

S+ || -T

FIAP

# PYTHON JOURNEY

MACHINE & DEEP LEARNING

# ANA RAQUEL

## PROFESSORA

- Tecnólogo em banco de dados pela faculdade FIAP.
- MBA em inteligência artificial pela FIAP.
- Mais de 8 anos de experiência como profissional na área de dados tendo atuado em diversos projetos de Banco de Dados, BI, Analytics e Data Science.
- Cientista de dados na FIAP.

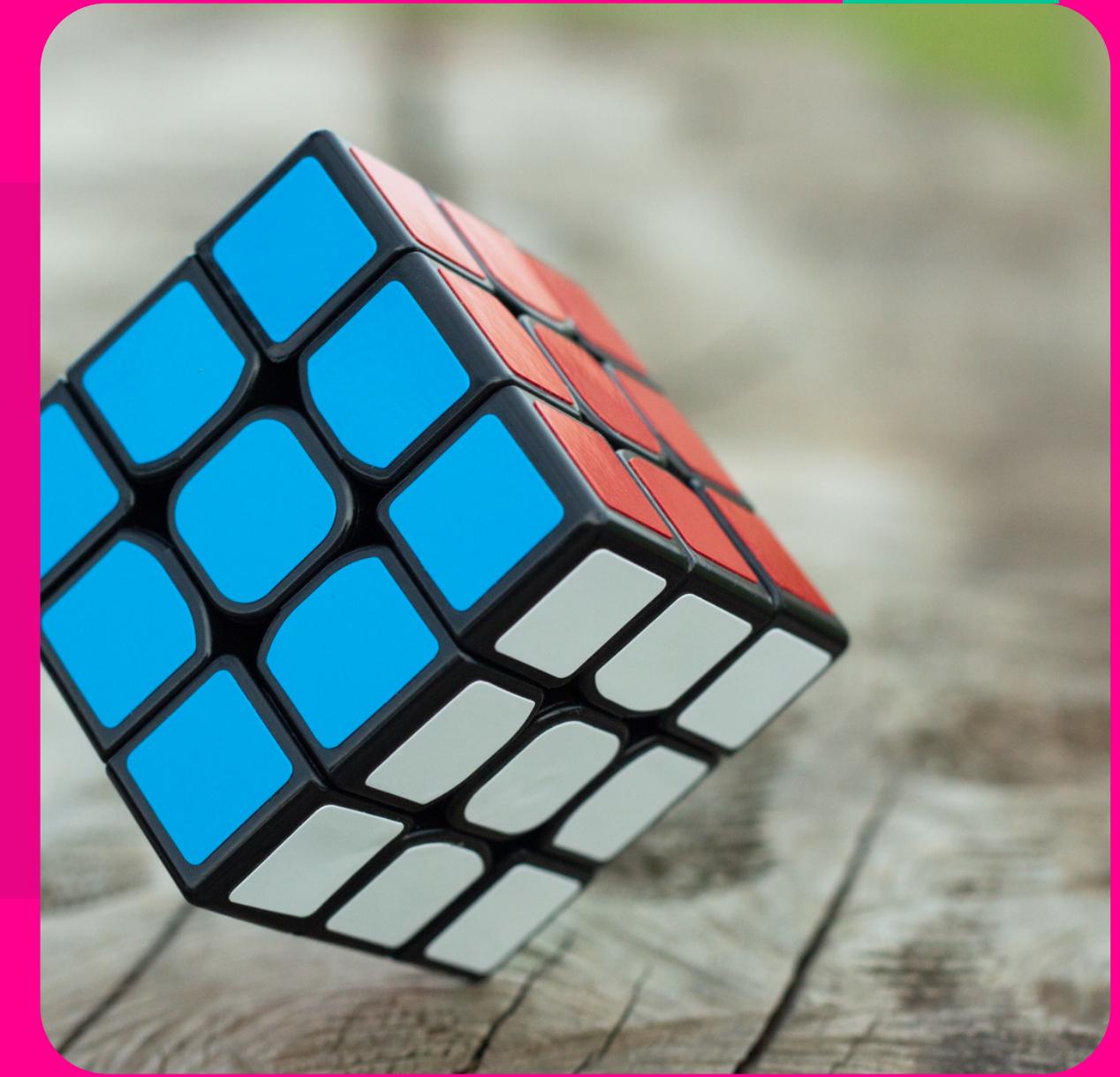


# REDES NEURAIS CONVOLUCIONAIS

# DEEP LEARNING

# MAS O QUE É UMA REDE NEURAL CONVOLUCIONAL?

Diferente da rede neural de multcamads, na rede neural convolucional os neurônios **não são totalmente conectados**. Cada neurônio da rede aprende características diferentes.



## CLASSIFICAÇÃO DE IMAGENS

**MAS POR QUE CADA  
NEURÔNIO APRENDE  
CARACTERÍSTICAS  
DIFERENTES?**

**MAS POR QUE CADA  
NEURÔNIO APRENDE  
CARACTERÍSTICAS  
DIFERENTES?**

**VAMOS ENTENDER COMO FUNCIONA O  
CÉREBRO...**

# LABRADOR OU FRANGO FRITO?



CÃO OU  
ROSCA?



# SHEEPDOG OU ESFREGÃO?



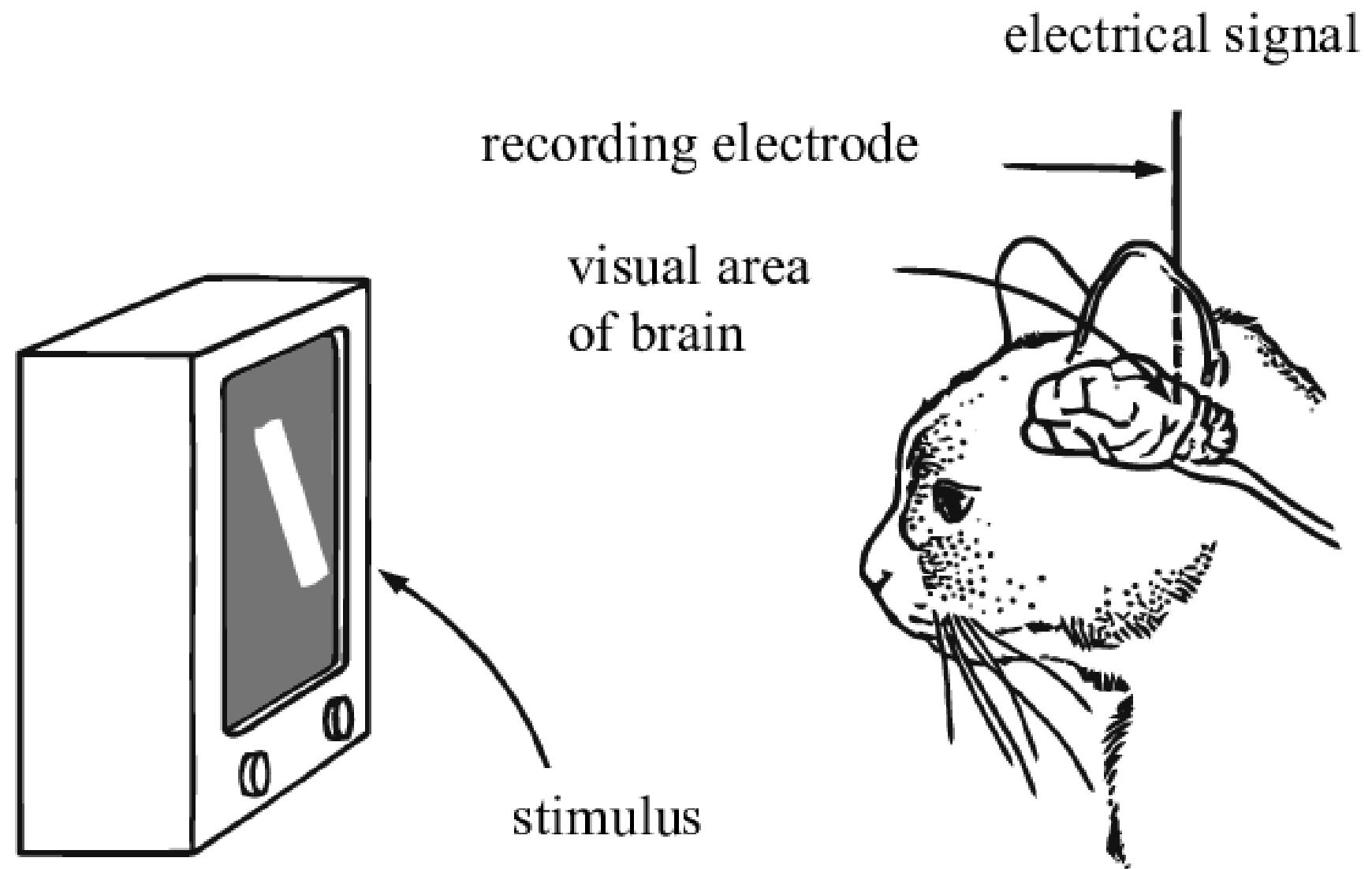


MAÇA OU  
CORUJA?



