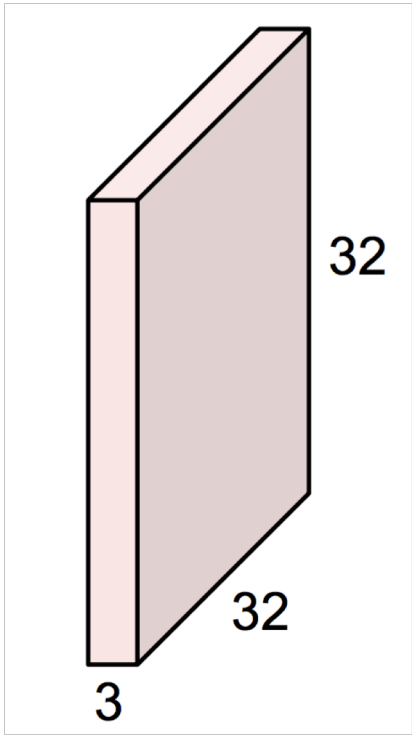


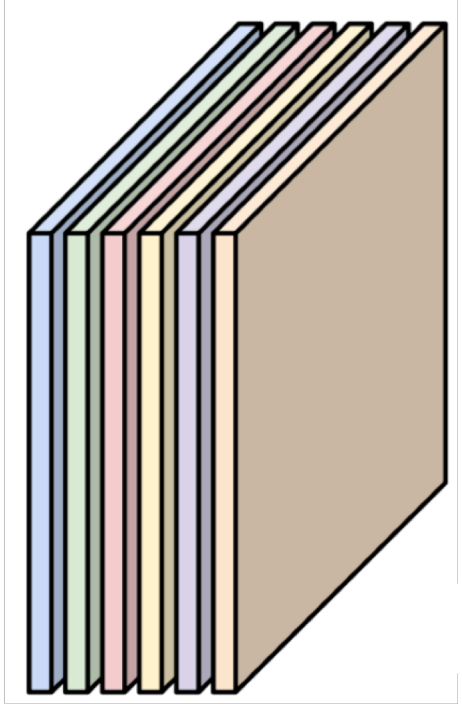
Volume de entrada:
$32 \times 32 \times 3$

10 filtros 5×5, stride 1, pad 2

Número de parâmetros desta camada?

Vamos praticar?





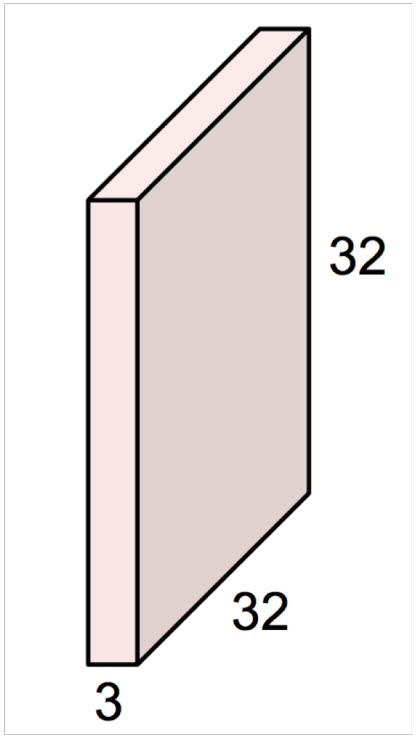
Volume de entrada:
$32 \times 32 \times 3$

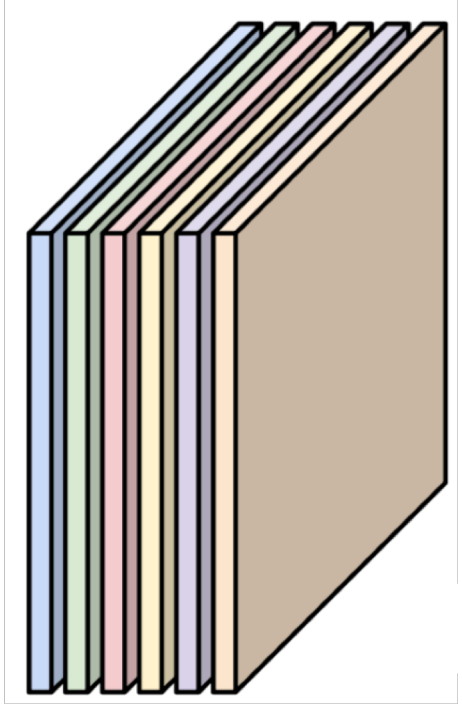
10 filtros 5×5, stride 1, pad 2

Número de parâmetros desta camada?

Cada filtro tem $5 \times 5 \times 3 + 1 = 76$ parâmetros ($+1$ por causa do *bias*)

Vamos praticar?





Volume de entrada:
$32 \times 32 \times 3$

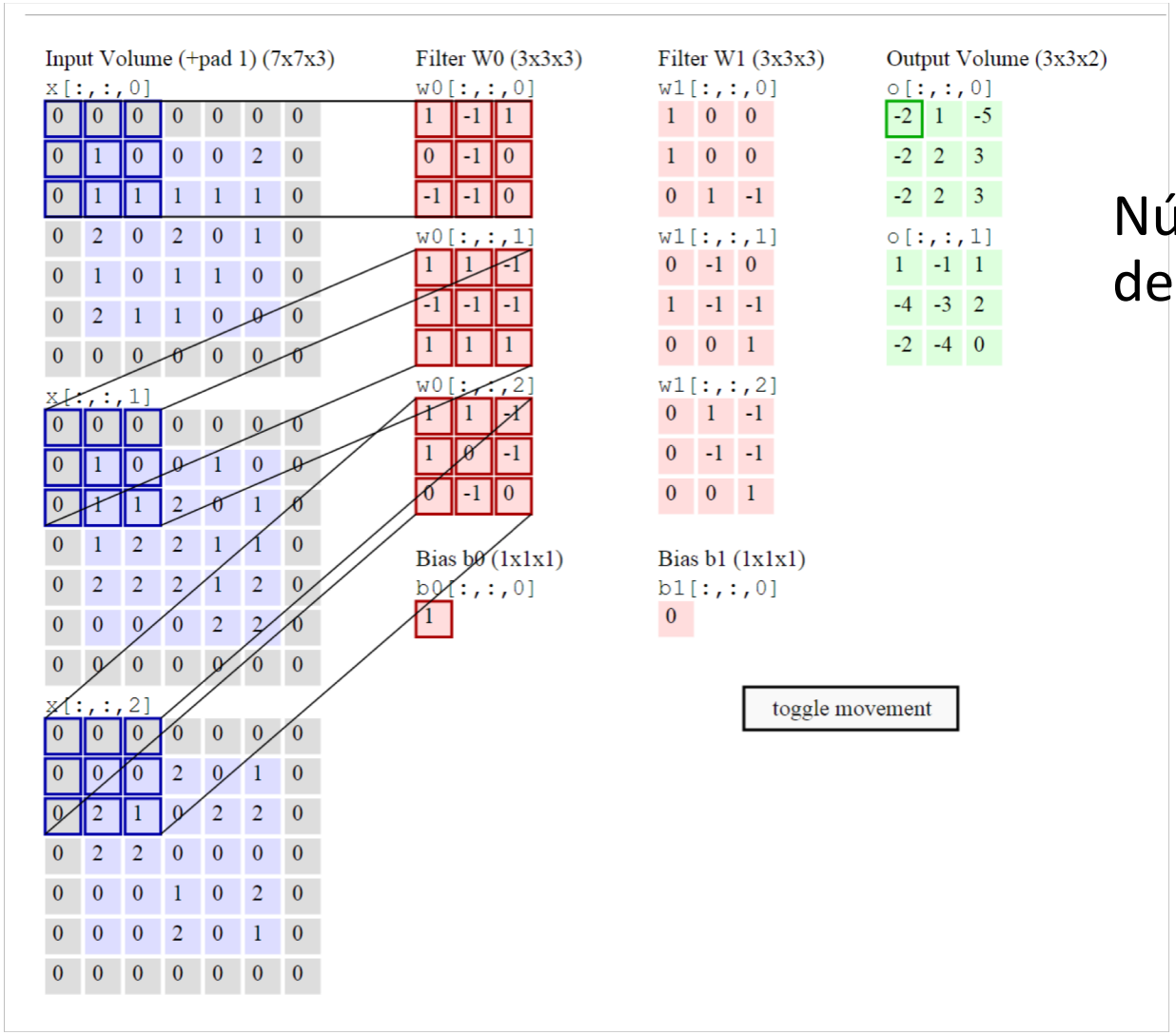
10 filtros 5×5, stride 1, pad 2

Número de parâmetros desta camada?

Cada filtro tem $5 \times 5 \times 3 + 1 = 76$ parâmetros (+1 por causa do *bias*)

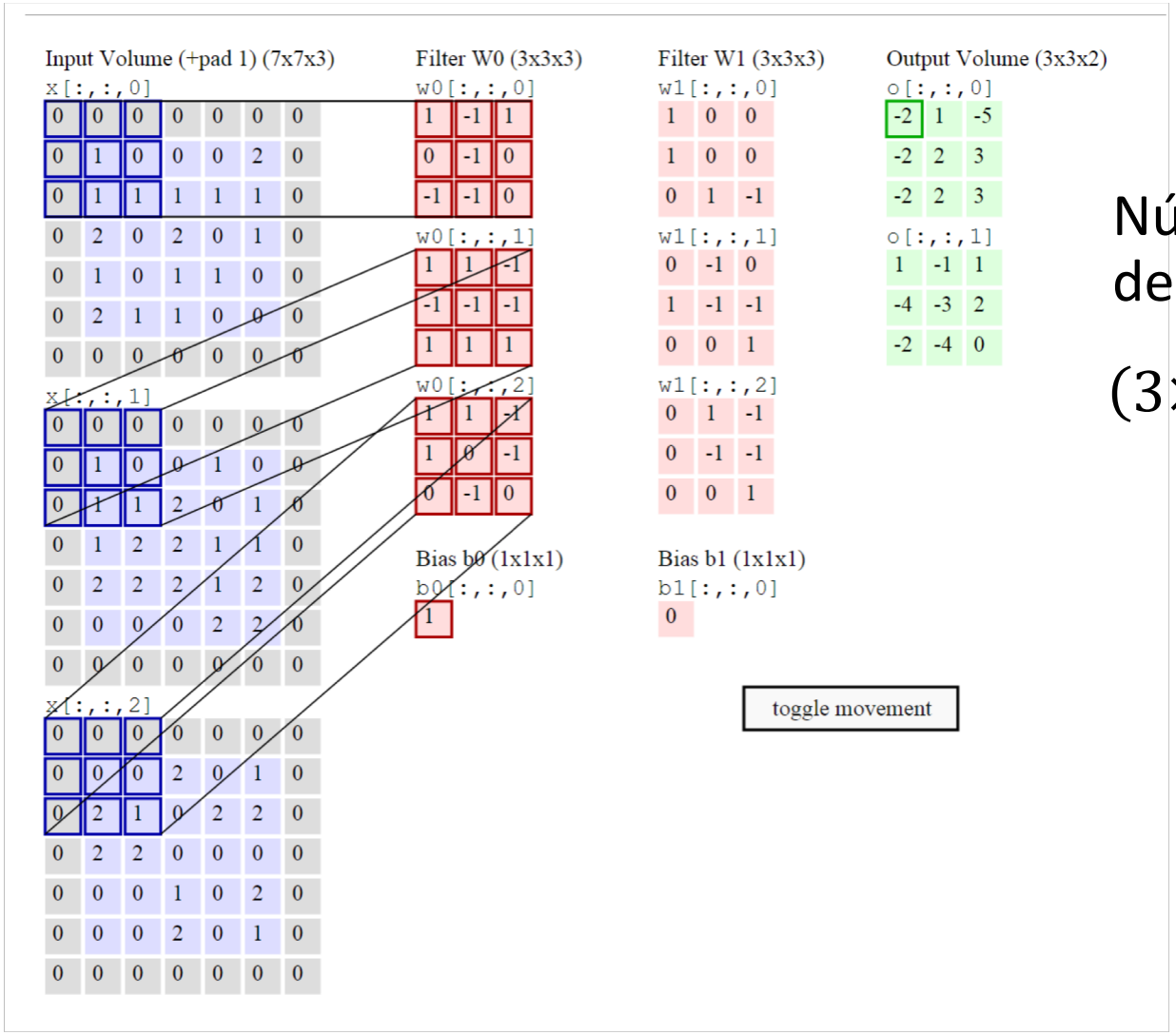
$$\mathbf{76 \times 10 = 760}$$

Vamos praticar?



Número de parâmetros desta camada?

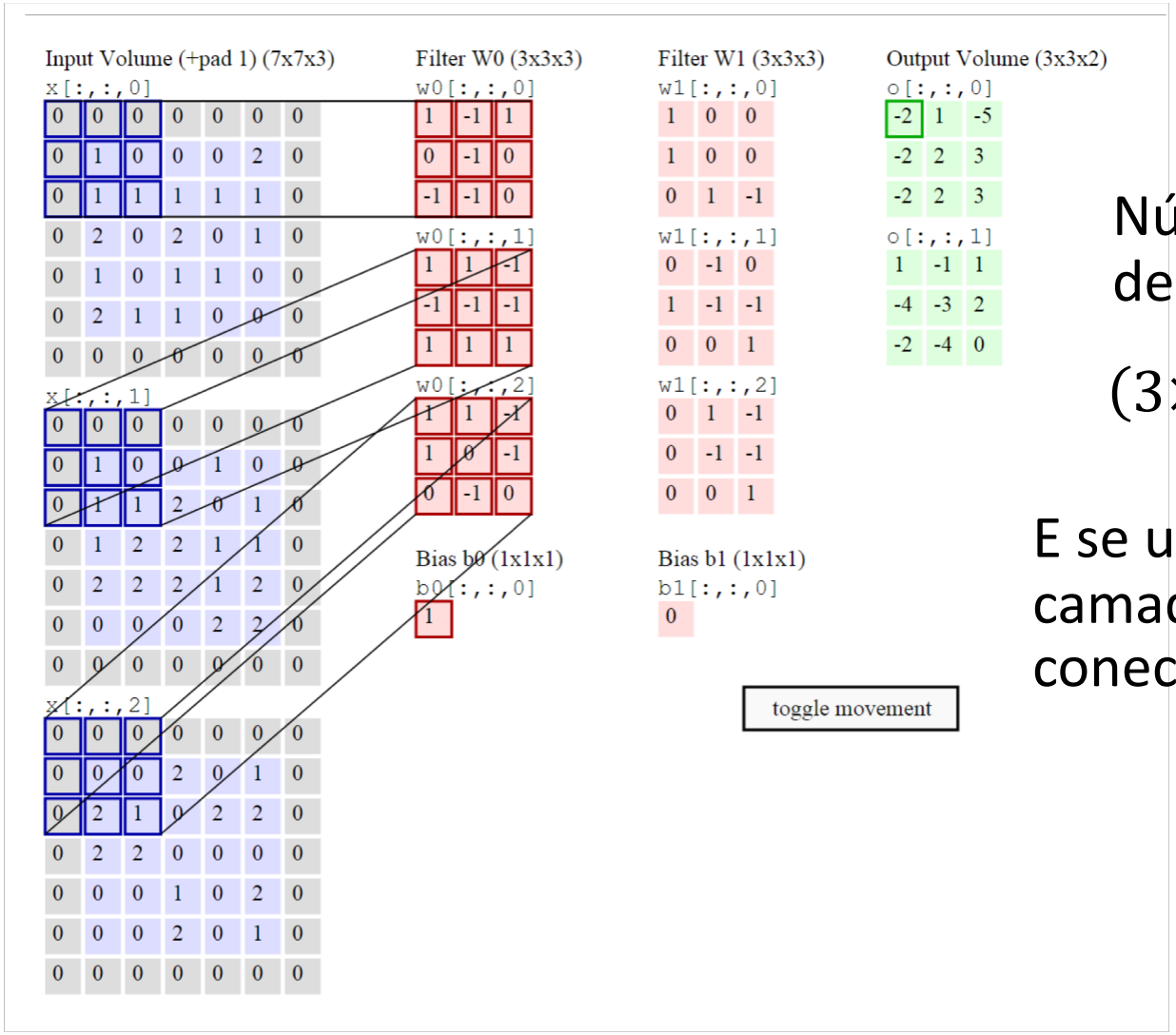
Vamos praticar?



Número de parâmetros desta camada?

$$(3\times3\times3+1)\times 2 = 56$$

Vamos praticar?

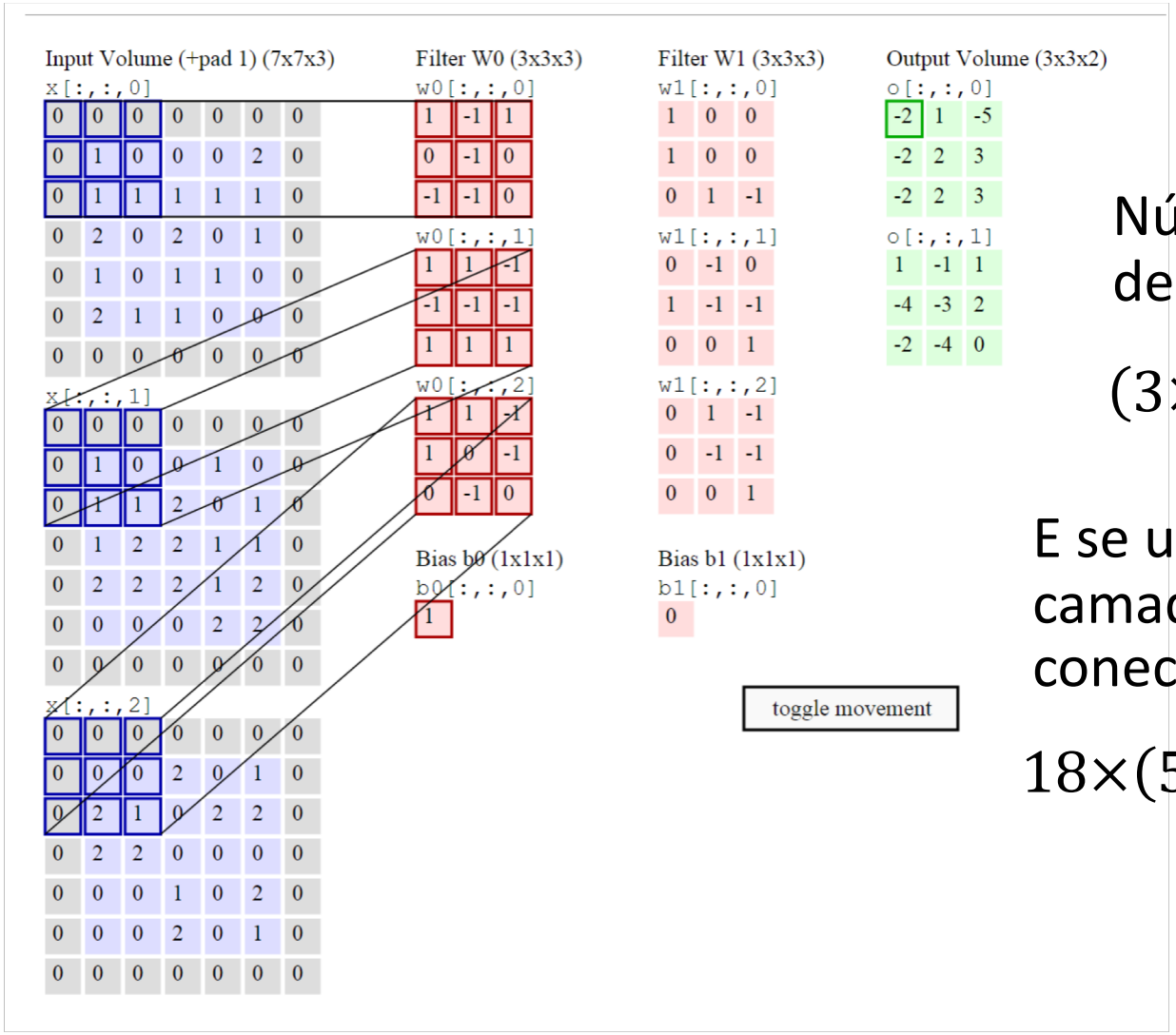


Número de parâmetros desta camada?

$$(3\times3\times3+1)\times 2 = 56$$

E se utilizássemos uma camada totalmente conectada com 18 neurônios?

