CURSO BÁSICO DE INTELIGÊNCIA ARTIFICIAL E BATE-PAPO COM CONVIDADOS ESPECIAIS

INTELIGÊNCIA ARTIFICIAL PARA TODOS

DE 08/06 A 12/08

COM OS PROFESSORES DO LABORATÓRIO ARIA/UFPB: TELMO FILHO, THAÍS GAUDENCIO E YURI MALHEIROS

Centro de Informática UFPB

Departamento de ESTATÍSTICA

aria artificial intelligence applications

- CURSO SEM PRÉ-REQUISITOS
- HTTP://ARIA.CI.UFPB.BR/IAPARATODOS
- INSCRIÇÃO PARA CERTIFICADO - DE 01/06 A 07/06: HTTP://BIT.LY/SIGEVENTOS
- ENCONTROS: SEGUNDAS E QUARTAS
- HORÁRIO: 19:00 ÀS 20:00

aria.ci.ufpb.br

SE INSCREVE E JÁ APERTA NO SININHO, QUE VOCÊS PASSAM A RECEBER AS NOTIFICAÇÕES.

NOSSOS ENCONTROS DURARÃO 1 HORA E, ASSIM QUE POSSÍVEL, DEIXAREMOS OS VÍDEOS GRAVADOS NO CANAL.

NÃO PRECISA SE PREOCUPAR EM ESTAR LIGADO ÀS 19:00, MAS ESTANDO, ROLA TIRAR DÚVIDA E PARTICIPAR, O QUE JÁ DEIXA A AULA MAIS ANIMADA.

SOBRE O MATERIAL DE ACOMPANHAMENTO: O ALUNO PRECISA SE LOGAR EM:
CLASSROOM.GOOGLE.COM
DEPOIS, CLICAR EM PARTICIPAR DA TURMA (ÍCONE COM UM MAIS +)

POR FIM, ENTRAR COM O CÓDIGO DA TURMA: PXV3ANW

TAMBÉM EM: HTTPS://ARIA.CI.UFPB.BR/IA-PARA-TODOS-MATERIAL/
ESPERAMOS QUE VOCÊS REALMENTE CURTAM O CURSO E APROVEITEM-NO AO MÁXIMO. NÃO DEIXEM DE INTERAGIR CONOSCO, TAMBÉM, POR E-MAIL OU MENSAGEM NO NOSSO INSTAGRAM (@APRENDIZAGEMDEMAQUINA)

Redes Neurais Convolucionais

Yuri Malheiros

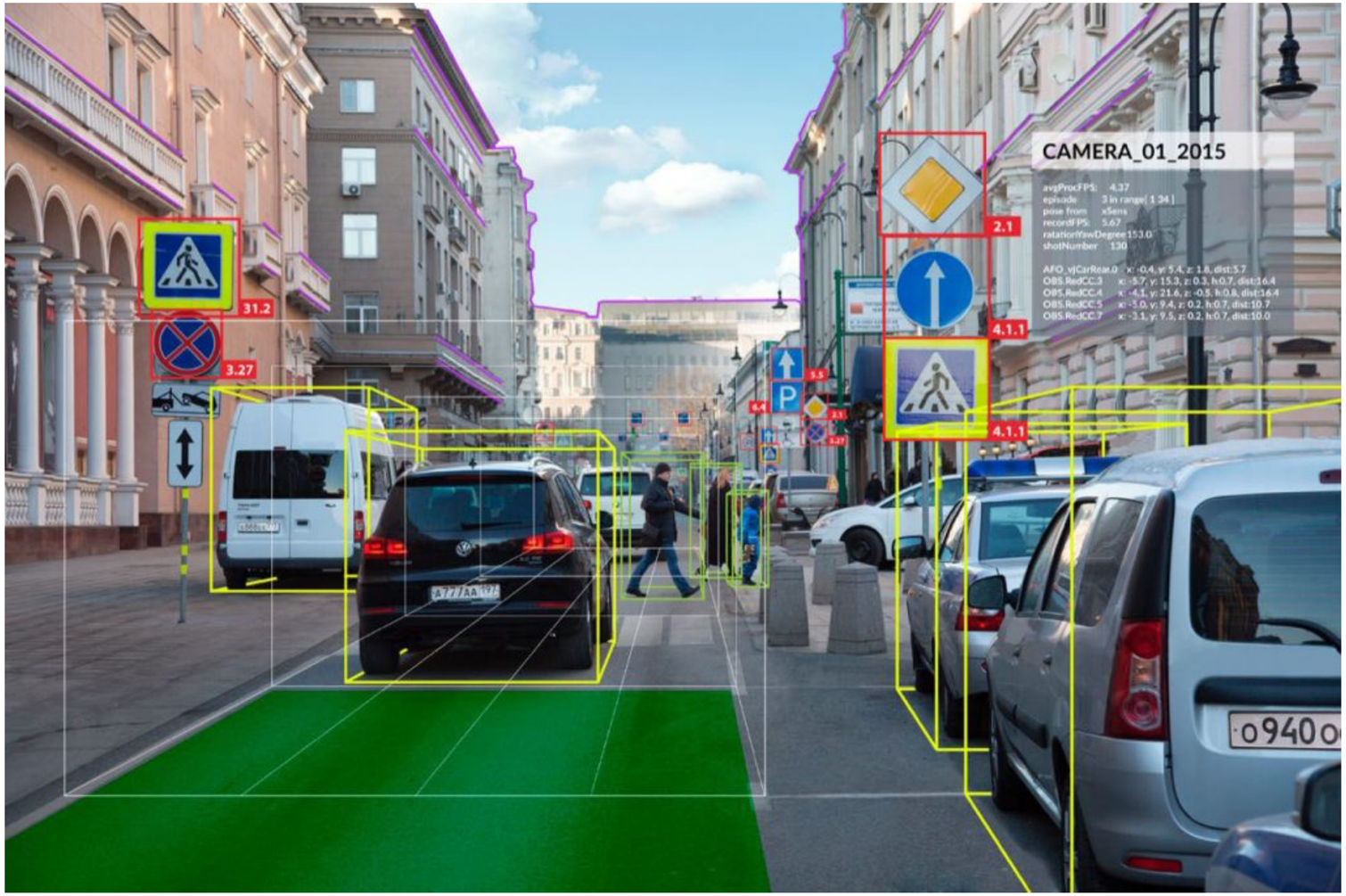
Visão Computacional

Visão computacional é o campo da ciência que estuda como os computadores podem enxergar e entender o conteúdo de imagens e vídeos

Isto inclui adquirir, processar e analisar imagens ou vídeos

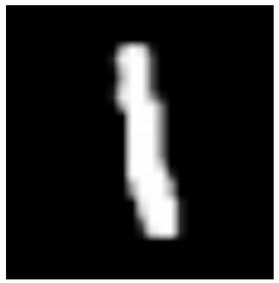
Problemas ligados a visão computacional muitas vezes são resolvidos facilmente por pessoas, mas não pelas máquinas

Visão Computacional



Visão Computacional

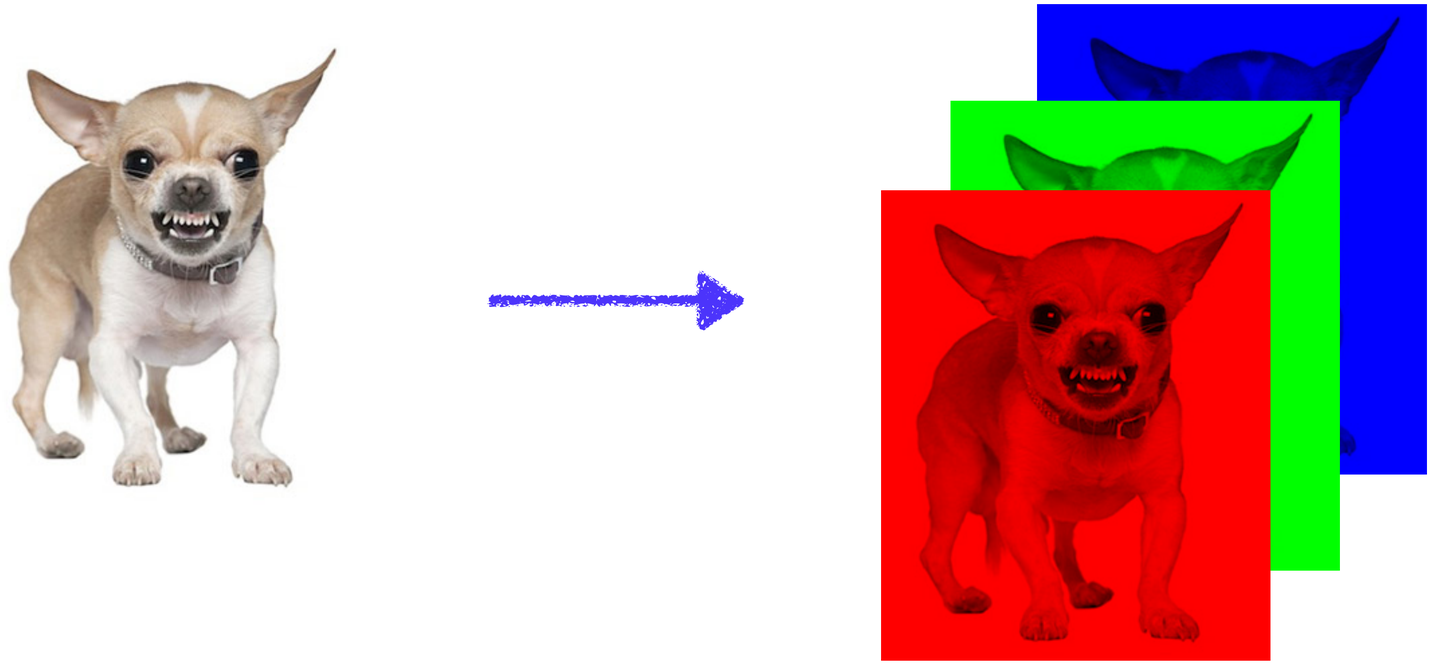
Uma imagem é representada como uma matriz



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

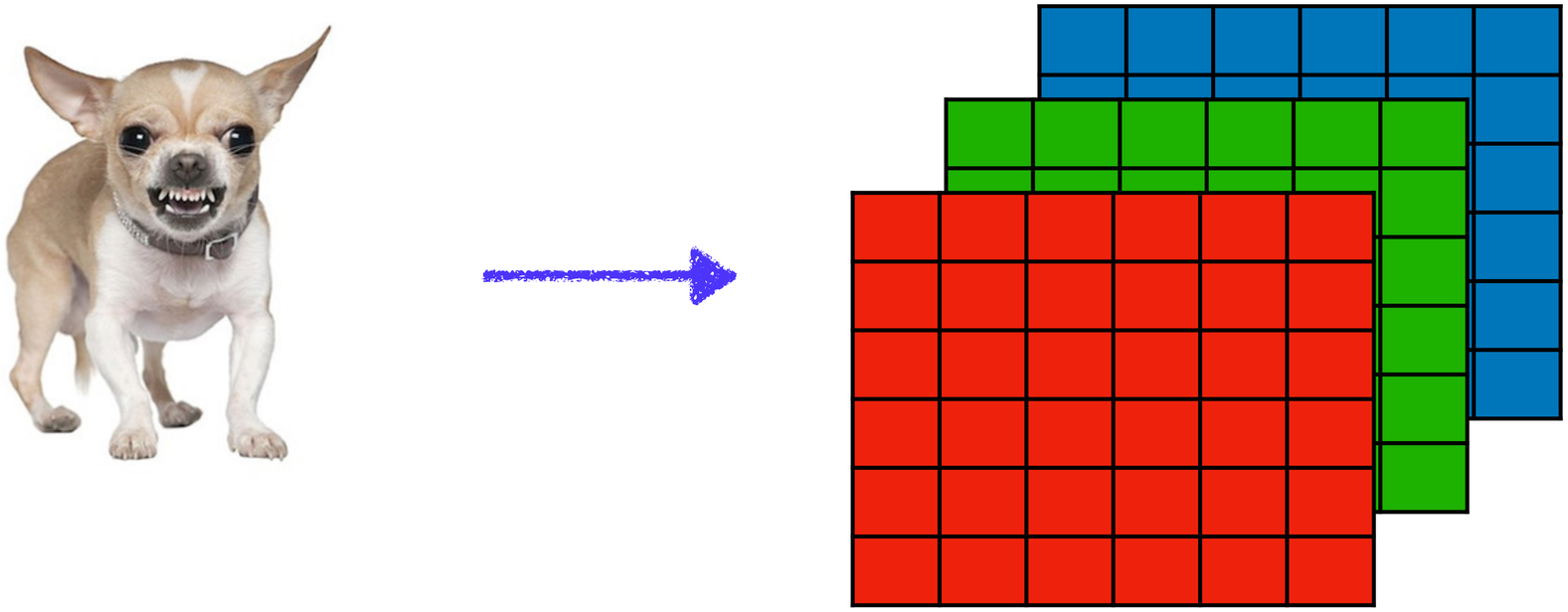
Visão Computacional

Uma imagem colorida possui 3 canais (red, green e blue)



Visão Computacional

Uma imagem colorida possui 3 canais (red, green e blue)

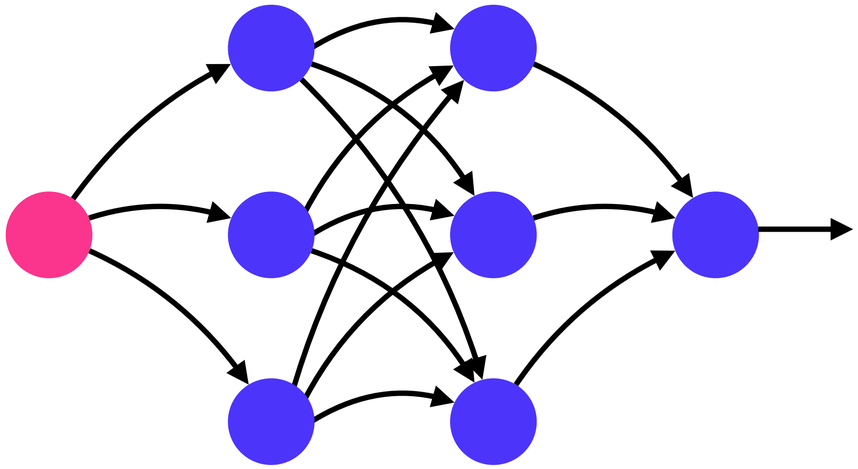


Classificação de Imagem

Cachorro ou gato?