**FRANGO CRU  
OU DONALD  
TRUMP?**

**COMO IDENTIFICAMOS QUE  
AS IMAGENS SÃO  
DIFERENTES?**

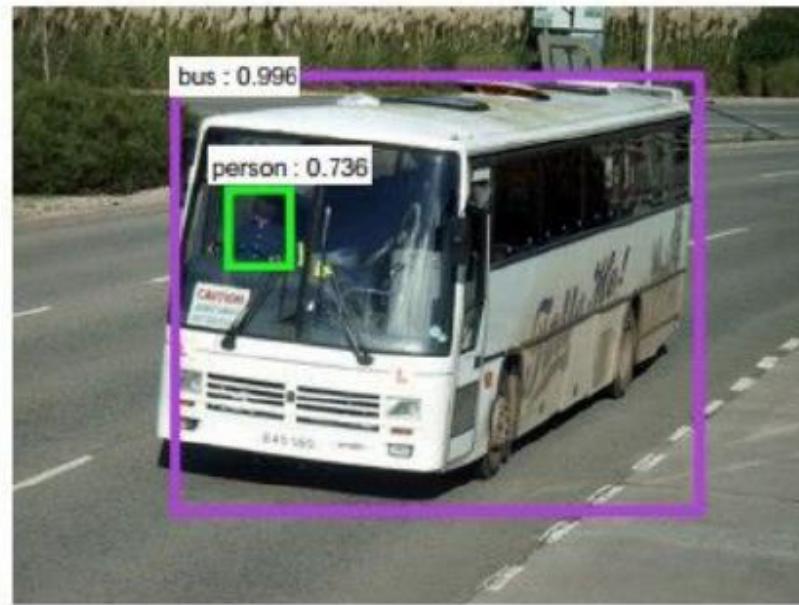
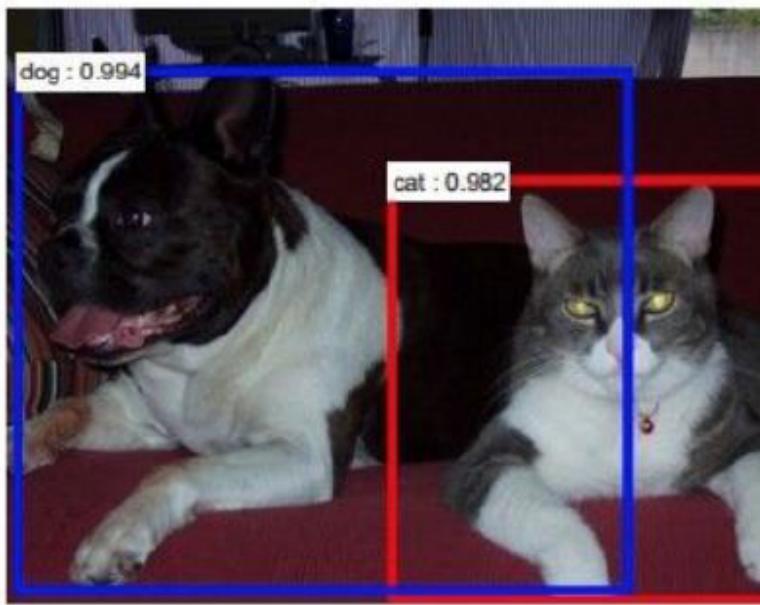
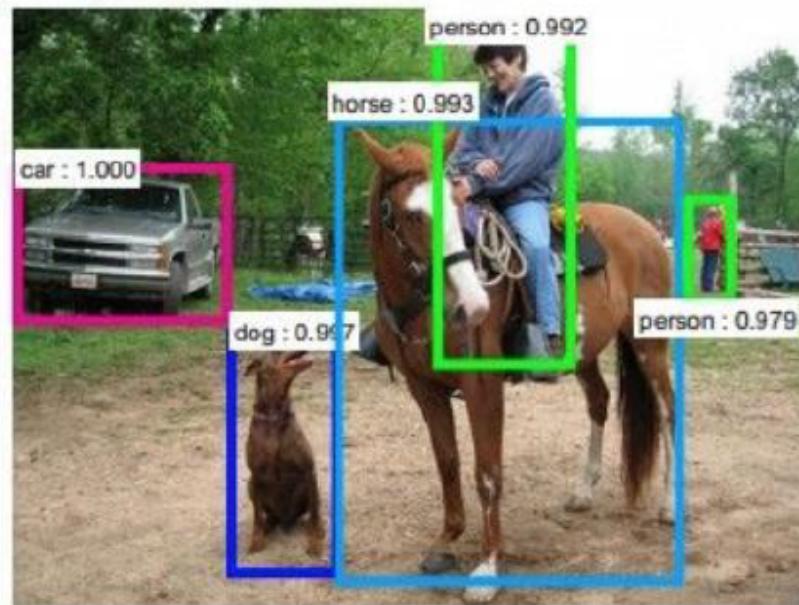


Hubel e Wiesel descobriram que os neurônios do córtex visual de um gato **são ativados juntos quando são expostos a algumas linhas ou curvas.**

# CONCLUÍMOS QUE...

Inspirado no córtex visual, saber **distinguir** diferentes imagens é possível quando **identificamos e comparamos diferentes características em cada ponto específico na imagem.**

# REDES NEURAIS CONVOLUCIONAIS



```
gpu\Py\35\tensorflow\core\common_runtime\gpu\gpu_device.cc:976] DMA: 0
2017-08-31 17:28:22.422919: I C:\tf_jenkins\home\workspace\rel-win\windows-gpu\Py\35\tensorflow\core\common_runtime\gpu\gpu_device.cc:986] O_
2017-08-31 17:28:22.423348: I C:\tf_jenkins\home\workspace\rel-win\windows-gpu\Py\35\tensorflow\core\common_runtime\gpu\gpu_device.cc:1045] Creating TensorFlow device (/gpu:0) -> (device: 0, name: GeForce GTX 1080 Ti, pci bus id: 0000:02:00.0)

vehicle_detector.py - J:\Object-detect\models\object_detection\vehicle_detector.py (3.5.0)
File Edit Format Run Options Window Help
B size, in inches, of the output images.
IMAGE_SIZE = (12, 8)
with detection_graph.as_default():
    (graph=detection_graph) as sess:
        sess.resize(grab_screen(region=(0,40,1280,745)), (WIDTH,HEIGHT))
        sess.resize(grab_screen(region=(0,40,1280,745)), (800,450))
        cv2.cvtColor(screen, cv2.COLOR_BGR2RGB)
mensions since the model expects images to have shape: [1, None
panded = np.expand_dims(image_np, axis=0)
r = detection_graph.get_tensor_by_name('image_tensor:0')
represents a part of the image where a particular object was dection_graph.get_tensor_by_name('detection_boxes:0')
e represent how level of confidence for each of the objects.
shows on the result image, together with the class label.
tection_graph.get_tensor_by_name('detection_scores:0')
tection_graph.get_tensor_by_name('detection_classes:0')
ons = detection_graph.get_tensor_by_name('num_detections:0')
tection.
res, classes, num_detections = sess.run(
    scores, classes, num_detections),
ct=(image tensor: image_np expanded)
tion of the results of a detection.
visualize_boxes_and_labels_on_image_array(
p,
cv2(boxes),
cv2(classes).astype(np.int32),
cv2(scores),
y_index,
normalized_coordinates=True,
ickness=8)

>window', cv2.resize(image_np, (800,450)))
Key(25) & 0xFF == ord('q'):
troyAllWindows()
```