Vamos praticar?



Número de parâmetros desta camada?

$$(3\times3\times3+1)\times 2 = 56$$

E se utilizássemos uma camada totalmente conectada com 18 neurônios?

$$18\times(5\times5\times3)+18 = 1368$$

Resumo da Camada de Convolução

- Recebe como **entrada** um volume de tamanho $L_1 \times A_1 \times C_1$

- Precisa de **4 hiperparâmetros**:
  - Número de filtros K
  - Tamanho espacial dos filtros F
  - *Stride* (passo) S
  - Quantidade de zero-padding P

- Produz um **volume** de tamanho $L_2 \times A_2 \times C_2$ onde:
  - $L_2 = \frac{L_1 + (2 \times P) - F}{S} + 1$
  - $A_2 = \frac{A_1 + (2 \times P) - F}{S} + 1$
  - $C_2 = K$

- Produz $F \times F \times C_1$ pesos por filtro, totalizando $(F \times F \times C_1) \times K$ pesos e K biases

- No volume resultante, a c-ésima fatia ($L_2 \times A_2$) é resultado de uma convolução válida do c-ésimo filtro sobre o volume de entrada com stride S e transladado pelo c-ésimo bias

Resumo da Camada de Convolução

- Recebe como **entrada** um volume de tamanho $L_1 \times A_1 \times C_1$
- Precisa de **4 hiperparâmetros**:
  - Número de filtros K
  - Tamanho espacial dos filtros F
  - *Stride* (passo) S
  - Quantidade de zero-padding P

Valores frequentemente utilizados:

K =(potências de 2 → $32, 64, 128, \ldots$)

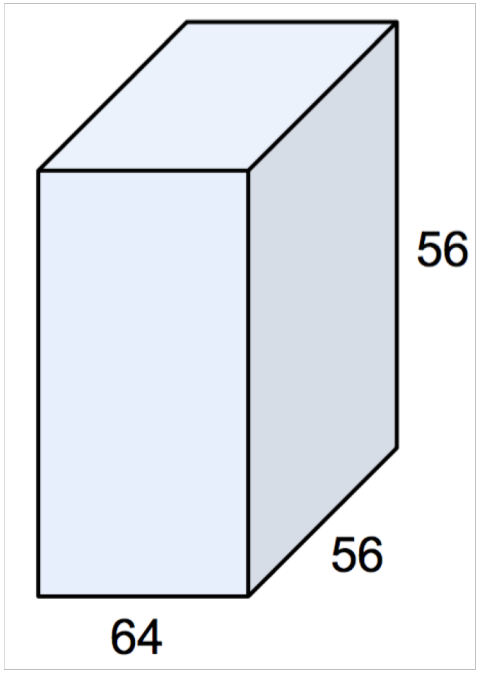
$F = 3, S = 1, P = 1$

$F = 5, S = 1, P = 2$

$F = 1, S = 1, P = 0$

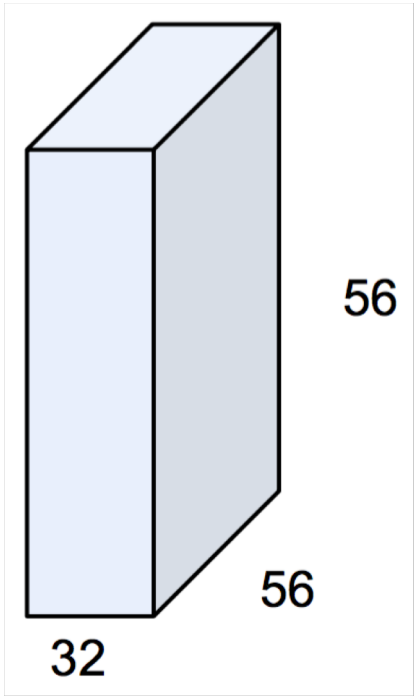
- Produz um **volume** de tamanho $L_2 \times A_2 \times C_2$ onde:
  - $L_2 = \frac{L_1 + (2 \times P) - F}{S} + 1$
  - $A_2 = \frac{A_1 + (2 \times P) - F}{S} + 1$
  - $C_2 = K$
- Produz $F \times F \times C_1$ pesos por filtro, totalizando $(F \times F \times C_1) \times K$ pesos e K biases
- No volume resultante, a c-ésima fatia ($L_2 \times A_2$) é resultado de uma convolução válida do c-ésimo filtro sobre o volume de entrada com stride S e transladado pelo c-ésimo bias

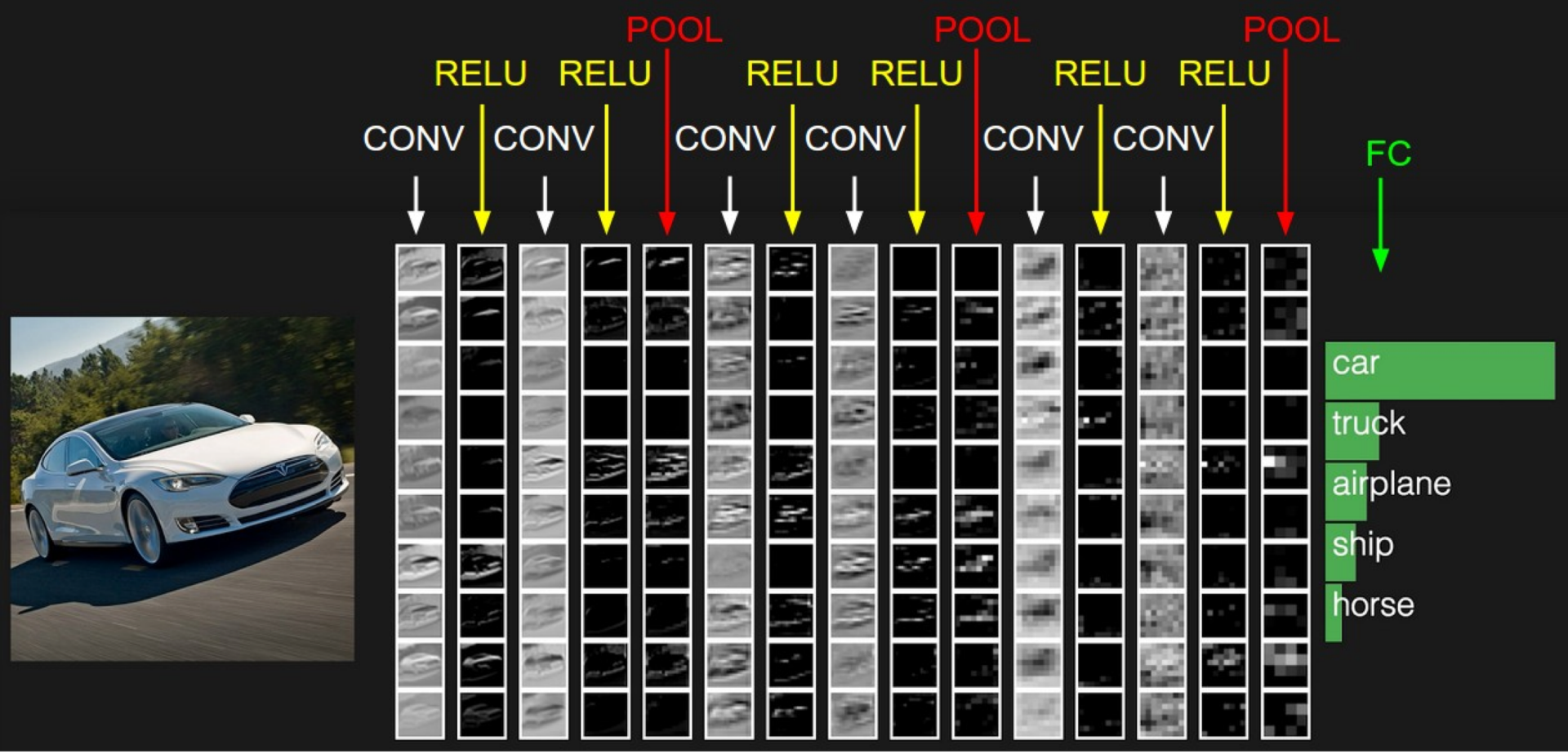
(faz sentido utilizar convoluções 1×1!!)



CONV 1×1
com 32 filtros

Cada filtro tem
tamanho $1\times1\times64$

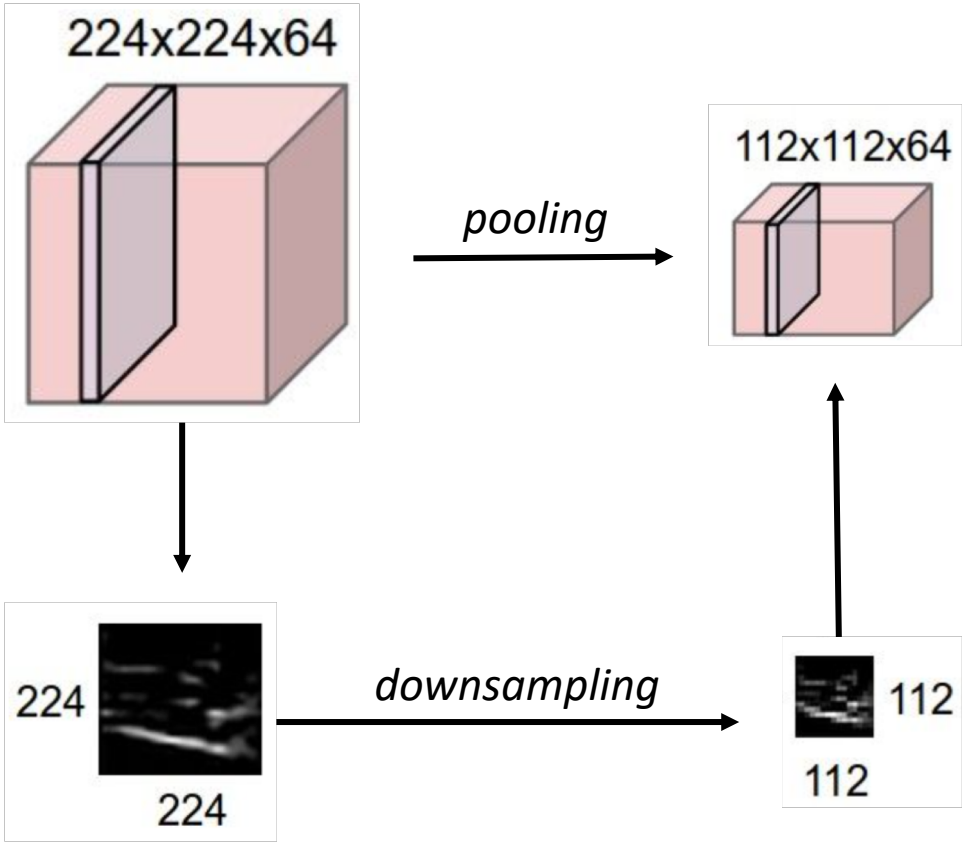




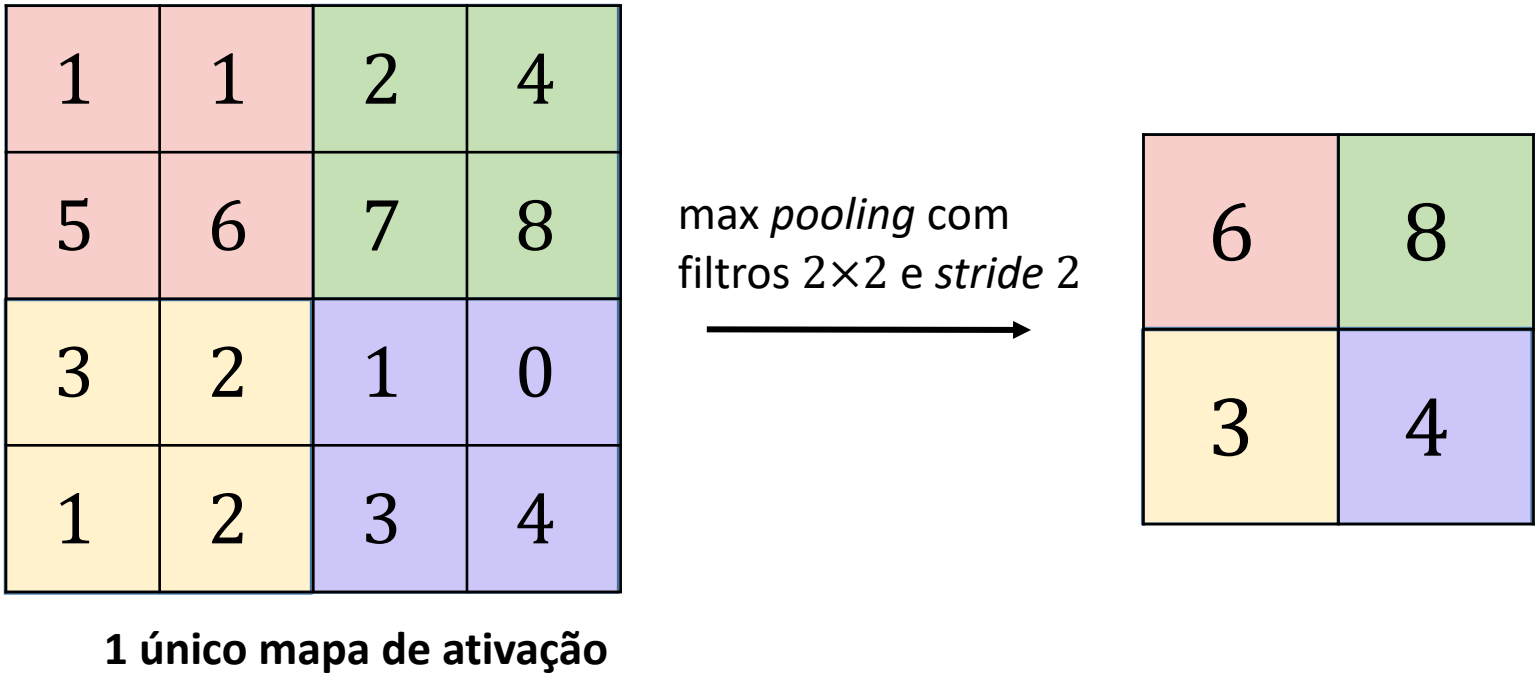
Próxima camada: Pooling!

Camada de *Pooling*

- **Reduz os volumes** para que se tornem gerenciáveis
- Atua sobre cada mapa de ativação de **maneira independente**

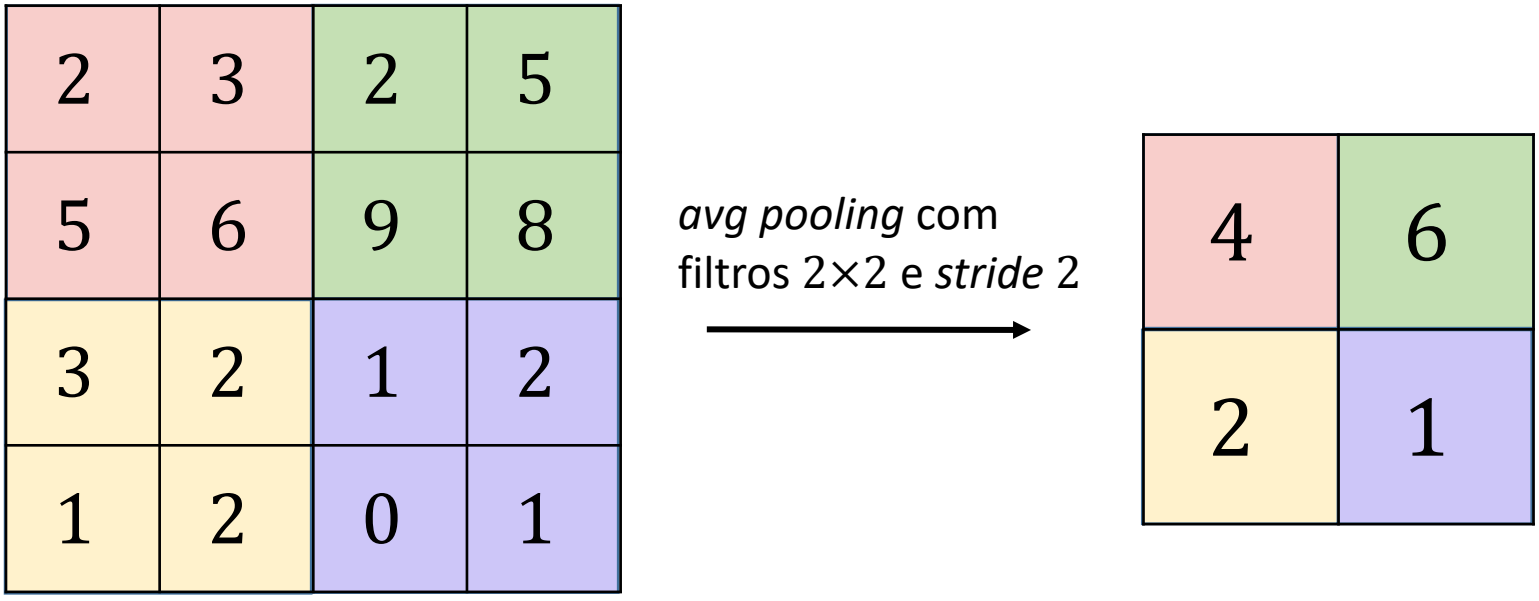


MAX *Pooling*



1 único mapa de ativação

AVERAGE (AVG) *Pooling*



1 único mapa de ativação

Resumo da Camada de *Pooling*

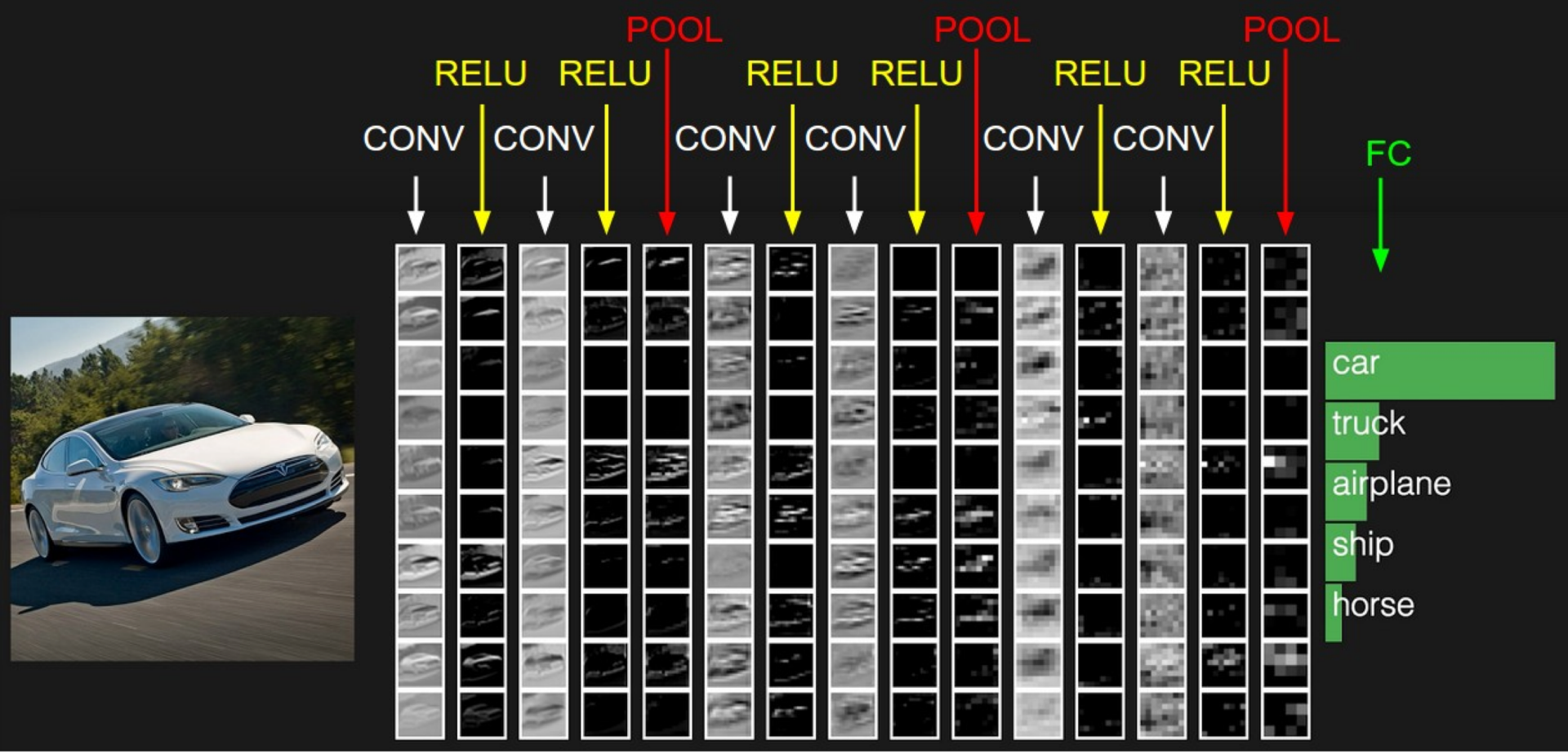
- Recebe como **entrada** um volume de tamanho $L_1 \times A_1 \times C_1$
- Precisa de **2 hiperparâmetros**:
  - Tamanho espacial dos filtros F
  - *Stride* (passo) S
- Produz um **volume** de tamanho $L_2 \times A_2 \times C_2$ onde:
  - $L_2 = \frac{L_1 - F}{S} + 1$
  - $A_2 = \frac{A_1 - F}{S} + 1$
  - $C_2 = C_1$
- **Não introduz parâmetros** pois computa função fixa sobre a entrada
- Note que **não é comum** utilizar *zero-padding* para camadas de *pooling*