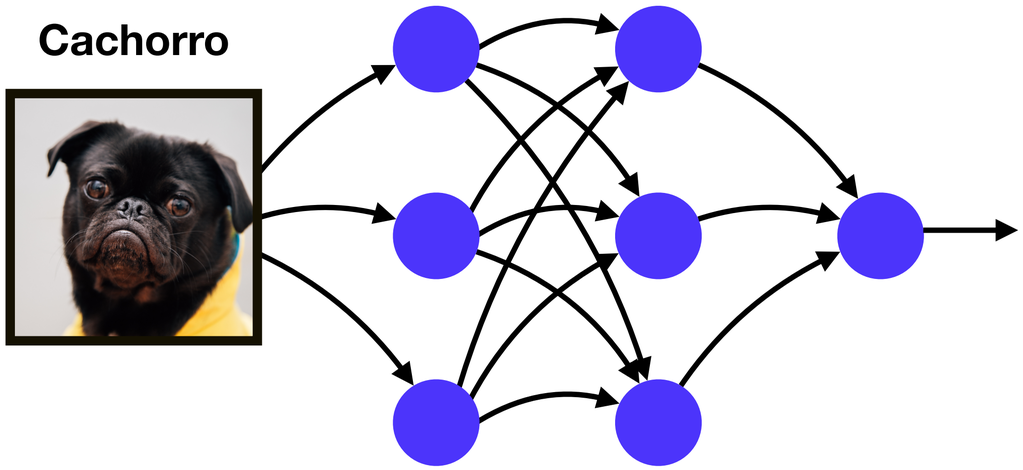
Classificação de Imagem

Podemos treinar uma rede neural com várias imagens de cachorros e gatos



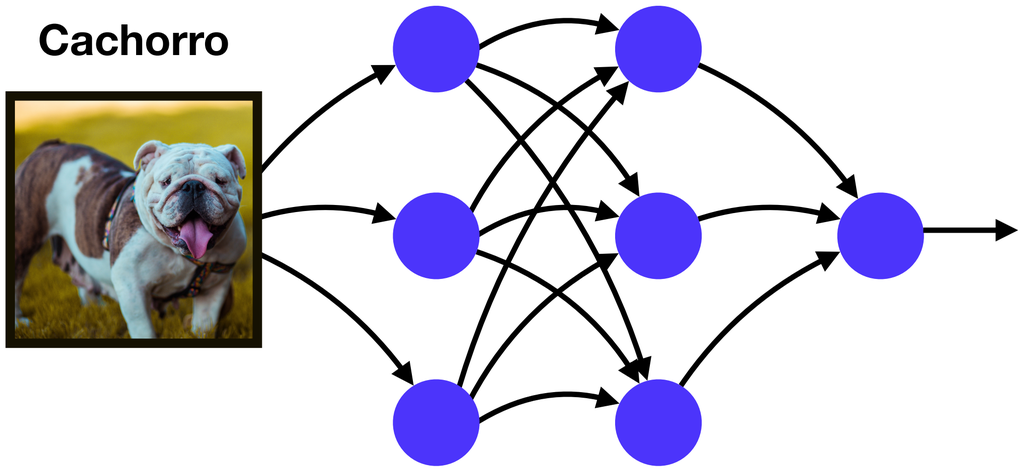
Classificação de Imagem

Podemos treinar uma rede neural com várias imagens de cachorros e gatos



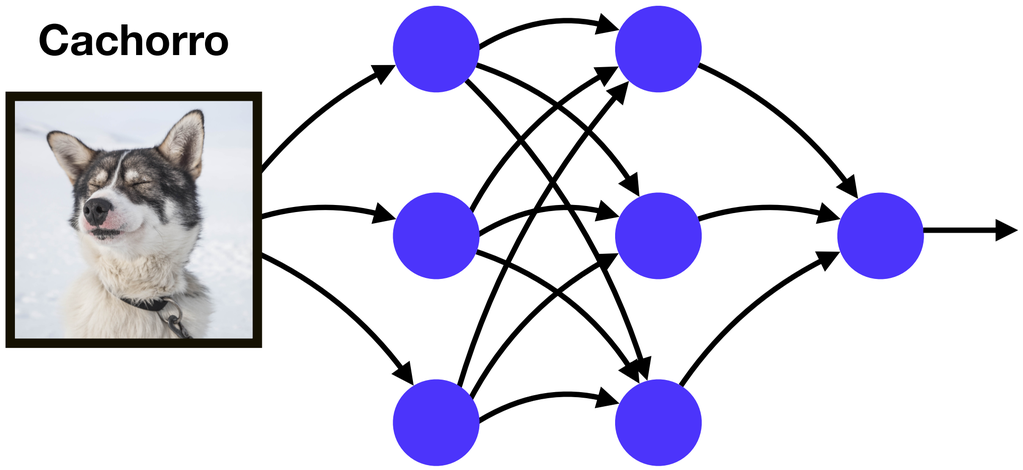
Classificação de Imagem

Podemos treinar uma rede neural com várias imagens de cachorros e gatos



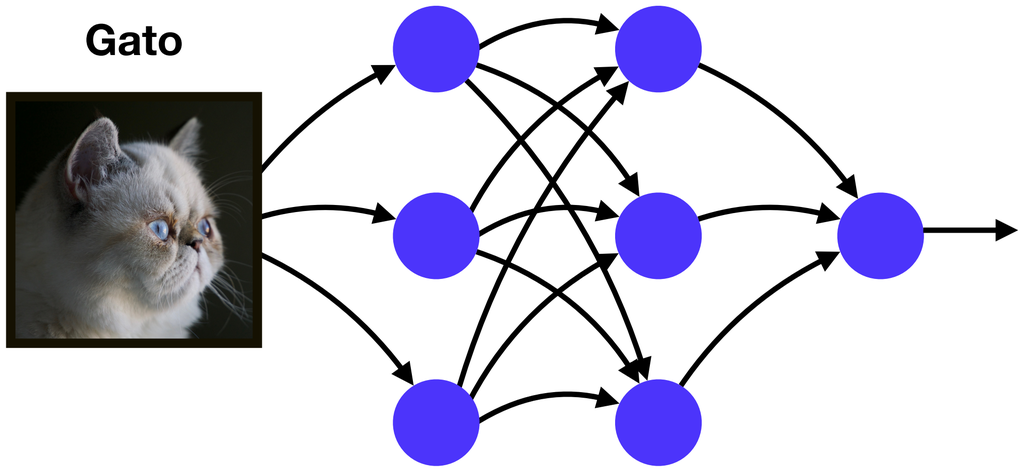
Classificação de Imagem

Podemos treinar uma rede neural com várias imagens de cachorros e gatos



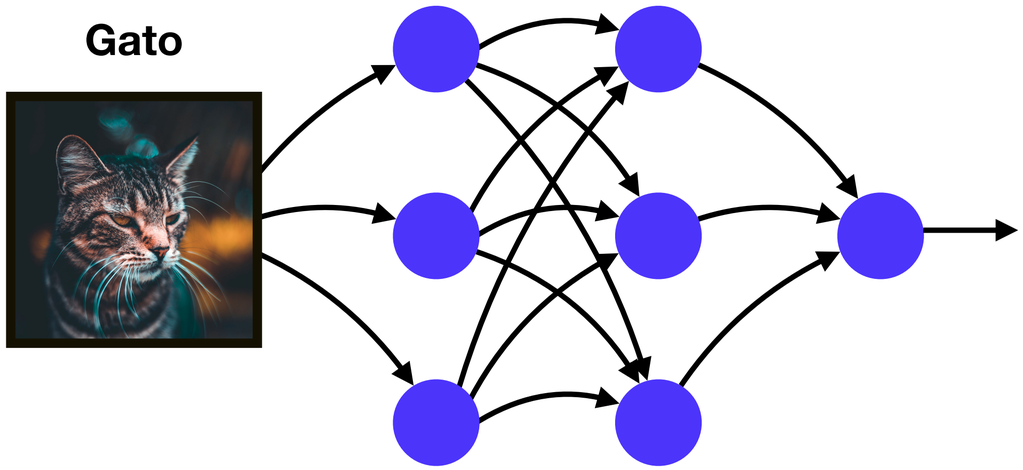
Classificação de Imagem

Podemos treinar uma rede neural com várias imagens de cachorros e gatos



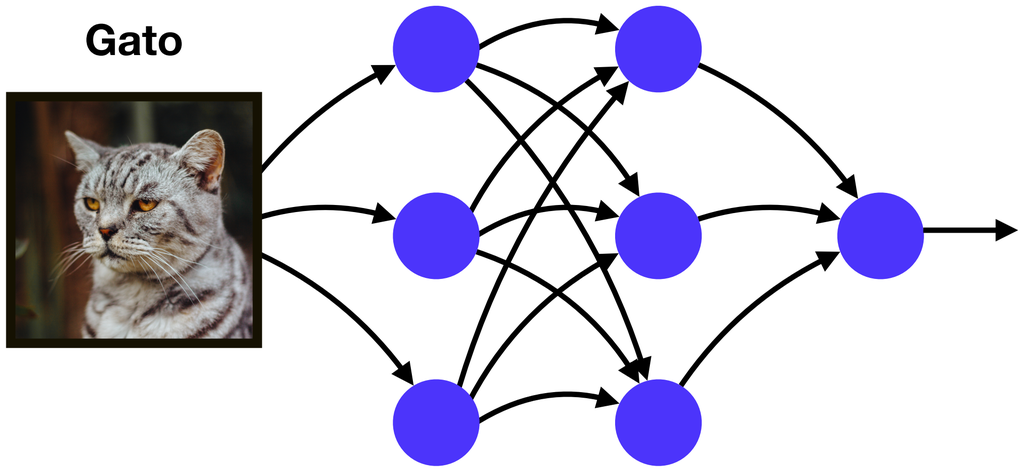
Classificação de Imagem

Podemos treinar uma rede neural com várias imagens de cachorros e gatos



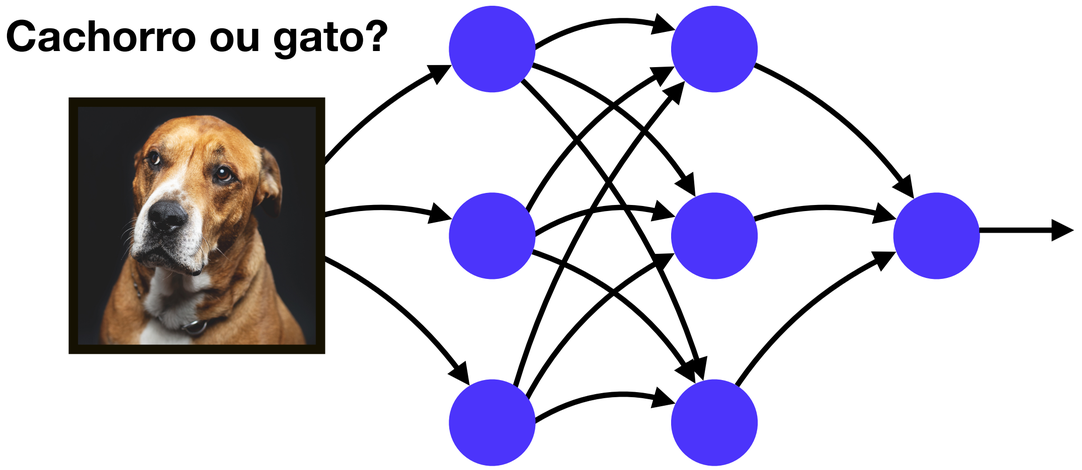
Classificação de Imagem

Podemos treinar uma rede neural com várias imagens de cachorros e gatos



Classificação de Imagem

Para depois classificar novas entradas



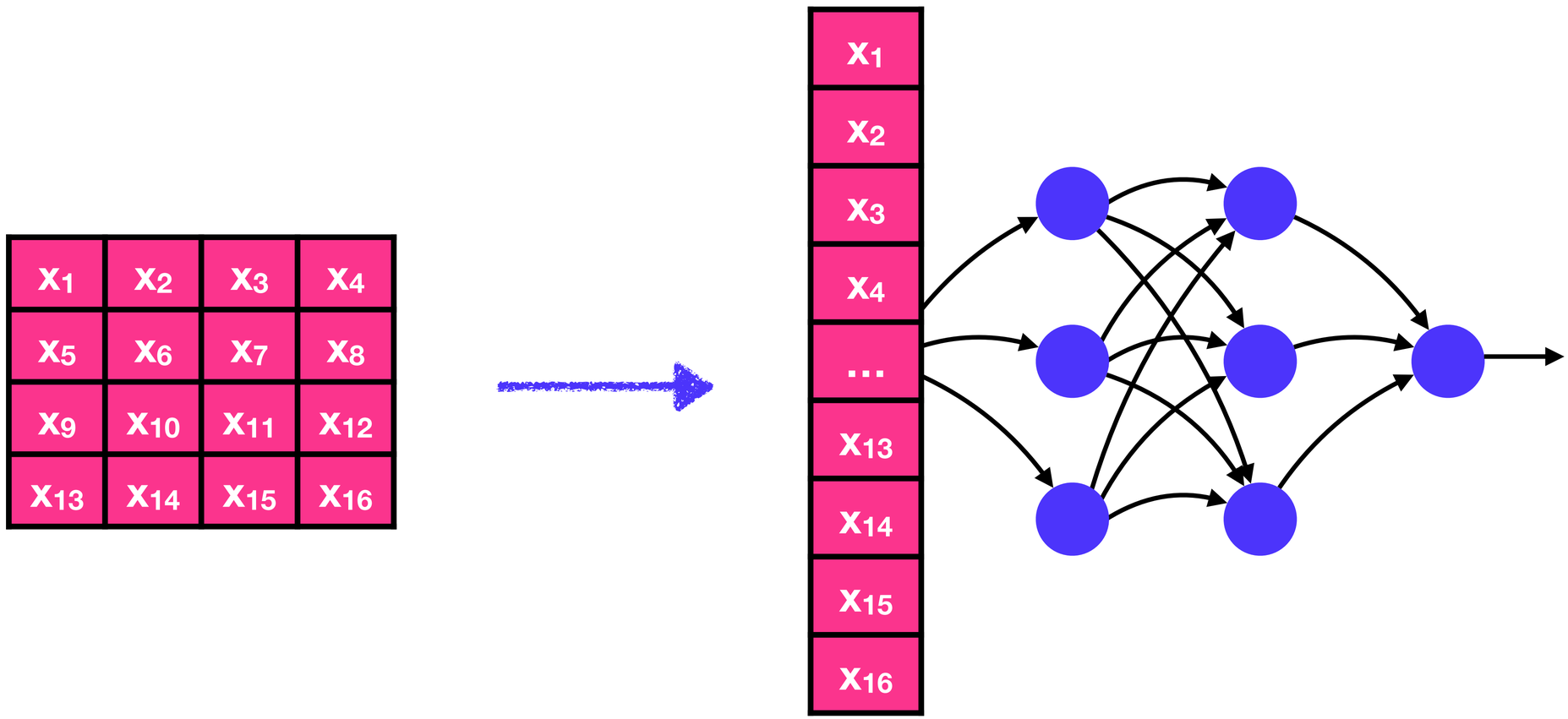
Classificação de Imagem

Como passar uma imagem para uma rede neural?