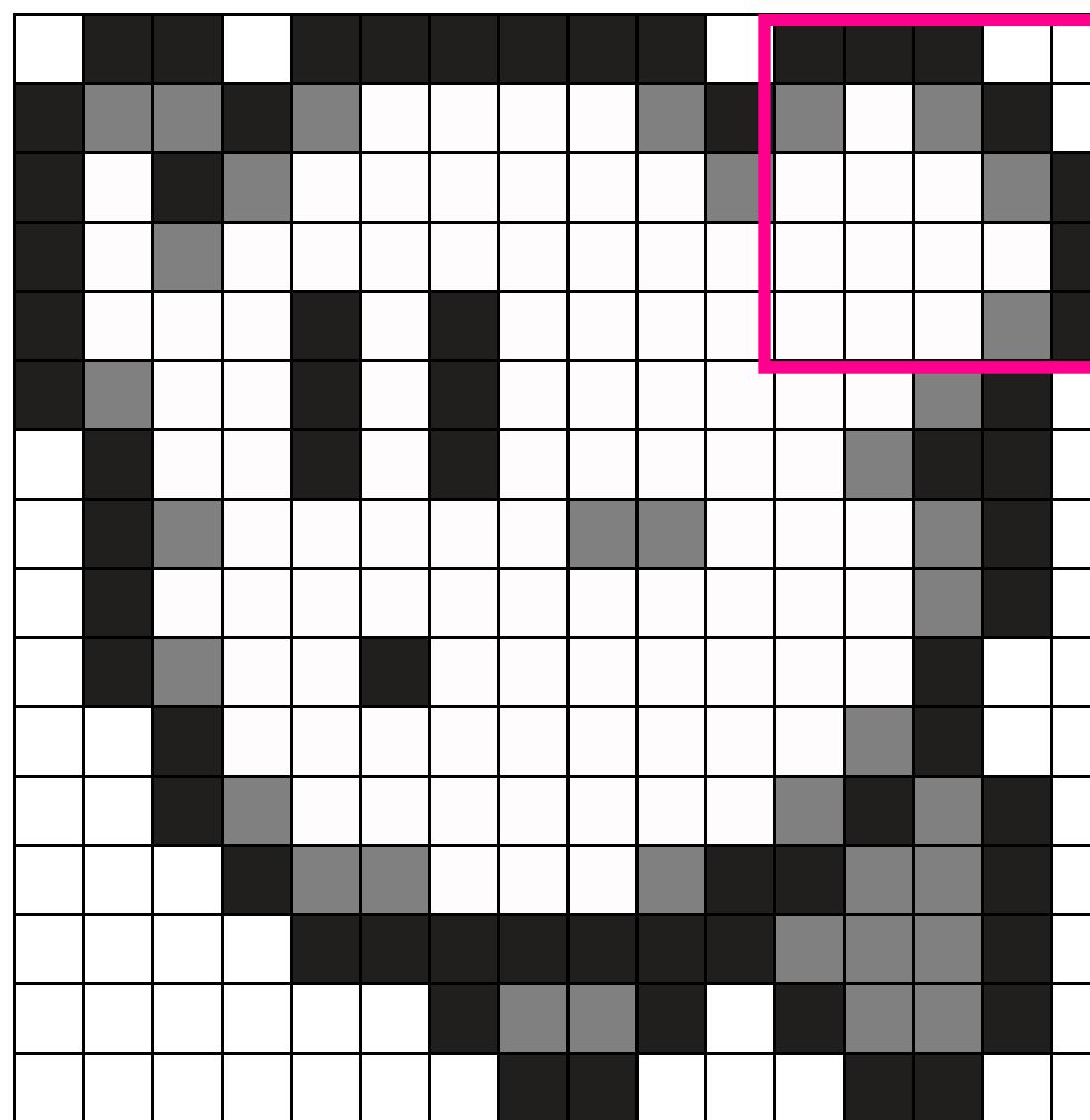
# REDES NEURAIS CONVOLUCIONAIS



# E COMO O COMPUTADOR IDENTIFICA IMAGENS?

# IMAGENS PRETA E BRANCA (GRAYSCALE)

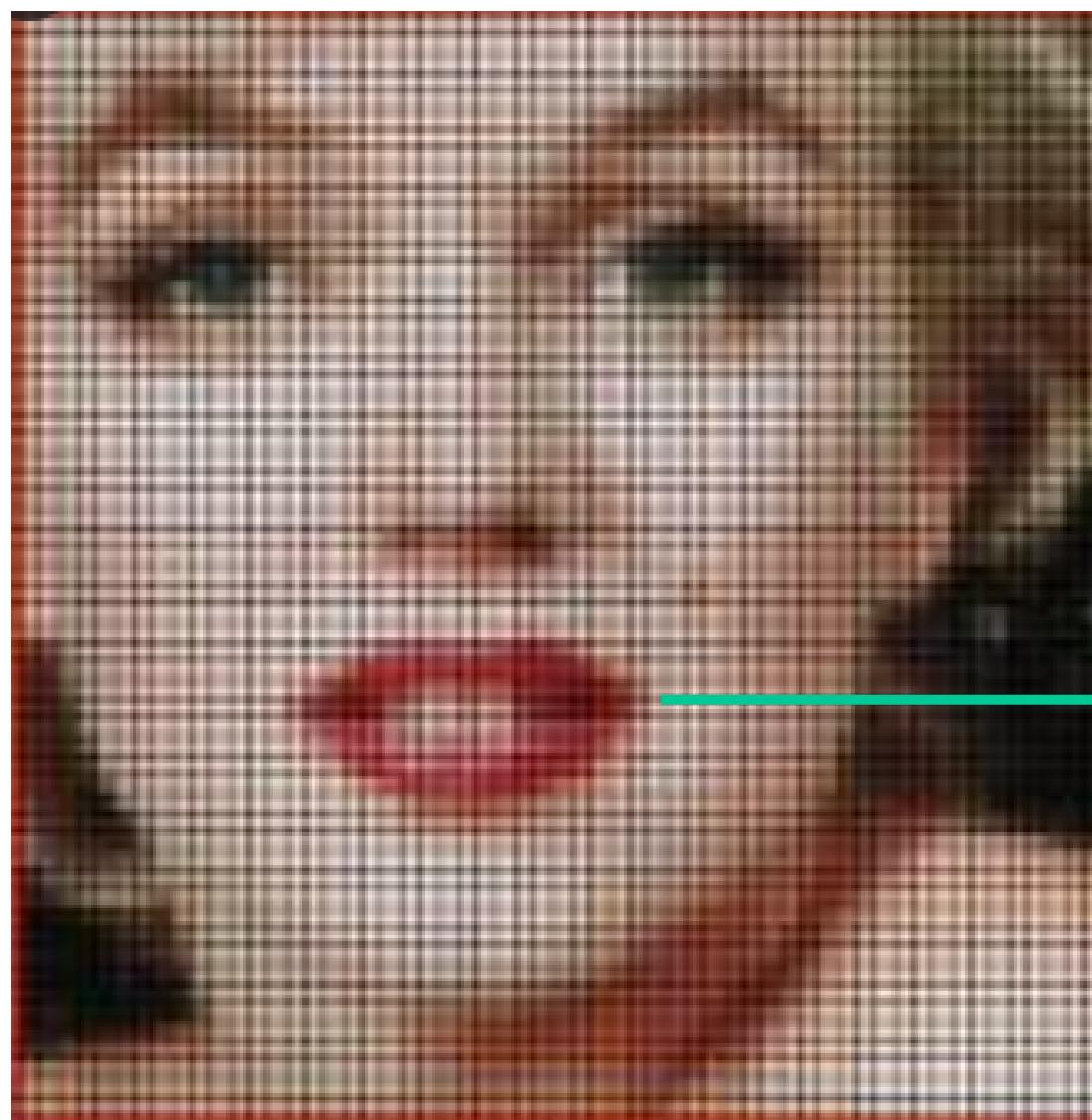
Representada por uma matriz 2D, onde cada posição da imagem é composta por elementos que variam de 0 (preto) até 255 (branco).



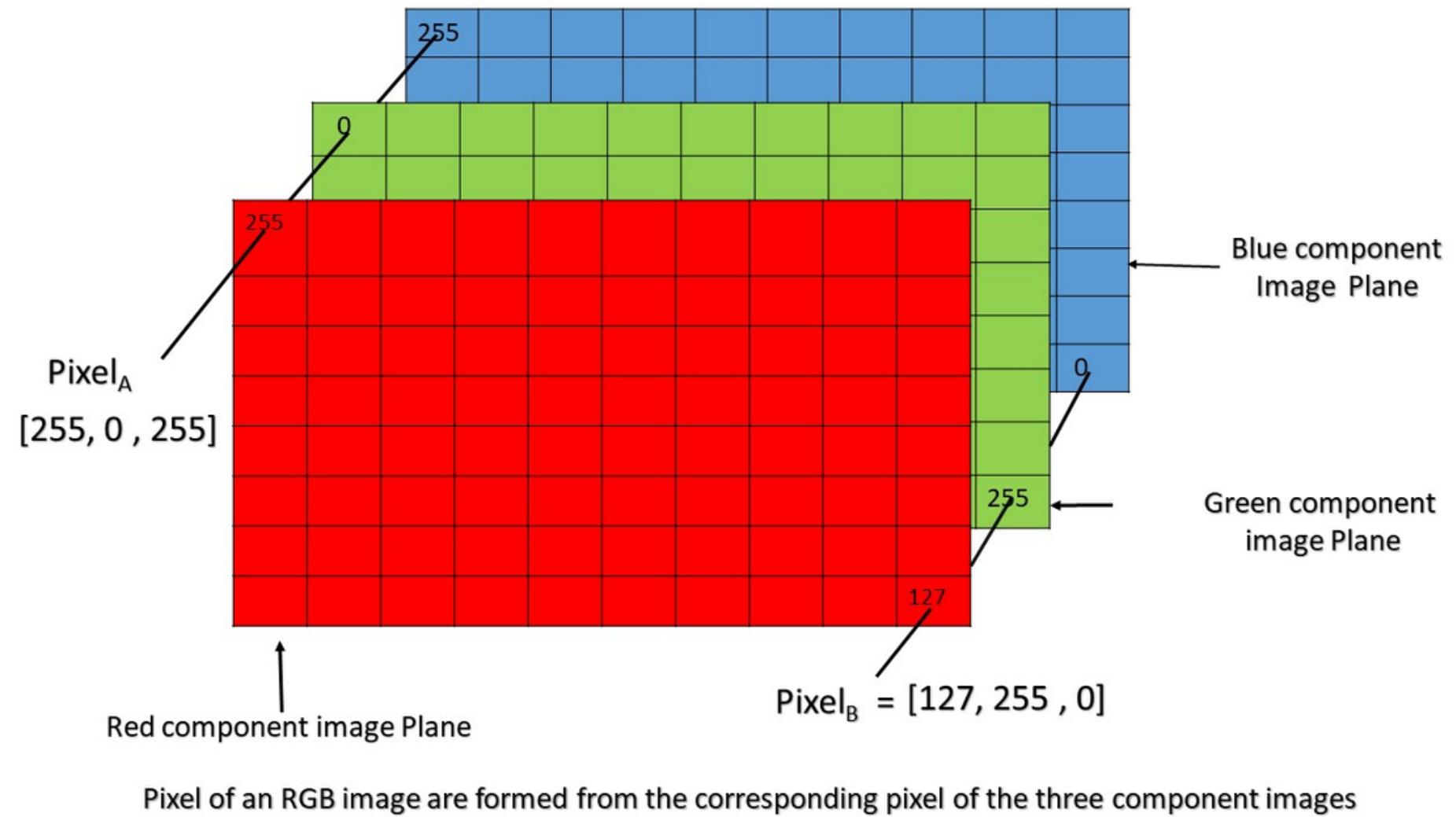
0	0	0	255	255
170	255	170	0	255
255	255	255	170	0
255	255	255	255	0
255	255	255	170	0

# IMAGENS COLORIDA (RGB)

Representada por uma matriz 3D, combinando cores vermelho, verde e azul. O propósito principal do sistema RGB é a reprodução de cores em dispositivos eletrônicos como monitores de TV, computadores, mídias digitais, celulares em geral.

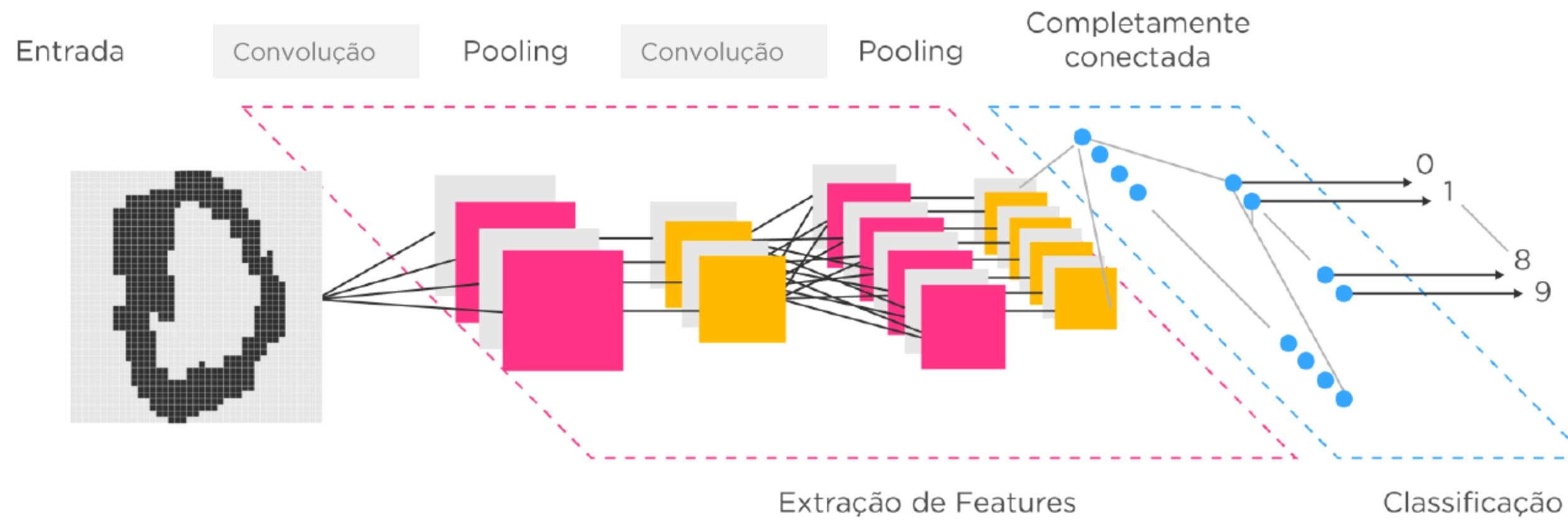


(255,0,0)