Resumo da Camada de *Pooling*

- Recebe como **entrada** um volume de tamanho $L_1 \times A_1 \times C_1$
- Precisa de **2 hiperparâmetros**:
  - Tamanho espacial dos filtros F
  - *Stride* (passo) S

Valores frequentemente utilizados:
- $F = 2,\ S = 2$
- $F = 3,\ S = 2$

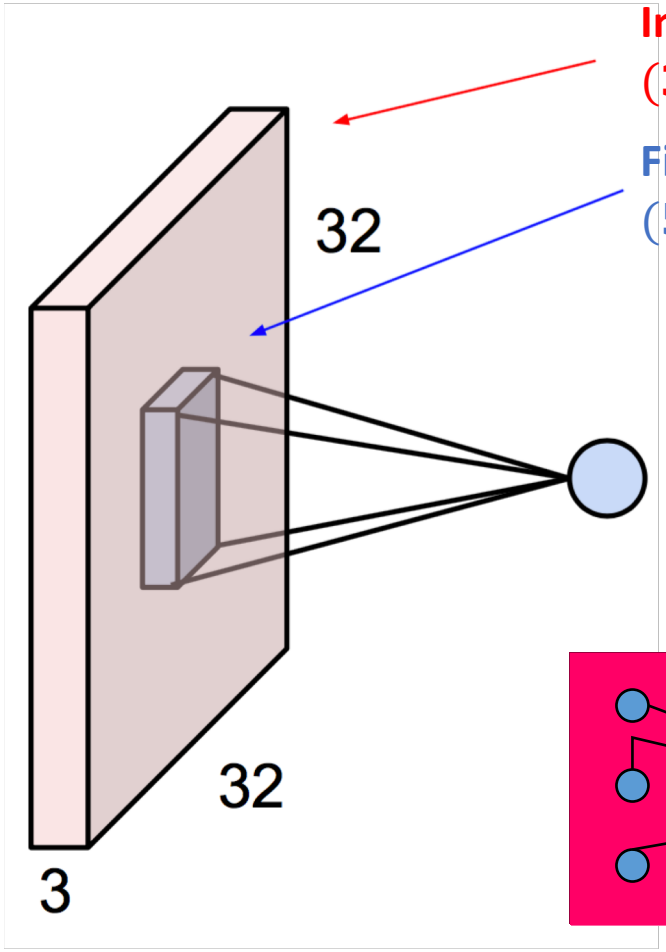
- Produz um **volume** de tamanho $L_2 \times A_2 \times C_2$ onde:
  - $L_2 = \frac{L_1 - F}{S} + 1$
  - $A_2 = \frac{A_1 - F}{S} + 1$
  - $C_2 = C_1$
- **Não introduz parâmetros** pois computa função fixa sobre a entrada
- Note que **não é comum** utilizar *zero-padding* para camadas de *pooling*

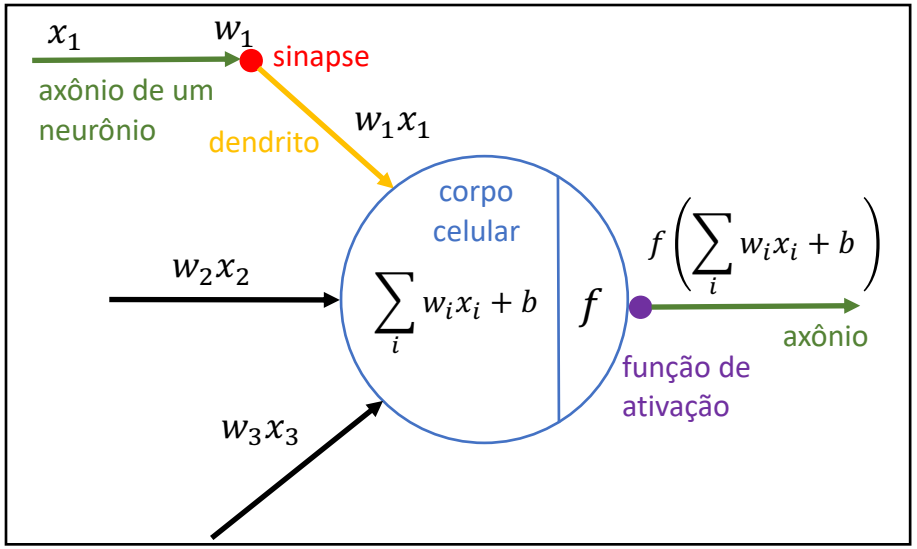


Finalmente: FC (Fully-Connected Layer)

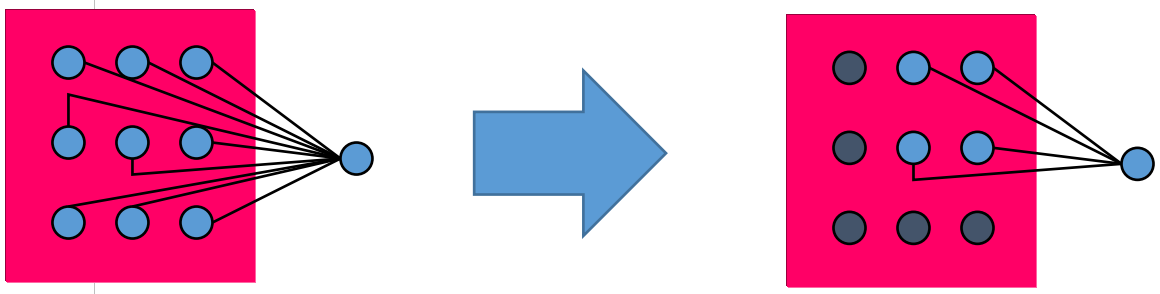
Uma rede convolucional pode conter uma (ou várias) camadas totalmente conectadas, onde os neurônios estão conectados com toda a entrada (redes neurais tradicionais)

Intuição e Analogia c/ Cortex Visual

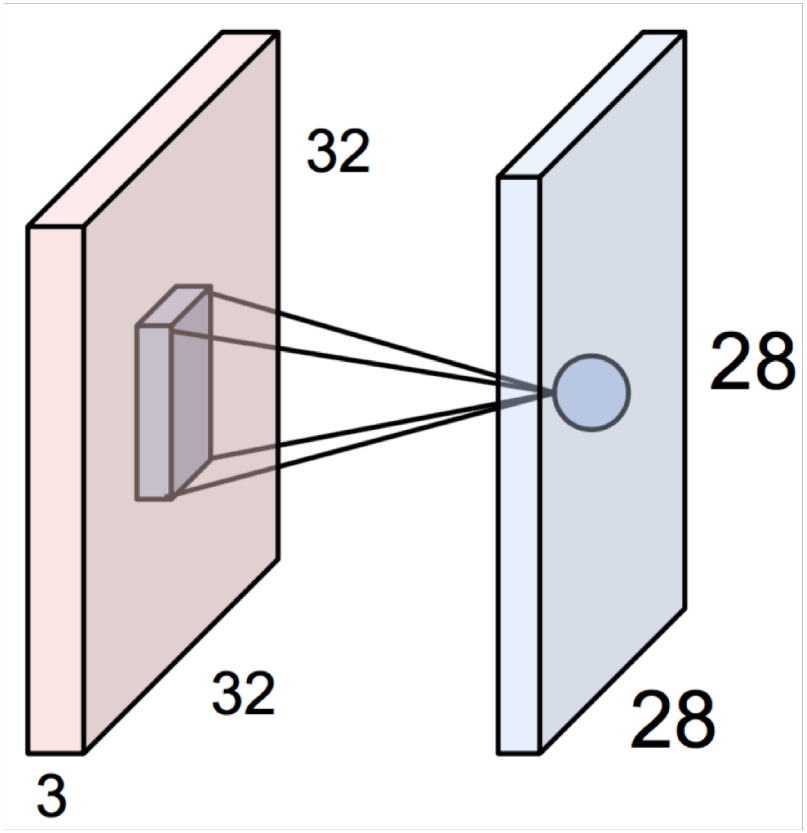


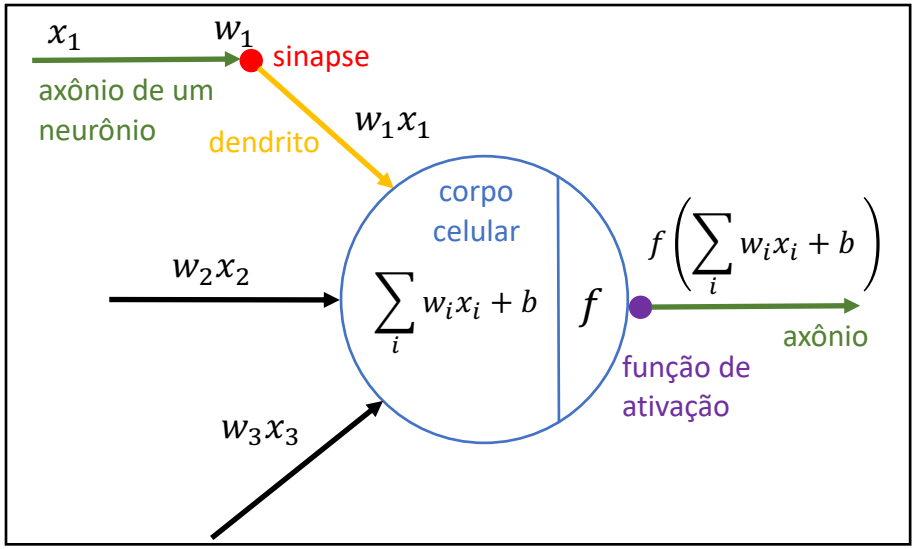


É simplesmente um neurônio com conectividade local!!



Intuição e Analogia c/ Cortex Visual





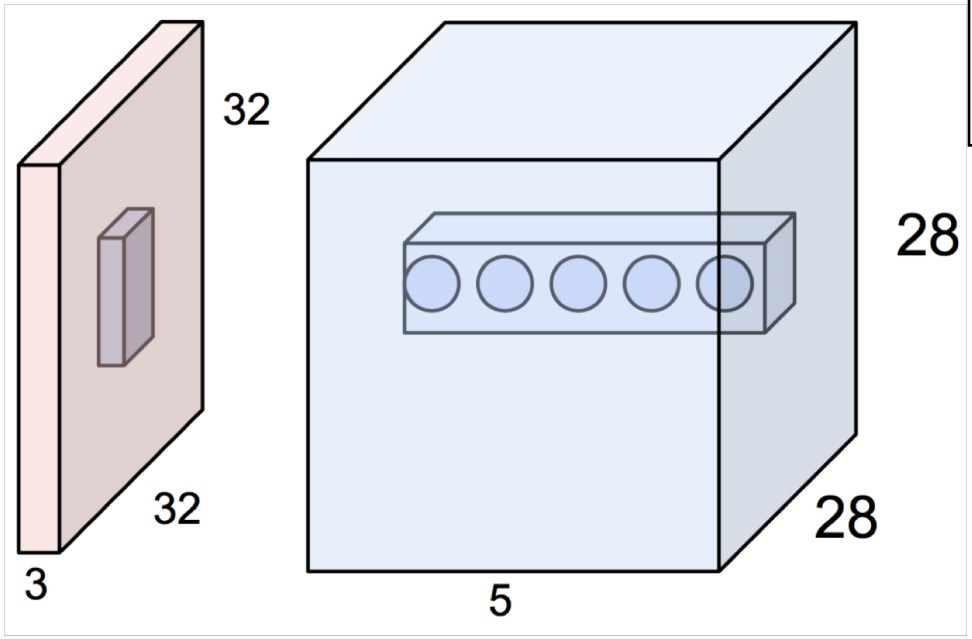
Um mapa de ativação é um conjunto de saída de neurônios:

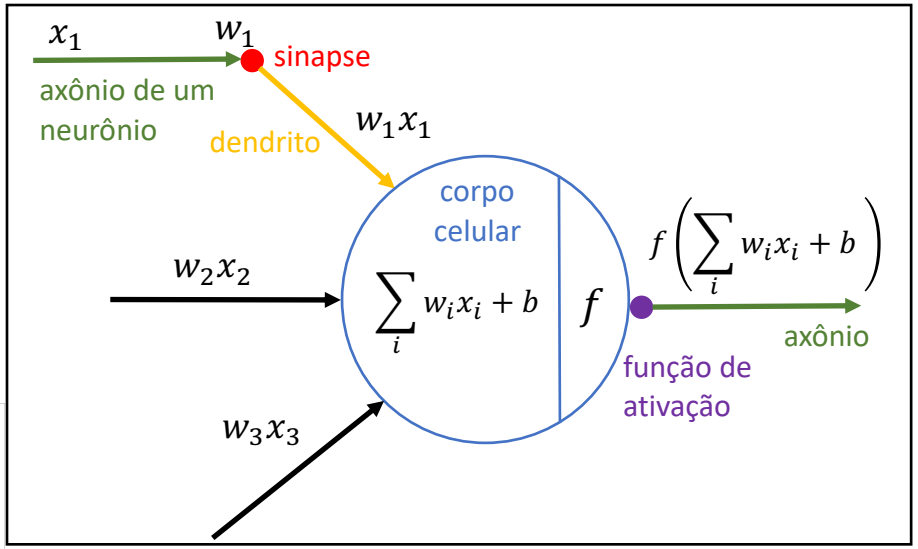
1. Cada um conectado a uma pequena região da entrada
2. Todos eles compartilham parâmetros

Filtro 5×5 = campo receptor de tamanho 5×5 para cada neurônio

Intuição e Analogia c/ Cortex Visual

Ex: camada de convolução com 5 filtros → neurônios dispostos em um grid 3D (28×28×5)



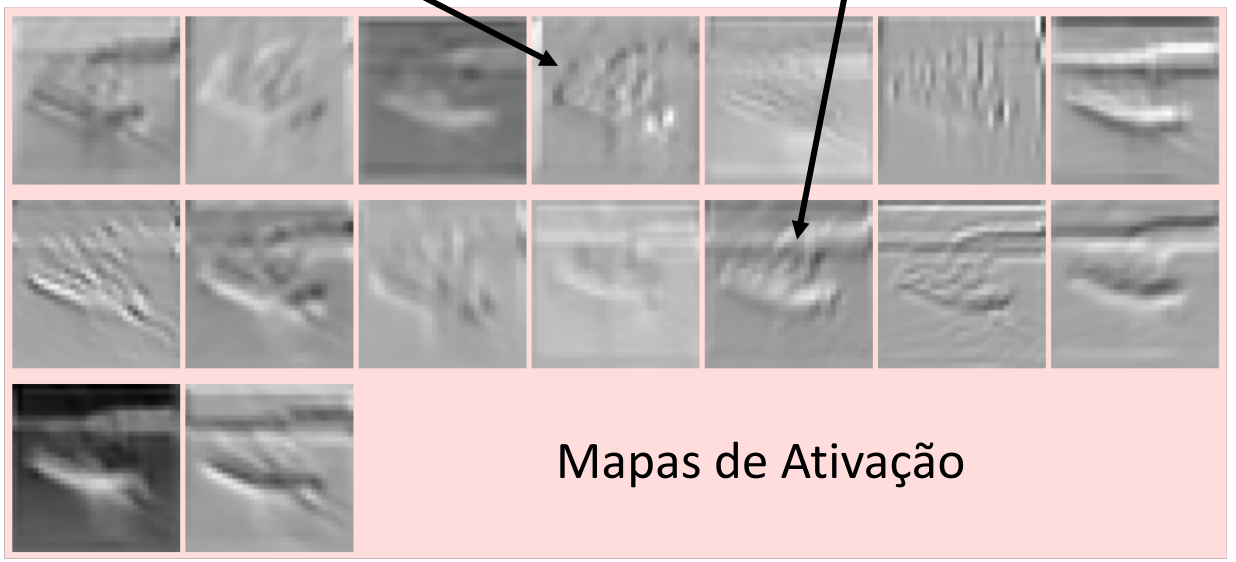


Existem 5 neurônios responsáveis pelo processamento da mesma região do volume de entrada

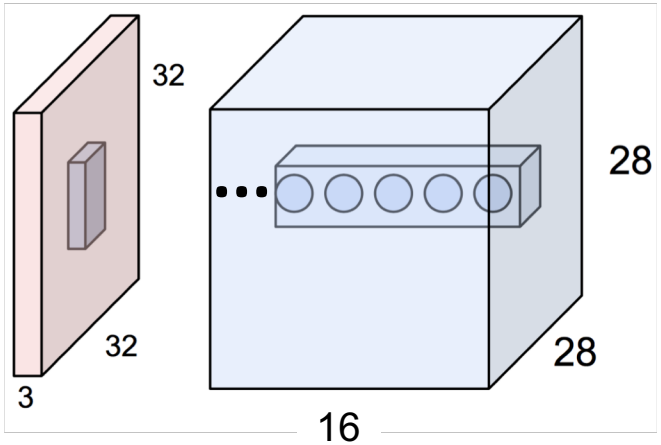
Intuição e Analogia c/ Cortex Visual

Ex: imagem 32×32×3
16 filtros 5×5×3

Filtros:

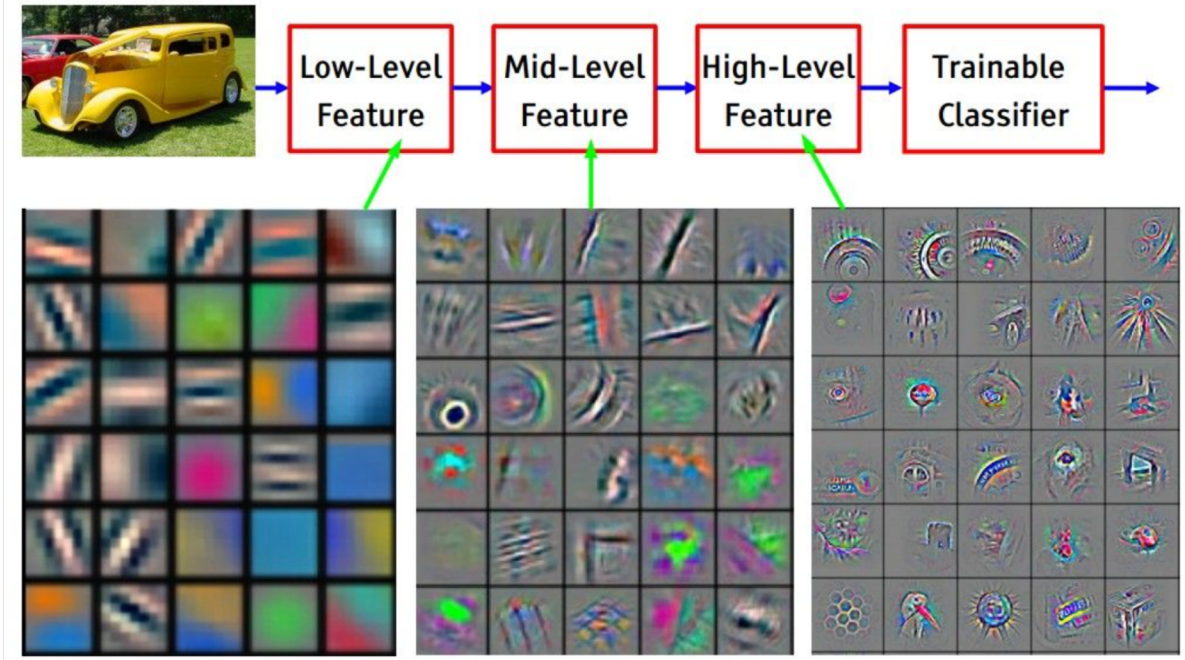


Mapas de Ativação

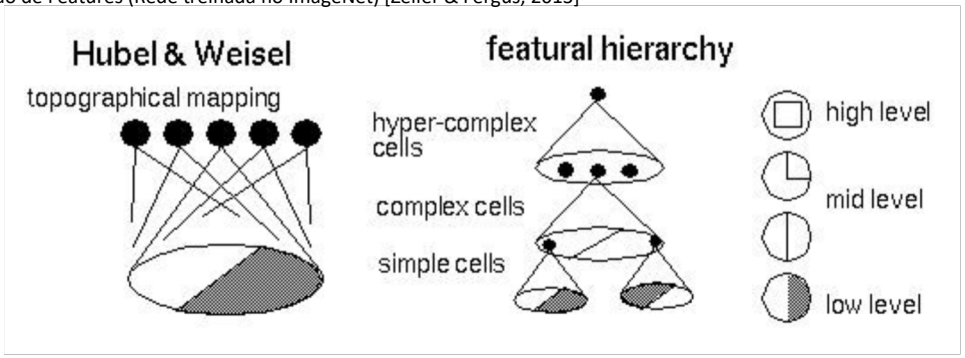


Intuição e Analogia c/ Cortex Visual

Slides Yann LeCun



Visualização de Features (Rede treinada no ImageNet) [Zeiler & Fergus, 2013]



Exemplos de Arquiteturas: LeNet-5

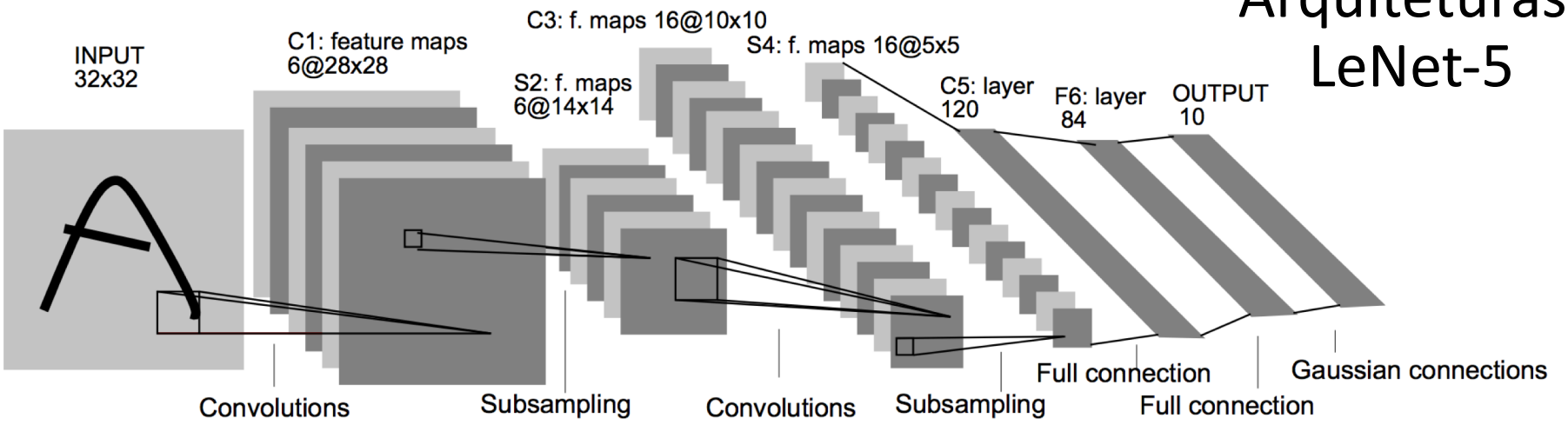


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Filtros Convolucionais: 5×5
Stride: 1

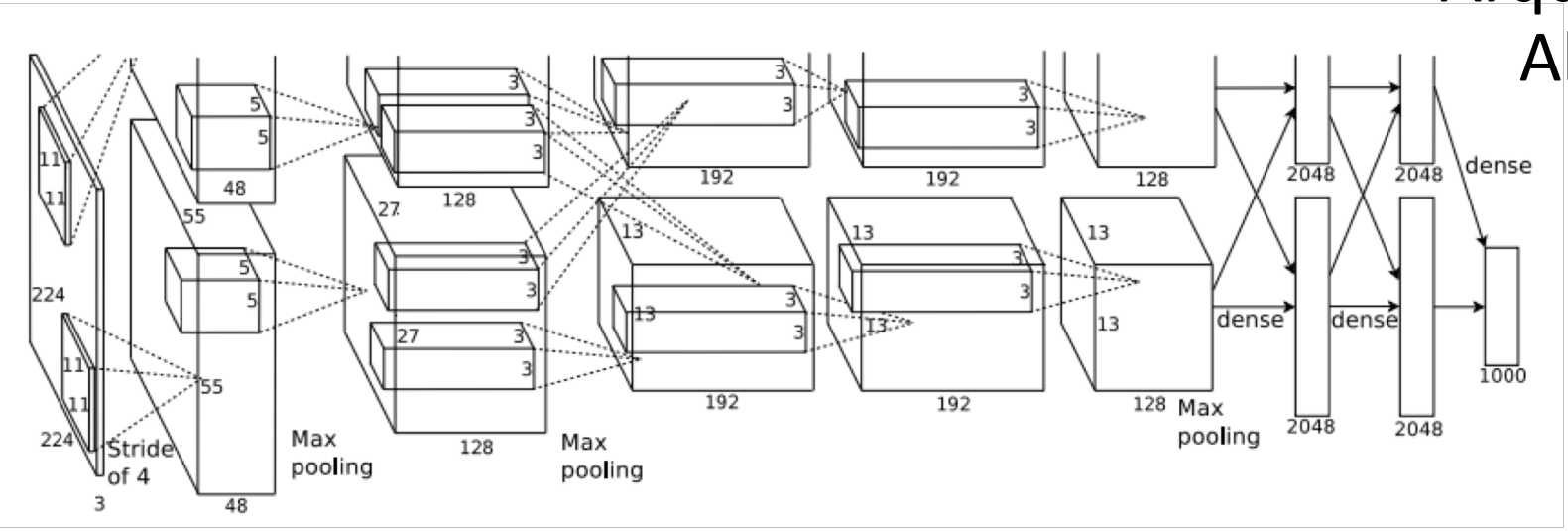
Filtros de Pooling: 2×2
Stride: 2

Arquitetura: [CONV-POOL-CONV-POOL-FC1-FC2-FC3]

Y. LeCun, L. D. Jackel, L. Bottou, A. Brunot, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, U. A. Muller, E. Sackinger, P. Simard, and V. Vapnik. Comparison of learning algorithms for handwritten digit recognition. In F. Fogelman and P. Gallinari, editors, *International Conference on Artificial Neural Networks*, pages 53-60, Paris, 1995.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, november 1998.

Exemplos de Arquiteturas: AlexNet



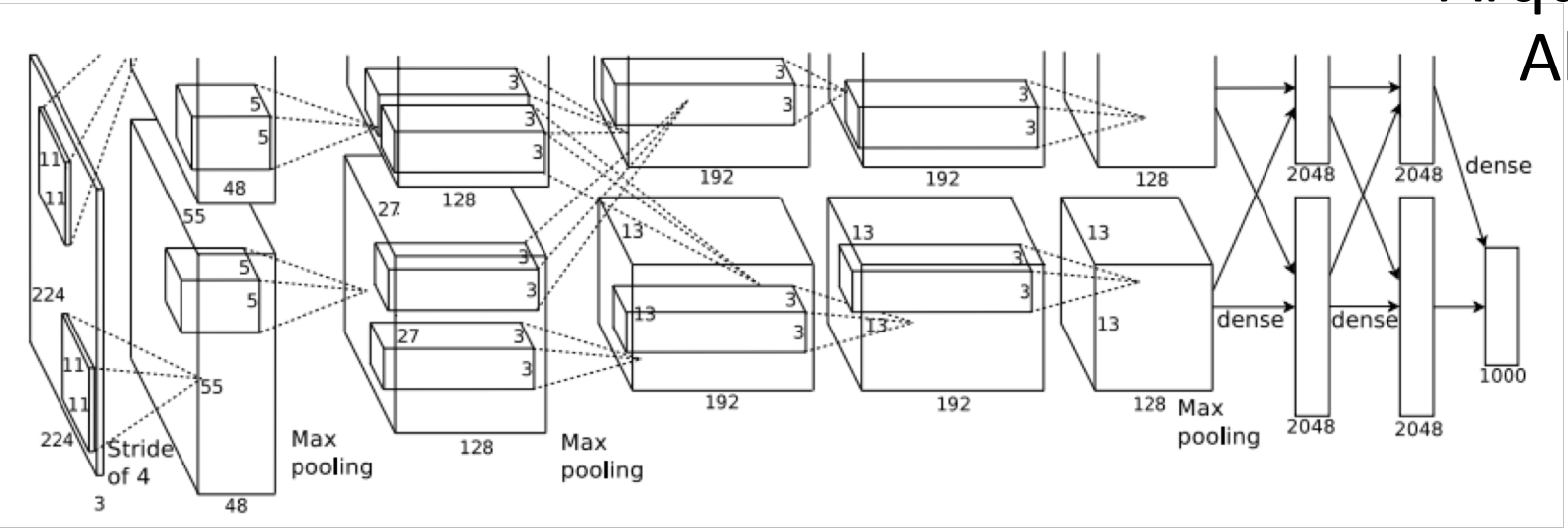
Entrada: imagens 227×227×3

(CONV1): 96 filtros 11×11 com stride de 4

Volume de saída?

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems , pages 1097–1105, 2012.

Exemplos de Arquiteturas: AlexNet



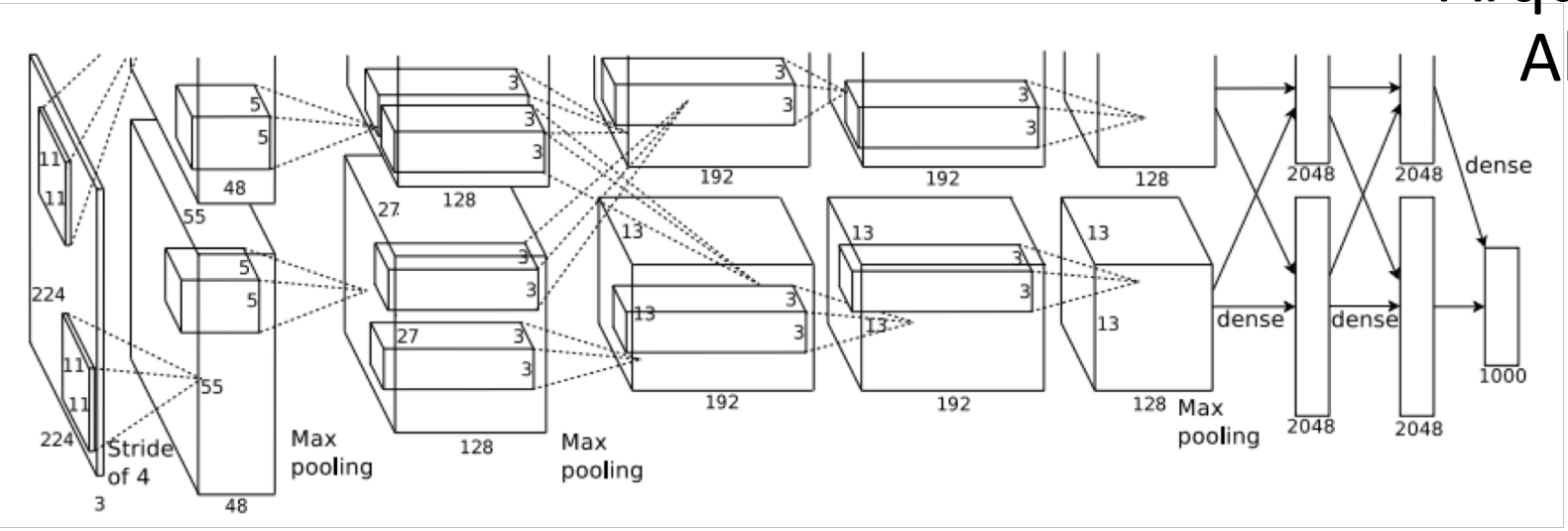
Entrada: imagens 227×227×3

(CONV1): 96 filtros 11×11 com stride de 4

Volume de saída: 55×55×96

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems , pages 1097–1105, 2012.

Exemplos de Arquiteturas: AlexNet



Entrada: imagens 227×227×3

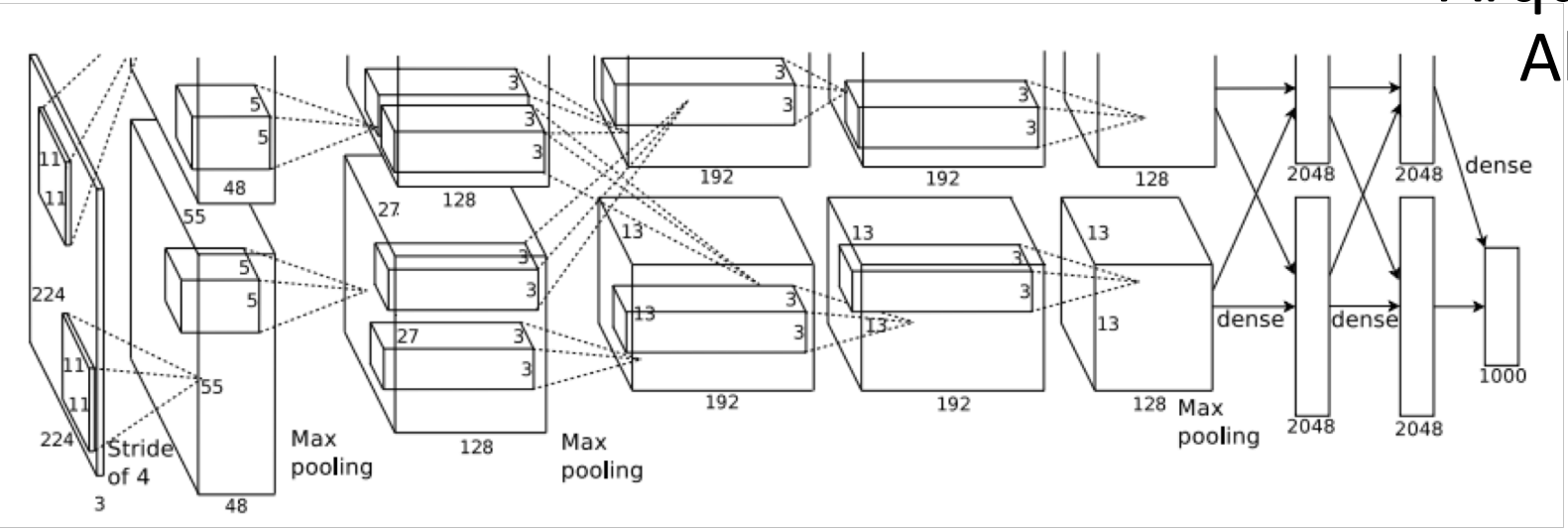
(CONV1): 96 filtros 11×11 com stride de 4

Volume de saída: 55×55×96

Total de parâmetros nesta camada?

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems , pages 1097–1105, 2012.

Exemplos de Arquiteturas: AlexNet



Entrada: imagens 227×227×3

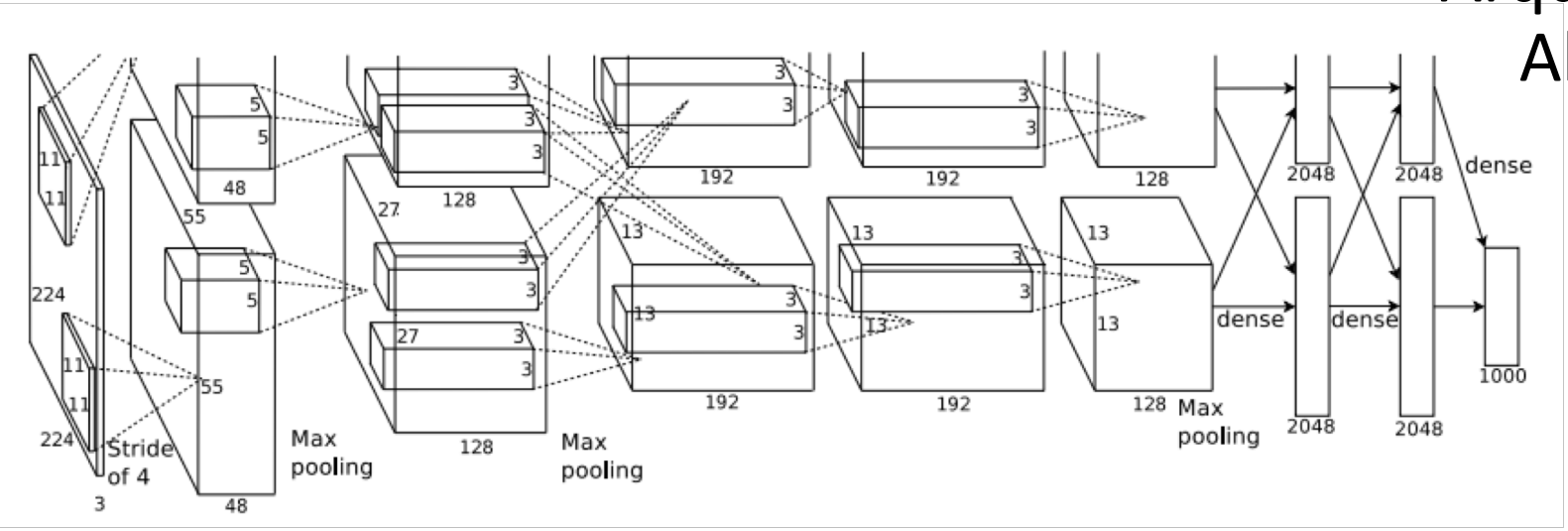
(CONV1): 96 filtros 11×11 com stride de 4

Volume de saída: 55×55×96

Total de parâmetros nesta camada: 34,848 + 96 biases

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems , pages 1097–1105, 2012.