A rede recebe entradas numéricas, ela não recebe uma matriz

Classificação de Imagem

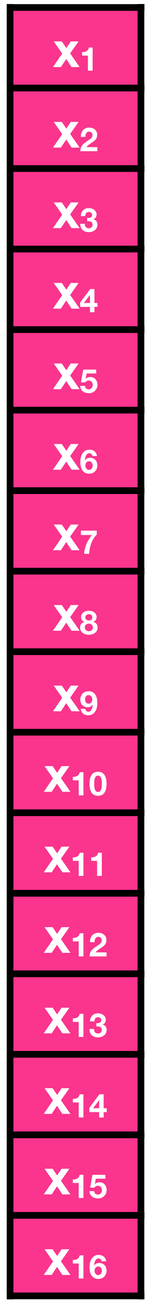
Podemos passar cada valor da matriz como uma entrada



Classificação de Imagem

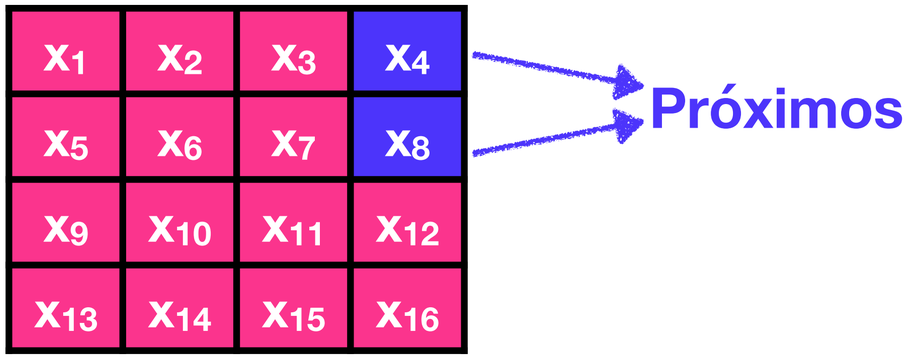
Uma limitação dessa abordagem é que perdemos a noção espacial da imagem

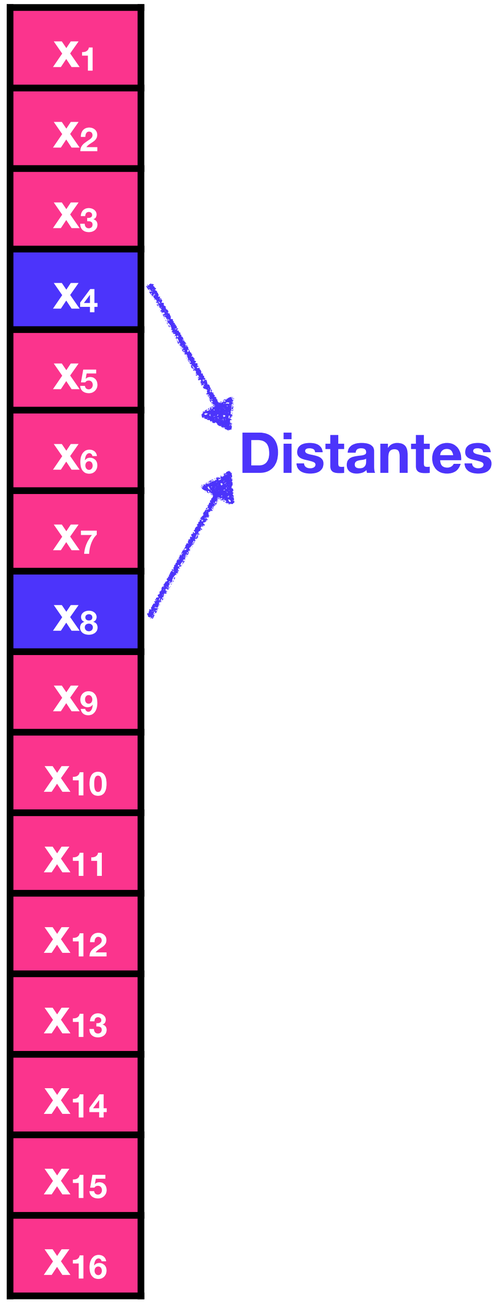
x_1	x_2	x_3	x_4
x_5	x_6	x_7	x_8
x_9	x_{10}	x_{11}	x_{12}
x_{13}	x_{14}	x_{15}	x_{16}



Classificação de Imagem

Uma limitação dessa abordagem é que perdemos a noção espacial da imagem





Classificação de Imagem

Outra limitação é que o número de entradas e pesos será muito grande

Uma imagem colorida 1000x1000 tem 3 milhões de valores

Se a primeira camada intermediária possuir 10 neurônios, teremos 30 milhões de pesos só na primeira camada!

Classificação de Imagem

Podemos processar previamente as imagens e depois passar para rede neural:

- Detectar arestas
- Detectar cantos
- Detectar regiões

Detecção de arestas

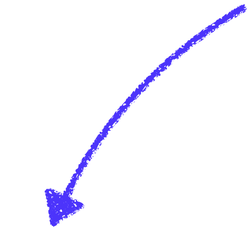
Detecção de arestas

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

Detecção de arestas

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

Convolução



*

Detecção de arestas

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

*

Filtro

Detecção de arestas

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

*

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

Detecção de arestas

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

*

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

=

Detecção de arestas

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

*

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

=

Detecção de arestas

3x1

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

*

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

=

Detecção de arestas

3x1 + 1x1

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

*

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

=

Detecção de arestas

$$3x1 + 1x1 + 2x1$$

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

*

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

=

Detecção de arestas

$$3x1 + 1x1 + 2x1 + 0x0$$

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

*

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

=

Detecção de arestas

$$3x1 + 1x1 + 2x1 + 0x0 + 5x0$$

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

*

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

=

Detecção de arestas

$$3x1 + 1x1 + 2x1 + 0x0 + 5x0 + 7x0$$

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

*

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

=

Detecção de arestas

3x1 + 1x1 + 2x1 + 0x0 + 5x0 + 7x0 + 1x-1

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

*

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

=

Detecção de arestas

3x1 + 1x1 + 2x1 + 0x0 + 5x0 + 7x0 + 1x-1 + 8x-1

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

*

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

=

Detecção de arestas

3x1 + 1x1 + 2x1 + 0x0 + 5x0 + 7x0 + 1x-1 + 8x-1 + 2x-1

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

*

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

=

Detecção de arestas

3x1 + 1x1 + 2x1 + 0x0 + 5x0 + 7x0 + 1x-1 + 8x-1 + 2x-1 = -5

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

*

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

=

-5			

Detecção de arestas

$$0x1 + 5x1 + 7x1 + 1x0 + 8x0 + 2x0 + 2x{-1} + 9x{-1} + 5x{-1} = -5$$

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

*

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

=

-5	-4		

Detecção de arestas

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

*

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

=

-5	-4	0	

Detecção de arestas

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

*

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

=

-5	-4	0	8

Detecção de arestas

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

*

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

=

-5	-4	0	8
-10			

Detecção de arestas

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

*

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

=

-5	-4	0	8
-10	-2		

Detecção de arestas

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

*

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

=

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

Detecção de arestas

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

*

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

=

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

Detecção de arestas

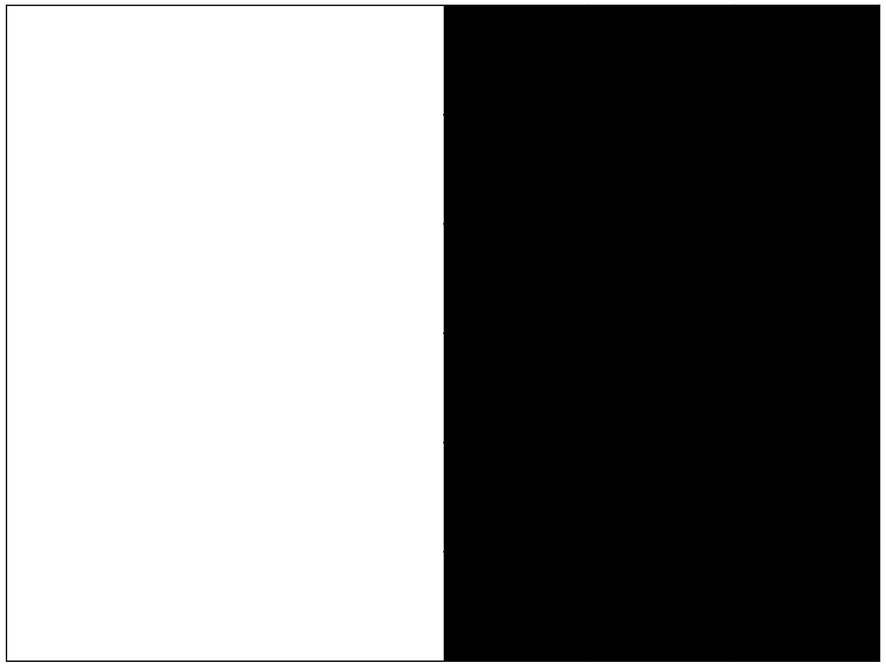
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

*

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

=

Detecção de arestas



*

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

=

Detecção de arestas

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

=

Detecção de arestas

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	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 arestas

10x1 + 10x1 + 10x1 + 10x0 + 10x0 + 10x0 + 10x-1 + 10x-1 + 10x-1 = 0

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	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 arestas

10x1 + 10x1 + 10x1 + 10x0 + 10x0 + 10x0 + 10x0 + 10x0 + 10x0 = 30

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

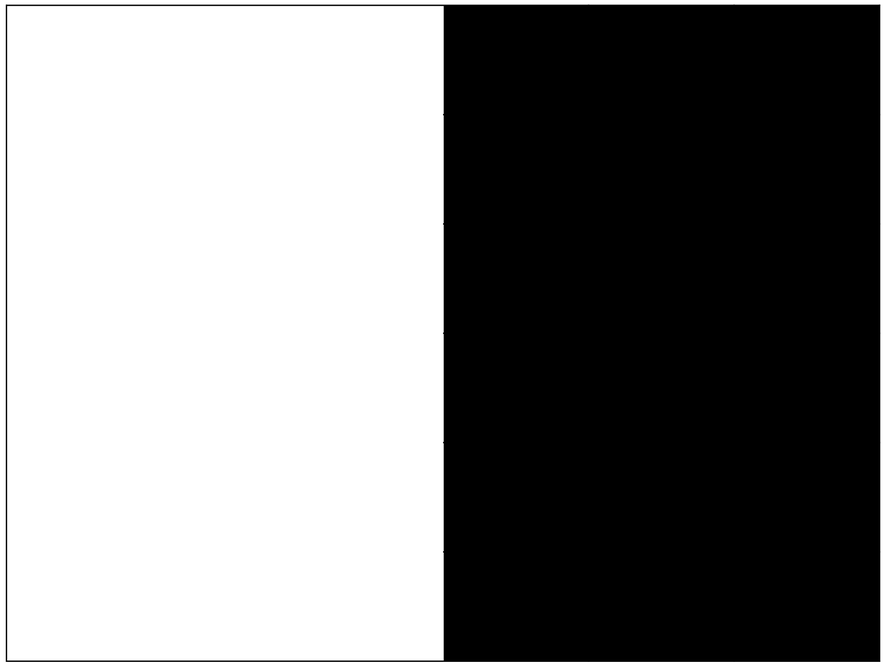
*

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

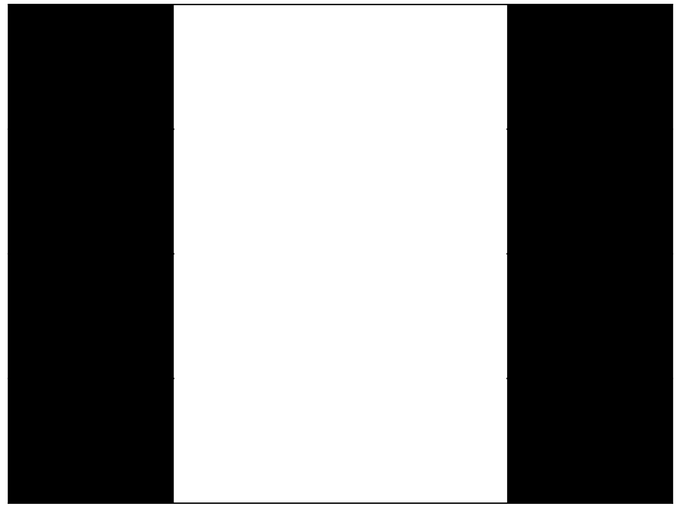
Detecção de arestas



*

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

=



Detecção de arestas

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	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 arestas

Filtros diferentes detectam características diferentes:

Arestas verticais

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

Arestas horizontais

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

Redes Neurais Convolucionais

Podemos tentar criar filtros manualmente escolhendo valores

Quais filtros criar?

Como saber se os valores escolhidos são bons?

Não sabemos previamente!