# **ARQUITETURA REDES NEURAIS CONVOLUCIONAIS**

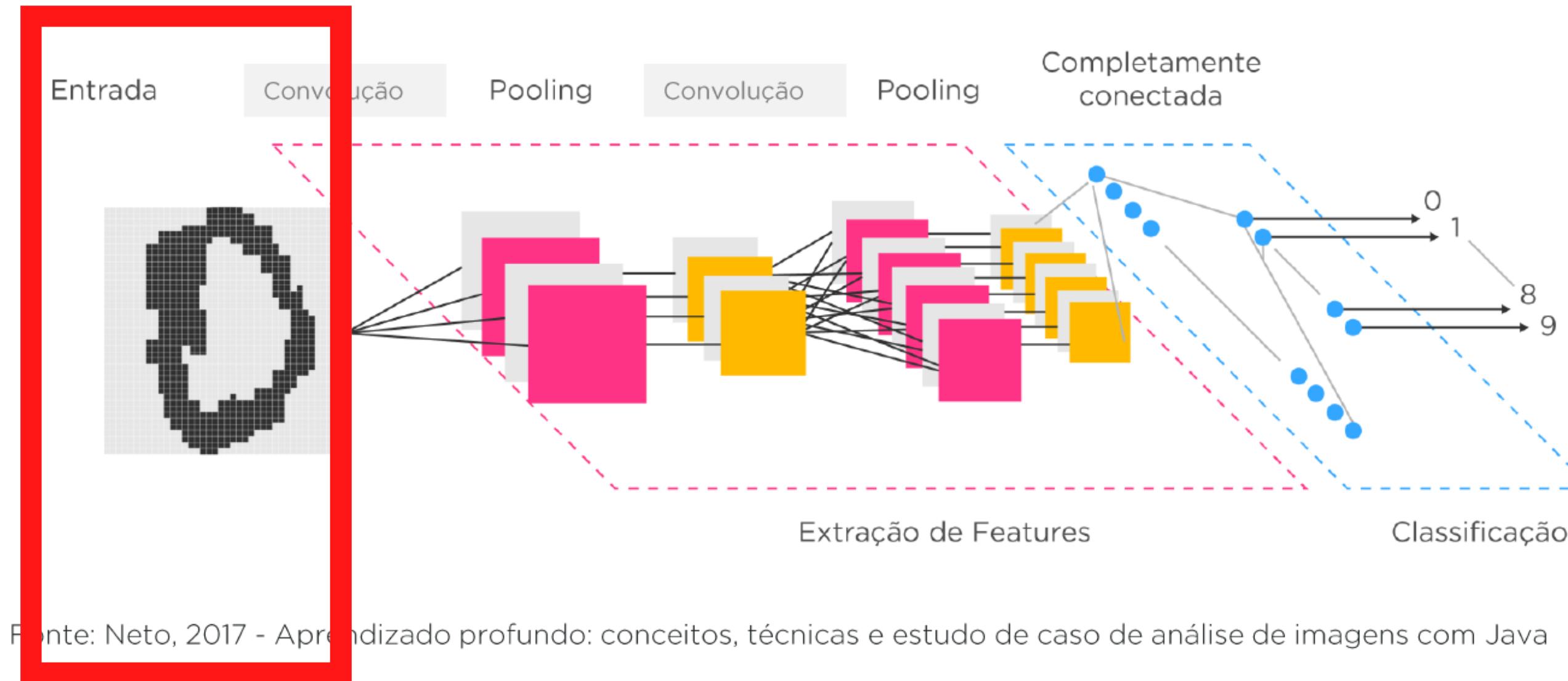
# ARQUITETURA REDES NEURAIS CONVOLUCIONAIS



Fonte: Neto, 2017 - Aprendizado profundo: conceitos, técnicas e estudo de caso de análise de imagens com Java

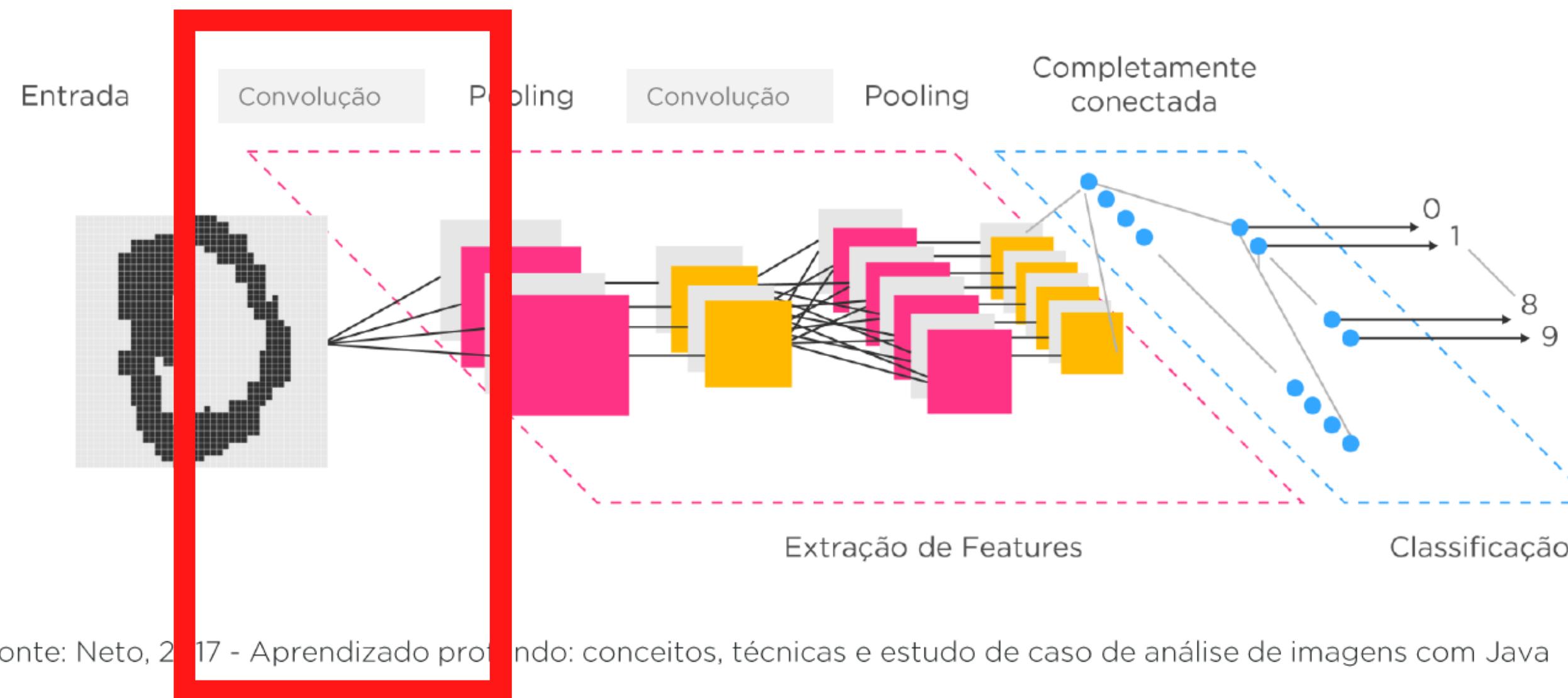
# ENTRADA:

A entrada basicamente recebe uma imagem.



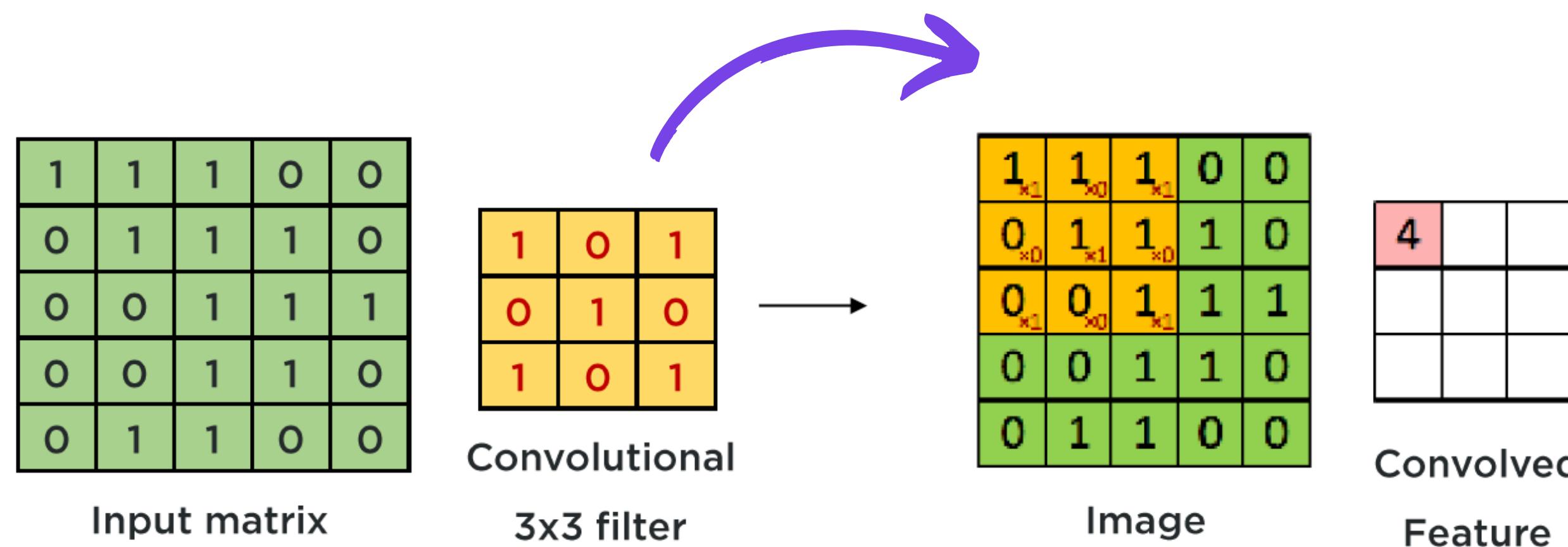
# CONVOLUÇÃO:

"Detector de atributos", essa etapa vai mapear a região de uma imagem para posteriormente termos um "mapa de atributos" (feature map). Funciona basicamente como um filtro (kernel).



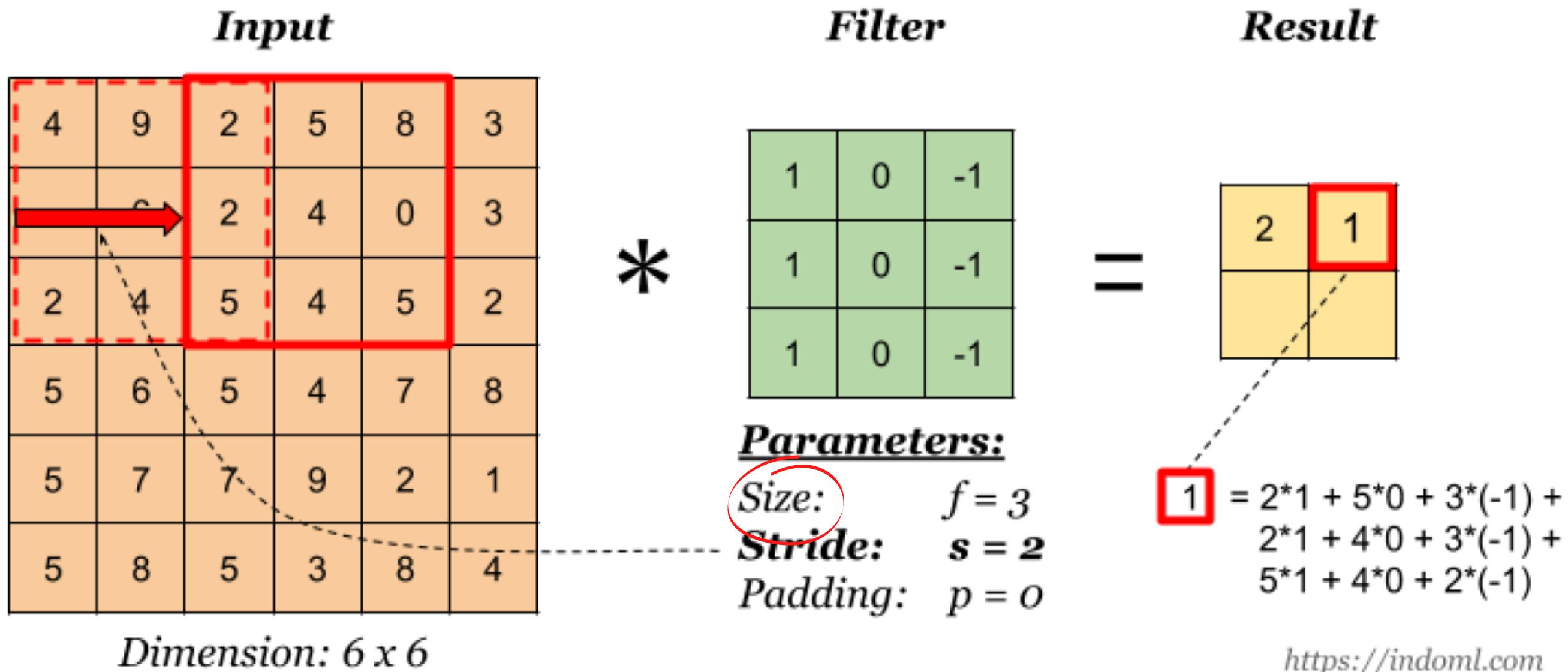
# CONVOLUÇÃO:

O filtro nada mais é do que a multiplicação de matrizes. Esse processo é realizado diversas vezes por toda a imagem recebida. Nessa etapa temos um multiplicador que se chama "**stride (deslocamento)**" (normalmente com o valor = 1) cuja principal função é realizar a transformação da imagem.