Exemplos de Arquiteturas: AlexNet



Entrada: imagens 227×227×3

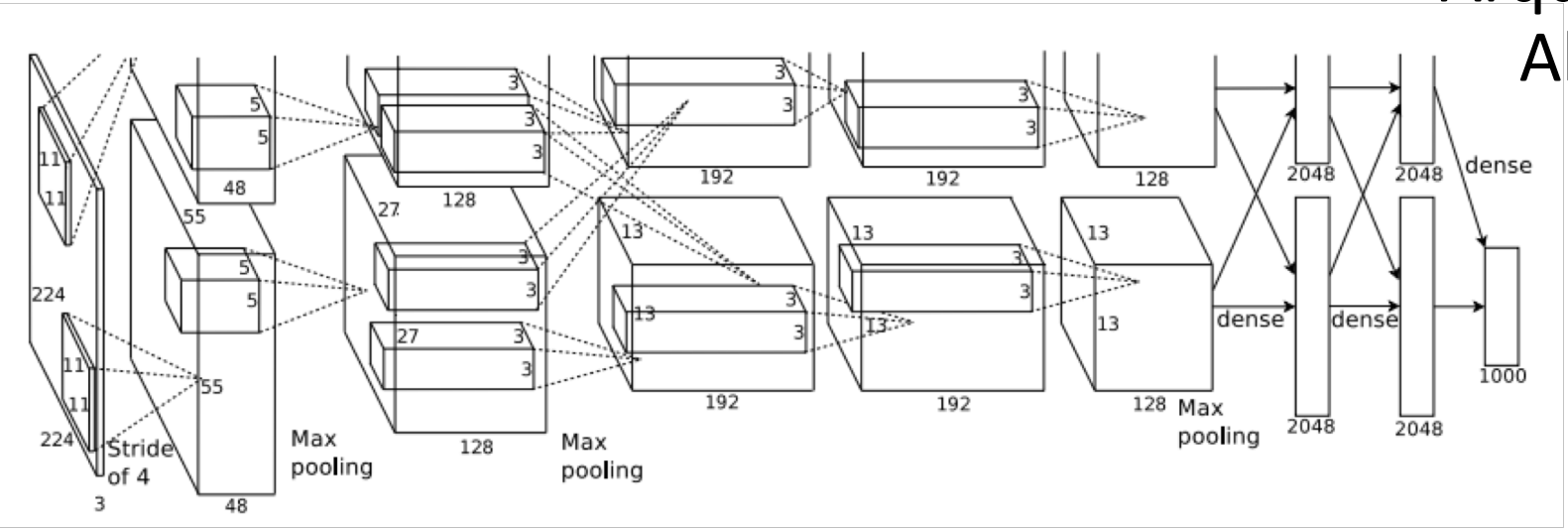
(CONV1): 96 filtros 11×11 com stride de 4

Volume de saída: 55×55×96

(POOL1): filtros 3×3 com stride de 2

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems , pages 1097–1105, 2012.

Exemplos de Arquiteturas: AlexNet



Entrada: imagens 227×227×3

(CONV1): 96 filtros 11×11 com stride de 4

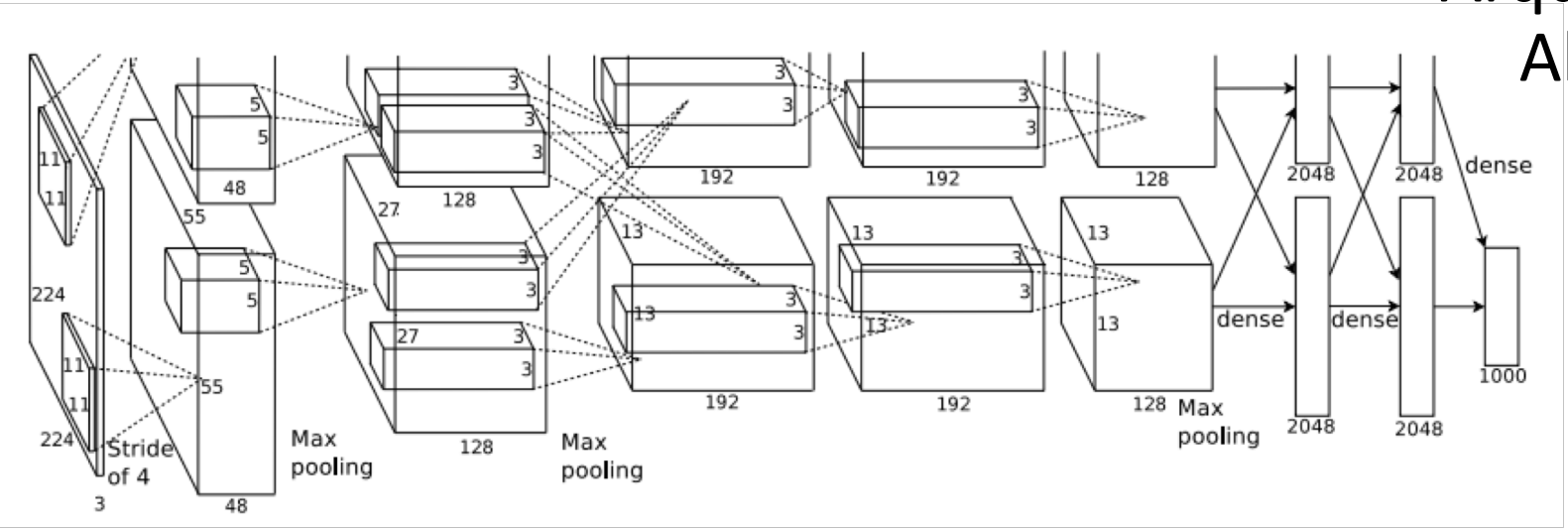
Volume de saída: 55×55×96

(POOL1): filtros 3×3 com stride de 2

Volume de saída?

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems , pages 1097–1105, 2012.

Exemplos de Arquiteturas: AlexNet



Entrada: imagens 227×227×3

(CONV1): 96 filtros 11×11 com stride de 4

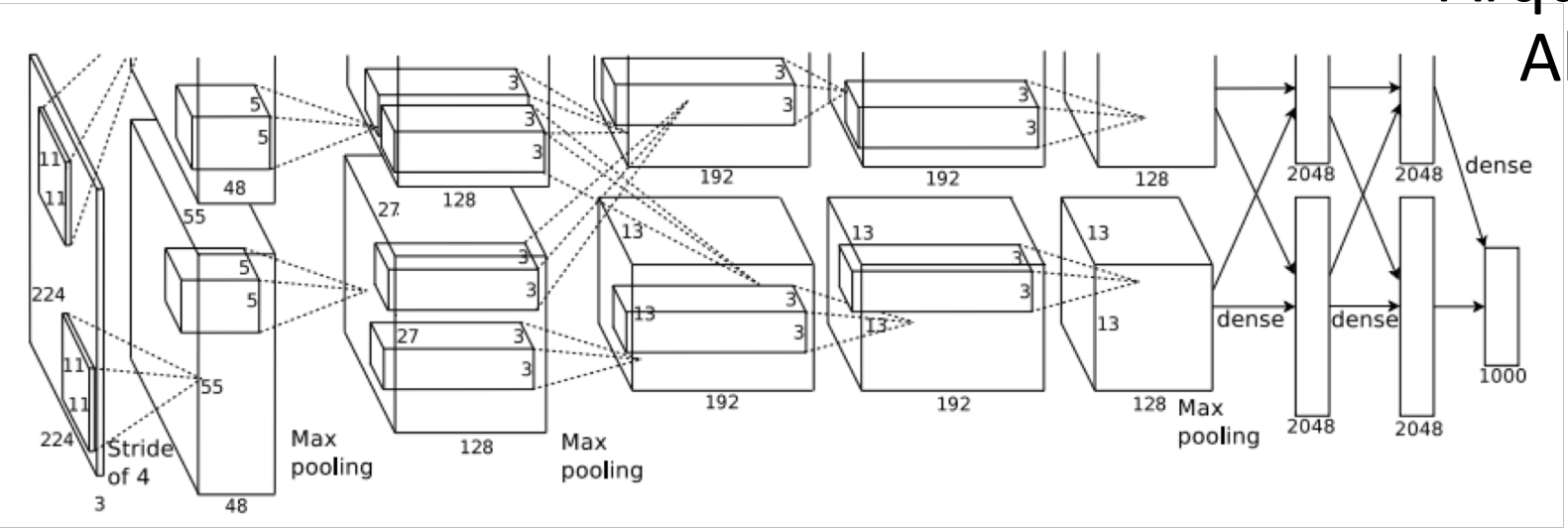
Volume de saída: 55×55×96

(POOL1): filtros 3×3 com stride de 2

Volume de saída: 27×27×96

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems , pages 1097–1105, 2012.

Exemplos de Arquiteturas: AlexNet



Entrada: imagens 227×227×3

(CONV1): 96 filtros 11×11 com stride de 4

Volume de saída: 55×55×96

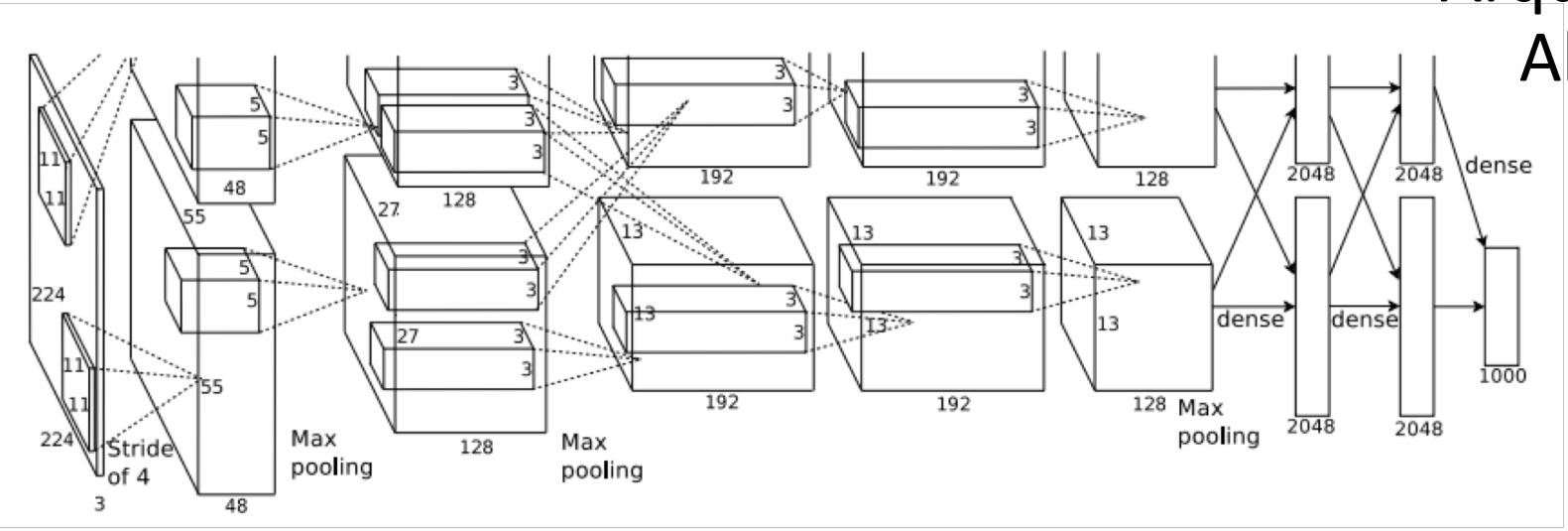
(POOL1): filtros 3×3 com stride de 2

Volume de saída: 27×27×96

Total de parâmetros nesta camada?

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems , pages 1097–1105, 2012.

Exemplos de Arquiteturas: AlexNet



Entrada: imagens 227×227×3

(CONV1): 96 filtros 11×11 com stride de 4

Volume de saída: 55×55×96

(POOL1): filtros 3×3 com stride de 2

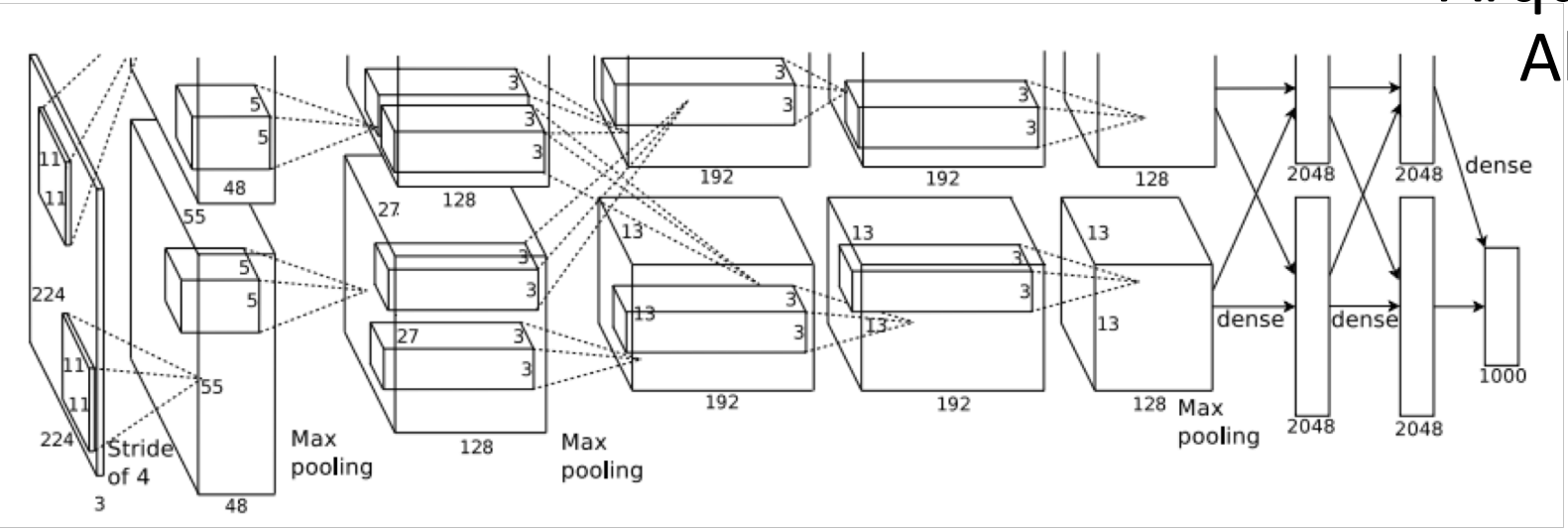
Volume de saída: 27×27×96

Total de parâmetros nesta camada: zero!

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems , pages 1097–1105, 2012.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems , pages 1097–1105, 2012.

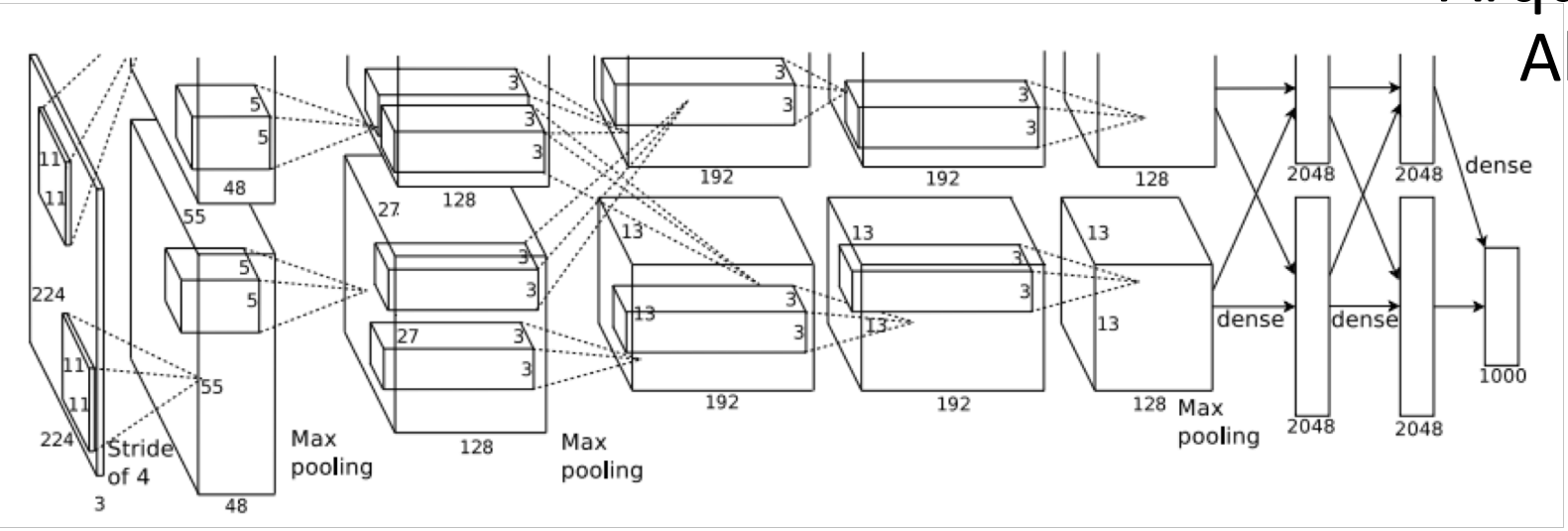
Exemplos de Arquiteturas: AlexNet



Arquitetura completa (versão simplificada):
Entrada: $[227\times227\times3]$
(CONV1) 96 filtros 11×11, stride 4, zero-padding 0: $[55\times55\times96]$
(MAX POOL 1) filtros 3×3, stride 2: $[27\times27\times96]$
(NORM 1) camada de normalização de contraste: $[27\times27\times96]$
(CONV2) 256 filtros 5×5, stride 1, zero-padding 2: $[27\times27\times256]$
(MAX POOL 2) filtros 3×3, stride 2: $[13\times13\times256]$
(NORM 2) camada de normalização de contraste:$[13\times13\times256]$
(CONV3) 384 filtros 3×3, stride 1, zero-padding 1: $[13\times13\times384]$
(CONV4) 384 filtros 3×3, stride 1, zero-padding 1: $[13\times13\times384]$
(CONV5) 256 filtros 3×3, stride 1, zero-padding 1: $[13\times13\times256]$
(MAX POOL 3) filtros 3×3, stride 2: $[6\times6\times256]$
(FC6) conecta os $6\times6\times256 = 9216$ a 4096 neurônios
(FC7) conecta os 4096 neurônios a outros 4096
(FC8) conecta os 4096 neurônios a 1000 neurônios (classes)

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems , pages 1097–1105, 2012.