Redes Neurais Convolucionais

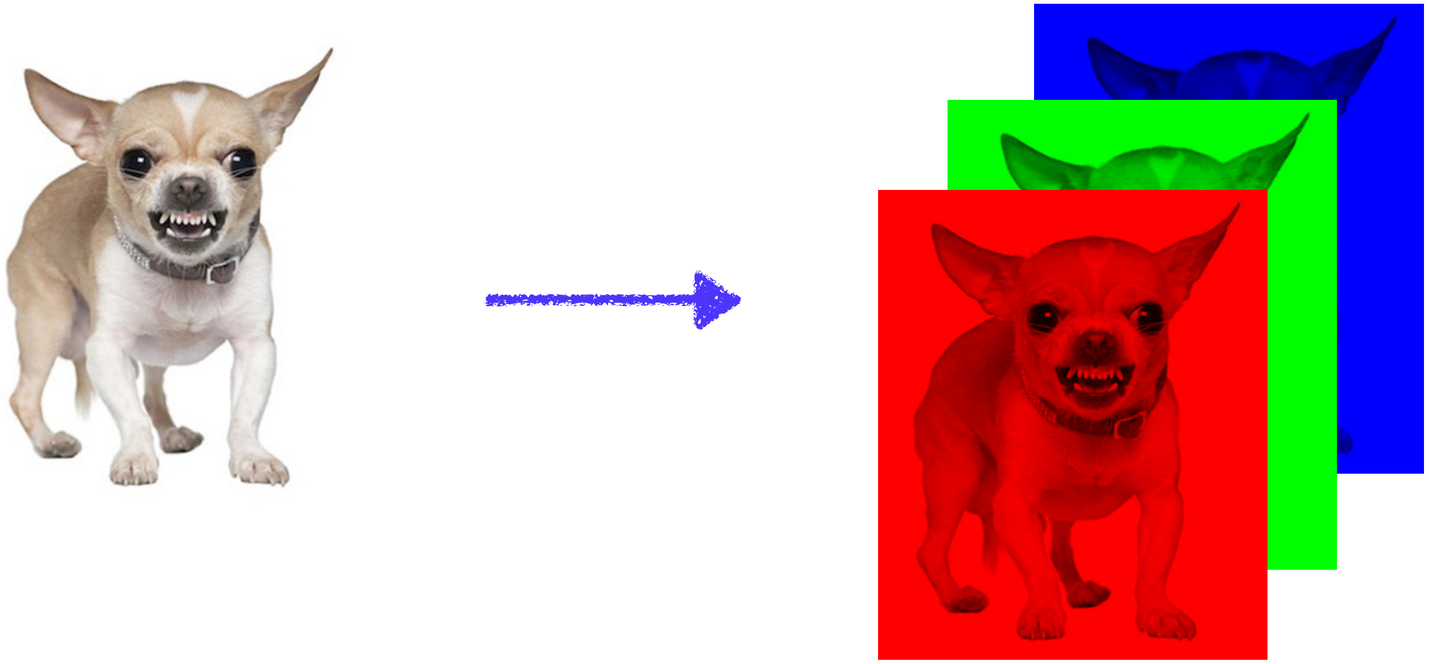
3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

*

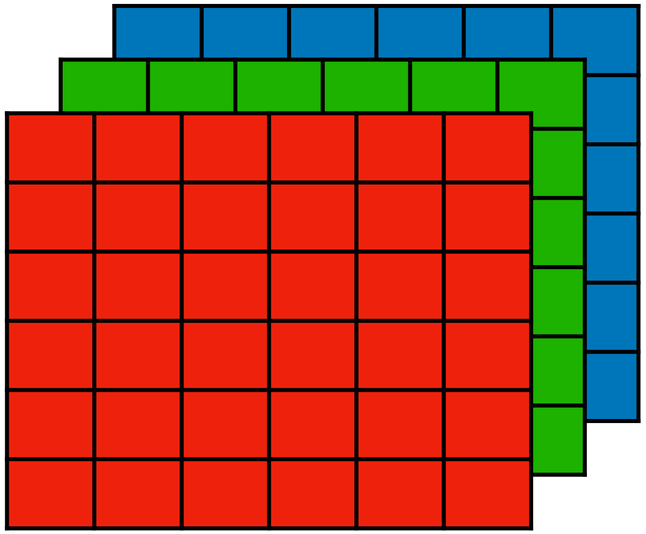
w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Vamos tratar os valores dos filtros como pesos que serão aprendidos pela rede neural através do backpropagation