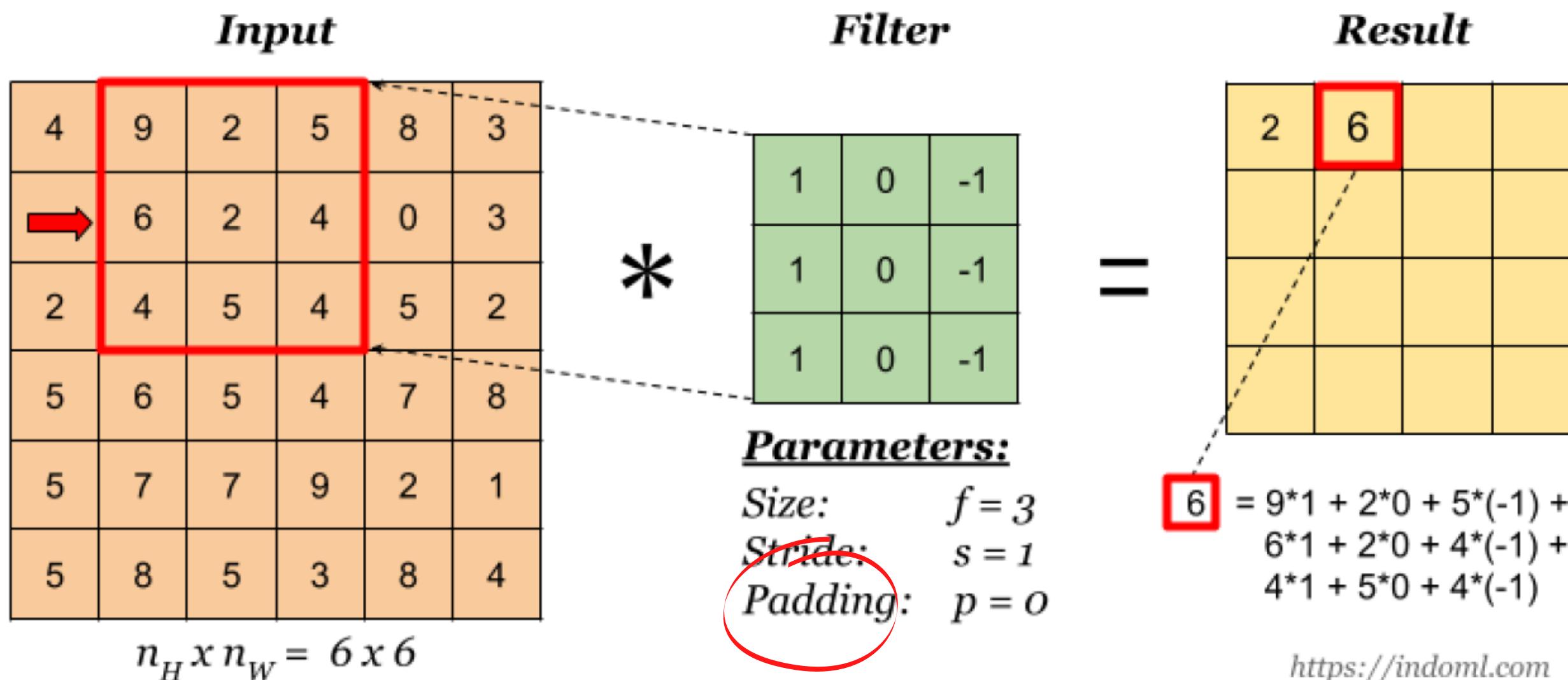
Toda essa transformação irá resultar uma imagem com uma dimensionalidade menor.

## Funcionamento do stride:



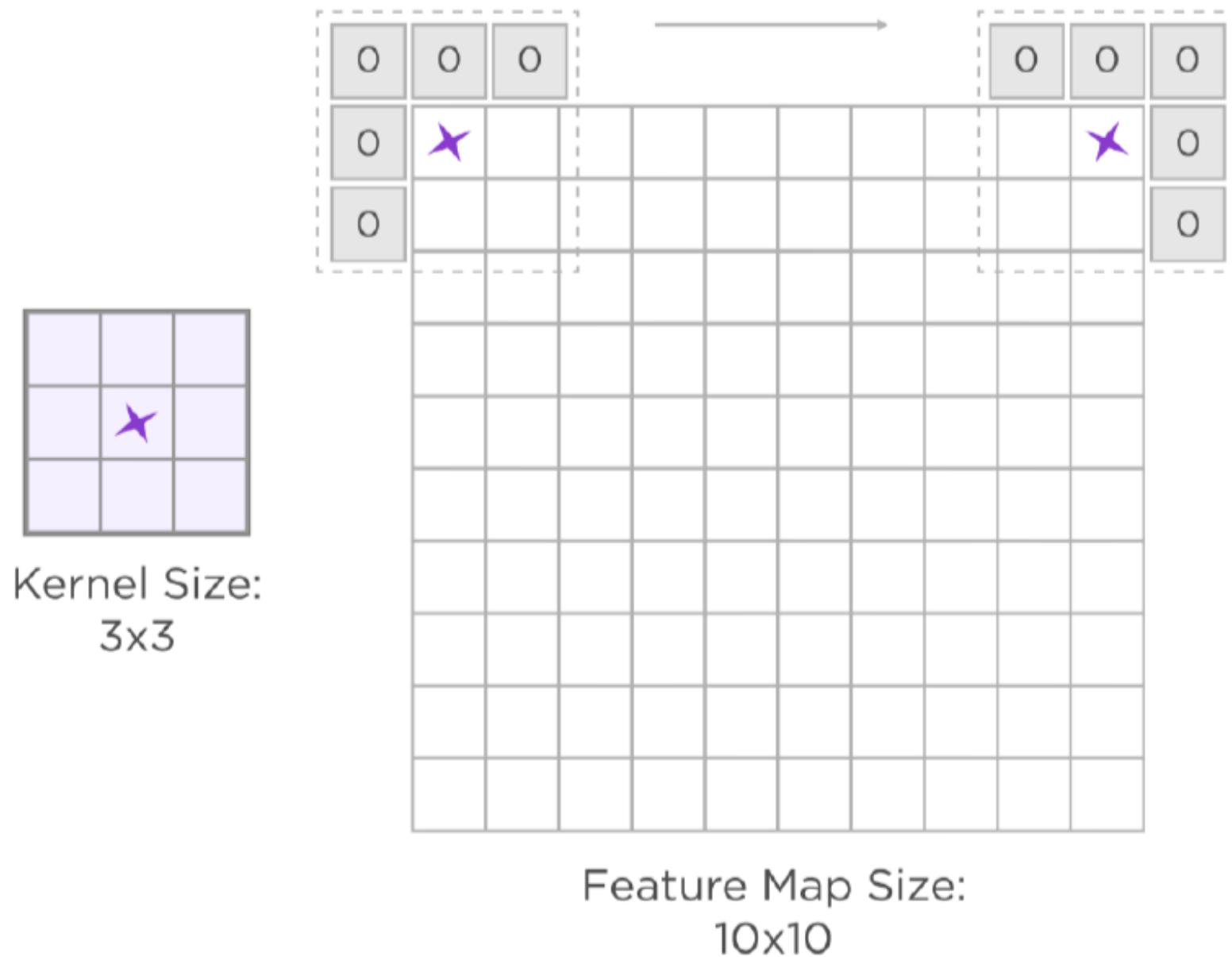
<https://indoml.com>

## Funcionamento do filtro:



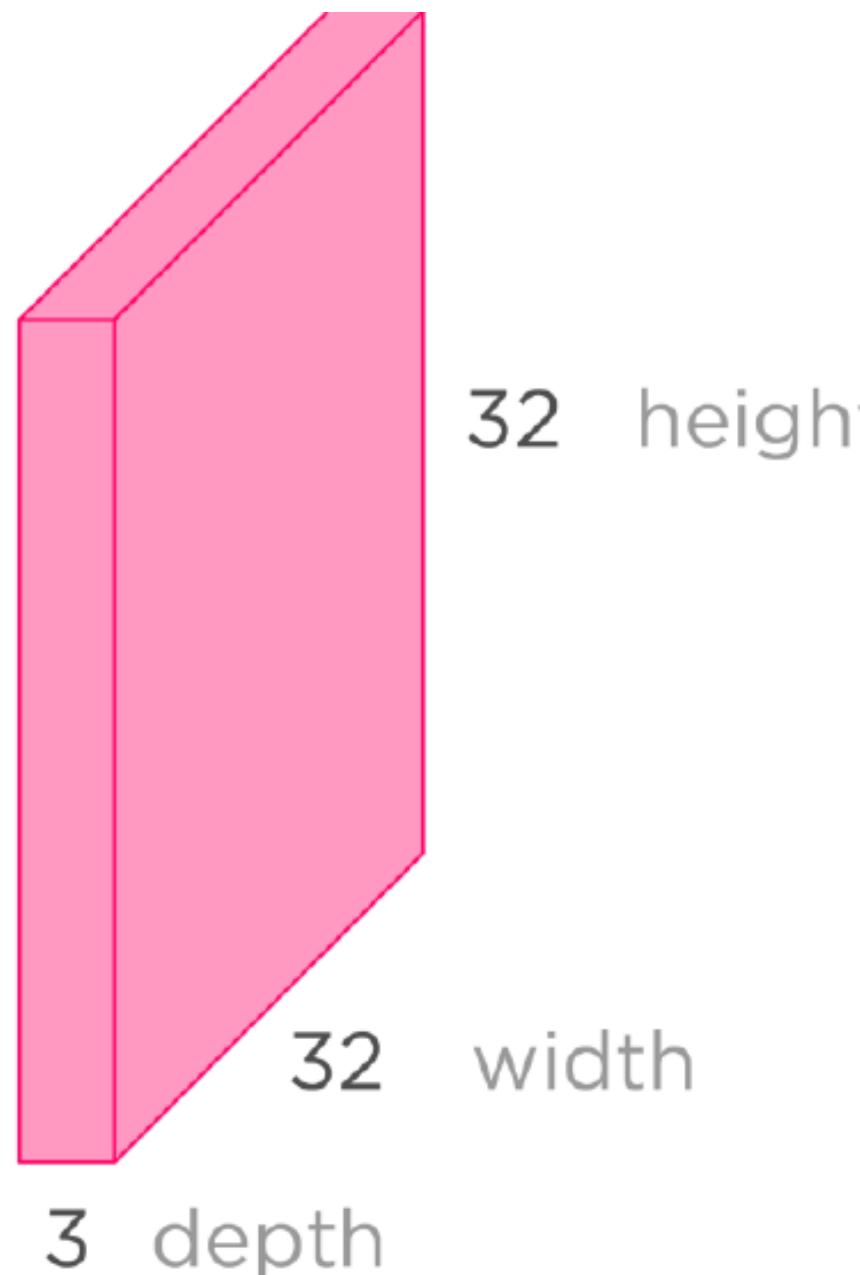
<https://indoml.com>

Muitas convoluções podem impactar na assertividade da CNN se o tamanho da imagem for muito reduzido. Para contornar esse cenário, normalmente é utilizado o conceito de **Padding** (distância entre o conteúdo de um elemento e suas bordas).