Exemplos de Arquiteturas: AlexNet



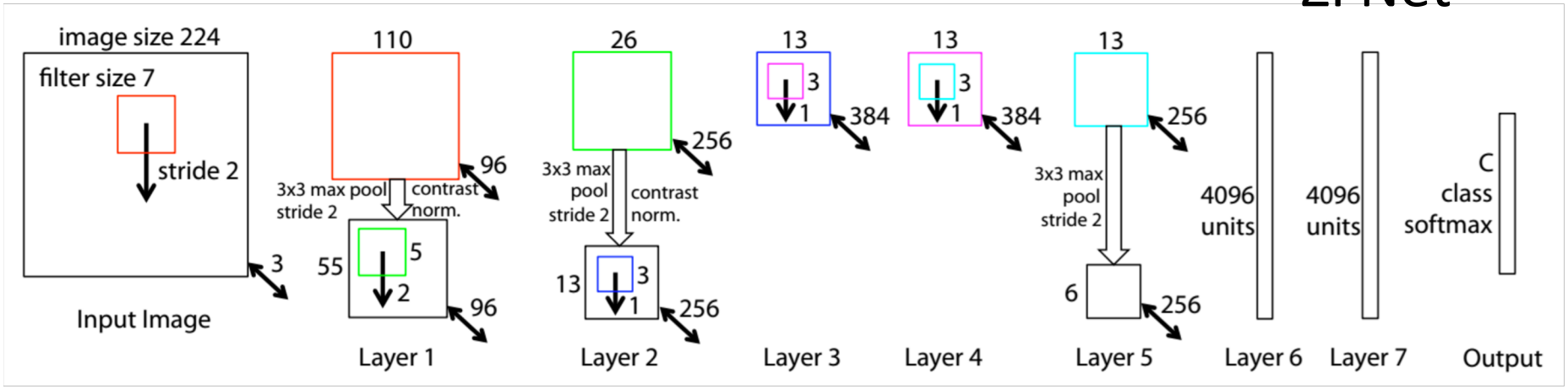
Arquitetura completa (versão simplificada):
Entrada: $[227\times227\times3]$
(CONV1) 96 filtros 11×11, stride 4, zero-padding 0: $[55\times55\times96]$
(MAX POOL 1) filtros 3×3, stride 2: $[27\times27\times96]$
(NORM 1) camada de normalização de contraste: $[27\times27\times96]$
(CONV2) 256 filtros 5×5, stride 1, zero-padding 2: $[27\times27\times256]$
(MAX POOL 2) filtros 3×3, stride 2: $[13\times13\times256]$
(NORM 2) camada de normalização de contraste:$[13\times13\times256]$
(CONV3) 384 filtros 3×3, stride 1, zero-padding 1: $[13\times13\times384]$
(CONV4) 384 filtros 3×3, stride 1, zero-padding 1: $[13\times13\times384]$
(CONV5) 256 filtros 3×3, stride 1, zero-padding 1: $[13\times13\times256]$
(MAX POOL 3) filtros 3×3, stride 2: $[6\times6\times256]$
(FC6) conecta os $6\times6\times256 = 9216$ a 4096 neurônios
(FC7) conecta os 4096 neurônios a outros 4096
(FC8) conecta os 4096 neurônios a 1000 neurônios (classes)

Detalhes da AlexNet:

- primeira vez do ReLU
- utilizava camadas NORM (não são mais usadas)
- data augmentation (detalhes na disciplina RNP)
- dropout: 0.5
- tamanho do batch: 128
- SGD + Momentum 0.9
- Taxa de aprendizado: 10^{-2}, reduzida por 10 manualmente quando acurácia de val estabiliza
- Regularização L2: 5×10^{-4}
- Ensemble com 7 AlexNets: 18.2% → 15.4%

Matthew D. Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks. In ECCV, 2014.

Exemplos de Arquiteturas: ZFNet



Muito similar à AlexNet, com as seguintes mudanças:

- CONV1: alteração de (11×11 stride 4) para (7×7 stride 2)
- CONV3,4,5: alteração de (384, 384 e 256 filtros) para (512, 1024, 512) filtros

Erro top-5 ImageNet: 15.4% → 14.8%

Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In ICLR, 2015.

Exemplos de Arquiteturas: VGGNet

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Apenas convoluções 3×3 (stride 1, pad 1) e *max pooling* 2×2 (stride 2)

melhor modelo

Erro top-5 ImageNet:
7.0% (uma única rede)
6.8% (ensemble com 2 redes)

Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In ICLR, 2015.

Exemplos de Arquiteturas: VGGNet

	C	D
ght 's	16 weight layers	16 weight layers
× 224 RGB image		
-64 **-64**	conv3-64 conv3-64	conv3-64 conv3-64
maxpool		
128 **128**	conv3-128 conv3-128	conv3-128 conv3-128
maxpool		
256 256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**
maxpool		
512 512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**
maxpool		
512 512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**
maxpool		
FC-4096		
FC-4096		
FC-1000		
soft-max		

Cálculo do # de Parâmetros: (sem contar os biases)

Entrada: [224×224×3] **#params:** 0

(CONV1: 64) [224×224×64] **#params:** (3×3×3)×64 = 1,728

(CONV2: 64) [224×224×64] **#params:** (3×3×64)×64 = 36,864

(POOL 1) [112×112×64] **#params:** 0

(CONV3: 128) [112×112×128] **#params:** (3×3×64)×128 = 73,728

(CONV4: 128) [112×112×128] **#params:** (3×3×128)×128 = 147,456

(POOL 2) [56×56×128] **#params:** 0

(CONV5: 256) [56×56×256] **#params:** (3×3×128)×256 = 294,912

(CONV6: 256) [56×56×256] **#params:** (3×3×256)×256 = 589,824

(CONV7: 256) [56×56×256] **#params:** (3×3×256)×256 = 589,824

(POOL 3) [28×28×256] **#params:** 0

(CONV8: 512) [28×28×512] **#params:** (3×3×256)×512 = 1,179,648

(CONV9: 512) [28×28×512] **#params:** (3×3×512)×512 = 2,359,296

(CONV10: 512) [28×28×512] **#params:** (3×3×512)×512 = 2,359,296

(POOL 4) [14×14×512] **#params:** 0

(CONV11: 512) [14×14×512] **#params:** (3×3×512)×512 = 2,359,296

(CONV12: 512) [14×14×512] **#params:** (3×3×512)×512 = 2,359,296

(CONV13: 512) [14×14×512] **#params:** (3×3×512)×512 = 2,359,296

(POOL 5) [7×7×512] **#params:** 0

(FC1) [1×1×4096] **#params:** (7×7×512)×4096 = 102,760,448

(FC2) [1×1×4096] **#params:** 4096×4096 = 16,777,216

(FC3) [1×1×1000] **#params:** 4096×1000 = 4,096,000

 # total de parâmetros: 138,344,128 ×4 bytes ≈ 550MB

Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In ICLR, 2015.

Exemplos de Arquiteturas: VGGNet

	C	D
ght 's	16 weight layers	16 weight layers
× 224 RGB image		
-64 **-64**	conv3-64 conv3-64	conv3-64 conv3-64
maxpool		
128 **128**	conv3-128 conv3-128	conv3-128 conv3-128
maxpool		
256 256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**
maxpool		
512 512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**
maxpool		
512 512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**
maxpool		
FC-4096		
FC-4096		
FC-1000		
soft-max		

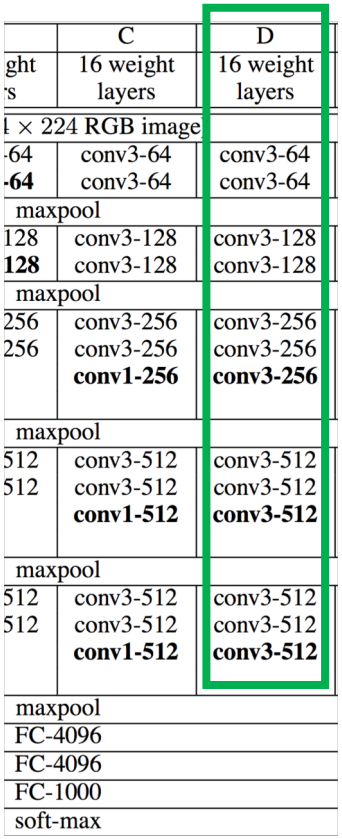
Cálculo de memória dos dados:

Entrada: [224×224×3] $224\times224\times3 \approx 150k$
(CONV1: 64) [224×224×64] $224\times224\times64 \approx 3.2M$
(CONV2: 64) [224×224×64] $224\times224\times64 \approx 3.2M$
(POOL 1) [112×112×64] $112\times112\times64 \approx 800k$
(CONV3: 128) [112×112×128] $112\times112\times128 \approx 1.6M$
(CONV4: 128) [112×112×128] $112\times112\times128 \approx 1.6M$
(POOL 2) [56×56×128] $56\times56\times128 \approx 400k$
(CONV5: 256) [56×56×256] $56\times56\times256 \approx 800k$
(CONV6: 256) [56×56×256] $56\times56\times256 \approx 800k$
(CONV7: 256) [56×56×256] $56\times56\times256 \approx 800k$
(POOL 3) [28×28×256] $28\times28\times256 \approx 200k$
(CONV8: 512) [28×28×512] $28\times28\times512 \approx 400k$
(CONV9: 512) [28×28×512] $28\times28\times512 \approx 400k$
(CONV10: 512) [28×28×512] $28\times28\times512 \approx 400k$
(POOL 4) [14×14×512] $14\times14\times512 \approx 100k$
(CONV11: 512) [14×14×512] $14\times14\times512 \approx 100k$
(CONV12: 512) [14×14×512] $14\times14\times512 \approx 100k$
(CONV13: 512) [14×14×512] $14\times14\times512 \approx 100k$
(POOL 5) [7×7×512] $7\times7\times512 \approx 25k$
(FC1) [1×1×4096] 4096
(FC2) [1×1×4096] 4096
(FC3) [1×1×1000] 1000

total de memória dos dados: $\approx 15M \times 4$ bytes $\approx$ 60MB/imagem (apenas forward! backward $\approx 2\times$)

Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In ICLR, 2015.

Exemplos de Arquiteturas: VGGNet



	C	D
ght 's	16 weight layers	16 weight layers
l × 224 RGB image		
-64 **-64**	conv3-64 conv3-64	conv3-64 conv3-64
maxpool		
128 **128**	conv3-128 conv3-128	conv3-128 conv3-128
maxpool		
256 256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**
maxpool		
512 512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**
maxpool		
512 512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**
maxpool		
FC-4096		
FC-4096		
FC-1000		
soft-max		

Entrada: $[224 \times 224 \times 3]$
(CONV1: 64) $[224 \times 224 \times 64]$
(CONV2: 64) $[224 \times 224 \times 64]$
(POOL 1) $[112 \times 112 \times 64]$
(CONV3: 128) $[112 \times 112 \times 128]$
(CONV4: 128) $[112 \times 112 \times 128]$
(POOL 2) $[56 \times 56 \times 128]$
(CONV5: 256) $[56 \times 56 \times 256]$
(CONV6: 256) $[56 \times 56 \times 256]$
(CONV7: 256) $[56 \times 56 \times 256]$
(POOL 3) $[28 \times 28 \times 256]$
(CONV8: 512) $[28 \times 28 \times 512]$
(CONV9: 512) $[28 \times 28 \times 512]$
(CONV10: 512) $[28 \times 28 \times 512]$
(POOL 4) $[14 \times 14 \times 512]$
(CONV11: 512) $[14 \times 14 \times 512]$
(CONV12: 512) $[14 \times 14 \times 512]$
(CONV13: 512) $[14 \times 14 \times 512]$
(POOL 5) $[7 \times 7 \times 512]$
(FC1) $[1 \times 1 \times 4096]$
(FC2) $[1 \times 1 \times 4096]$
(FC3) $[1 \times 1 \times 1000]$

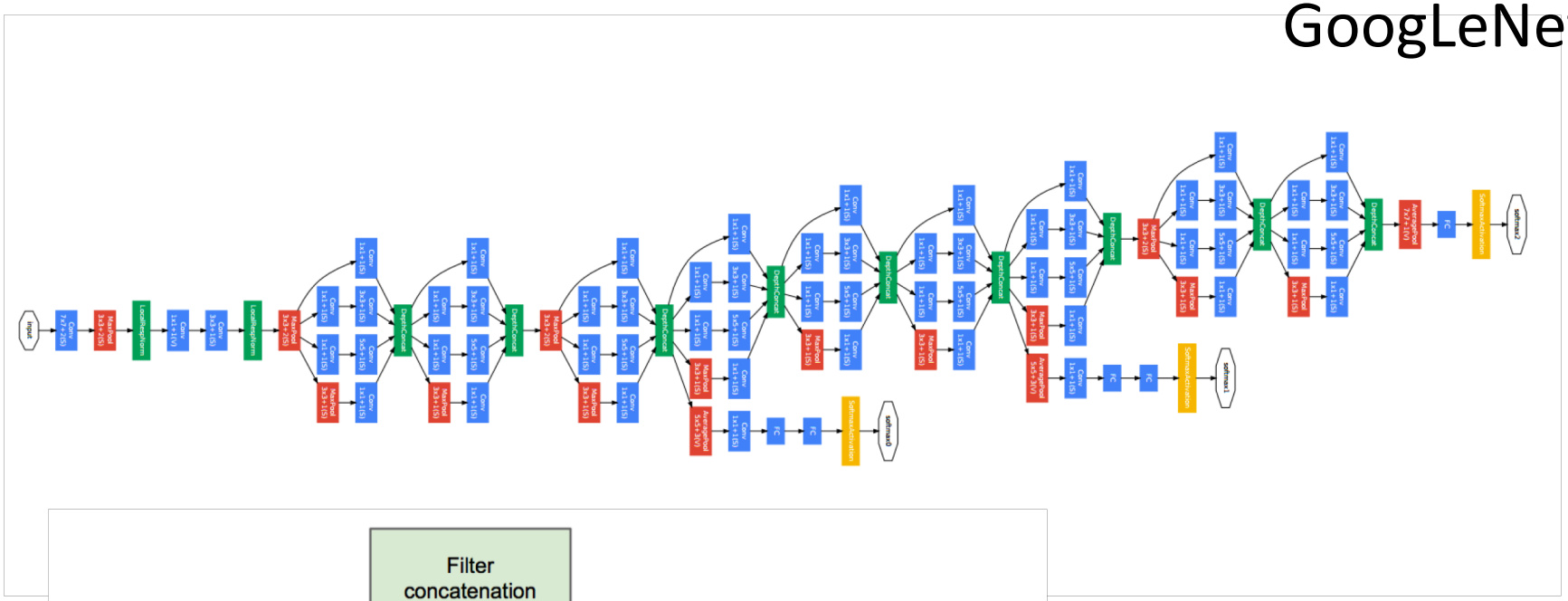
≈ 550MB de tamanho do modelo (parâmetros)

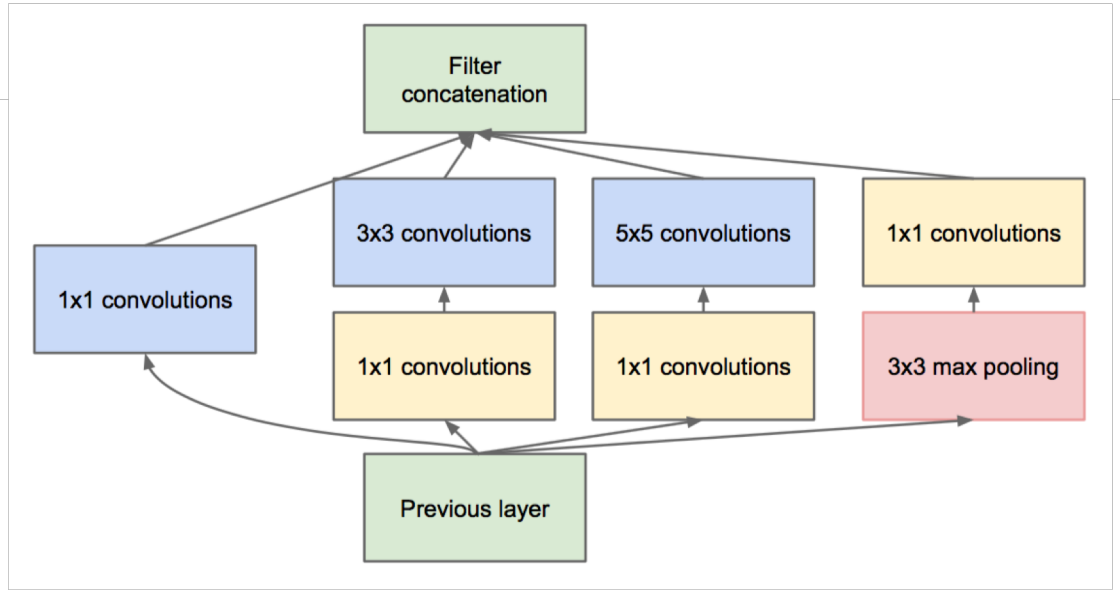
≈ 60MB por imagem (dados)

Considerando um batch de 128 imagens:
$550 + (60 \times 128) \approx 8.2$GB de espaço em GPU
(apenas *forward*)

Szegedy, Christian, et al. **Going deeper with convolutions**. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015

Exemplos de Arquiteturas: GoogLeNet





Módulo de Inception

Szegedy, Christian, et al. **Going deeper with convolutions**. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015

Exemplos de Arquiteturas: GoogLeNet

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Apenas $\approx 5M$ de parâmetros!!

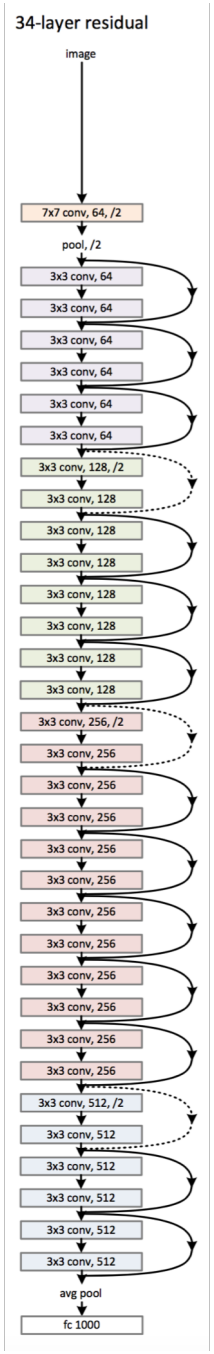
12× menos parâmetros que AlexNet
27× menos parâmetros que VGG

Apenas 1 FC!! (ligando features com classes)

Erro top-5 ImageNet:
7.9% (uma única rede)
6% (ensemble com 7 redes)

Average Pooling em vez de esticar *features* pós convoluções

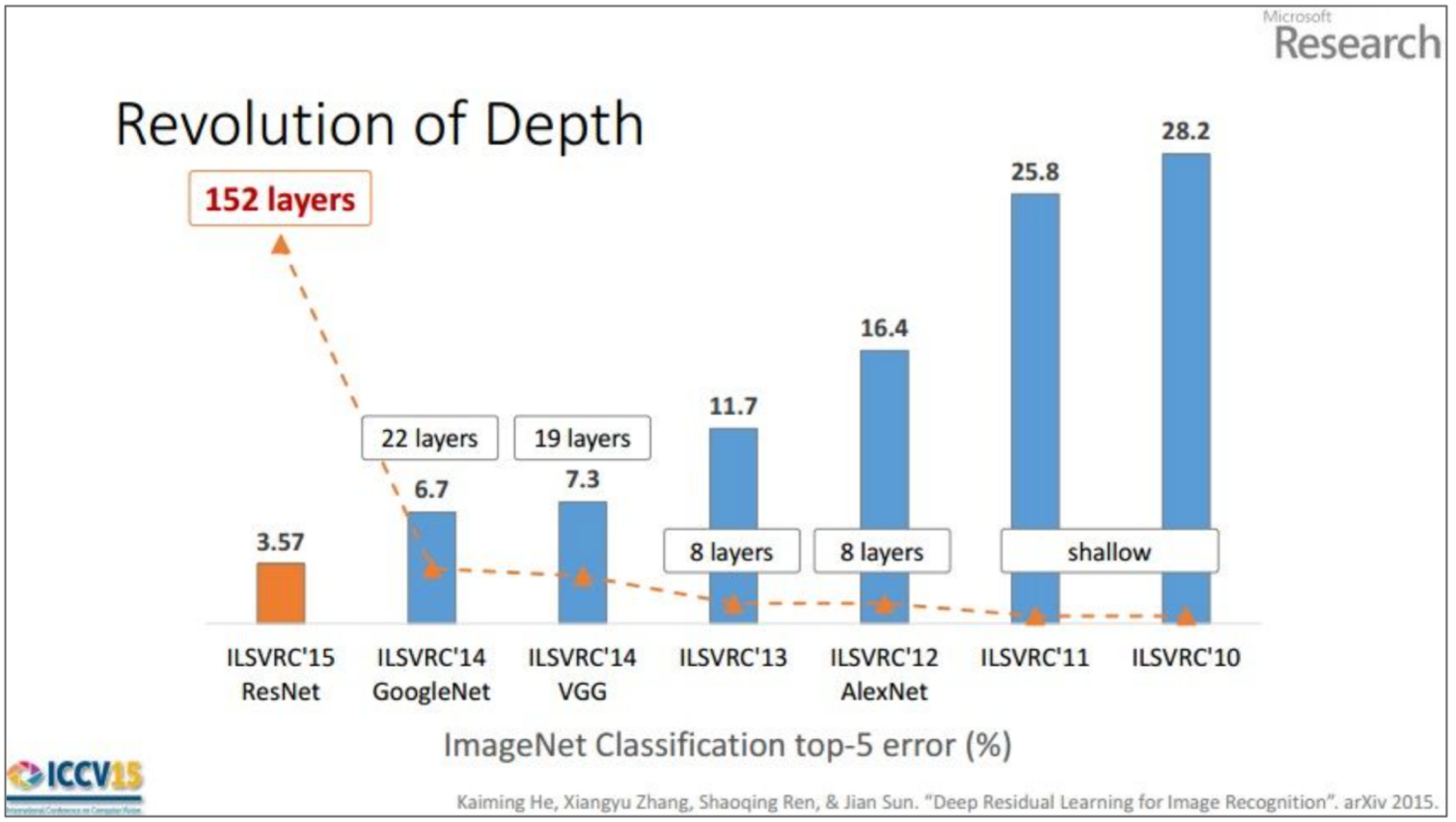
He, Kaiming, et al. **Deep Residual Learning for Image Recognition**. arXiv preprint arXiv:1512.03385, 2015.