Convolução 3D

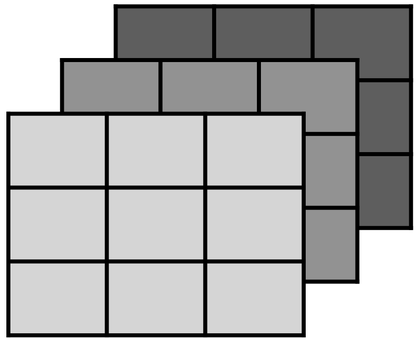


Convolução 3D



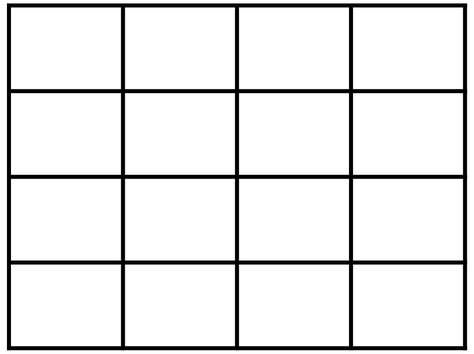
6 x 6 x 3

*



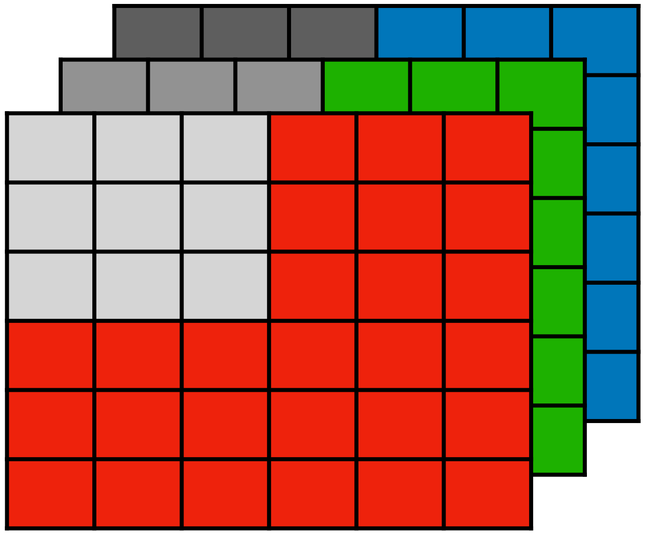
3 x 3 x 3

=



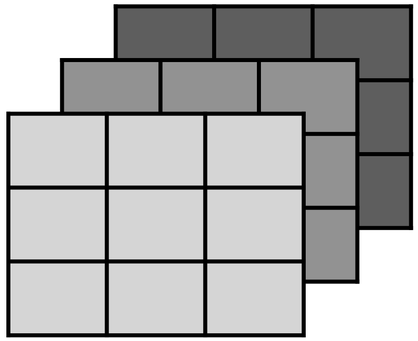
4 x 4

Convolução 3D



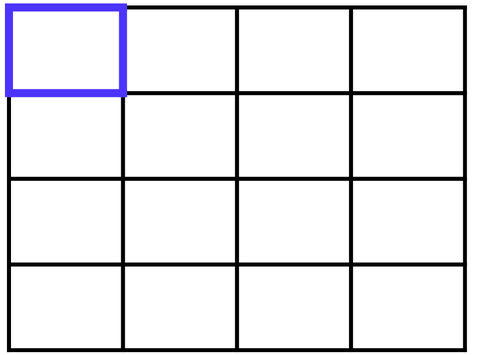
6 x 6 x 3

*

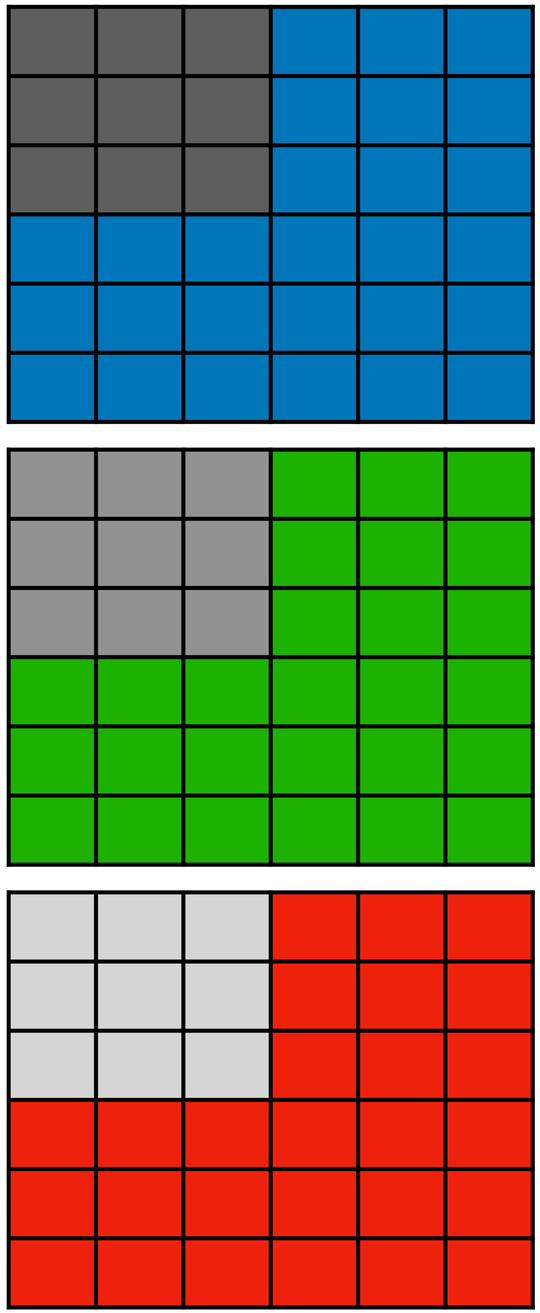


3 x 3 x 3

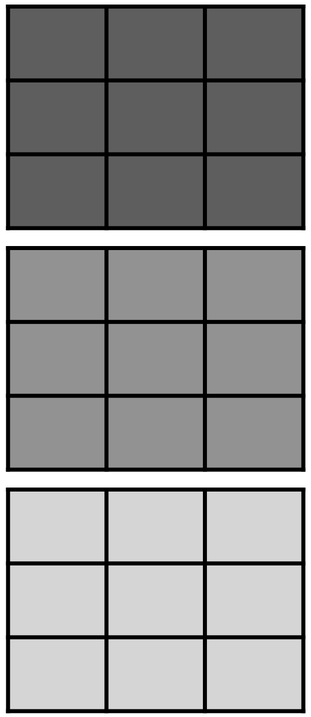
=



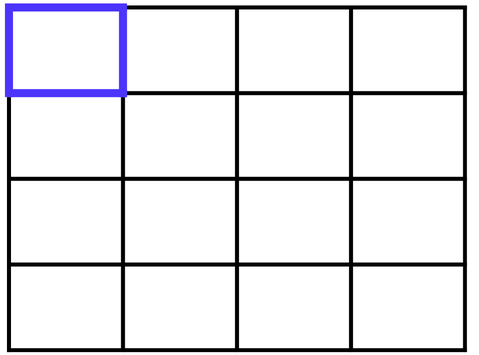
4 x 4



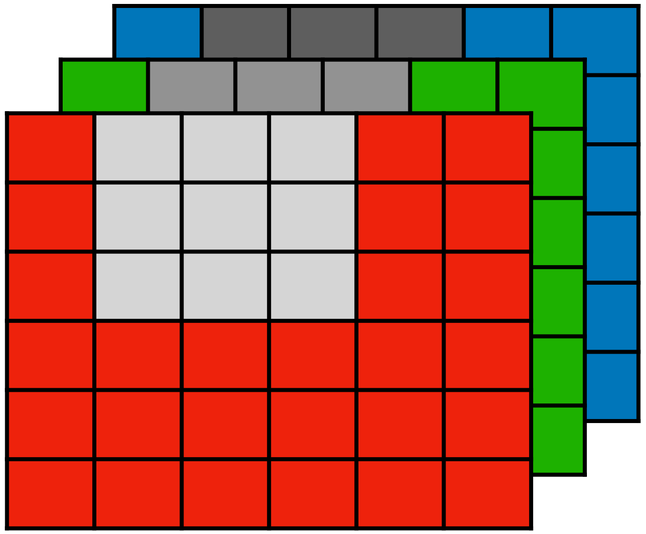
*



=

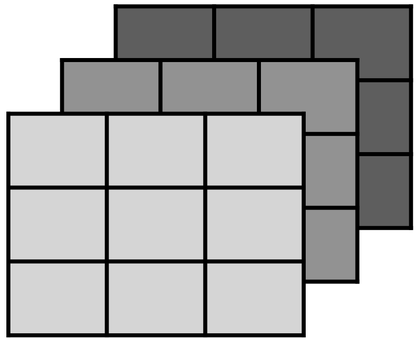


Convolução 3D



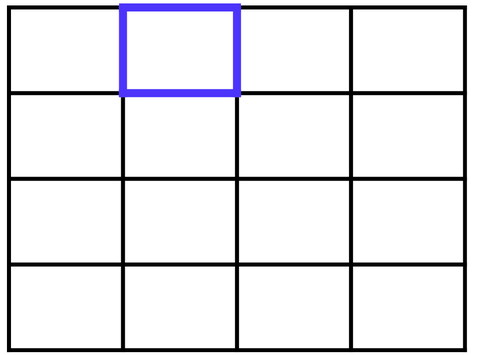
6 x 6 x 3

*



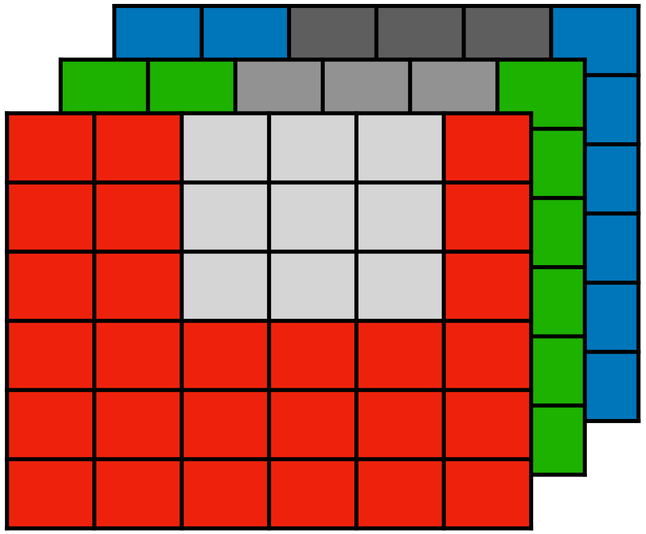
3 x 3 x 3

=



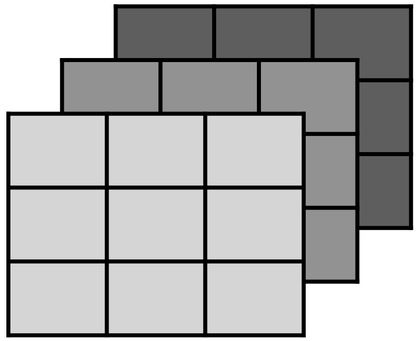
4 x 4

Convolução 3D



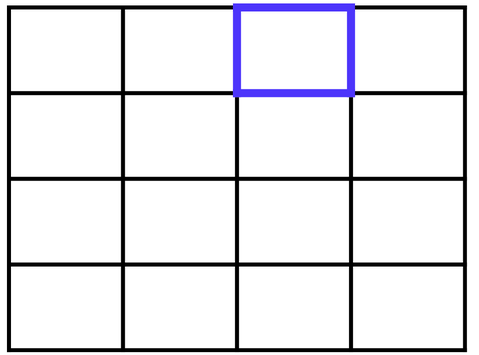
6 x 6 x 3

*



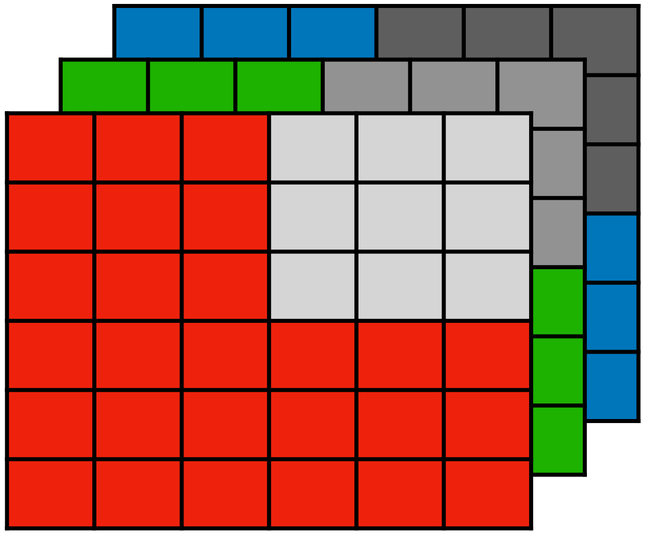
3 x 3 x 3

=



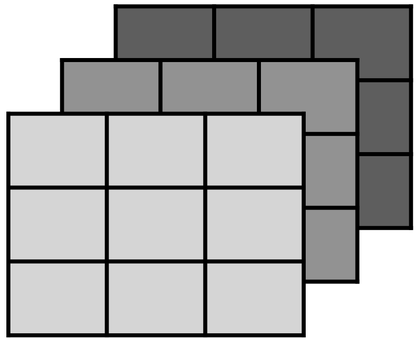
4 x 4

Convolução 3D



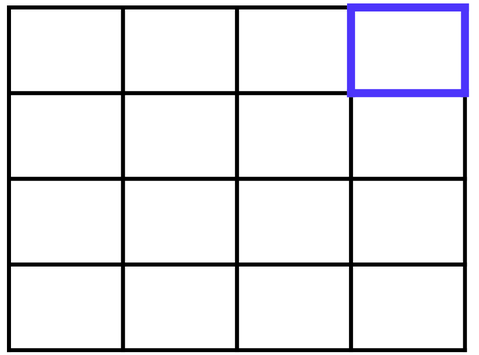
6 x 6 x 3

*



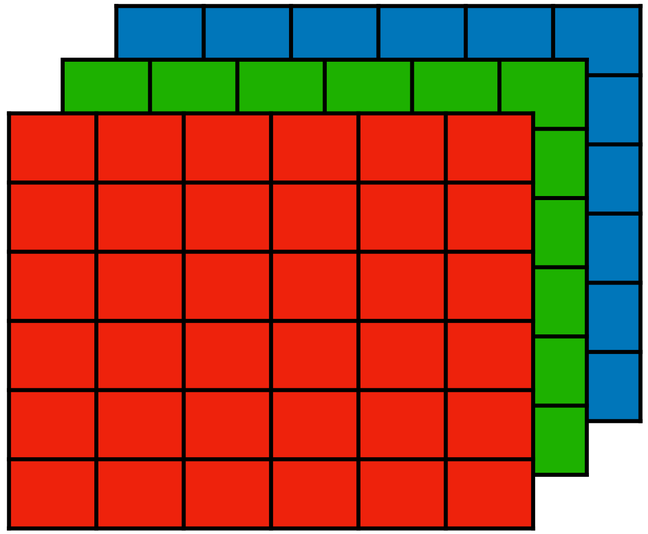
3 x 3 x 3

=



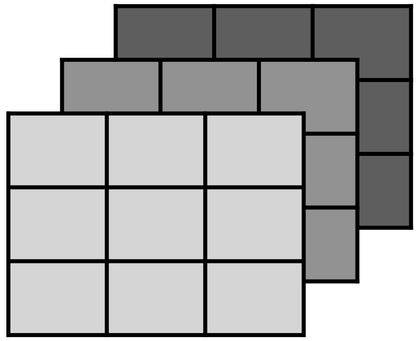
4 x 4

Convolução 3D



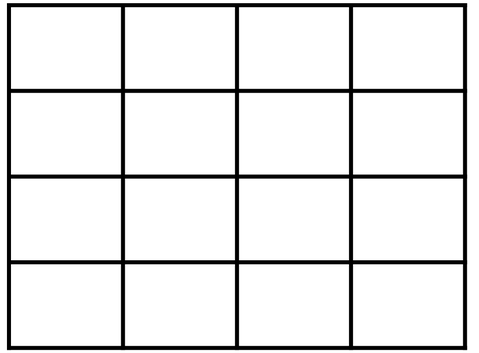
6 x 6 x 3

*



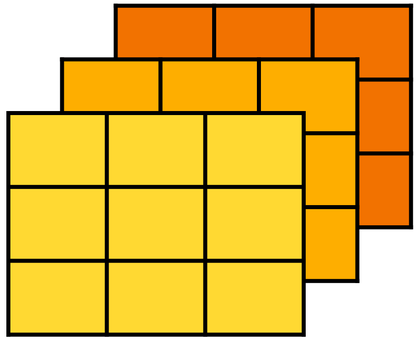
3 x 3 x 3

=



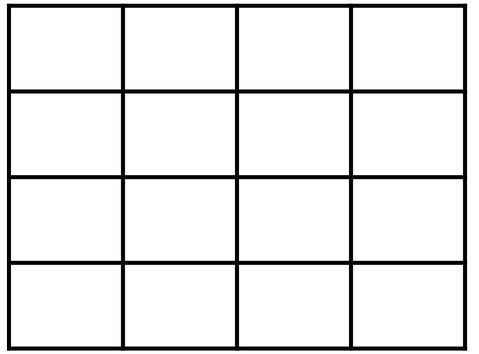
4 x 4

*



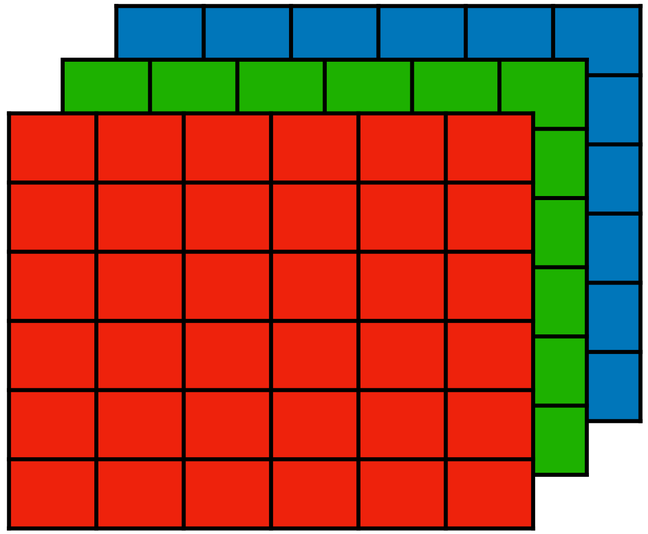
3 x 3 x 3

=



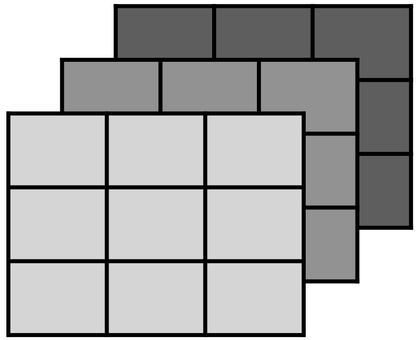
4 x 4

Convolução 3D



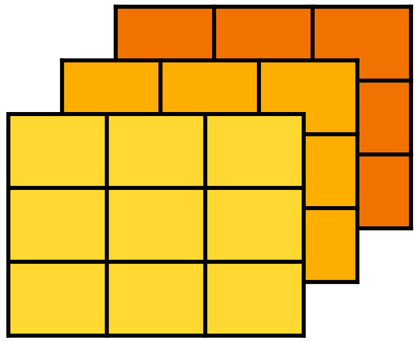
6 x 6 x 3

*



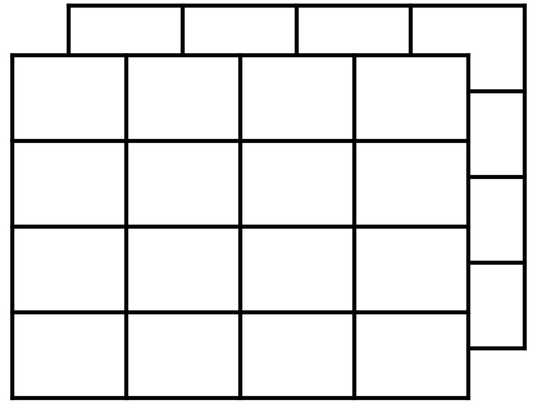
3 x 3 x 3

*



3 x 3 x 3

=



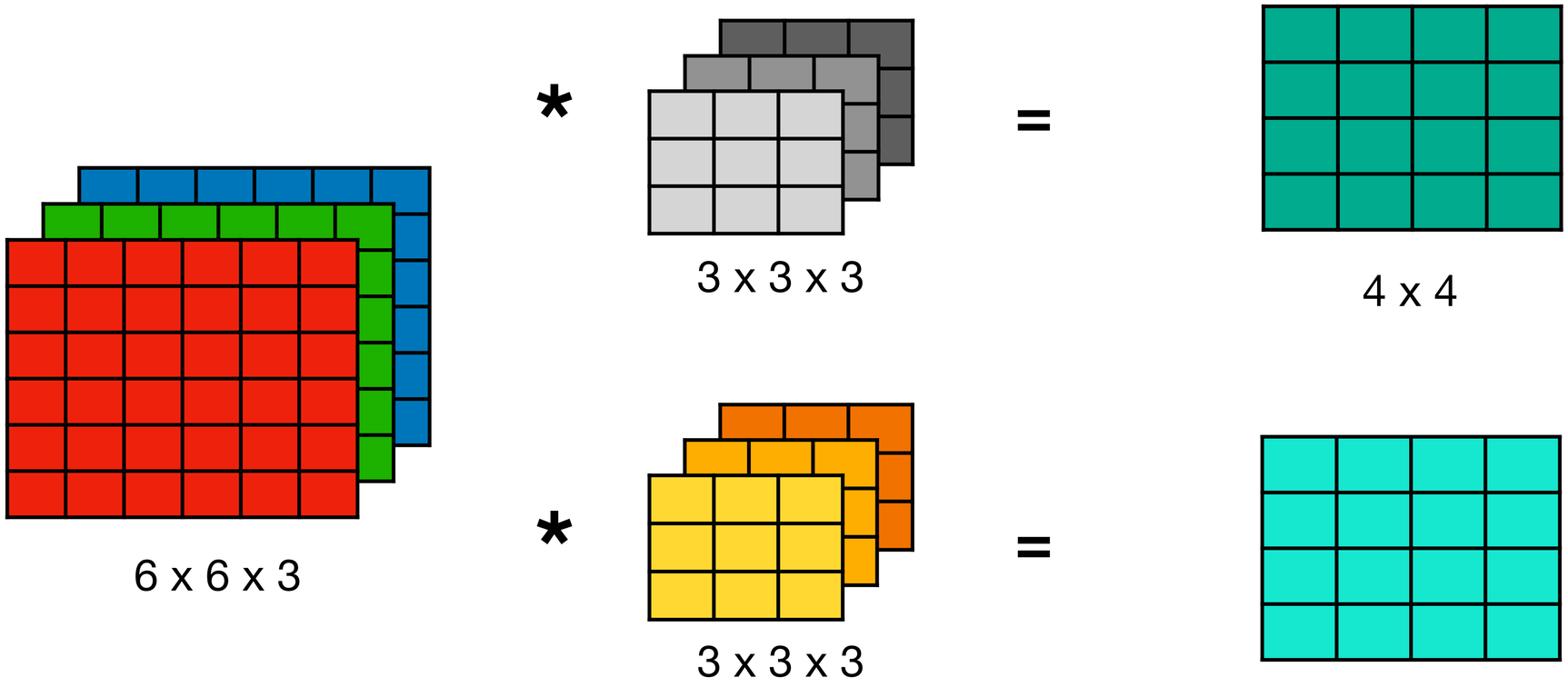
4 x 4 x 2

Redes Neurais Convolucionais

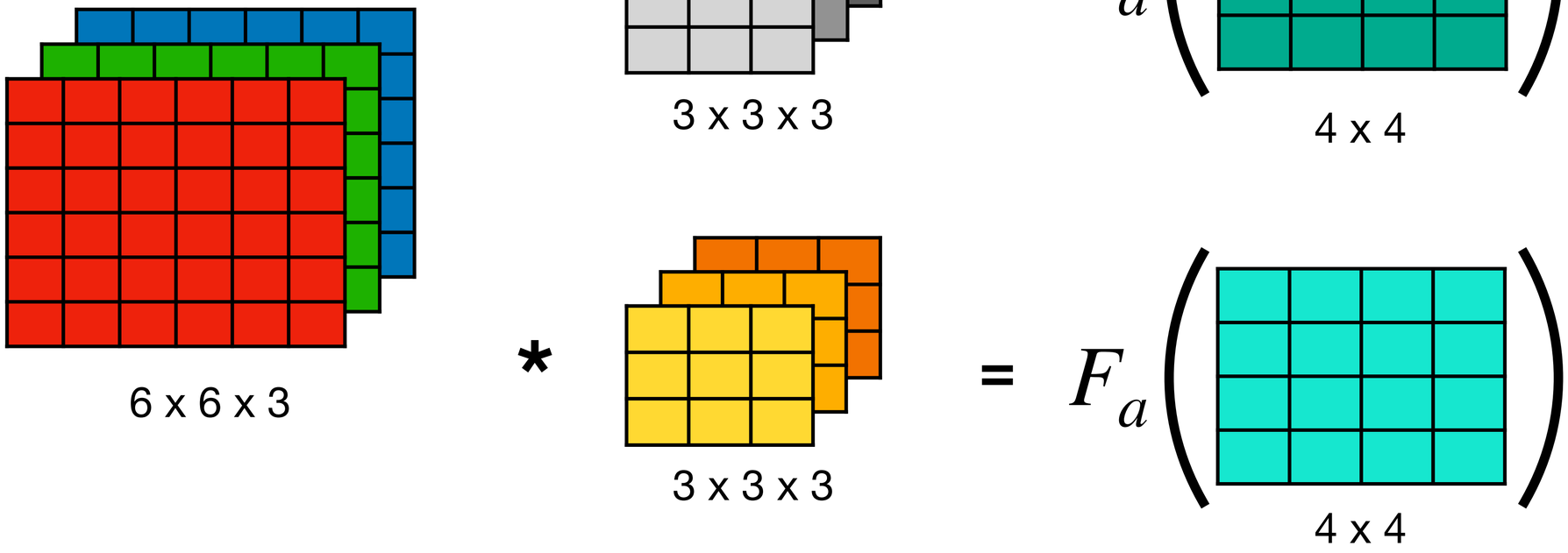
Temos todas as peças para entender e construir uma rede neural convolucional