O objetivo do padding é manter a dimensionalidade da imagem.

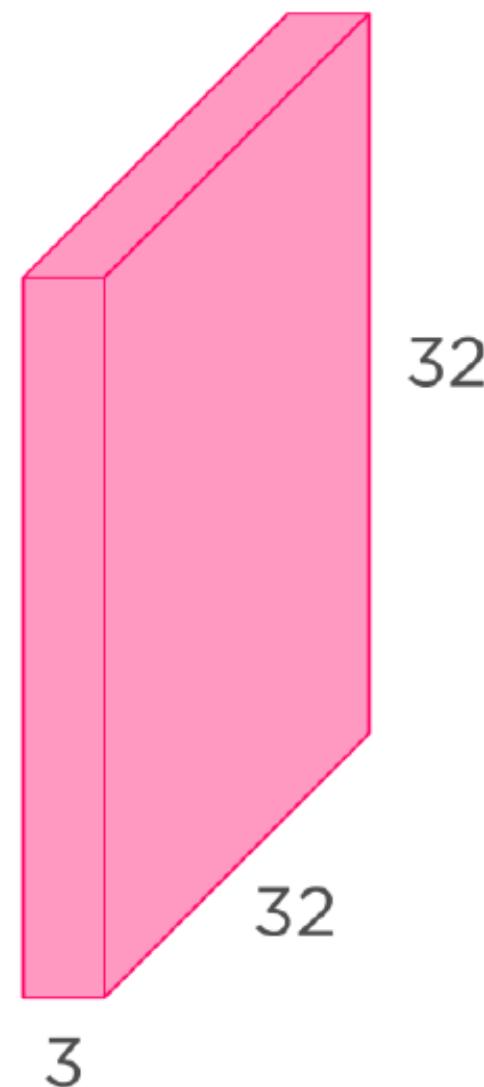
# CONVOLUÇÃO:



- Imagem  $32 \times 32 \times 3 \rightarrow$  Mantém a estrutura espacial preservada

# CONVOLUÇÃO:

32x32x3 image

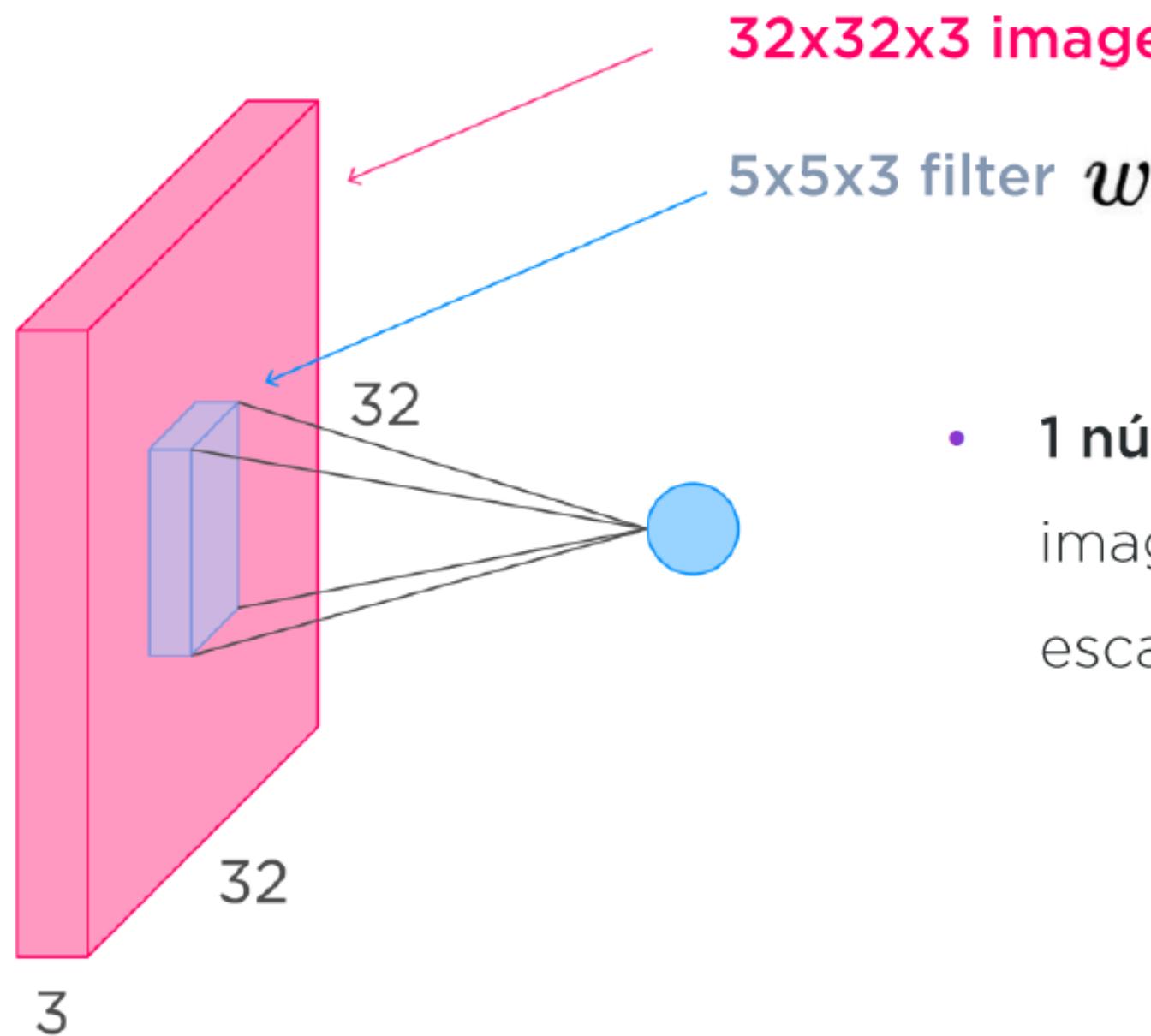


5x5x3

A 3D blue cube representing a 5x5x3 filter kernel.

- Filtro
- Convolve o filtro com a imagem.
- Por exemplo: "Se desliza a imagem espacialmente, computando produtos escalares".

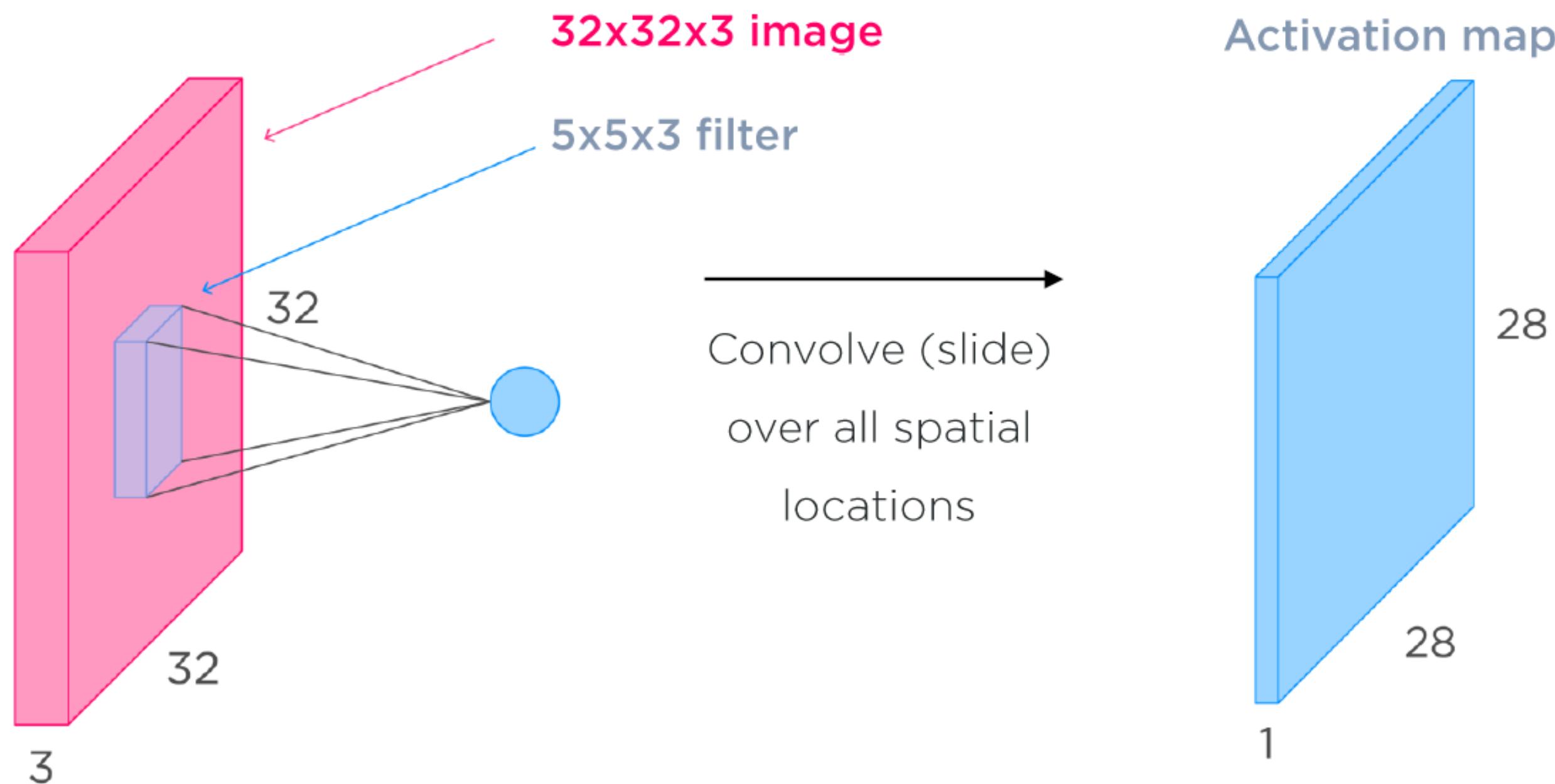
# CONVOLUÇÃO:



- **1 número:** Pedaço de  $5 \times 5 \times 3$  da imagem (ou seja,  $5 * 5 * 3 =$  produto escalar de 75 dimensões + bias)

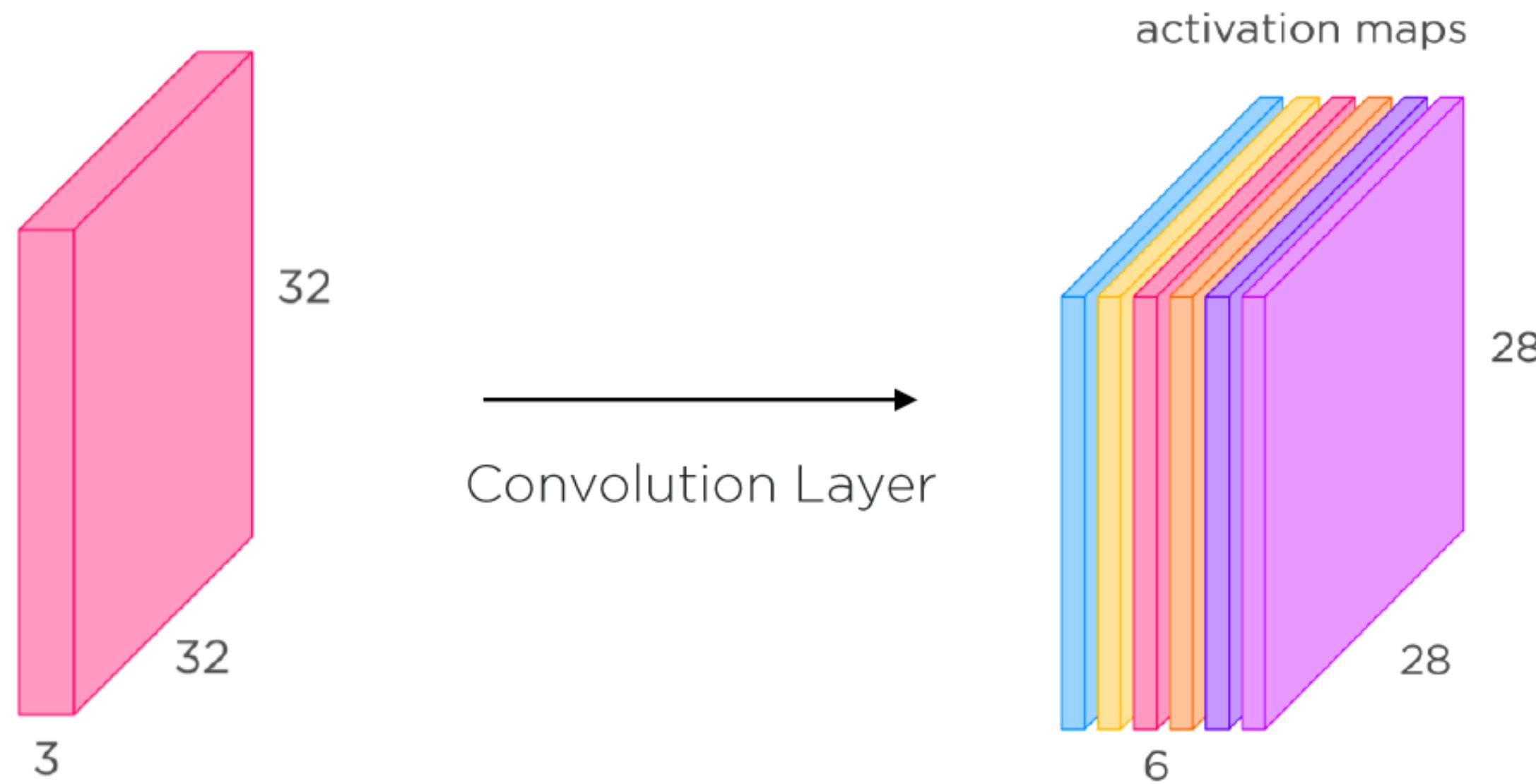
$$w^T x + b$$

# CONVOLUÇÃO:



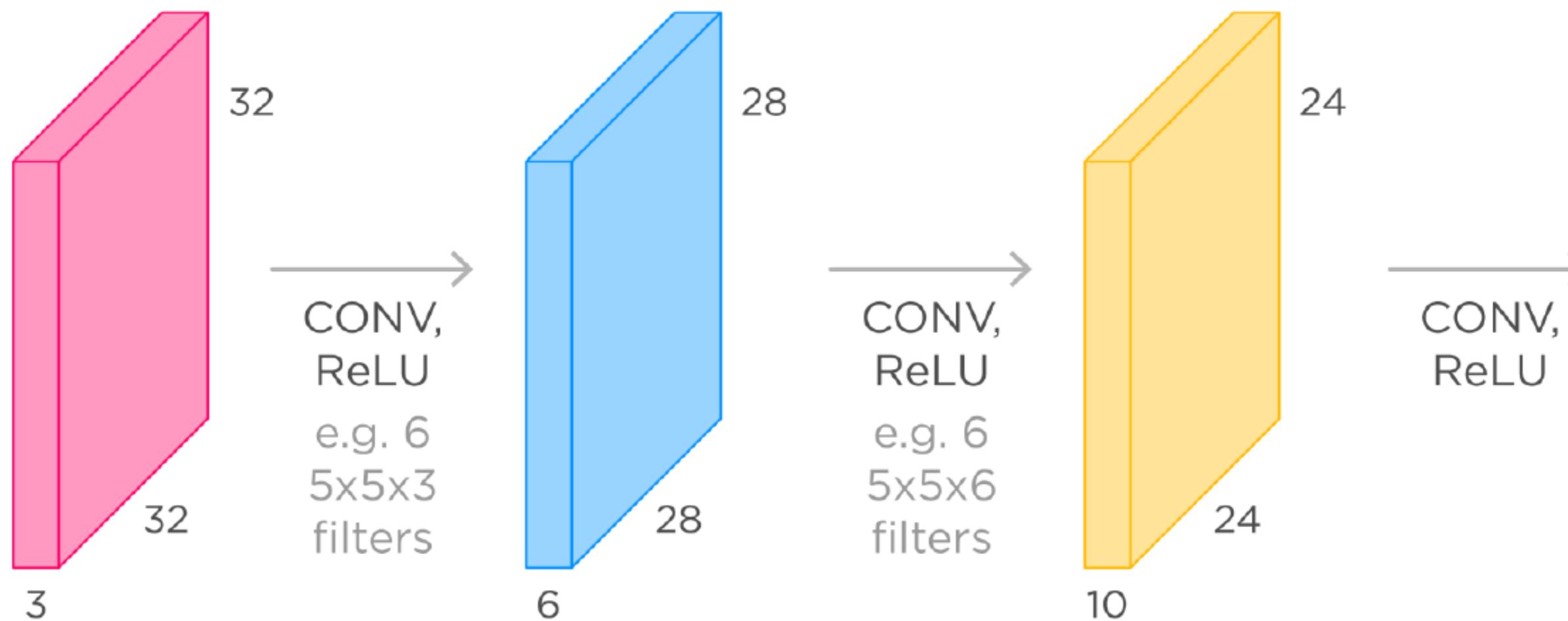
# CONVOLUÇÃO:

A cada medida que realizamos essas extrações de características das imagens, vamos adicionando novos mapas de filtros. Exemplo: 6 filtros de 5x5 teríamos 6 mapas de ativação.



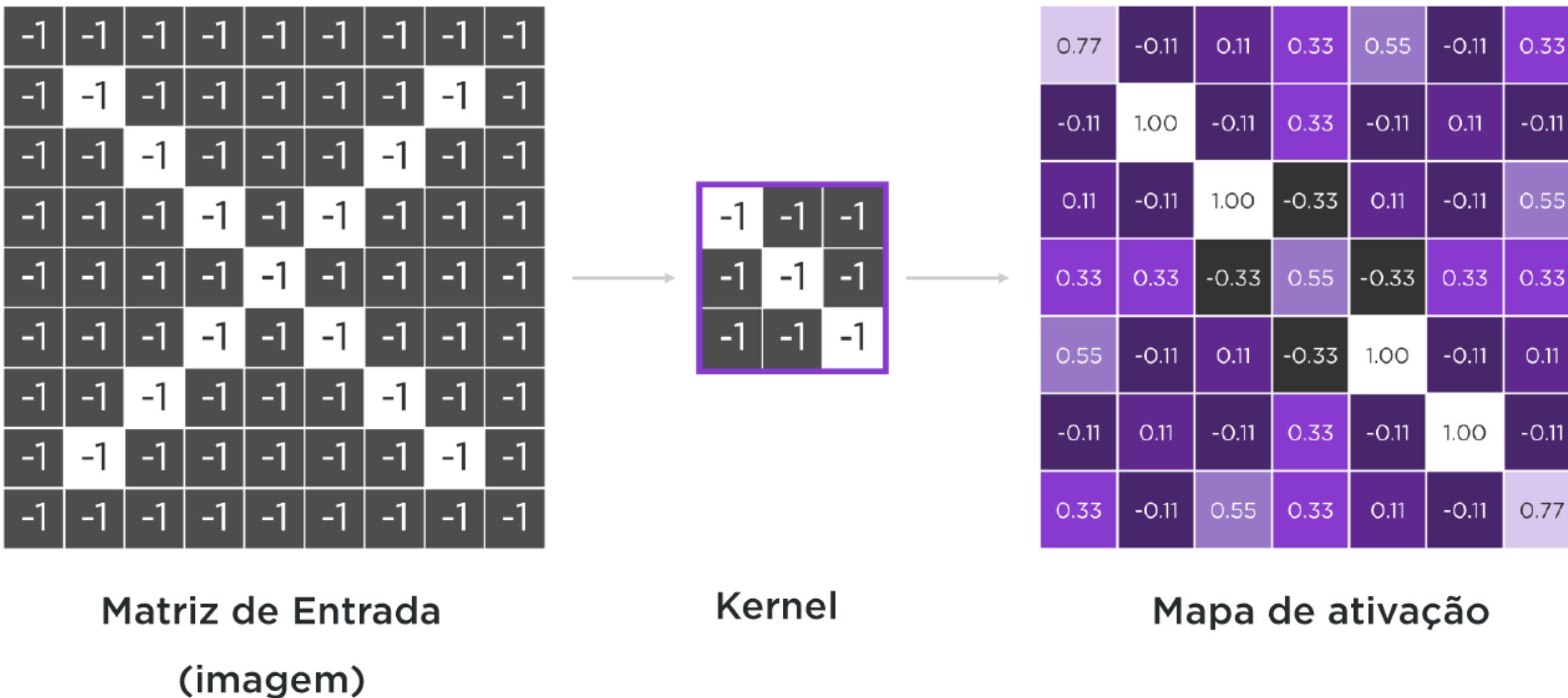
# CONVOLUÇÃO:

Entre as camadas de mapa de atributos, vamos ter uma **função de ativação** para realizar a classificação das imagens.