Exemplos de Arquiteturas: ResNet

He, Kaiming, et al. **Deep Residual Learning for Image Recognition**. arXiv preprint arXiv:1512.03385, 2015.

Exemplos de Arquiteturas: ResNet

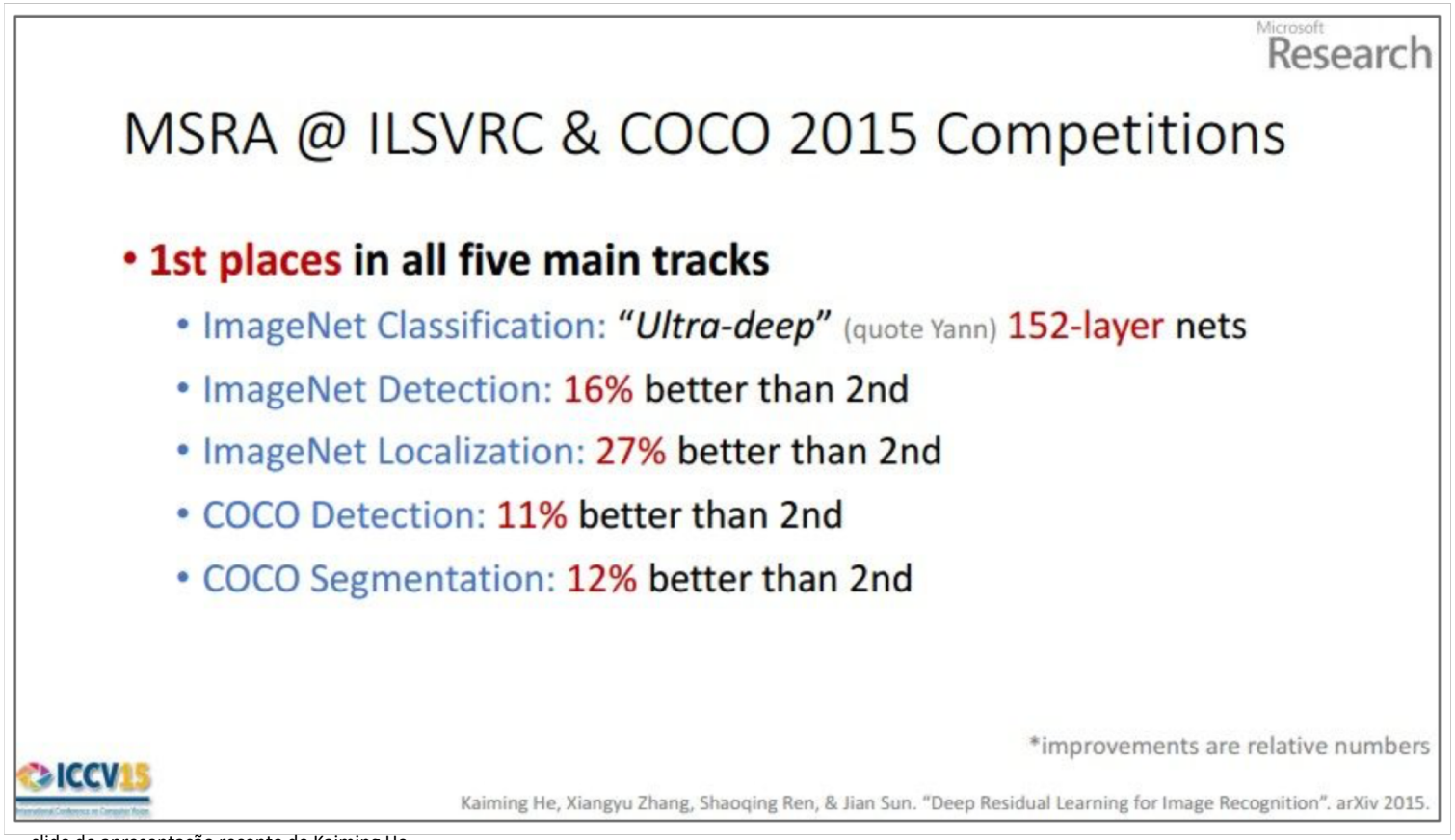


slide de apresentação recente de Kaiming He

He, Kaiming, et al. **Deep Residual Learning for Image Recognition**. arXiv preprint arXiv:1512.03385, 2015.

Exemplos de Arquiteturas: ResNet

Erro top-5 ImageNet:
4.5% (uma única rede)
3.6% (ensemble com 6 redes)



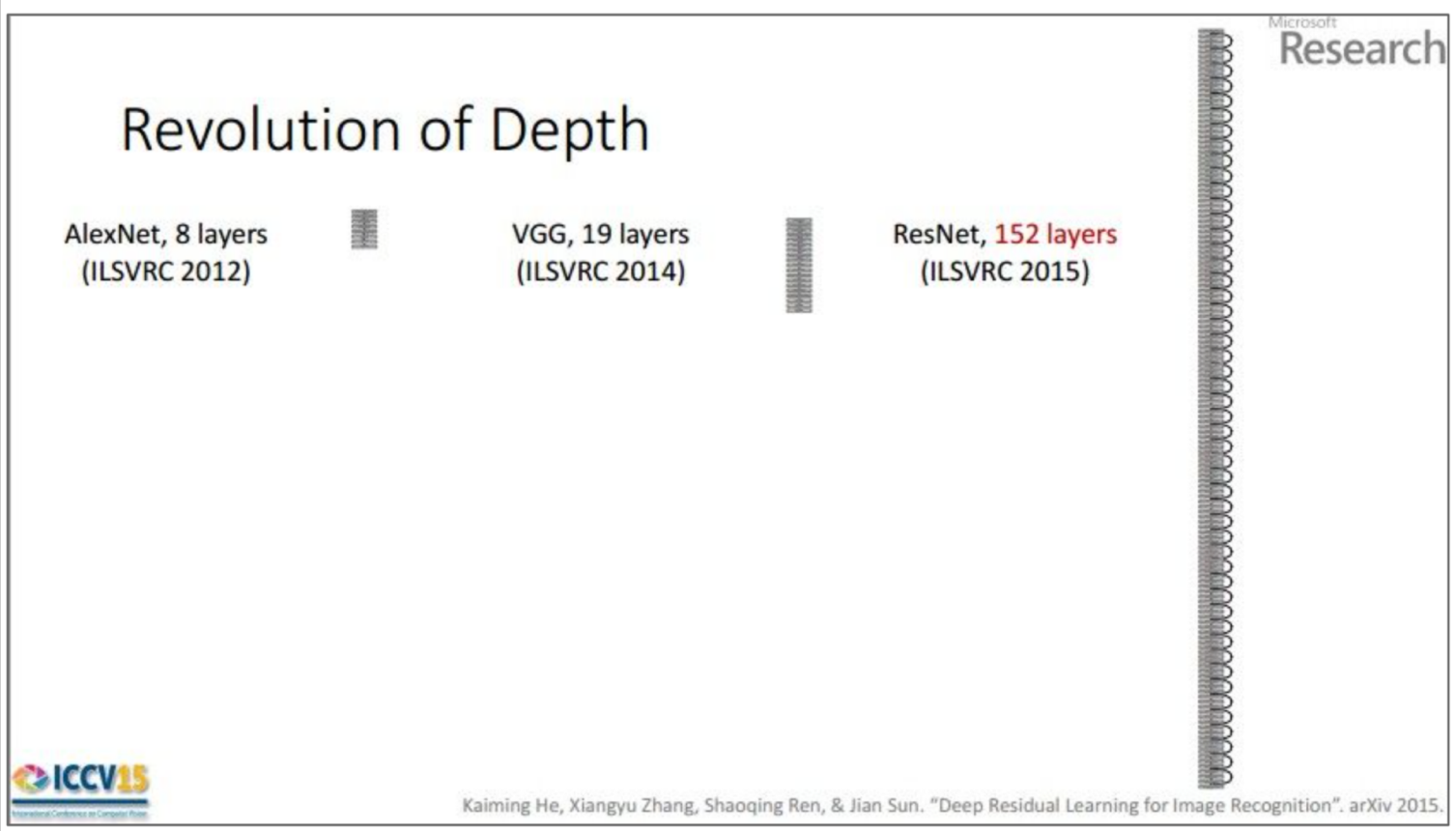
slide de apresentação recente de Kaiming He

He, Kaiming, et al. **Deep Residual Learning for Image Recognition**. arXiv preprint arXiv:1512.03385, 2015.

Exemplos de Arquiteturas: ResNet

2-3 semanas para treinar em máquina com 8 GPUs!!!

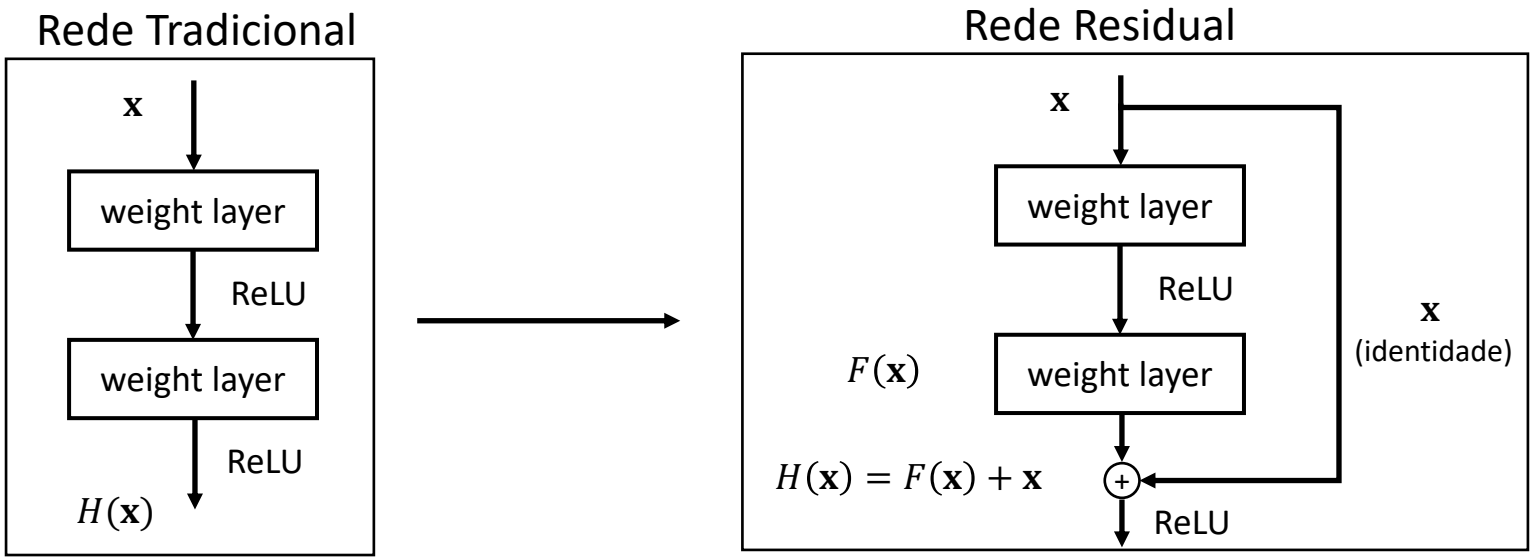
Para classificar é mais rápida que VGG!! (mesmo com 8× mais camadas)



slide de apresentação recente de Kaiming He

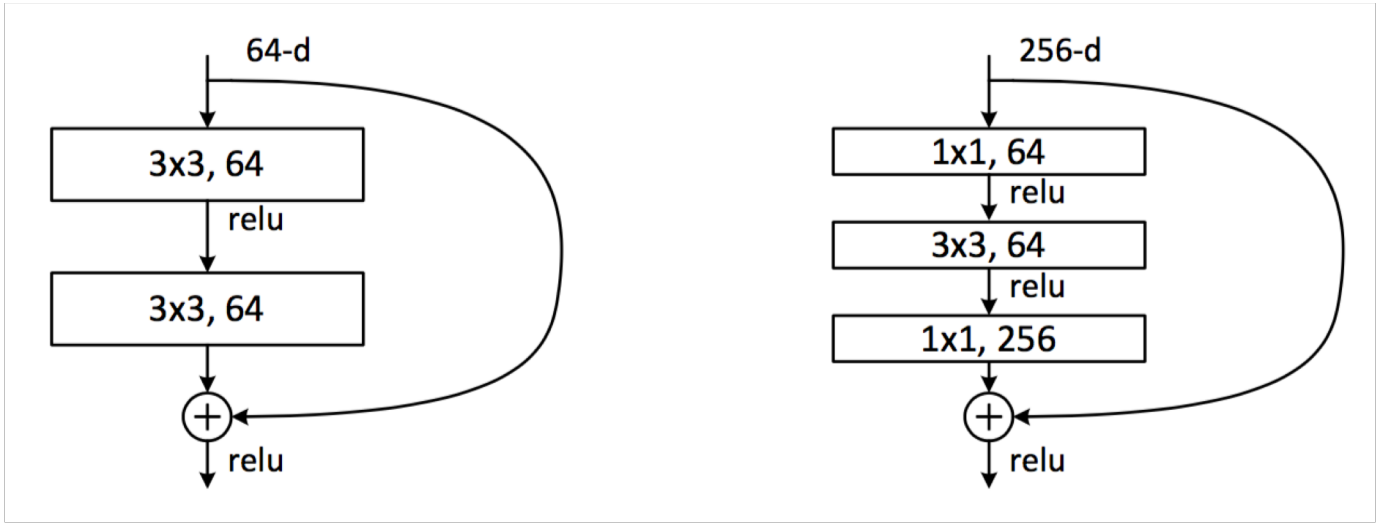
He, Kaiming, et al. **Deep Residual Learning for Image Recognition**. arXiv preprint arXiv:1512.03385, 2015.

Exemplos de Arquiteturas: ResNet



He, Kaiming, et al. **Deep Residual Learning for Image Recognition**. arXiv preprint arXiv:1512.03385, 2015.

Exemplos de Arquiteturas: ResNet

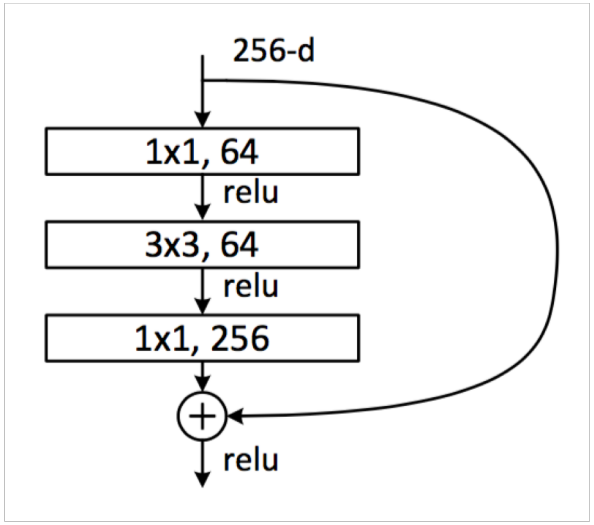


Toda com filtros 3×3

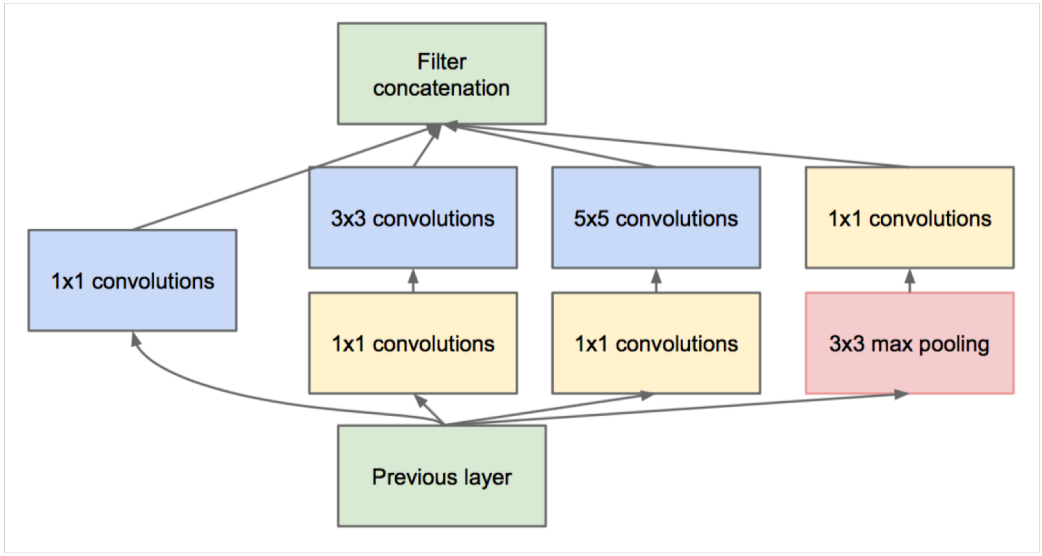
Versão *bottleneck*
(apenas ResNet-50/101/152)

He, Kaiming, et al. **Deep Residual Learning for Image Recognition**. arXiv preprint arXiv:1512.03385, 2015.