Iniciaremos com uma camada

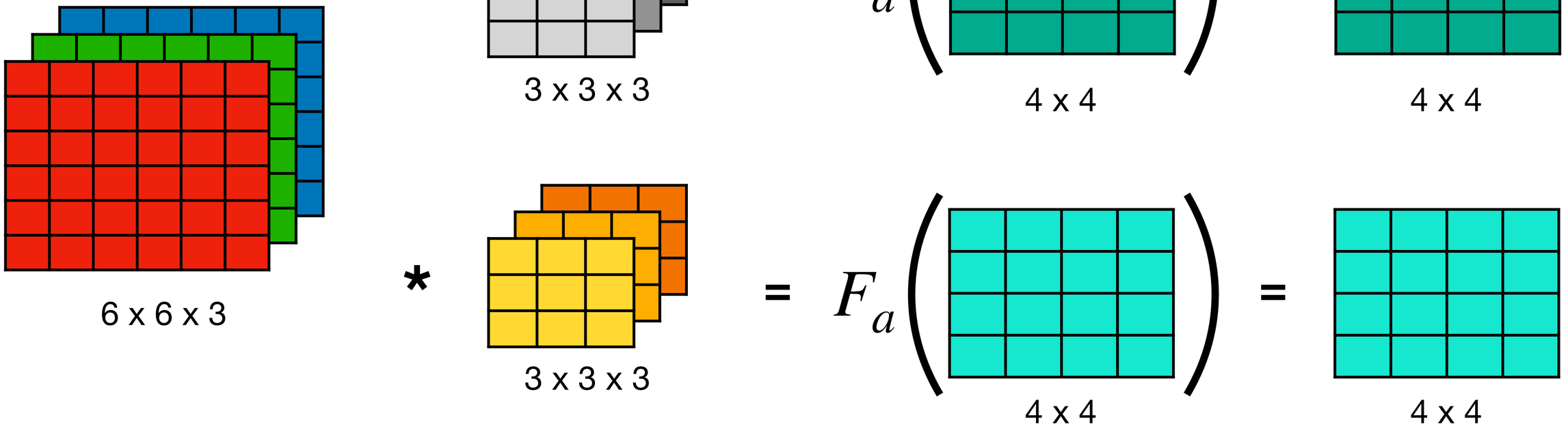
Redes Neurais Convolucionais



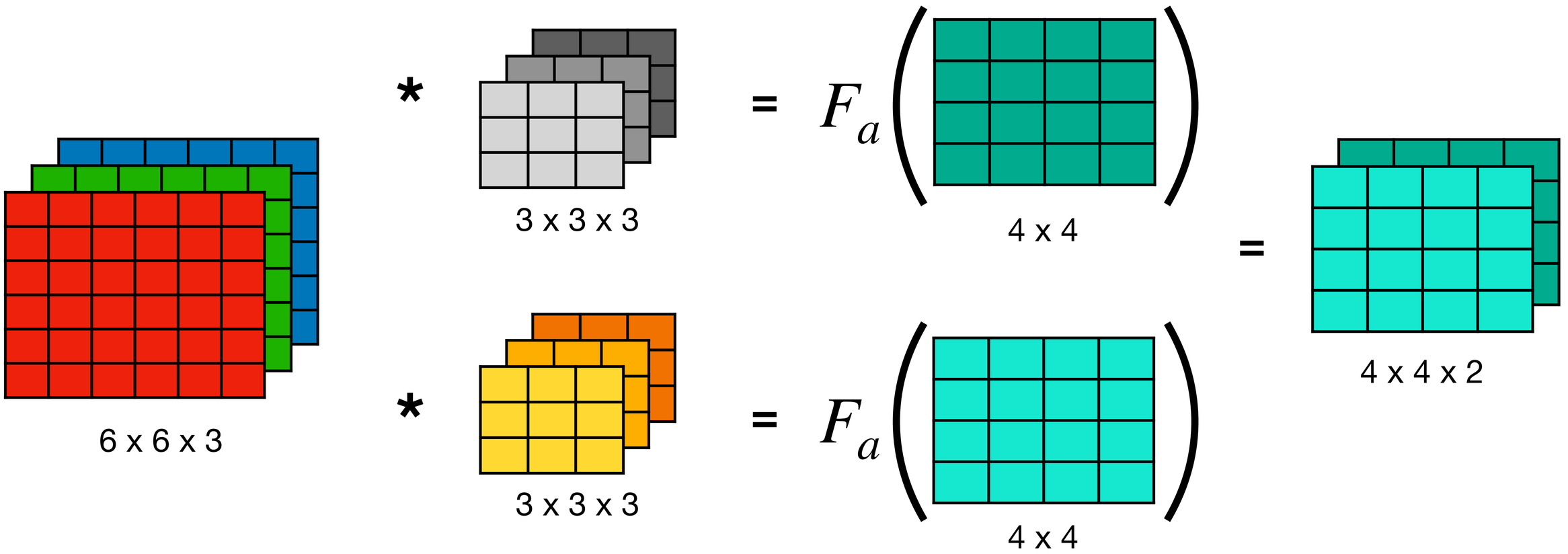
Redes Neurais Convolucionais



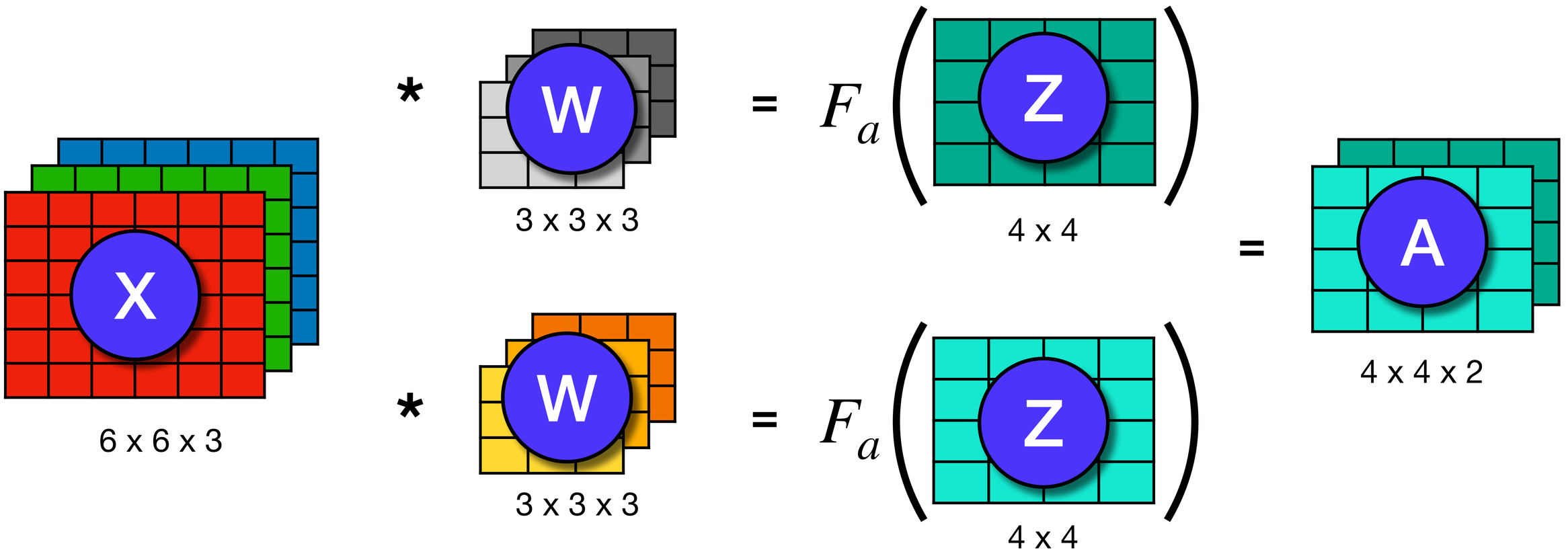
Redes Neurais Convolucionais



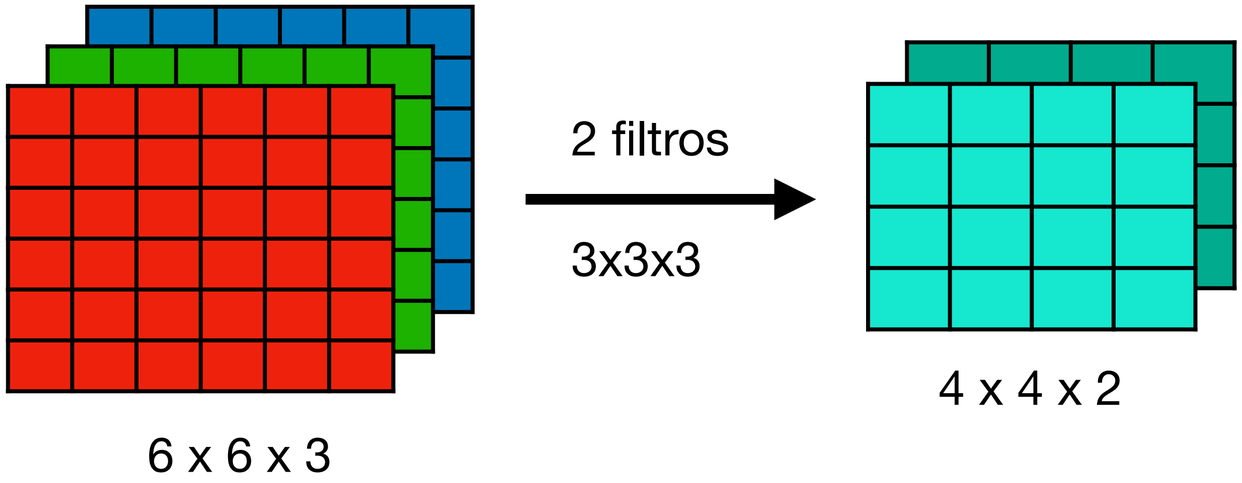
Redes Neurais Convolucionais



Redes Neurais Convolucionais

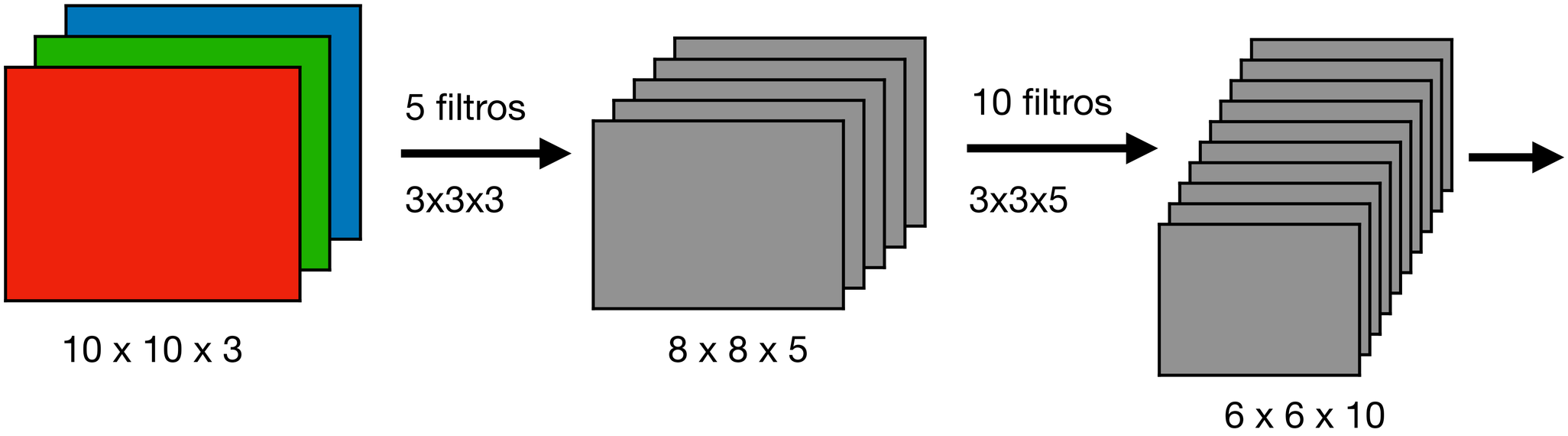


Redes Neurais Convolucionais



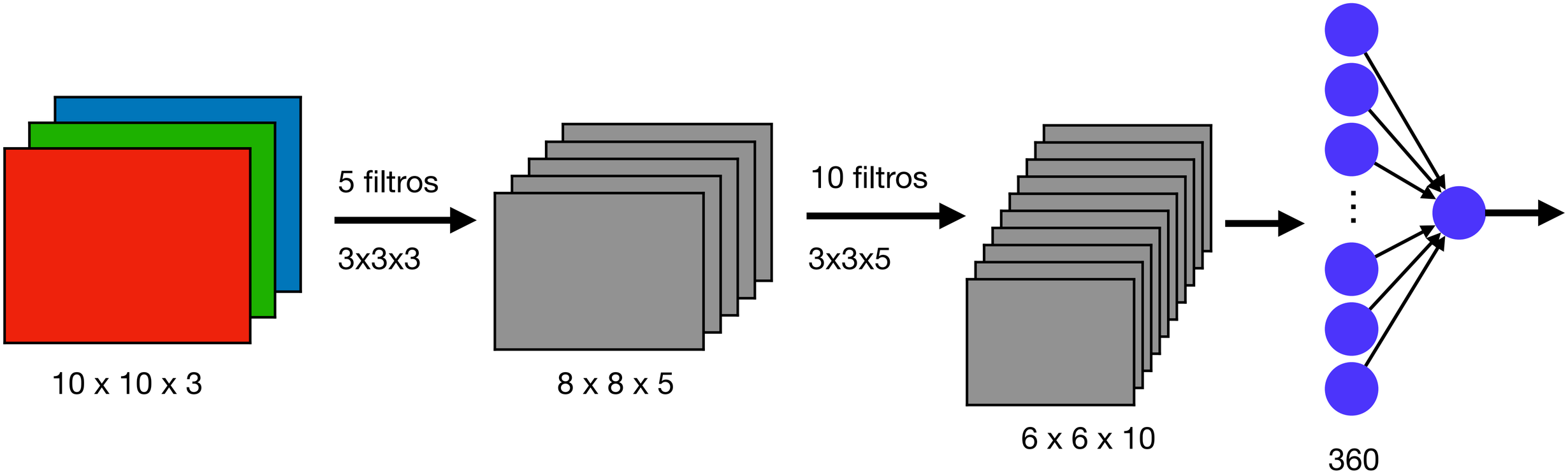
Redes Neurais Convolucionais

Uma rede convolucional com duas camadas:



Redes Neurais Convolucionais

No final, usamos uma camada totalmente conectada:

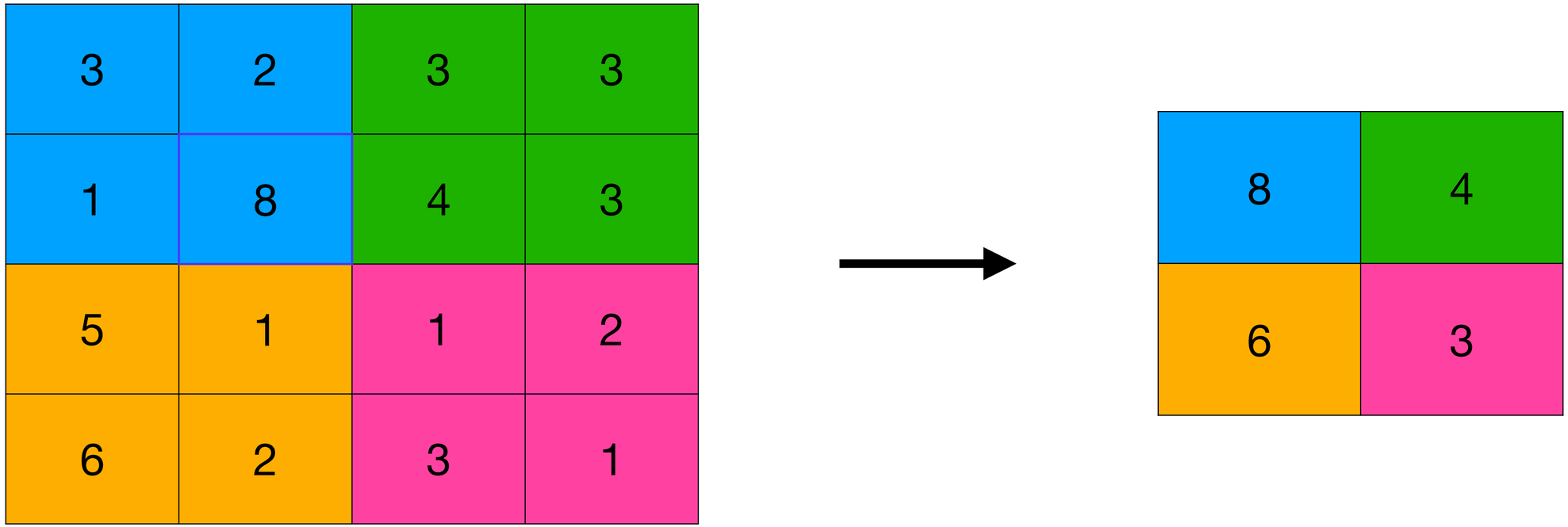


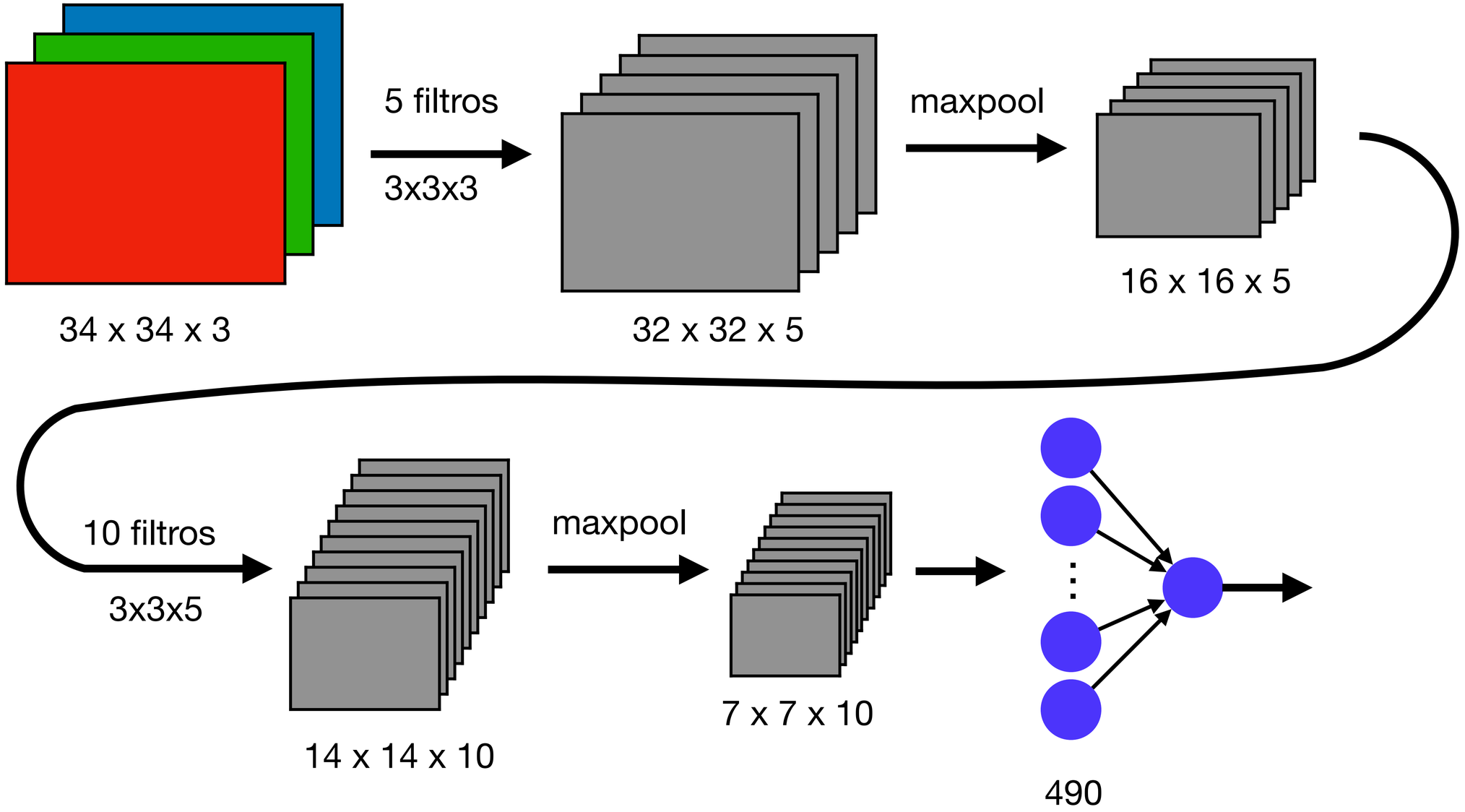
Pooling

A operação de pooling é usada para aumentar a velocidade e diminuir a quantidade de memória usada

Ela também realça características encontradas pelos filtros

Max Pooling




5 filtros
3x3x3
maxpool
34 x 34 x 3
32 x 32 x 5
16 x 16 x 5
10 filtros
3x3x5
maxpool
14 x 14 x 10
7 x 7 x 10
490

Redes Neurais Convolucionais

Nas Redes Neurais Convolucionais é muito comum usarmos o padrão:

Convolução ➡ pooling ➡ convolução ➡ pooling ➡ …
➡ convolução ➡ pooling ➡ camada totalmente conectada

Redes Neurais Convolucionais

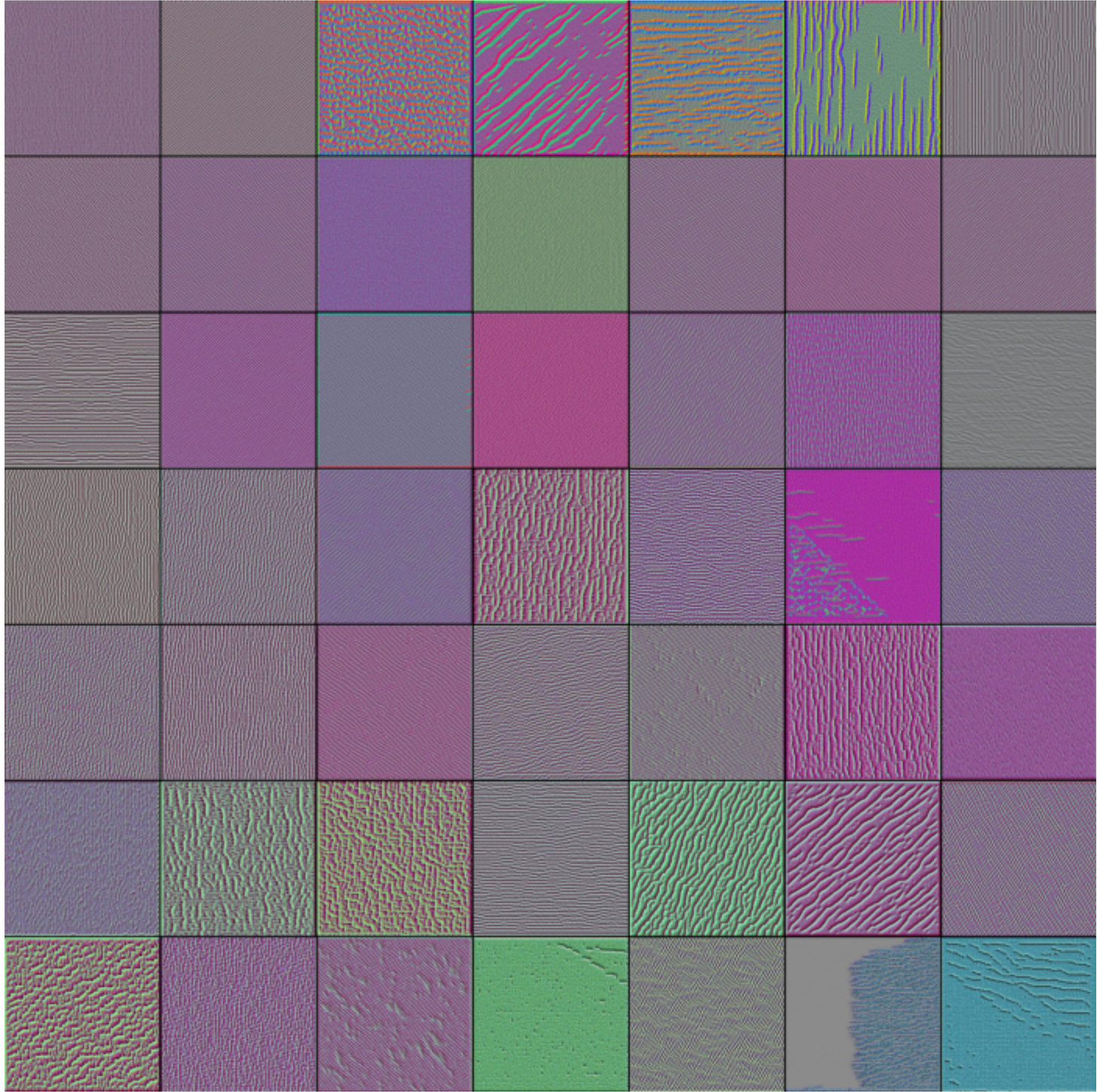
As Redes Neurais Convolucionais, ao longo de suas camadas, vão aprendendo a detectar padrões cada vez mais complexos

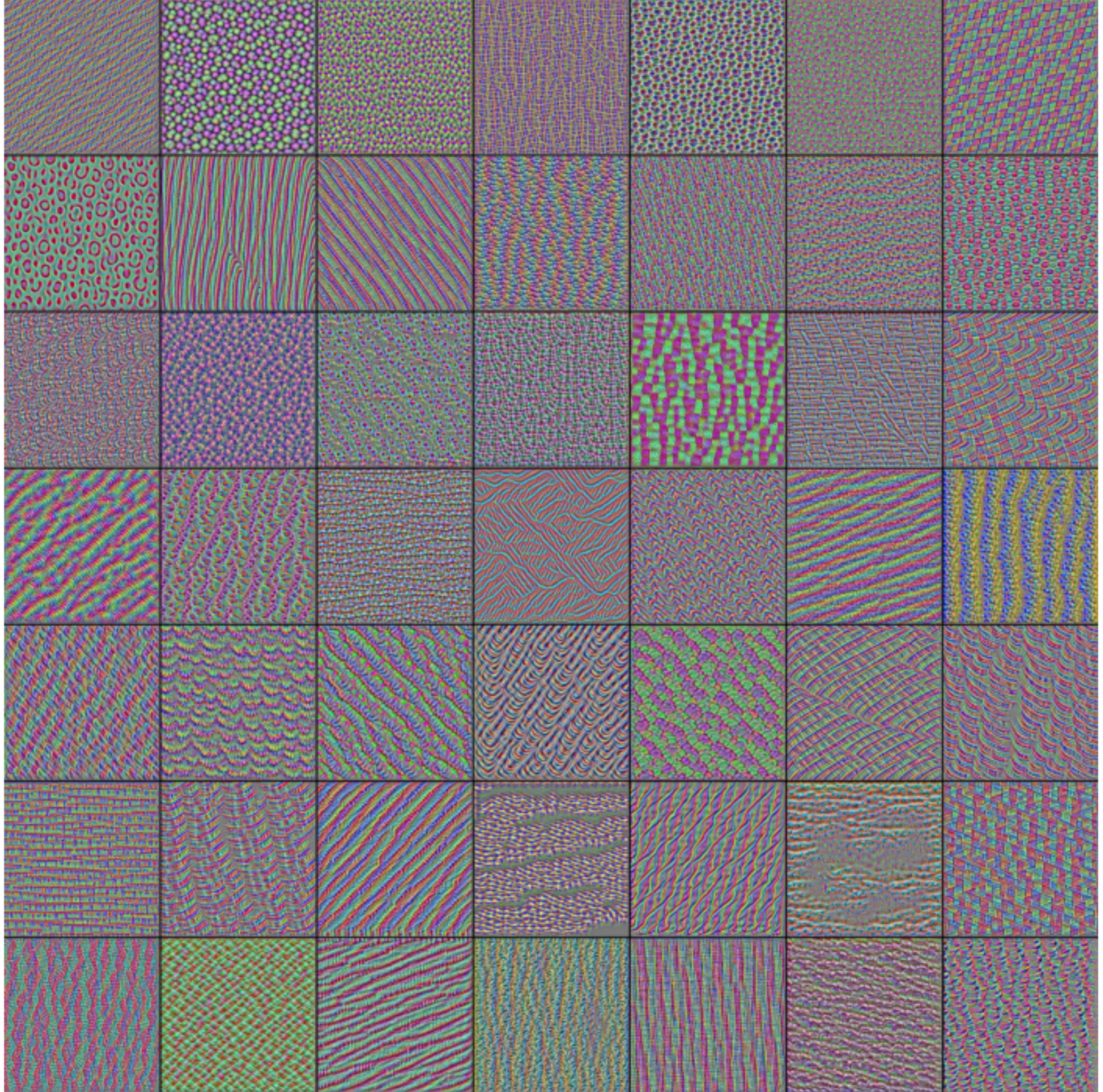
É possível visualizar esses padrões através da criação de imagens artificialmente para maximizar uma das saídas de uma determinada camada