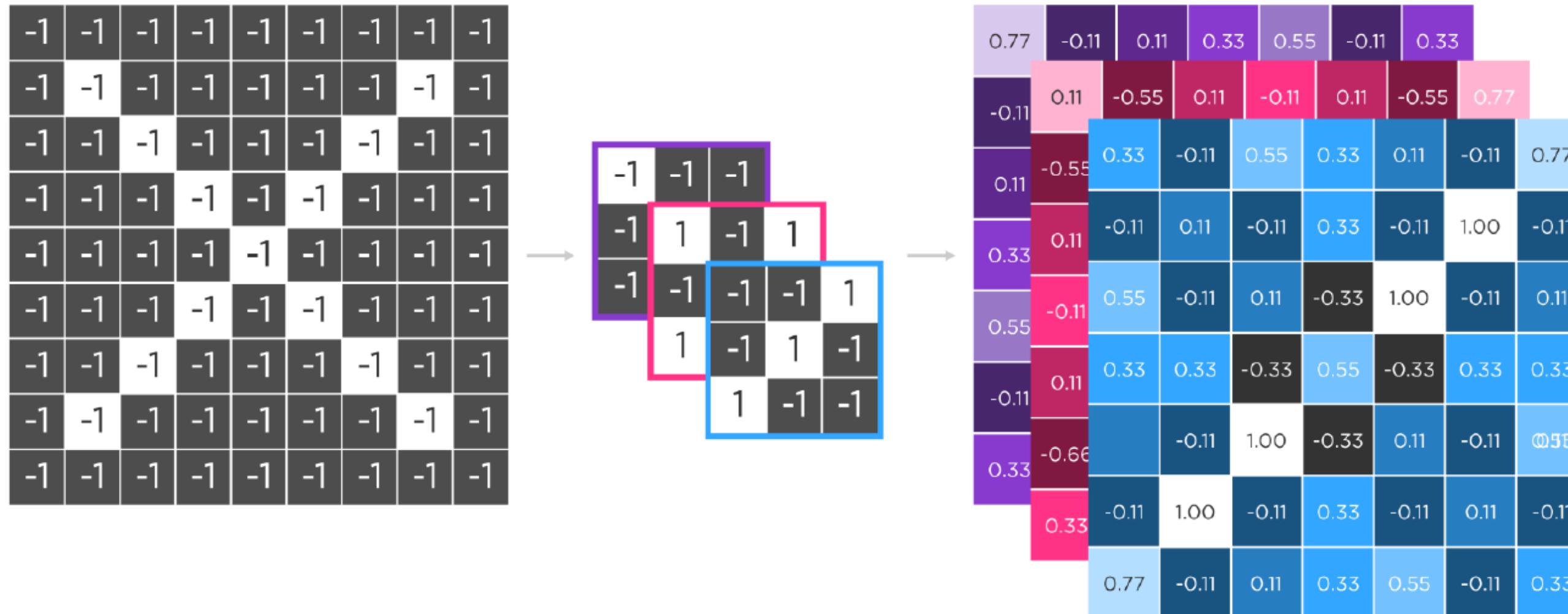


Então nossa rede se torna uma sequência de camadas de convolução, intercaladas por funções de ativação.

# CONVOLUÇÃO:

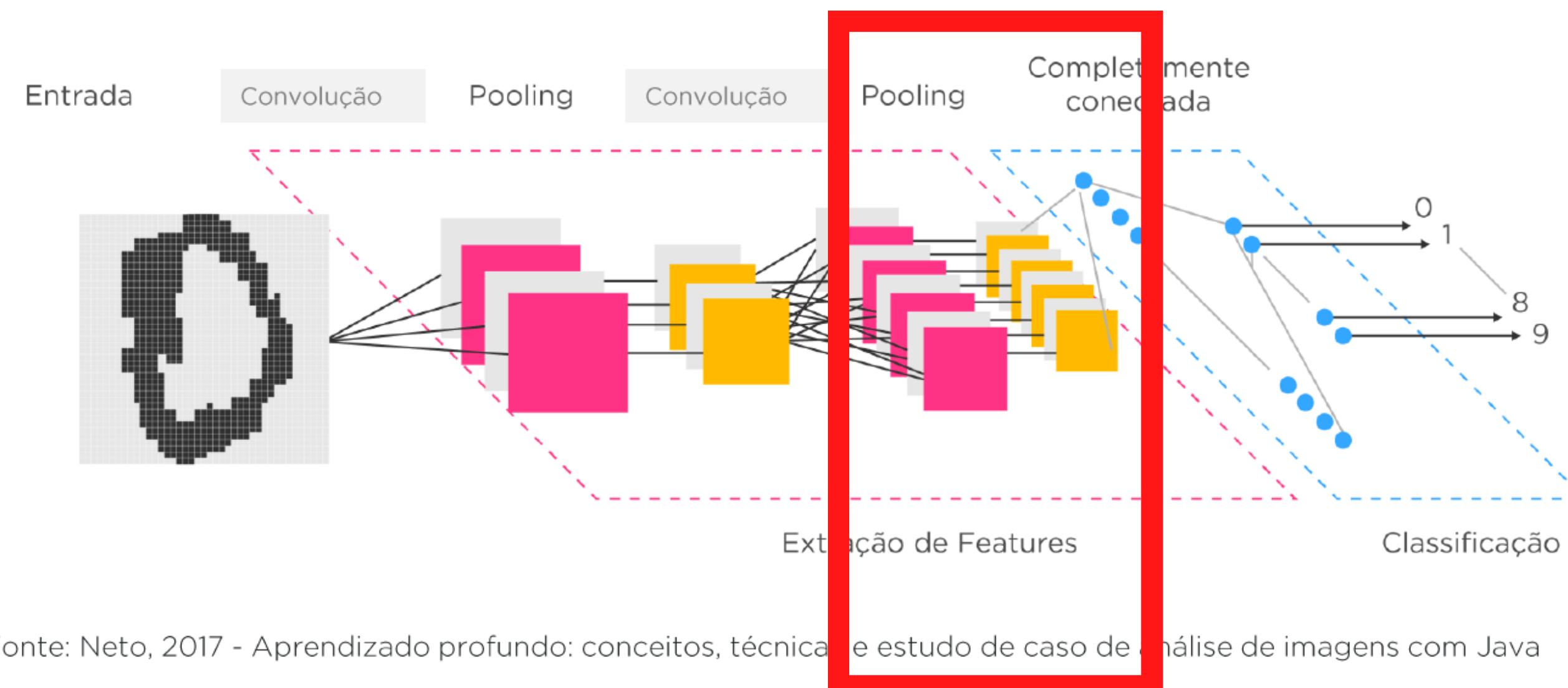


# CONVOLUÇÃO:



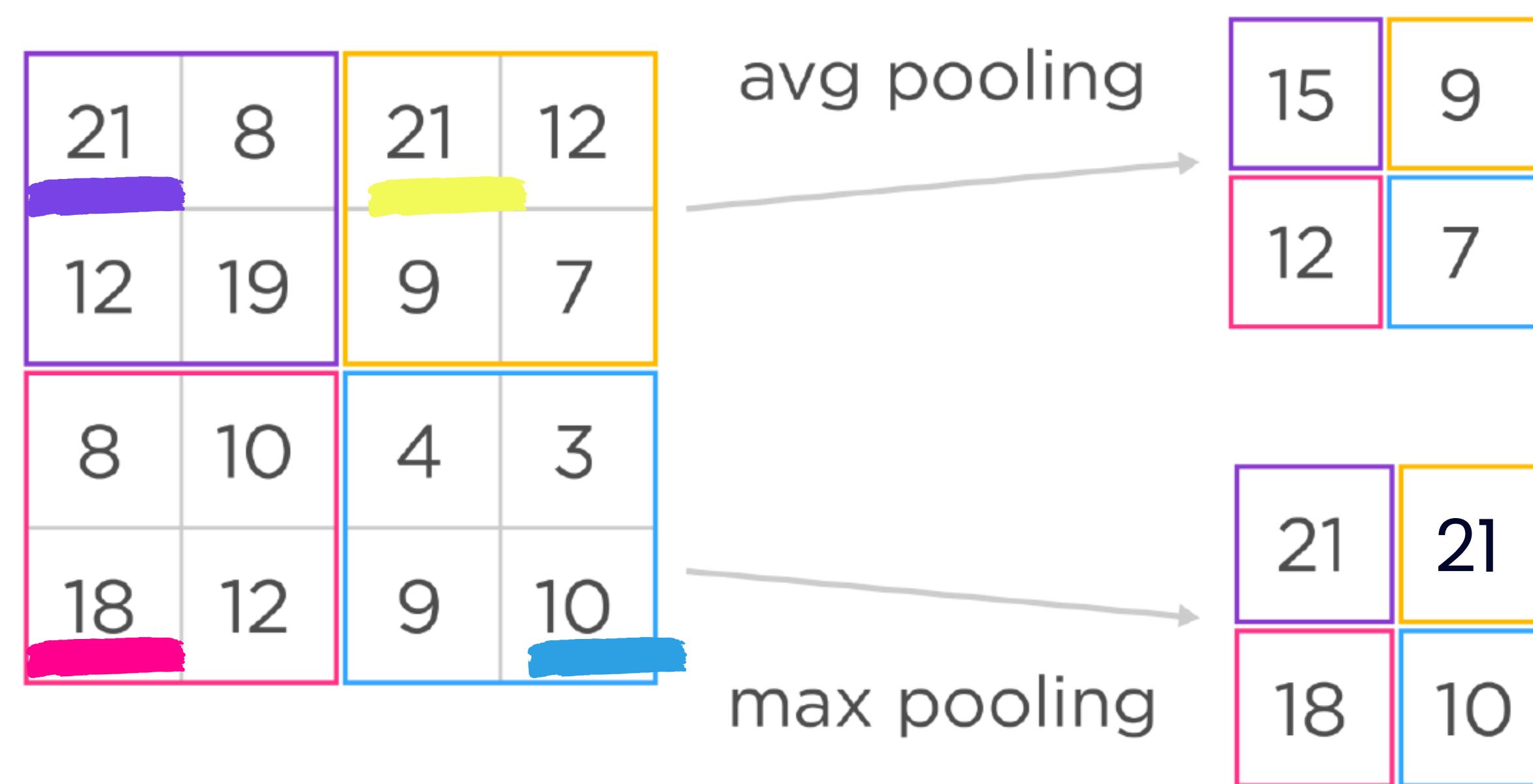
# POOLING:

Processo de redução de dimensionalidade da imagem. A principal motivação dessa operação no modelo, é de diminuir sua variância a pequenas alterações e também de reduzir a quantidade de parâmetros treinados pela rede.



## POOLING:

A operação max pooling retira o maior (max) elemento de determinada região da matrix. Em seguida é realizado um novo calculo de stride (assim como na convolução).



# HANDS ON!

Classificando dígitos escritos a mão.



# DESAFIO