Exemplos de Arquiteturas: ResNet



Versão *bottleneck*
(apenas ResNet-50/101/152)



Mesmo truque utilizado para reduzir complexidade na GoogLeNet

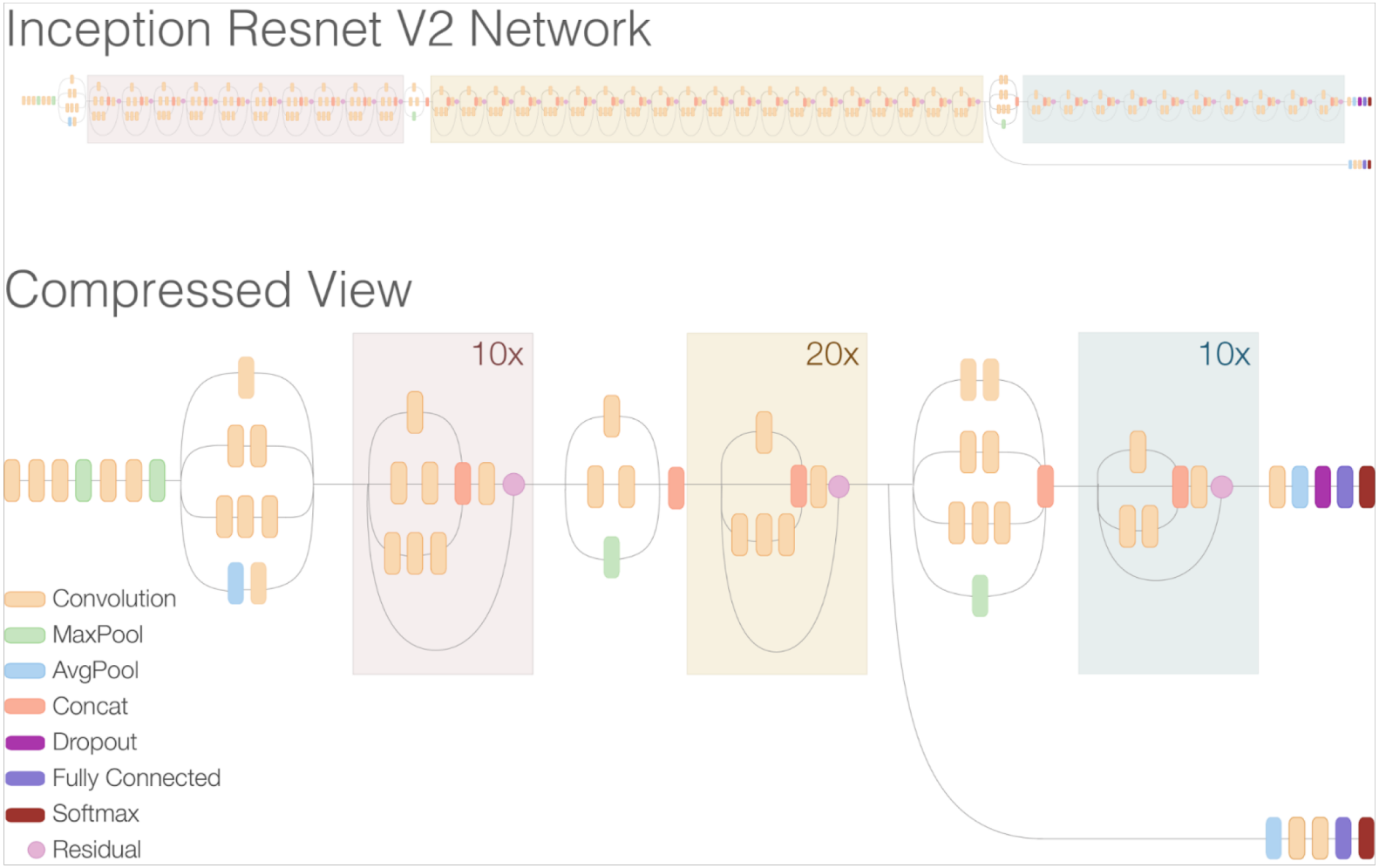
He, Kaiming, et al. **Deep Residual Learning for Image Recognition**. arXiv preprint arXiv:1512.03385, 2015.

Exemplos de Arquiteturas: ResNet

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

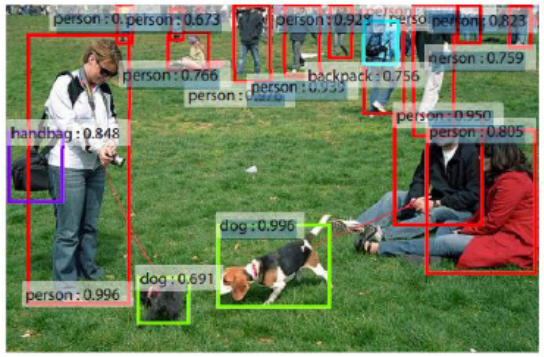
Exemplos de Arquiteturas: Inception-ResNet-v2

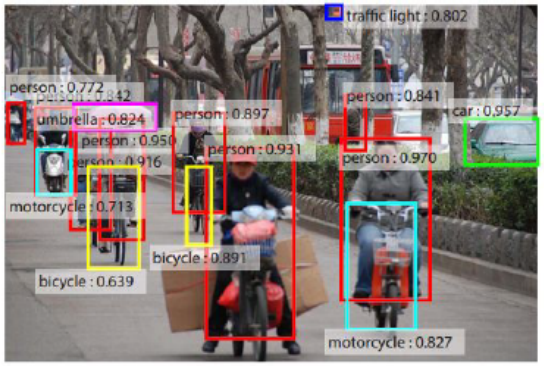
Erro top-5 ImageNet:
3.7% (uma única rede)
3.1% (ensemble com 4 redes)



Aplicações

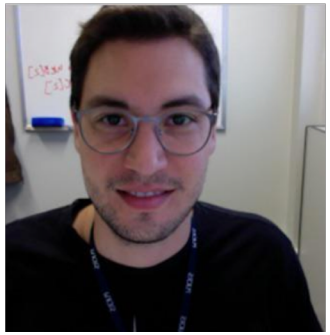
Localização e Detecção de Objetos



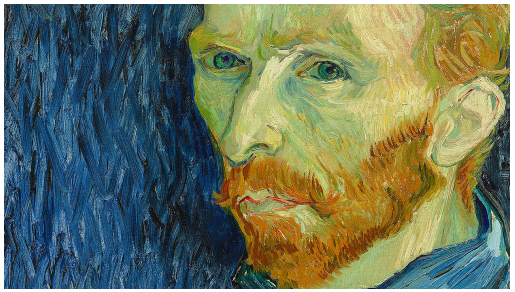


Aplicações

Transferência de Estilo Visual



+



=

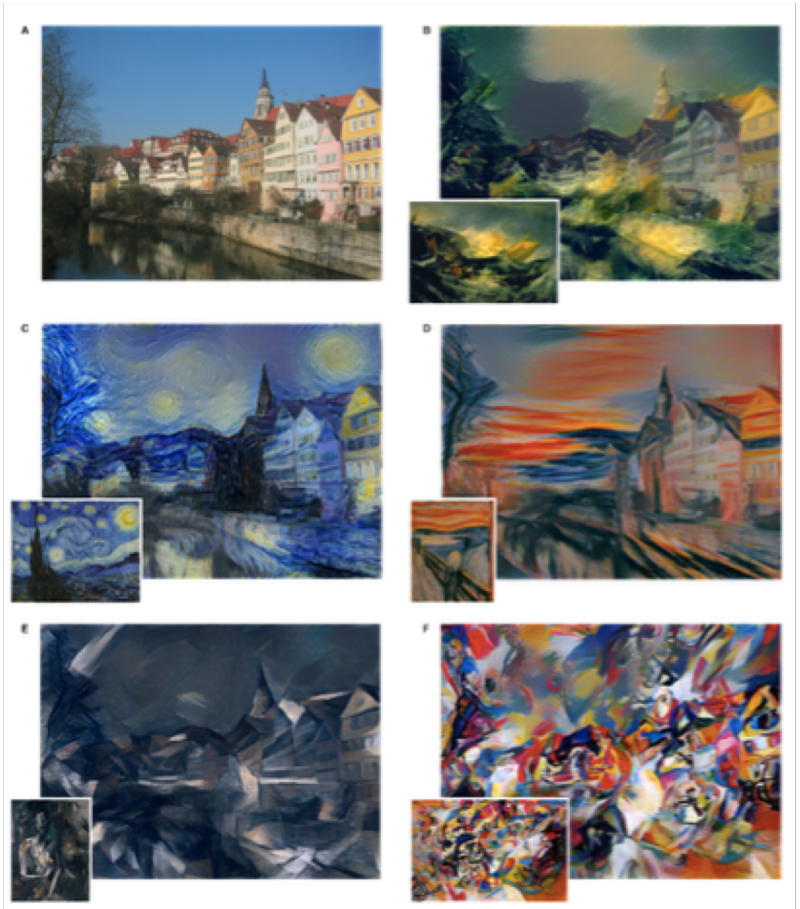


Figure 2: Images that combine the content of a photograph with the style of several well-known artworks. The images were created by finding an image that simultaneously matches the content representation of the photograph and the style representation of the artwork (see Methods). The original photograph depicting the Neckarfront in Tübingen, Germany, is shown in **A** (Photo: Andreas Praefcke). The painting that provided the style for the respective generated image is shown in the bottom left corner of each panel. **B** *The Shipwreck of the Minotaur* by J.M.W. Turner, 1805. **C** *The Starry Night* by Vincent van Gogh, 1889. **D** *Der Schrei* by Edvard Munch, 1893. **E** *Femme nue assise* by Pablo Picasso, 1910. **F** *Composition VII* by Wassily Kandinsky, 1913.

Leon A. Gatys, Alexander S. Ecker, Matthias Bethge. **A Neural Algorithm of Artistic Style.** CoRR abs/1508.06576 (2015)

Aplicações

Transferência de Estilo Visual

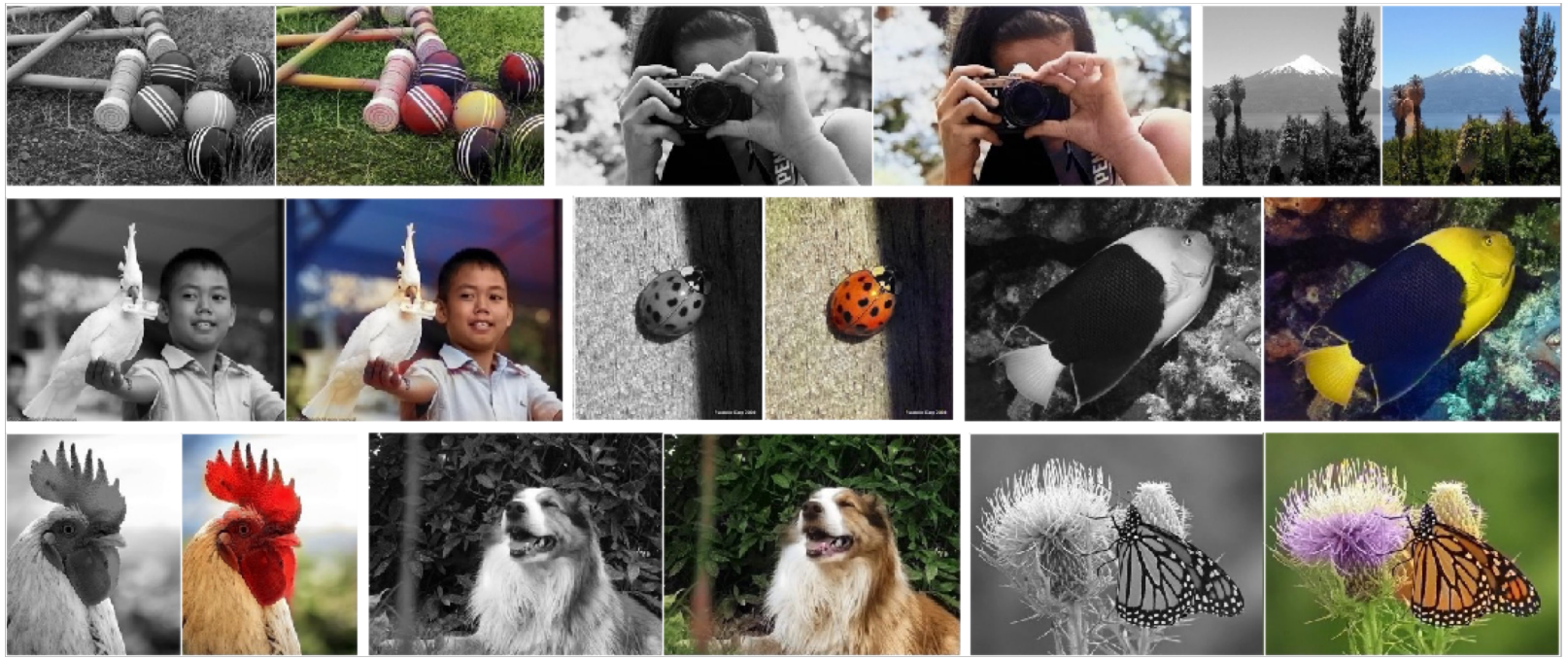
+

=

Leon A. Gatys, Alexander S. Ecker, Matthias Bethge. **A Neural Algorithm of Artistic Style**. CoRR abs/1508.06576 (2015)

Aplicações

Colorização de Imagens



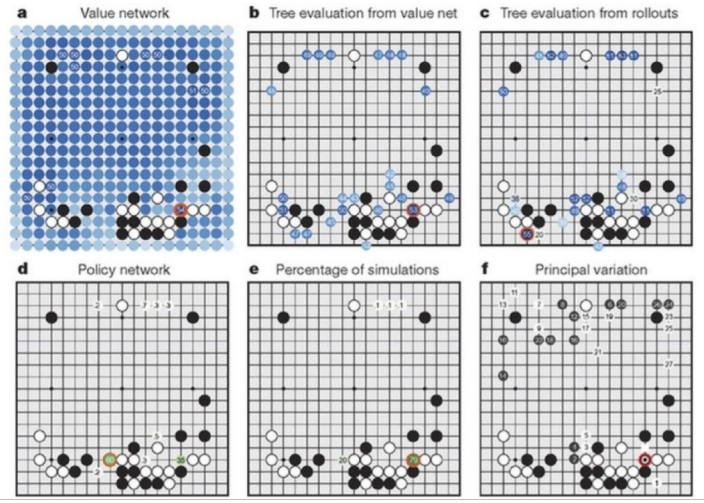
Zhang, Isola, Efros. Colorful Image Colorization. In ECCV, 2016.

Aplicações

AlphaGo

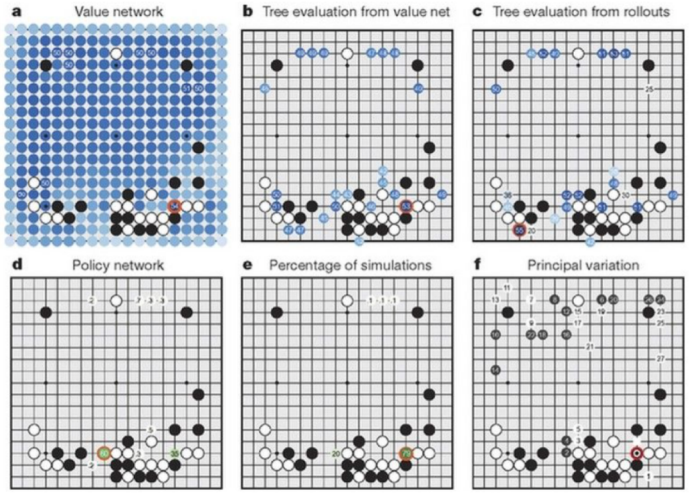
IA para Jogos

Silver, David, et al. **Mastering the game of Go with deep neural networks and tree search.** *Nature* 529.7587 (2016): 484-489.



Aplicações

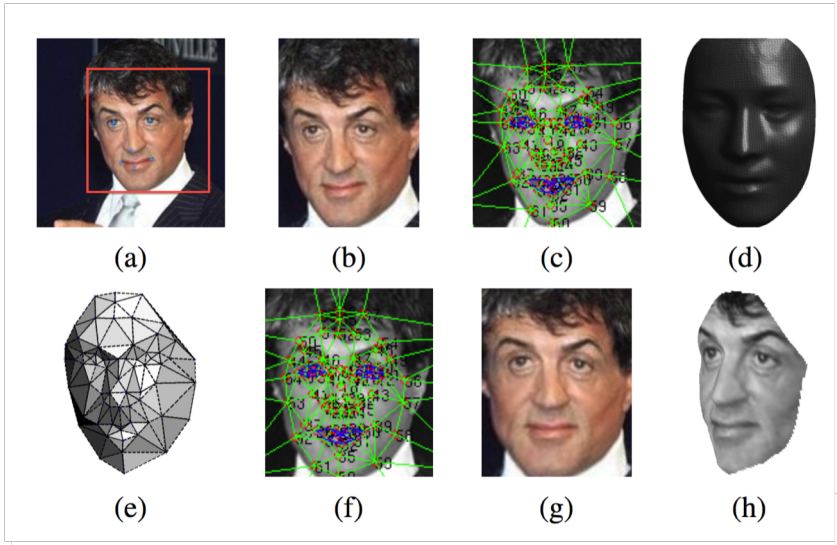
AlphaGo



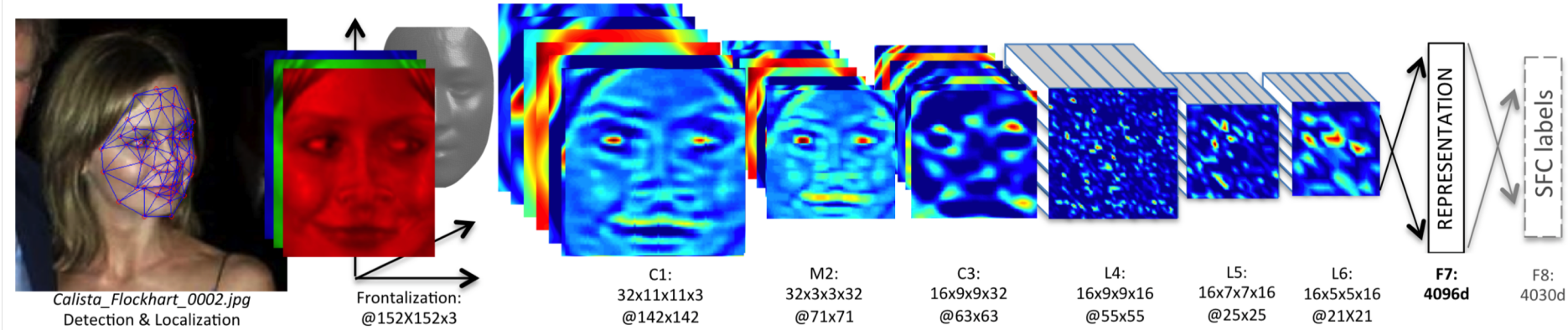
”Policy” Network:

Entrada: $19 \times 19 \times 48$
CONV1: 192 filtros 5×5, stride 1, zero-padding 2 → $19 \times 19 \times 192$
CONV2..12: 192 filtros 3×3, stride 1, zero-padding 1 → $19 \times 19 \times 192$
CONV3: 1 filtro 1×1, stride 1, zero-padding 0 → 19×19 (mapa de probabilidades de movimentações promissoras)

Aplicações



Reconhecimento Facial



Taigman et al. (2014) – DeepFace: Closing the Gap to Human-Level Performance in Face Verification

Resumo da Aula de Hoje

- Redes Convolucionais são pilhas de camadas CONV, POOL e FC
- Há tendência em utilizar filtros pequenos (3×3) e arquiteturas profundas
- Há tendência em reduzir/eliminar camadas POOL e FC
- Arquitetura típica é dada pela seguinte expressão:
  - $[(CONV - RELU)\times N - POOL?]\times M - (FC - RELU)\times K, SOFTMAX$
  - onde N geralmente vai até 5, M é GRANDE e $0 \leq K \leq 2$
  - Note que GoogLeNet e ResNet já mudam bastante este paradigma!