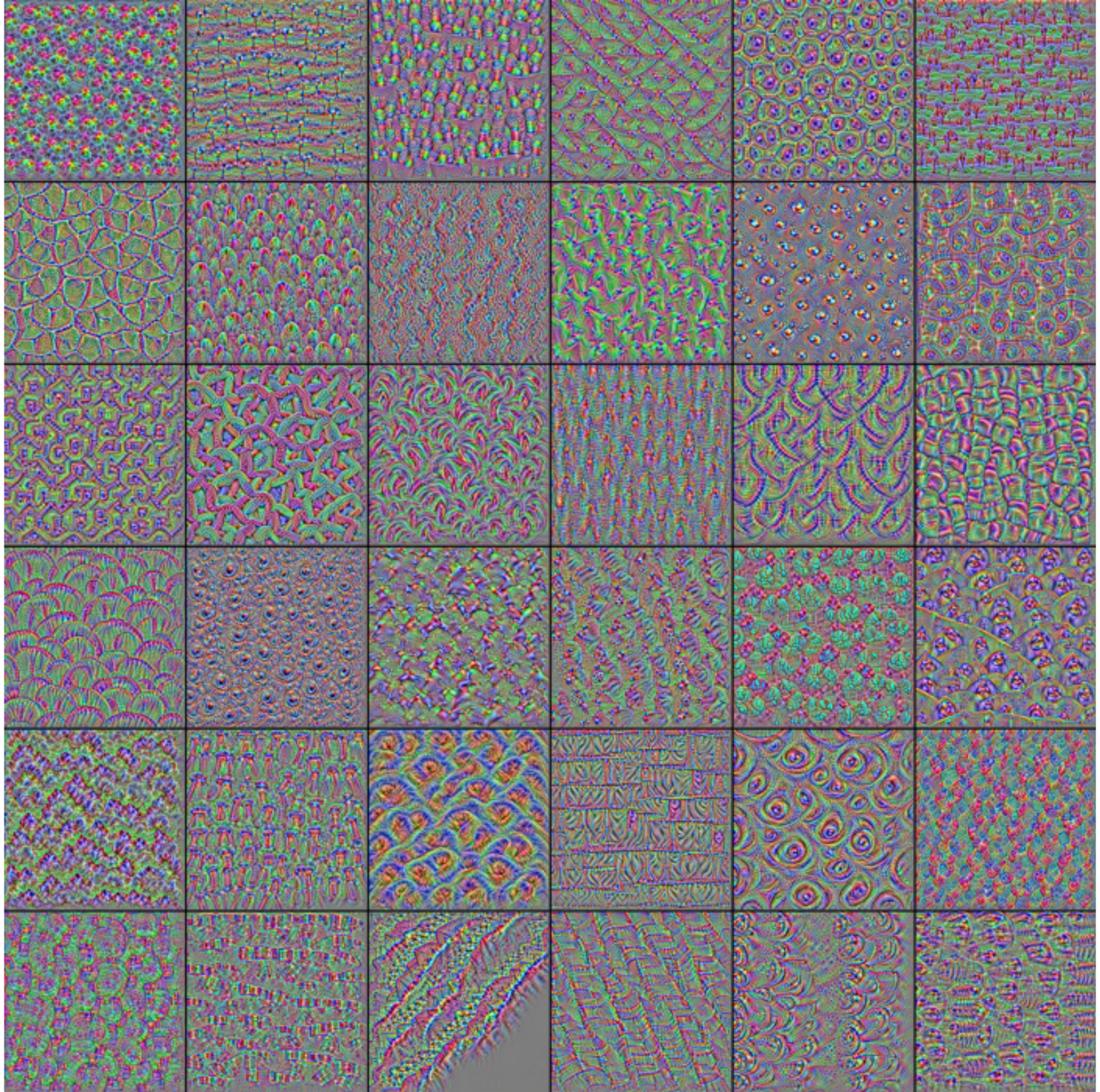
Redes Neurais Convolucionais

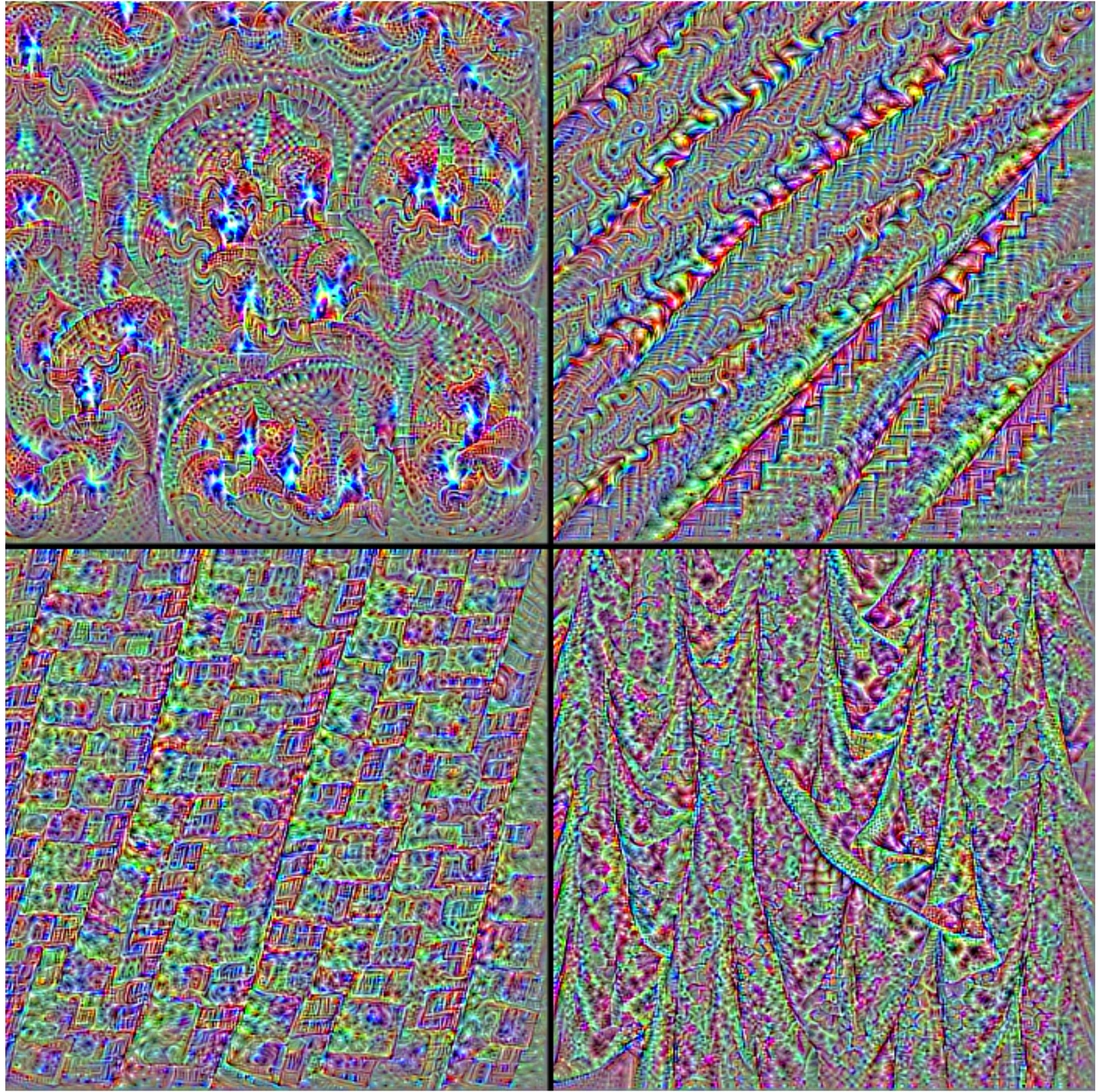
A seguir veremos algumas imagens que mostram os padrões aprendidos pelo modelo VGG-16

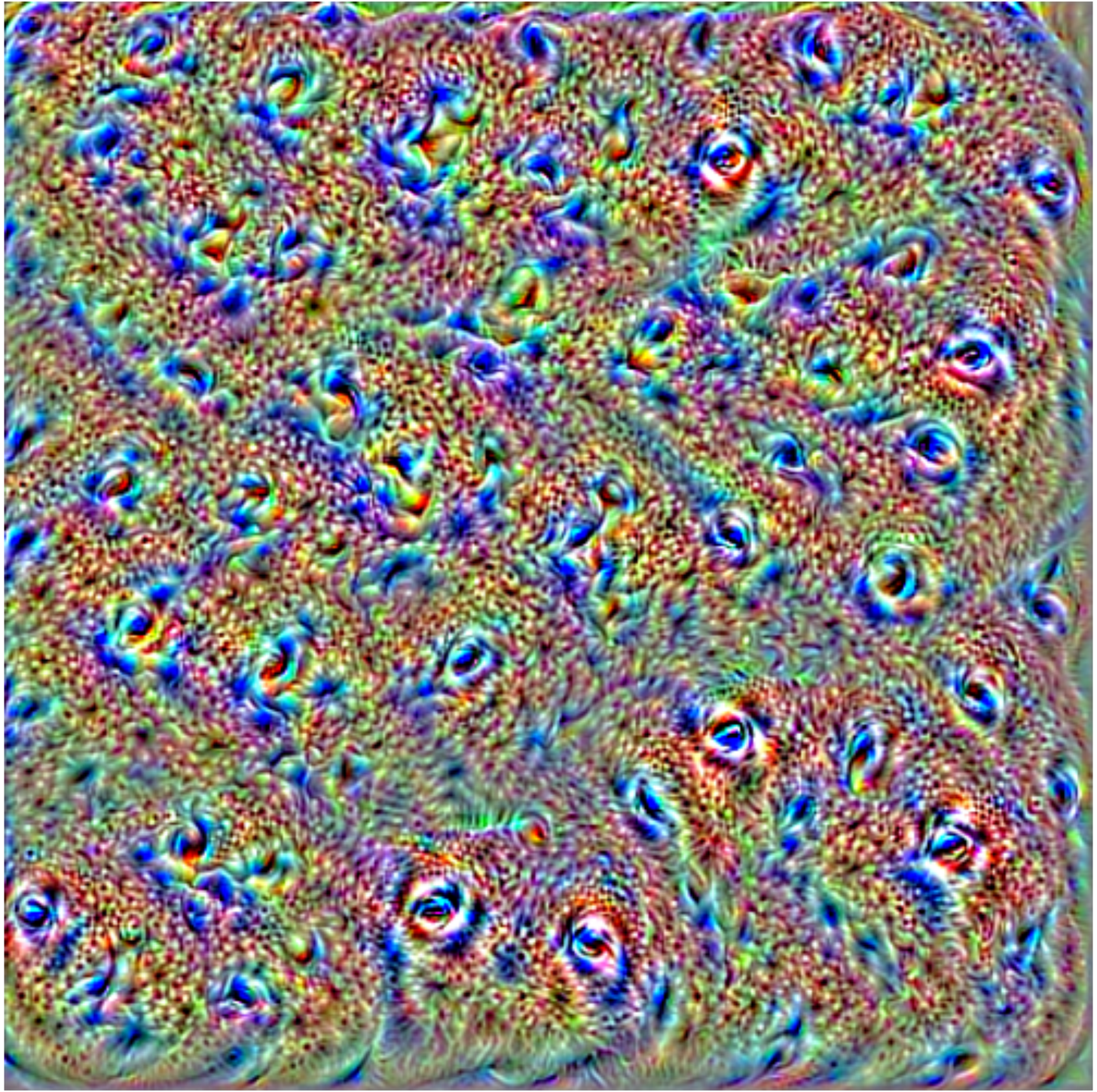
Fonte: https://towardsdatascience.com/feature-visualization-on-convolutional-neural-networks-keras-5561a116d1af











```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10))

model.compile(optimizer='adam', loss="categorical_crossentropy", metrics=['accuracy'])

history = model.fit(train_images, train_labels, epochs=10,
                    validation_data=(test_images, test_labels))

test_loss, test_acc = model.evaluate(test_images, test_labels)
```

MUNDO BANCÁRIO E IA

BATE-PAPO COM RYCHARD GUEDES

Cientista de Dados no Itaú e Professor na Let's Code, Rychard é Especialista em Ciência de Dados pelo ITA e Engenheiro Eletricista pelo IFPB. Sua trilha em dados teve início na Austrália durante estágio no NICTA. Depois disso, estagiou no setor de Gestão da Informação no Tribunal de Contas do Estado da Paraíba, apoiando as decisões da auditoria utilizando dados. Em 2018, ingressou no Itaú como Cientista de Dados, trabalhando com modelagem de crédito.

@aprendizagemdemaquina